國 立 清 華 大 學

博 士 論 文

基於數位化變更型邏輯映射混沌系統的安全傳輸

Digital Modified-Logistic Map Based Systems for Secure Communications

系所別　：資訊工程學系 博士班

學號姓名：948301　陳世梁　(Shih-Liang Chen)

指導教授：黃婷婷　博士　(Dr. TingTingHwang)

中華民國　九十九年 十一月

摘要

混沌系統具有不規則，非周期性，無法預測和對於初始條件相當敏感的特質。而這些特質符合與密碼學上混亂和擴散的特性。因此，近年來混沌系統在密碼學上的應用被廣泛的討論與研究。然而，當混沌系統數位化的過程中，原先所保有的混沌特質產生了變化，這種現象又稱為動態特性的降低。一個明顯的例子就是數位化的混沌系統容易產生出一個短周期的輸出軌道。而一個短周期的軌道在統計學的角度上則是容易被分析且不適合應用於密碼系統。在這一篇論文中，我們將研究動態退化的現象並且提出數個方法來提升數位化混沌系統的隨機品質。主要的研究內容如下。首先，我們提出了強化型邏輯映射混沌系統。此系統擁有比傳統邏輯映射混沌系統更大範圍的可用參數，而且這個可用參數範圍內不會存在短周期的參數。基於強化型邏輯映射混沌系統，我們更提出了強化型多維度混沌系統，使其具有更多的可用參數來應用於安全傳輸系統。第二，我們提出了變化型邏輯映射混沌系統。此系統明顯的增加了單位時間的輸出量與隨機品質。此外我們串接數個變化型邏輯映射混沌系統來建立多變化型邏輯映射混沌系統，使其 可容易擴張，並可以快速的產生具有高複雜度與長週期特性的混沌數列。最後，在本論文的第三部分則是針對偽隨機變數產生器應用提出了數位化變更型邏輯映射混沌系統。在這個系統中我們使用了參數選擇與擾動技術，我們有效的減少了系統的計算量並提高了輸出的複雜度。在現行的偽隨機亂數數列測試平台測試結果顯示，相比於先前所提出的混沌偽隨機亂數產生器，我們的系統使用了較低的硬體成本產生了較高隨機品質的偽隨機數列。

Digital Modified-Logistic Map Based Systems for Secure Communications

Student: Shih-Liang Chen
Advisor: Prof. TingTing Hwang

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30013

Nov. 2010

Contents

List of Figures

List of Tables

Abstract

The orbit of a chaotic system is irregular, aperiodic, unpredictable, and sensitive to initial conditions. These characteristics coincide with the confusion and diffusion properties in cryptography. In recent years, chaotic systems have been studied for secure communications.

However, when a chaotic system is digitalized, it results in some unexpected behaviors due to limited precision. It is known as dynamical degradation. An obvious phenomenon is that an orbit enters a cycle with unpredictable length. The orbit with short cycle length has poor quality of randomness because it can be easily analyzed from statistical point of view.

In this dissertation, we focus on improving quality of randomness for digitalized logistic maps. New modified logistic map and techniques are proposed to improve the degree of complexity for secure communications and pseudo random number generation. The main achievements of this dissertation are as follows.

First, we propose a Robust Logistic Map (RLM) which has a larger parameter space than classical logistic map. Moreover, there are no *windows* with short period-length in the parameter space. Based on RLM, a Robust Hyper-Chaotic System (RHCS) is constructed for secure-communication systems with large parameter space.

Second, we propose a Variational Logistic Map (VLM) to significantly increase the throughout and quality of randomness of RLM. Moreover, a Multiple Variational Logistic Map (MVLM) is proposed for fast chaotic sequence generator. Because of the regular architecture of MVLM, it is easy to scale up the system degree to provide long output sequence with high degree of complexity and large key space for secure communications.

Pseudo Random Number Generators (PRNGs) are often an important component in secure communications. In the third part of this dissertation, we propose a PRNG based on a Digitalized Modified Logistic Map (DMLM). Two techniques, constant parameter selec-

tion and output scrambling are employed to reduce the computation cost and to increase the complexity of the PRNG. Compared to previous digitalized chaotic systems based PRNGs, our DMLM-PRNG has better quality of randomness and lower hardware cost.

Each of our system mentioned above has been implemented. Comparisons between our systems and previous work are conducted in terms of hardware cost and throughput. Moreover, the quality of randomness is demonstrated by statistical analysis.

Chapter 1

Overview

Modern communication frameworks, such as Internet and mobile-phone networks, have greatly increased the activities and possibilities of communications. With increasing communication activities, the security issues become more and more important. Thus, a lot of research activities focus on cryptographic techniques to provide secure communications.

A general secure-communication scheme is shown in Figure 1.1. In this scheme, message is transmitted by the *Transmitter* through channels after *Source Encoding*, *Encryption* and *Channel Encoding* & *Modulation*. The *Receiver* recovers the message by reversing these steps.

Basically, the security of the communication is provided by the encryption of the message. The encryption system scrambles the message so that it is unreadable by the non-authorized opponent. Some secret information, *key*, is shared by the *Transmitter* and the *Receiver* for encryption and decryption progress. It should be unknown to the non-authorized opponent. Depending on the management strategy of keys, crypto (encryption and decryption) systems are typically cataloged into two types, the *symmetric* (as known as *secret-key*) crypto systems and the *asymmetric* (as known as *public-key*) crypto systems.

In the symmetric crypto system, such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES), the message is encrypted and decrypted with the same key shared by the transmitter and the receiver, while in the asymmetric crypto system, such

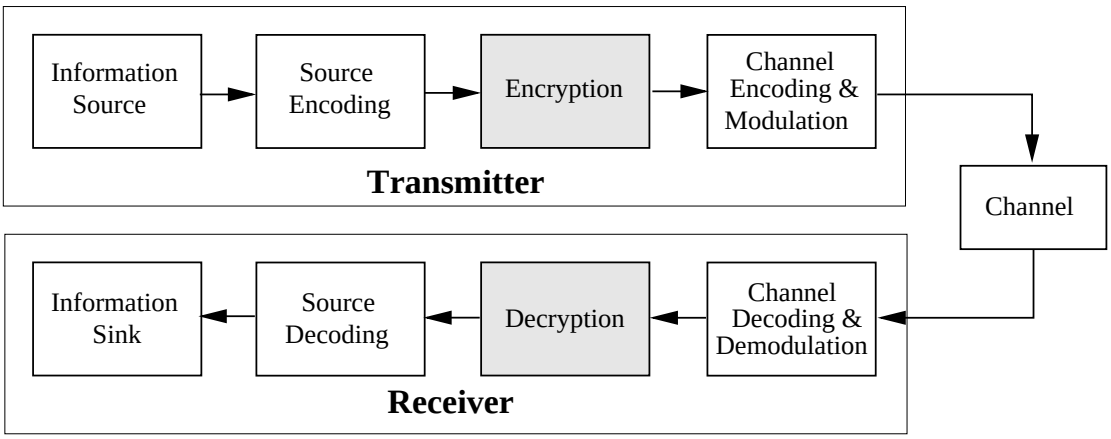


Figure 1.1: General secure-communication scheme.

as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC), the message is encrypted by the transmitter with the receiver's public key. The receiver will decrypt the message by the receiver's private key which is unknown to others. As compared to a asymmetric crypto system, a symmetric one usually has lower computation cost and higher throughput, but lower security level. In a practical embodiment, before message transmission, a asymmetric crypto system can be used to exchange a key, then a symmetric crypto system uses the exchanged key to transmit the message with high throughput. Please refer to chapters 3 and 9 in [18] for the detailed descriptions. In this research, we focus on the secure communications which are based on symmetric crypto systems.

An orbit generated by a chaotic system is irregular, aperiodic, unpredictable and sensitive to initial conditions [1]. These characteristics coincide with the confusion and diffusion properties in cryptography. Thus, since 1990s, chaotic systems have been used in secure communications. These chaotic-cryto systems can be divided into two forms: analog and digital. [2–15].

The analog secure-communication is based on chaos synchronization. The main idea is to mask messages by chaotic signals, then the messages can be recovered by a synchronized

chaotic system [16]. The digital secure-communication is based on the chaos theory. It is based on the random-like behavior of the orbit generated iteratively by a chaotic system [2].

Because the orbit generated by a digital chaotic map is deterministic (computed by computers) and sensitive to initial conditions (parameters and initial values), the pseudo random number generator (PRNG) is a natural application of digital chaotic maps. PRNGs are widely used in many applications, such as numerical analysis, integrated circuit testing, computer games. It is also an important component in secure communications [13, 14, 17].

However, chaotic behaviors become unpredictable when a chaotic system is digitalized. The dynamic degradation is caused by the limited precision used to compute the orbit. The randomness quality of the output is greatly reduced, and the system behavior can be easily analyzed. Hence, a lot of researchers address the problem and proposed methods to improve the output complexity of digital chaotic systems. [11]. Besides complexity of the output, efficiency is also an important issue for a chaos-based ciphers. One may increase the level of complexity by applying complicated chaotic systems, however, it is non-practical because of large computation cost.

In this dissertation, we will propose digital modified logistic maps and techniques to increase the quality of randomness with good compromise between security and efficiency. The proposed systems are suitable for digital secure communications and PRNGs. In the following sections, we will introduce previous work and the organization of this dissertation.

1.1 Previous Work

In analog chaotic secure-communications, chaotic signals are used as masking streams to carry information which can be recovered by the chaotic synchronization behavior between the transmitter and the receiver [19]. It is based on the concept that a chaotic system (drive system) can be synchronized with a separate chaotic system (response system), provided

that the conditional Lyapunov exponents of the difference equations between the drive and response systems are all negative proposed by Pecora and Carrol [16].

Besides, digital chaotic systems are also widely used for generating digital signals for security systems [2, 3, 5, 6, 12, 14, 20]. Among others, Matthews [2] proposed the first chaotic-crypto system based on a logistic map implemented on the computer. At the same time, Wheeler [21] commented that Matthews' system can indeed generate unpredictable sequences. However, with short precision, the system will have a small number of total states.

When chaotic systems are digitalized, it becomes "pseudo-chaotic" because the dynamic behavior is reduced. For example, the cycle length of a orbit is short. Because of the limited precision, there has truncation error for each computation. Two points of the orbit with a difference which is smaller than the truncation error will become the same one after truncating. The short output cycle leads to non-uniform output distribution. In this case, the output is easy to be analyzed and attacked by enumerating all states of output.

Wheeler [21] suggested that digital chaotic system implemented with more digits can solve the problem of short output cycle length. Also, multi-dimensional system constructed by coupled maps [8, 13] and timing-based reseeding method [12] are also proposed to increase the complexity and output cycle length.

Although using higher precision and coupled maps can increase the output cycle length and complexity, it still can not solve another issue that is the small parameter space for chaotic maps. Álvarez [22] pointed out that the usable region of parameter values is a weakness of the discrete-time chaotic system. The chaotic behavior of the system is dependent on the parameters. Unfortunately, all parameters are not equally strong. Some of them will result in *windows*. Note that here a *window* is defined as the chaotic orbit of a non-linear system visualized as periodic on computers (see e.g. [1, p. 356]). The length of orbit generated by the parameter in *window* is fixed no matter how large the precision is

increased to compute the orbit. The remaining parameter space may easily be attacked by brute-force enumeration method because of smaller parameter space. For example, previous systems using logistic maps work only when parameter γ is equal or close to 4 [12]. This constraint makes the key space of a security system smaller than applications require.

Because of low computation cost, a logistic map, serves as popular map to generate chaotic sequence for security systems [5, 9, 12]. In this dissertation, we will propose modified digital logistic maps for secure communications and PRNGs. Compared to a classical logistic map, our maps have larger parameter space and better quality of randomness. Moreover, techniques for increasing the cycle length is proposed for our chaotic systems. With low computation cost and good quality of randomness, proposed systems are suitable for secure communications and PRNGs. The overview of this dissertation is described in the next section.

1.2 Dissertation Overview

In Chapter 2, from our review of previous work, we deduce that to effectively use chaotic maps in digital encryption, a system must meet the following three criteria. First, the length of digital precision must be long enough to prevent the system from being attacked by state enumeration. Second, the parameter space must be large enough for practical use. Finally, the re-construction of the chaotic system must be infeasible using current computational technology.

To solve these problems, we propose a Robust Hyper-Chaotic Encryption-Decryption System (RHCEDS) for secure communications. An RHCEDS consists of two Robust Hyper-Chaotic Systems (RHCS) for the transmitter and the receiver. An RHCS is constructed by coupling robust logistic chaotic maps [23], one carrier map and several hidden maps, so that it has more than one positive Lyapunov exponent. Thus, the RHCS has a higher degree of complexity than traditional discrete-time secure-communication systems

because the former uses multiple coupled chaotic maps rather than a single one [24]. The new proposed system RHCEDS has a large parameter space which grows along with system precision. Hence, the re-construction of our system is not feasible by current computational technology. The statistical analysis of the RHCS shows that the system has good quality of randomness.

In Chapter 3, we will propose a Variational Logistic Map (VLM) with un-restricted parameter space, and can be implemented at lower cost as compared with classical logistic map. Then, we design a Multi-VLM (MVLM) system constructed by VLMs to have output sequence with higher degree of complexity and larger key space than a single VLM. An MVLM constructed by four 32-bit VLMs can generate sequence with cycle length more than 2^{128} with a 128-bit external key. We demonstrate that MVLM can generate output sequence with well quality of randomness with higher throughput and lower hardware cost as compared to previous work.

In Chapter 4, a nonlinear, Digitalized Modified-Logistic Map based Pseudo Random Number Generator (DMLM-PRNG) is proposed for randomness enhancement. Two techniques, constant parameter selection and output sequence scrambling are employed to reduce the computation cost without sacrificing the complexity of output sequence. Statistical test results show that with only one multiplication, DMLM-PRNG passes all cases in SP800-22. Moreover, it passes most of cases in *Crush*, one of the test suite of TesuU01. When compared to solutions based on digitized pseudo-chaotic maps previously proposed in the literature, in terms of randomness quality, our system is as good as a Rényi-map based PRNG and better than a logistic-map based PRNG. Moreover, compared to solutions based on Rényi-map based PRNG, DMLM-PRNG is better scalable to high digital resolutions with reasonable area overhead.

Finally, the conclusion and future work will be given in Chapter 5.

Chapter 2

Digital Secure-Communications Using Robust Hyper-Chaotic Systems

In this chapter, we propose a robust hyper-chaotic system that is suitable for digital secure-communications. The system consists of many coupled robust logistic maps that form a hyper-chaotic system. It has a higher degree of complexity than traditional discrete-time secure-communication systems that use only a single map. Moreover, the system has a very large parameter space which grows along with system precision. Hence, attacking the system by the method of map re-construction in current computation technology is not feasible. Statistical analysis shows that the system achieves very high security level. Finally, two hardware architectures (multiple-cycle and pipelined) are proposed for area and performance optimization, respectively.

The rest of this chapter is organized as follows. In Section 2.1, our target system Robust Hyper-Chaotic Systems (RHCS) and a Encryption/Decryption scheme Robust Hyper-Chaotic Encryption-Decryption System (RHCEDS) will be presented. In Section 2.2, the cryptanalysis will show that our system is suitable for secure communications. In Section 2.3, we present the hardware implementation to demonstrate our RHCEDS. Finally, summary is given in Section 2.4.

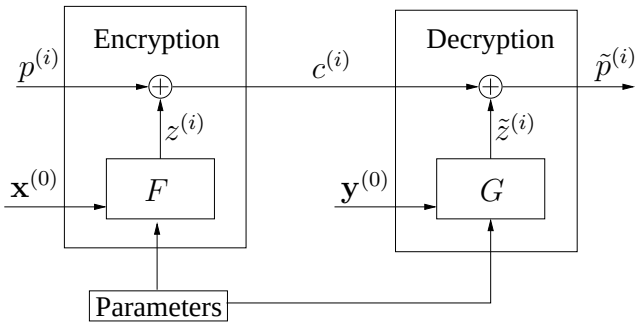


Figure 2.1: The architecture of RHCEDS.

2.1 Robust Hyper-Chaotic Encryption-Decryption System

The crypto system is defined as communications between the Encryption layer and the Decryption layer in a general secure-communication scheme. An architecture of crypto system is shown in Figure 2.1. Given an initial vector $\mathbf{x}^{(0)} = [x_1^{(0)}, \ldots, x_n^{(0)}]^\top$, parameters including an n-by-n stochastic matrix $\boldsymbol{C} = [c_{ij}]$ and a chaotic parameter vector $\mathbf{r} = [\gamma_1, \ldots, \gamma_n]^\top$, where $x_i^{(0)} \in \{(0,1)\backslash\{\frac{1}{2}\}\}$, $\gamma_i \geq 4$ for $i = 1, \ldots, n$ and $0 < c_{ij} < 1$ for $i, j = 1, \ldots, n$., the RHCEDS is constructed by two RHCSs, denoted by F and G, respectively. At the encryption end, a masking sequence $z^{(i)}$ is generated by the system $F(\mathbf{r}, \mathbf{x})$ and used for encrypting the plaintext $p^{(i)}$. At the decryption end, the receiver recovers the plaintext from the ciphertext $c^{(i)}$ by removing the mask $\tilde{z}^{(i)}$ generated by the system $G(\mathbf{r}, \mathbf{y})$.

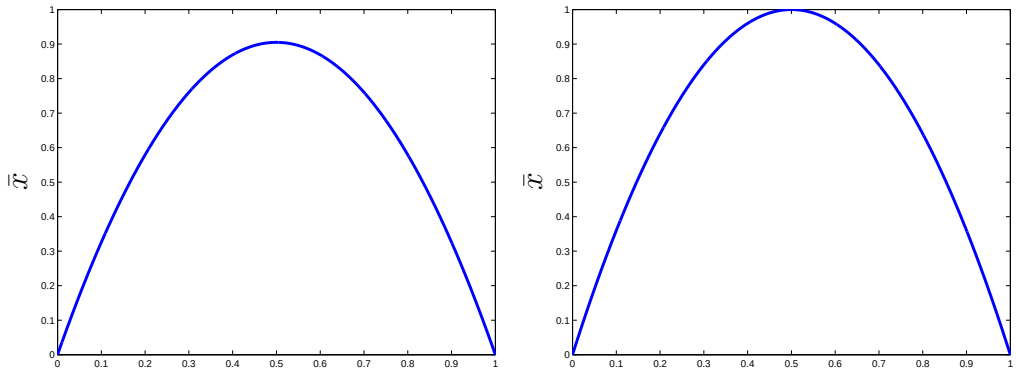
2.1.1 Robust Logistic Map

Before introducing the RHCS, we present a robust logistic map which is developed from a classical logistic map.

A classical logistic map is defined by

$$\bar{x} = \gamma x\,(1 - x)\,, \quad x \in [0, 1], \tag{2.1}$$

where γ is a parameter and $0 \leq \gamma \leq 4$. In Equation (2.1), when $3.57 < \gamma \leq 4$, it is a *chaos*



(a) $\gamma = 3.62$ (b) $\gamma = 4$

Figure 2.2: Classical logistic maps with $\gamma = 3.62$ and 4.

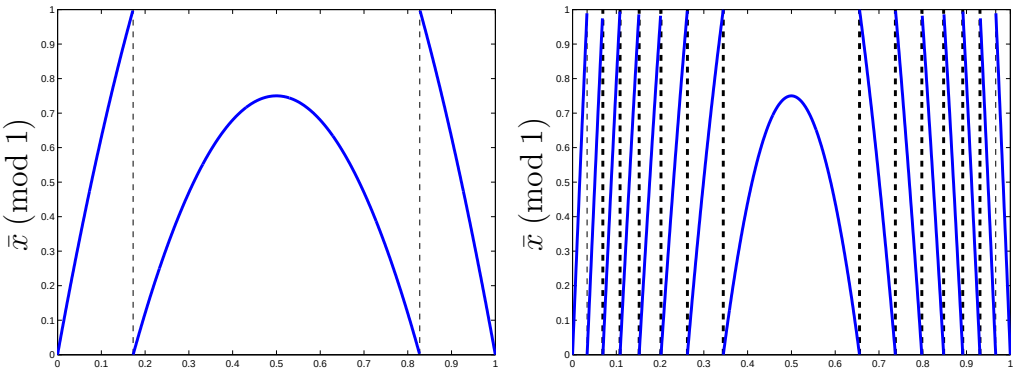
region and the generated sequence is non-periodic. However, the set of parameters γ that result in *windows* of Equation (2.1) is open and dense. Moreover, the chaotic attractor is not fully distributed within the range of 0 to 1 and its length is less than one. In this case, γ is easily detected by measuring the length of chaotic attractors. For example, in Figure 2.2(a), when $\gamma = 3.62$, the length of attractor is 0.594. The only useful case of Equation (2.1) is when $\gamma = 4$ because its chaotic attractor is fully distributed in the range of 0 to 1 as shown in Figure 2.2(b). Therefore, the selection of γ values is limited.

In order to increase the parameter space and to have a fully distributed map in [0,1], we propose a robust logistic function as follows:

$$L(\gamma, x) = \begin{cases} \gamma x(1-x) \pmod 1, & x \in I_{\text{ext}}, \\ \frac{\gamma x(1-x) \pmod 1}{\frac{\gamma}{4} \pmod 1}, & x \in I_{\text{int}}, \end{cases} \tag{2.2}$$

where $I_{\text{ext}} \in (0,1) \setminus I_{\text{int}}$ (do not belong I_{int}), $I_{\text{int}} = [\eta_1, \eta_2]$, $\eta_1 = \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{[\frac{\gamma}{4}]}{\gamma}}$ and $\eta_2 = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{[\frac{\gamma}{4}]}{\gamma}}$ in which $[w]$ is the greatest integer less than or equal w. A robust logistic map (RLM) is then defined by $x^{(i+1)} = L(\gamma, x^{(i)})$.

By this modification, we extend the γ range to a value more than 4. When $L(\gamma, x)$ is greater than 1, the first equation in Equation (2.2) is to shift the map value greater than 1 to the range of 0 to 1. Figure 2.3 shows that modular one operation keeps x invariant in



(a) $\gamma = 7$ (b) $\gamma = 31$

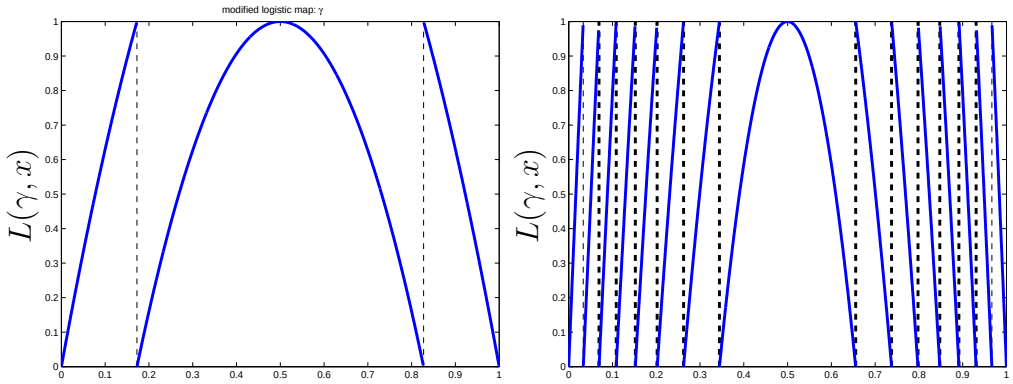
Figure 2.3: The mapping without normalization of x vs. $L(\gamma, x)$ with $\gamma = 7$ and 31.

[0,1]. However, when x is in the range I_{int}, the mapping is not fully distributed in [0,1], it results in *window* of the map. Therefore, when $L(\gamma, x)$ is less than 1, the second equation in Equation (2.2) is to scale the value to the range of 0 to 1. With both modular and scaling operations, Figure 2.4 shows that two maps are fully distributed in [0,1] with piecewise nonlinear map when $\gamma = 7$ and 31.

To understand if there are *windows* in our robust logistic map when $r \geq 4$, we analyze the map by numerical methods. First, we compute the Lyapunov exponents by the method in [25]. In Figure 2.5, Lyapunov exponents of Equation (2.2) are computed from $\gamma = 0$ to 16. It shows when $\gamma \geq 4$, Lyapunov exponents are all positive. Next, we compute the bifurcation diagram of $L(\gamma, x)$ from $\gamma = 0$ to 16. The result is shown in Figure 2.6. It shows that, when $\gamma \geq 4$, $L(\gamma, x)$ is fully distributed in the range of 0 to 1 and there is no *window*. These numerical results indicate that the robust logistic map is indeed chaotic with large parameter space when $\gamma \geq 4$.

2.1.2 Construction of Robust Hyper-Chaotic System

To solve the dynamical degradation and increase the output complexity, methods [20, 26–29] with coupled map lattice for multi-dimensional system were proposed. Hu *et al*. [26]



(a) $\gamma = 7$ (b) $\gamma = 31$

Figure 2.4: The mapping with normalization of x vs. $L(\gamma, x)$ with $\gamma = 7$ and 31.

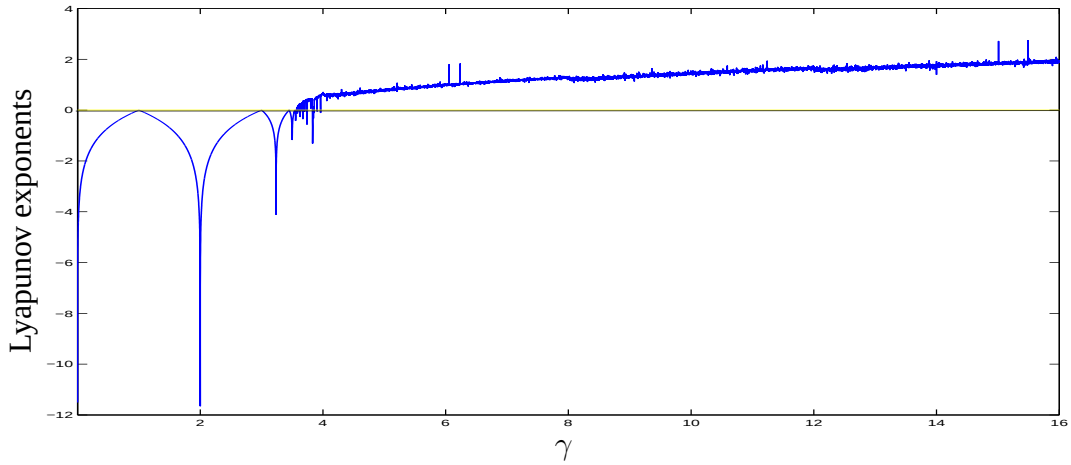


Figure 2.5: Lyapunov exponents vs. γ for $\gamma \in [0, 16]$.

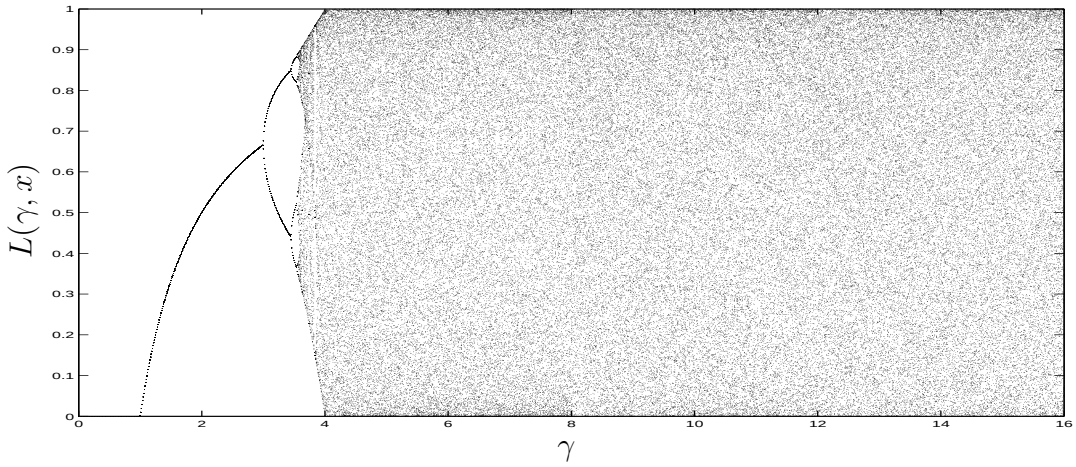


Figure 2.6: Bifurcation diagram of $L(\gamma, x)$ for $\gamma \in [0, 16]$.

presented a synchronous chaotic spread-spectrum CDMA system. Lu *et al*. [27] developed a spatiotemporally chaotic cryptosystem with one-way-coupled. Li *et al*. [20, 28, 29] generated multiple pseudo-random-bit sequences (or multiple keystreams) by spatiotemporal chaotic systems, logistic maps and skew tent maps. Their results showed that coupled chaotic maps can be a good candidate for generating chaotic sequence for security systems.

Based on a coupled map lattice a robust hyper-chaotic system (RHCS) can be constructed. The system is defined by

$$\mathbf{x}^{(i)} \quad = \quad F(\mathbf{r}, \mathbf{x}^{(i-1)}) := \mathbf{C}\mathcal{L}(\mathbf{r}, \mathbf{x}^{(i-1)}), \tag{2.3}$$

where $\mathbf{x}^{(i)} = [x_1^{(i)}, \ldots, x_n^{(i)}]^\top$, $\mathcal{L}(\mathbf{r}, \mathbf{x}^{(i-1)}) = \left[L(\gamma_1, x_1^{(i-1)}), \ldots, L(\gamma_n, x_n^{(i-1)})\right]^\top$, in which L is the robust logistic map defined in Equation (2.2), and

$$\mathbf{C} \quad = \quad \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

is a positive stochastic coupling matrix with all elements $0 < c_{ij} < 1$ and $\sum_j c_{ij} = 1$ for $i, j = 1, \ldots, n$. The masking sequence is defined by

$$z^{(i)} \quad = \quad x_1^{(i)}. \tag{2.4}$$

The system G is also an RHCS defined by

$$\mathbf{y}^{(i)} \quad = \quad G(\mathbf{r}, \mathbf{y}^{(i-1)}) := \mathbf{C}\mathcal{L}(\mathbf{r}, \mathbf{y}^{(i-1)}), \tag{2.5}$$

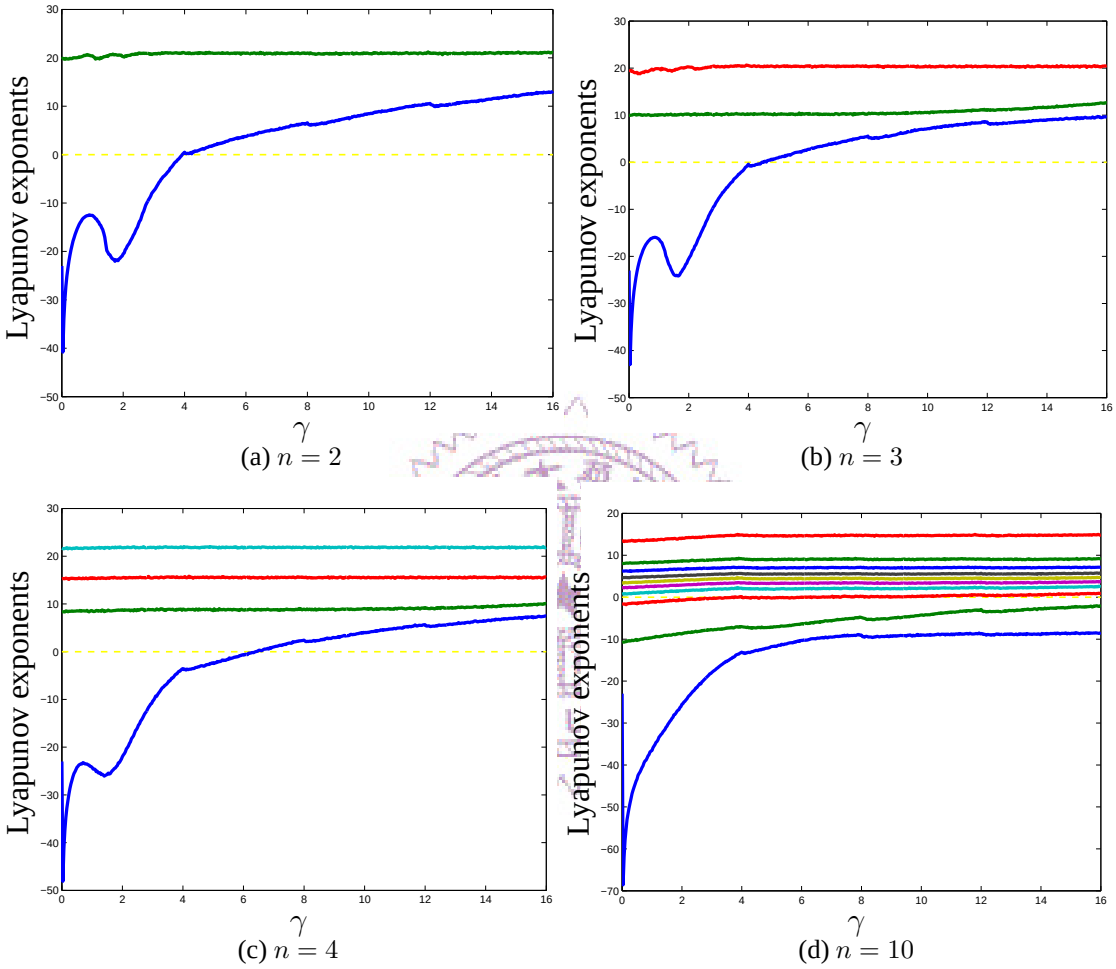
where $\mathbf{y}^{(i)} = [y_1^{(i)}, \ldots, y_n^{(i)}]^\top$ for $i > 0$. The unmasking sequence is defined by

$$\tilde{z}^{(i)} = y_1^{(i)}. \tag{2.6}$$

Note that F and G are hyper-chaotic systems in $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$, respectively, with the same parameters of $\mathbf{C}$ and $\mathbf{r}$.

The RHCS (F or G) is constructed by n-coupled robust logistic maps and each robust logistic map in the system has its own positive Lyapunov exponent. To understand if the dimension of the whole system in terms of the number of positive Lyapunov exponents is indeed increased, we analyze the RHCS by numerically. Since the higher dimension of the system, the more positive Lyapunov exponents the RHCS has. Hence, we expect that the behavior of the output masking sequence ($z^{(i)}$) is more complex. The number of coupled robust logistic maps being set to 2 (i.e., $n = 2$) is taken as our example. In this case, there are two parameters γ_1 and γ_2 for two robust logistic maps. In Figure 2.7(a), two Lyapunov exponents of 2-coupled robust logistic map are plotted for $\gamma_1 = 0$ to 16 with the scale of $\frac{1}{30}$, and a fixed $\gamma_2 = 29.6668$. The result shows when $\gamma_1 \geq 4$, two Lyapunov exponents are both positive, that is, the system is hyper-chaotic without *window*. Similarly, the number of Lyapunov exponents for $n = 3$, 4 and 10, where values of γ_i, $1 < i \leq n$ are fixed, and the range of γ_1 is from 0 to 16, are shown in Figure 2.7(b)(c)(d), respectively. We can see that the number of positive Lyapunov exponents of the system are increasing without *window* as n increased, provided that all γ_i in the system are larger than 4.

In order to encrypt and decrypt information correctly, the masking sequence $z^{(i)}$ must be identically synchronized to the unmasking sequence $\tilde{z}^{(i)}$. We first randomly create an initial vector $\mathbf{x}^{(0)}$ of the transmitter, and then send it to the receiver by replacing its initial vector $\mathbf{y}^{(0)}$ by $\mathbf{x}^{(0)}$. After this step, it holds that $z^{(i)} = \tilde{z}^{(i)}$ for $i > 0$. Then the RHCEDS



(a) $n = 2$ (b) $n = 3$

(c) $n = 4$ (d) $n = 10$

Figure 2.7: Lyapunov exponents vs. γ for $n = 2, 3, 4$ and 10.

is ready for information transmission. On the other hand, if the bandwidth of the channel is just one component of $\mathbf{x}^{(0)}$, then n steps are required to send n elements of the initial vector to the receiver. Therefore, after n steps, the vector $\mathbf{y}^{(0)}$ will be equal to $\mathbf{x}^{(0)}$.

2.1.3 Encryption & Decryption

In our secure-communication system, RHCEDS, the masking sequence of system F will be used as a mask to encrypt plaintext. In other words, the cryptograph system is similar to an one-time-pad block cipher. In this case, the randomness of the masking sequence directly affects the security level of the system. To enhance the randomness of the masking sequence, the ℓ most significant digits are hidden in communications, that is, these ℓ digits are dropped and not used in the encryption. The more hidden digits are used, the more difficult to analyze the encrypted information. However, the increased security is at the expense of more computing resource. In our experimental results, hiding two-digits is found to have good randomness, which is examined by a random number testing package, NIST SP 800-22 [30].

In summary, our secure-communication system, RHCEDS, is implemented as follows.

In Transmitter:

We use m digits to represent all real numbers in the system F including parameters $\mathbf{r}$ and $\mathbf{C}$, and the initial vector $\mathbf{x}^{(0)}$. Given $d = m - \ell \in \mathbb{N}$, for $i \geq 1$, the plaintext $\boldsymbol{p}$ is decomposed into a sequence of $\{p^{(i)}\}$ with the length of each $p^{(i)}$ equal to d digits. The encryption process is as follow:

$$\begin{aligned} z^{(i)} &= \left\lfloor x_1^{(i)} \right\rfloor_\ell, \\ c^{(i)} &= z^{(i)} \oplus p^{(i)}, \end{aligned}$$

where $\oplus$ is an XOR operation, and $\lfloor x \rfloor_\ell$ means dropping the first ℓ digits from x.

In Receiver:

In receiver, the decrypted sequence, $\tilde{\boldsymbol{p}}$, is as follow:

$$\begin{aligned} \tilde{z}^{(i)} &= \left\lfloor y_1^{(i)} \right\rfloor_\ell, \\ \tilde{p}^{(i)} &= \tilde{z}^{(i)} \oplus c^{(i)}. \end{aligned}$$

Since systems F and G have the same initial vector and $z^{(i)} = \tilde{z}^{(i)}$, we can correctly decode ciphertext, that is, $\tilde{\boldsymbol{p}} = \boldsymbol{p}$.

From the above descriptions, the properties of the RHCEDS can be summarized as follows:

- There are n^2 selections of parameters to form $\mathbf{r}$ and $\mathbf{C}$. The large parameter space makes the attacking by brute-force enumeration infeasible.
- For the same plaintext, the crypto system can generate different ciphertexts with different initial vectors.
- Incomplete carrier map is transmitted in the public channel. Therefore, it is hard to re-construct the map even under the assumption of "chosen plaintext" attack.

2.2 Cryptanalysis of RHCDES

The cryptanalysis of our system will be based on an example where the precision of the system is $m = 8$, and the number of coupled robust maps is 2. With $n = 2$, the masking stream generator F is shown in Equation (2.7).

$$\begin{cases} x_1^{(i)} &= c_{11} L(\gamma_1, x_1^{(i-1)}) + (1 - c_{11}) L(\gamma_2, x_2^{(i-1)}), \\ x_2^{(i)} &= (1 - c_{22}) L(\gamma_1, x_1^{(i-1)}) + c_{22} L(\gamma_2, x_2^{(i-1)}). \end{cases} \tag{2.7}$$

2.2.1 Parameter Space

Attackers may construct a chaotic map by identifying its unique orbit if the key space is small. Therefore, the parameter space must be large enough for practical use.

According to the bifurcation diagram in Figure 2.6 and Lyaponov exponents in Figure 2.5, we found that our robust logistic map has no $windows$ when $\gamma \geq 4$.

Therefore, we can judiciously choose a stochastic matrix $\mathbf{C}$ and $\mathbf{r}$ to create an n-dimensional system with at least two positive Lyapunov exponents. That is, the system (2.3) has no $window$, which guarantees that there is no scruple by picking the parameters to construct a hyper-chaotic system. Furthermore, the parameter space of the system (2.3) is large enough for any practical application. For example, in Equation (2.7), there are four parameters c_{11}, c_{22}, γ_1 and γ_2 and the total number of parameters that can be selected is $2^{4\times 32} = 2^{128}$. This parameter space is much larger than 2^{100} which is the suggested size for parameter selection in [11, 22].

Moreover, one important property of the parameter is worth noticing. The generated masking sequence has a very sensitive dependence on the parameters. Without this property, attackers can easily find the relationship between parameters and their corresponding masking sequences.

To show this property, an experiment is conducted. First, the masking stream generator F shown in Equation (2.7) is taken as an example. Next, a set of $\mathbf{C}$ and $\mathbf{r}$ parameters are selected as base to generate a base masking sequence S_{base}. Then, 200 γ_1 are generated by varying the least significant bits of base γ_1. With different γ_1 and the same γ_2 and $\mathbf{C}$, 200 masking sequences are generated where $S_{base\pm d\times 2^{-32}}$, $d = 1, \ldots, 100$ denote the masking sequences. Finally, we compute bit error rate (BER) between S_{base} and $S_{base\pm d\times 2^{-32}}$. The result is shown in Figure 2.8. It can be seen that the generated sequences are indeed different even with a small change by 2^{-32} in one parameter.

2.2.2 Re-construction

Attackers may plot the map by analyzing output sequences of a chaotic map. Unrolling a system is a method to compute the values of unknown parameters. In our system, for

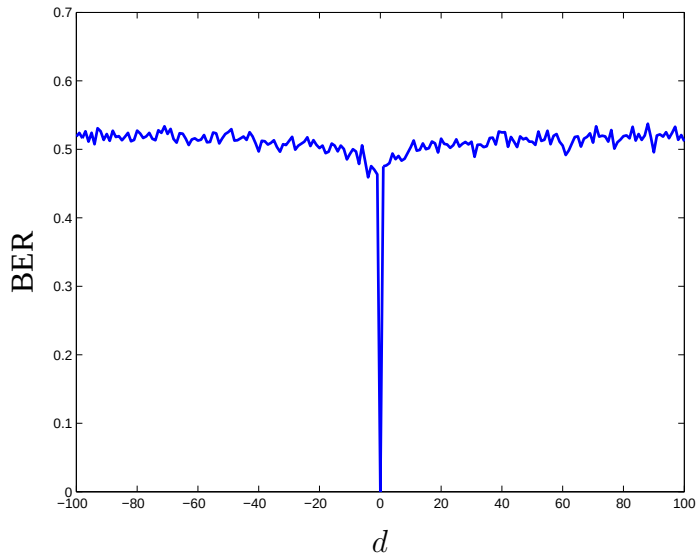


Figure 2.8: BER between S_{base} and $S_{base \pm d \times 2^{-32}}$.

example, when $i = 1$, Equation (2.7) has five unknown variables, $\gamma_1, \gamma_2, c_{11}, c_{22}$ and $x_2^{(1)}$. Unrolling the system to $i = 4$, attackers will have eight equations with additional three unknown variables, $x_2^{(2)}$, $x_2^{(3)}$ and $x_2^{(4)}$. Totally, eight equations are given to solve eight unknown variables. However, in RHCS, it is infeasible for an attacker to re-construct the map by unrolling because of the following two features of our system. First, the masking sequence $z^{(i)}$ is an incomplete output sequence of the system F. The most significant ℓ digits are dropped, that is, $z^{(i)} \neq x_1^{(i)}$. If there are four $x_1^{(i)}$ in the equations, each of $z^{(i)}$ drops j bits, the possible combinations of four $x_1^{(i)}$ are $(2^j)^4$. Second, mapping function is computed using the modular one operation in our robust logistic map. The piecewise non-linear map is not a one-to-one mapping. Given an output of L map, there are $\lceil \frac{\gamma}{4} \rceil \times 2$ possible inputs. There are eight L maps needed to be solved in this example. The combination of solutions are $(\lceil \frac{\gamma}{4} \rceil \times 2)^8$. Assuming that γ is less than 2,048, and j is 8, the attackers in total need to try $(2^8)^4 \times 1,024^8$ possible combinations of equations to solve the unknown variables taking the above two features into account. If we use a computer with 1 THz (Tera Hertz) CPU to run 10^{12} cases per second, then for the above example, it requires near

one million years to re-construct the system F. It is obvious that re-construction of RHCS is infeasible using current computation technology.

2.2.3 Statistical Analysis

To understand how precision affects randomness, we conduct randomness test for $m = 4$ to $m = 12$, where m denotes the number of digits (4-bit for one digit). SP800-22 testing package [30] is used in our analysis process to check the randomness of our system. The masking sequence of the system F is $\left\lfloor x_1^{(i)} \right\rfloor_2$ where the most significant 2 digits of the ${x_1}^{(i)}$ are dropped. Each test will produce a "p-value" from SP800-22 testing package. The higher p-value (a minimal default value is recommended by 0.01), the more random the test case. For each precision we choose three different γ_1 in the RHCS system while keeping the other parameters, γ_2, c_{11} and c_{22}, unchanged. For each γ_1, 100 sequences generated by RHCS with the length of 10^6 bits are fed to the testing package. As suggested in SP800-22, for each statistical test, the minimum pass rate of a well random source is 0.97 out of 100 binary sequences. Table 2.1 shows the result for $m = 4$ to 8. With this standard, we can see that when m is less than 8, the randomness is obviously alleviated. On the contrary, as shown Table 2.2, when m is larger than 8, the generated output sequences are indeed random.

2.3 System Demonstration

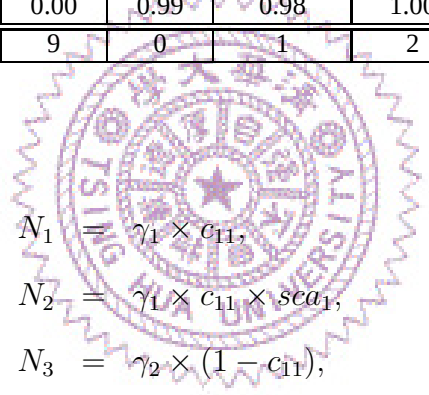
2.3.1 Architecture of Encryption System

To demonstrate the effectiveness of the system F, we implement it in hardware. In our design, the number of coupled robust logistic maps is selected to be 2.

Let sca_1 and sca_2 denote two scaling factors, $\frac{1}{\frac{\gamma_1}{4} \pmod 1}$ and $\frac{1}{\frac{\gamma_2}{4} \pmod 1}$, respectively. Since parameters γ_1, γ_2, sca_1, sca_2, c_{11}, and c_{22} are constants, $N_1, \ldots, N_8$ are precalculated to reduce computation cost, where

Table 2.1: The SP800-22 test results for $m = 2$ to 8 with $\gamma_2 =$ `1709.ffd3`, $c_{11} =$ `0.c8`, $c_{22} =$ `0.ce`

	$m = 4$			$m = 6$			$m = 8$		
γ_1(HEX)	`100`	`2d49`	`7b63`	`100.80`	`2d49.ff`	`7b63.3b`	`100.80`	`2d49.ff`	`7b63.3b`
Frequency	0.00	0.00	0.16	1.00	1.00	1.00	1.00	1.00	0.99
Block freq.	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.98
Cumulative	0.00	0.00	0.70	1.00	0.99	0.99	1.00	0.99	1.00
Runs	0.00	0.00	1.00	1.00	0.96	0.85	0.99	0.96	0.99
Longest run	0.93	0.00	0.00	0.99	0.98	0.98	0.97	0.98	0.98
Rank	0.99	1.00	0.88	0.99	0.99	1.00	0.98	0.99	0.98
FFT	0.00	0.00	0.00	0.99	1.00	0.98	0.99	1.00	0.98
Nonoverlap.	0.79	0.24	0.85	0.99	0.99	0.99	0.99	0.99	0.99
Overlap.	0.00	0.00	0.00	1.00	1.00	0.99	0.98	1.00	0.98
Universal	0.86	1.00	1.00	0.97	1.00	0.98	0.99	1.00	0.99
Apen	0.00	0.00	0.00	1.00	1.00	0.94	0.99	1.00	0.98
Random e.	1.00	0.00	0.98	1.00	0.99	0.99	0.97	0.99	0.99
Random e.v.	1.00	0.00	1.00	1.00	0.99	0.99	0.99	0.99	0.99
Linear Comp.	0.97	1.00	1.00	0.98	0.99	0.99	0.97	0.99	0.98
Serial	0.00	0.00	0.00	0.99	0.98	1.00	0.99	0.98	0.99
Fail Count	11	11	9	0	1	2	0	1	0



$$
\begin{aligned}
N_1 &= \gamma_1 \times c_{11}, \\
N_2 &= \gamma_1 \times c_{11} \times sca_1, \\
N_3 &= \gamma_2 \times (1 - c_{11}), \\
N_4 &= \gamma_2 \times (1 - c_{11}) \times sca_2, \\
N_5 &= \gamma_1 \times (1 - c_{22}), \\
N_6 &= \gamma_1 \times (1 - c_{22}) \times sca_1, \\
N_7 &= \gamma_2 \times c_{22}, \\
N_8 &= \gamma_2 \times c_{22} \times sca_2.
\end{aligned}
$$

The four conditions to determine if a modular or scaling operation is to be performed are: $\eta_1 = \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{[\frac{\gamma_1}{4}]}{\gamma_1}}$, $\eta_2 = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{[\frac{\gamma_1}{4}]}{\gamma_1}}$, $\eta_3 = \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{[\frac{\gamma_2}{4}]}{\gamma_2}}$ and $\eta_3 = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{[\frac{\gamma_2}{4}]}{\gamma_2}}$.

Table 2.2: The SP800-22 test results for $m = 10$ and 12 with $\gamma_2 =$ `1709.ffd3`, $c_{11} =$ `0.c8`, $c_{22} =$ `0.ce`

	$m = 10$			$m = 12$		
γ_1(HEX)	`100.80`	`2d49.ff`	`7b63.3b`	`100.80`	`2d49.ff`	`7b63.3b`
Frequency	0.99	0.98	1.00	1.00	0.99	1.00
Block freq.	1.00	0.98	0.99	0.99	0.98	0.99
Cumulative	0.99	0.98	1.00	1.00	0.99	1.00
Runs	0.99	0.98	0.99	0.99	0.99	0.99
Longest run	1.00	1.00	0.99	1.00	1.00	1.00
Rank	1.00	0.98	0.98	1.00	1.00	1.00
FFT	0.99	0.98	0.98	1.00	0.99	0.97
Nonoverlap.	0.99	0.99	0.99	0.99	0.99	0.99
Overlap.	0.99	0.99	0.97	0.99	0.99	0.98
Universal	0.99	1.00	0.99	0.98	0.98	0.99
Apen	0.98	1.00	1.00	0.99	0.99	1.00
Random e.	0.99	0.98	0.98	0.99	0.98	0.98
Random e.v.	0.99	0.99	0.99	0.99	0.99	1.00
Linear Comp.	1.00	0.97	1.00	1.00	0.97	0.98
Serial	0.99	0.98	0.98	0.99	1.00	1.00
Fail Count	0	0	0	0	0	0

Since γ_1 and γ_2 are given by the user and remain unchanged during operation, η_1, η_2, η_3, and η_4 are all input vectors to the system. Let $s_1(s_2)$ be a flag variable and $s_1 = 1(s_2 = 1)$ when $\eta_1 < x_1^{i-1} < \eta_2$ ($\eta_3 < x_2^{i-1} < \eta_4$) holds. The system F (Equation 2.7) can be rewritten as

$$x_1^{(i)} = \begin{cases} N_1 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_3 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 0, s_2 = 0, \\ N_2 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_3 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 1, s_2 = 0, \\ N_1 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_4 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 0, s_2 = 1, \\ N_2 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_4 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 1, s_2 = 1, \end{cases}$$

$$x_2^{(i)} = \begin{cases} N_5 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_7 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 0, s_2 = 0, \\ N_6 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_7 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 1, s_2 = 0, \\ N_5 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_8 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 0, s_2 = 1, \\ N_6 \times x_1^{(i-1)} \times (1 - x_1^{(i-1)}) + N_8 \times x_2^{(i-1)} \times (1 - x_2^{(i-1)}), & s_1 = 1, s_2 = 1. \end{cases}$$

The data flow of system F is shown in Figure 2.9. In this flow, 6 multiplications are required to generate one mask, $z^{(i)}$.

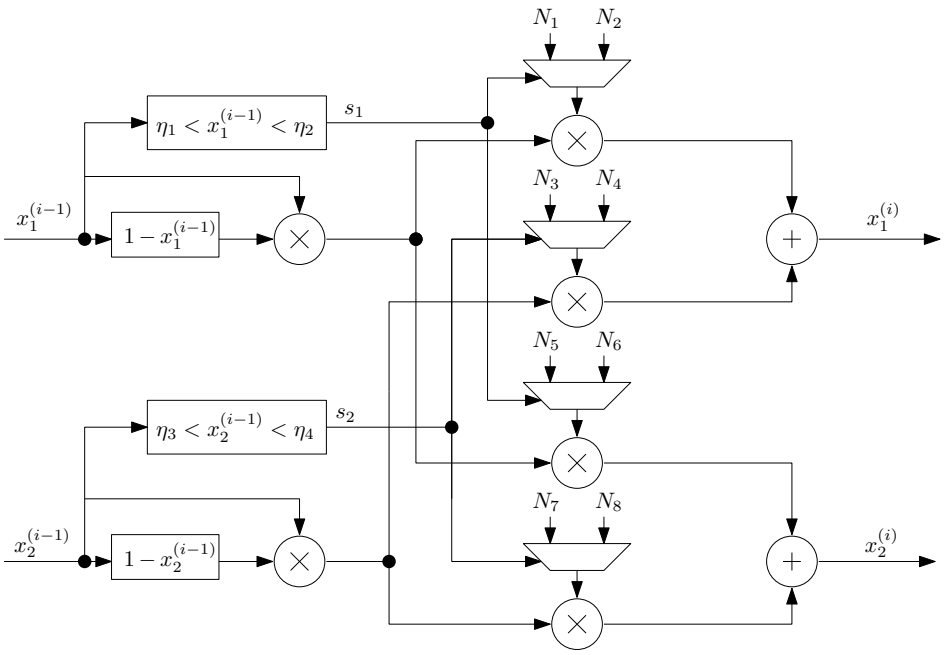


Figure 2.9: The data-flow of the mask generator.

To understand the tradeoff between area and performance, we will propose two architectures to implement system F. The first one is for area and the second one is for performance. Let us look at the first design. Since it is for area efficiency, multiple-cycle architecture is adopted where only one multiplier and one adder are used and all multiply and add operations use the same hardware at different cycle. Figure 2.10 shows the block diagram of system F in hardware. In this design, a two-stage pipelined multiplier is implemented. Hence, it requires 6 cycles to generate one mask. Besides the two-stage multiplier, the system has two registers, "RegA" and "RegB", for temporary data storage and four add/subtracters. Block "NEG" computes $NEG(x) = 1 - x$ and block "IntCheck" is used to check if the input is in I_{int} or not.

The second design is for performance efficiency. Pipelined architecture is adopted. The data flow of our system is partitioned into 2 stages separated by registers, and hence a 2-stage pipelined design. The data flow is shown in Figure 2.11 and the block diagram in

Figure 2.12. In this design, four multipliers are used and run in concurrency. One mask is generated at every cycle.

We describe our multiple-cycle and pipelined architectures in hardware description language (HDL), and then synthesize them by commercial tools. To be more specific, two designs are written in Verilog and synthesised by Synopsys Design Compiler with TSMC $.18\ um$ technology library. Area and timing information is obtained in gate-level netlist.

Moreover, we want to understand the hardware overhead when precision $m = 8$ is increased to $m = 12$. Implementations for $m = 8$ and $m = 12$ are performed. That is, all real numbers in the system is represented by 8 (12) digits. Then, in hexadecimal representation (one digit is 4 bits), the system operates in 32(48) bits. The number of hidden digits, ℓ is selected to be 2. With 2 hidden digits, the length of one masking stream is 24 (40) bits. Hence, the plaintext sequence will be divided into segments of length of 24 (40) bits.

Table 2.3 shows the synthesized results. When $m = 8$, the experimental results show that the transmitter F of multiple-cycle design achieves an encryption rate of 400M bits per second with 9.4K gate count. When implemented in the pipelined architecture, the system generates mask sequence at a rate of 2.4G bits. That is, our pipelined architecture is 600% faster than the multiple-cycle one. However, the area of pipelined architecture is 401% larger than that of multiple-cycle one. Moreover, by increasing $m = 8$ to $m = 12$, for multiple-cycle architecture, the system performance is 167% faster with 233% more area; for pipelined architecture, 1000% faster with 938% more area.

2.3.2 Example

We use the following parameters to demonstrate the system F with $m = 12$ and $n = 2$.

Table 2.3: The synthesized result of encryption system.

	$m = 8$		$m = 12$	
Architecture	multiple-cycle	pipelined	multiple-cycle	pipelined
Gate Count(k)	9.4	37.7	21.9	88.2
Throughput	1/6	1	1/6	1
Mask Length(bits)	24	24	40	40
Clock Frequency(Mhz)	100	100	100	100
Bits Per Second(M bits)	400	2400	667	4000
Area Ratio	1	4.01	2.33	9.38
Performance Ratio	1	6	1.67	10

$$
\begin{aligned}
x_1^{(0)} &= 0.26e7bf70710c \\
x_2^{(0)} &= 0.3cebe4e04ecb \\
\gamma_1 &= 15.0000000000 \\
\gamma_2 &= 23.0000000000 \\
c_{11} &= 0.fe0000000000 \\
c_{22} &= 0.fa0000000000
\end{aligned}
$$

Table 2.4 shows the encryption result of the plaintext “The Digital Encryption.” The plaintext is encoded into ASCII code format, and the data sequence will be encrypted by a masking sequence which is generated by F with the above parameters. The result also shows the receiver can recover the plaintext with the same parameters.

2.4 Summary

We have proposed a Robust Hyper-Chaotic Encryption-Decryption System composed of two RHCSs that is suitable for digital secure-communication. An RHCS consists of n-

Table 2.4: The encryption example.

Plaintext: The Digital Encryption. Plaintext in ASCII Code: 546865004469676974616c00456e6372797074696f6e2e Ciphertext: 5477bc5de59b7f735bac76c8a022ebaa4a763c2ed41b9d Decrypted plaintext: The Digital Encryption.

coupled robust logistic maps and has a large parameter space which grows along with system precision. Because multiple coupled robust chaotic maps rather than a single one are used, map re-construction of the RHCS system is not feasible by current computation technology. The result shows that the generated masking sequence has good randomness for stream cipher. Two hardware architectures (multiple-cycle and pipelined) have been proposed for area and performance optimization, respectively. The demonstration shows that RHCS can be easily realized in hardware.

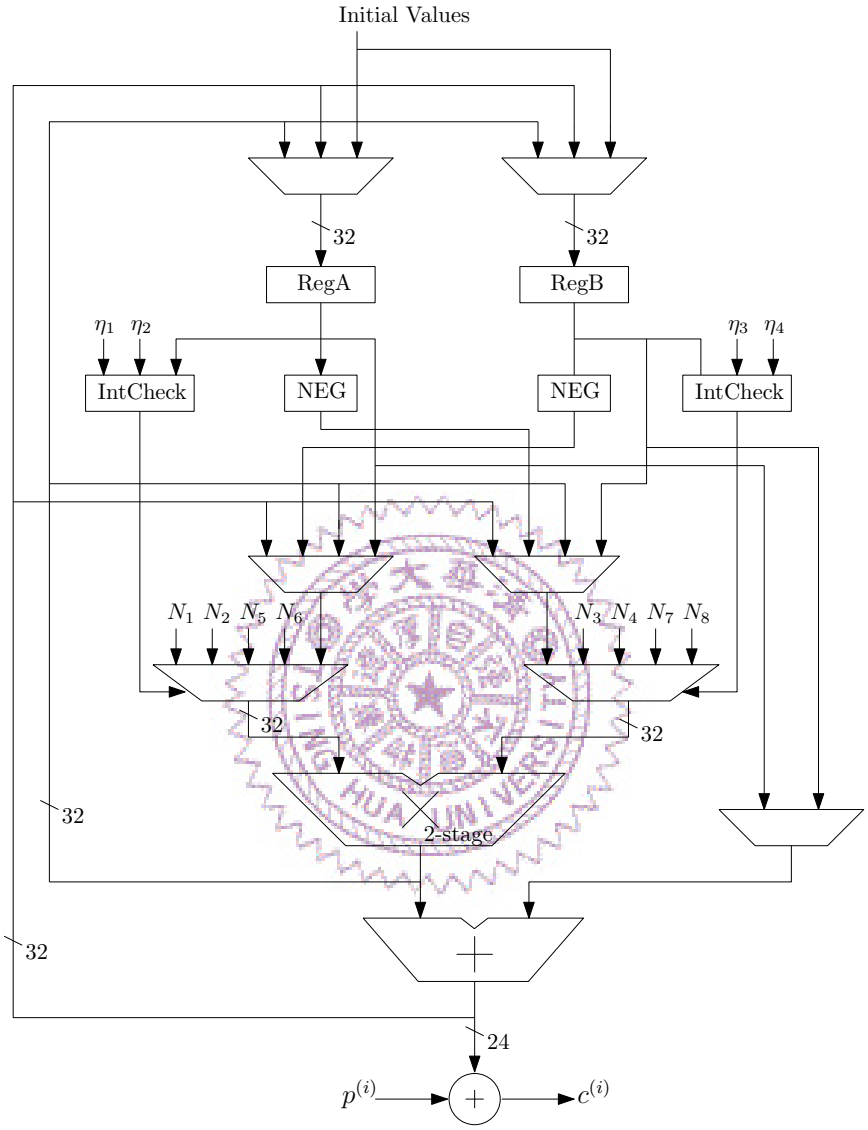


Figure 2.10: The architecture of multiple-cycle implementation.

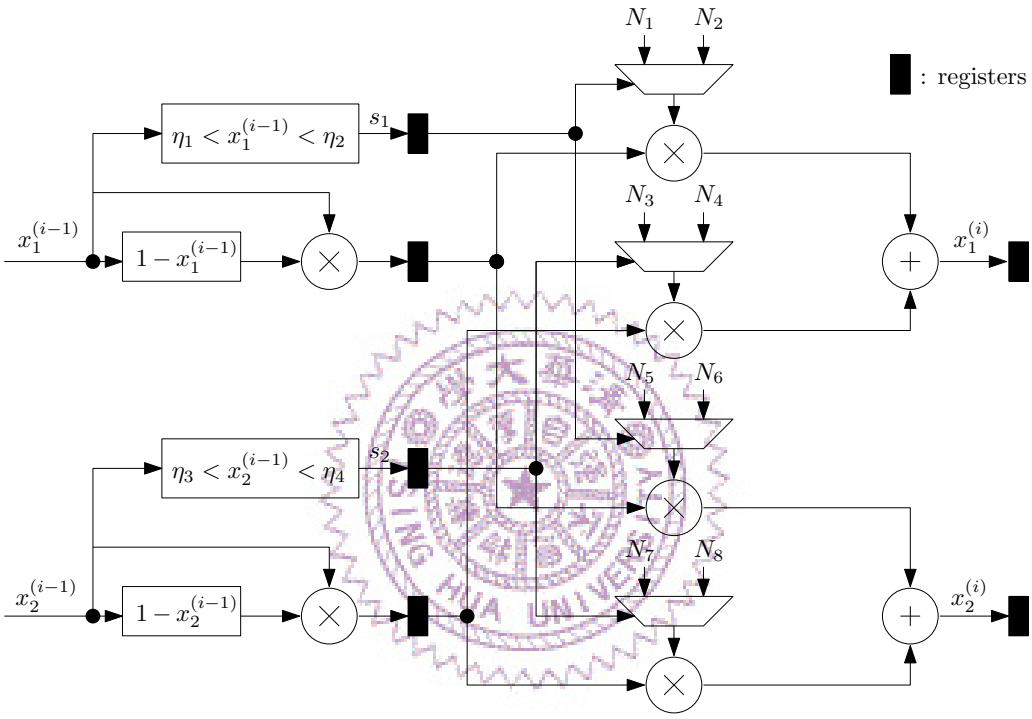


Figure 2.11: The pipelined data-flow of the mask generator.

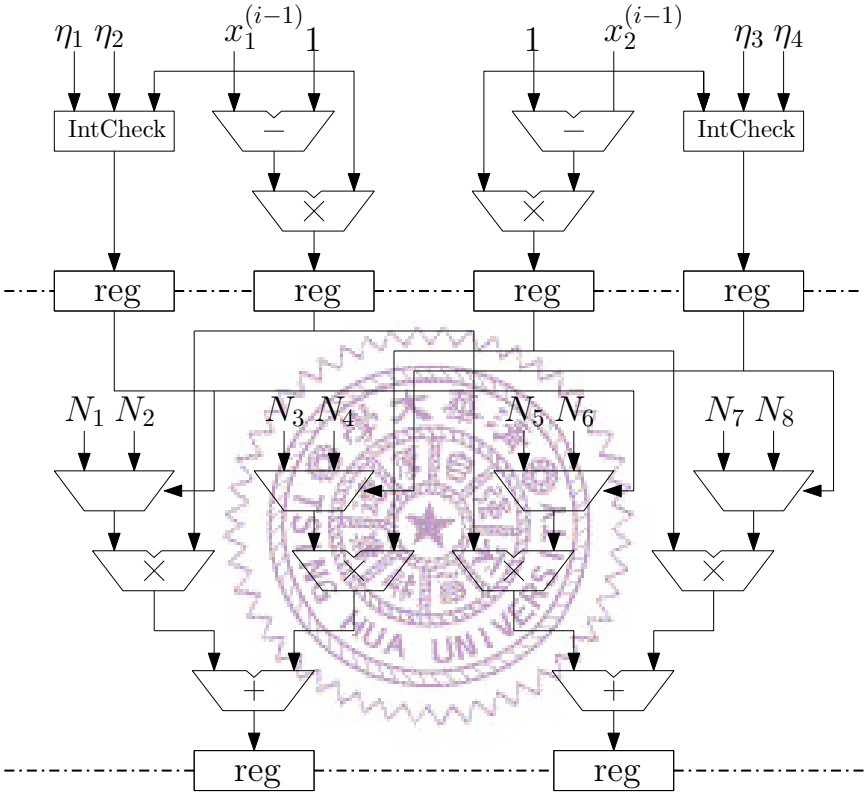


Figure 2.12: The architecture of pipelined implementation.

Chapter 3

A Fast Non-Linear Digital Chaotic Generator in Secure Communications

In Chapter 2, we proposed RLM to extend the parameter space for $\gamma > 4$ and a coupled hyper-chaotic system to generate more complex output sequence. Although the output sequence of the system has good quality of randomness, it needs up to three multiply operations to compute the next states [15]. The cost is larger than a classical logistic map where only two multiplications are used. Moveover, multiply operations are required to form a coupled system. These extra multiply operations limit the throughput of the system.

In this chapter, we will propose a Variational Logistic Map (VLM) with un-restricted parameter space, and can be implemented at lower cost as compared with classical logistic map. First, we show that the raw model of our VLM is a chaotic map by computing the discrete Lyapunov Exponents [17] for different parameters. Then, to verify the chaotic properties of our digitalized VLM, a set of numerical experiments including return map, output cycle length and output spectrum analysis are conducted. Moreover, SP800-22 [30] and TestU01 [31] are applied to verify the statistical properties of the proposed system.

Then, we design a Multi-VLM (MVLM) system constructed by VLMs to have output sequence with higher degree of complexity and larger key space than a single VLM. An MVLM constructed by four 32-bit VLMs can generate sequence with cycle length more

than 2^{128} with a 128-bit external key. We demonstrate that MVLM can generate output sequence with well quality of randomness in higher throughput and lower hardware cost as compared to robust hyper-chaotic systems (RHCS) [15].

Finally, cryptanalysis is conducted, we show that MVLM has large parameter space, long output cycle length, and is hard to reconstruct. From statistical point of view, outputs of MVLM with different keys have small correlations to each other. Moveover, MVLM passes all tests in SP800-22 and TestU01 which indicates MVLM has good quality of randomness.

The rest of this chapter is organized as follows. In Section 3.1, the Variational Logistic Map (VLM) is presented. In Section 3.2, we propose a scrambling method to scramble the output and parameter of VLM. In Section 3.3, architecture of MVLM will be shown. In Section 3.4, cryptanalysis will demonstrate that our system is suitable in secure communications. In Section 3.5, we present hardware implementation of MVLM system. Finally, summary is given in Section 3.6.

3.1 Variational Logistic Map (VLM)

Again, a classical logistic map is defined by

$$L(\gamma, x) = \gamma x(1-x), \quad x \in (0,1). \tag{3.1}$$

Most of chaotic behavior indexes such as the invariant set, Lyapunov exponent, topological, metric, and Renyi specific entropies show that the logistic map has complex behavior. However, these indexes are computed under real number definition and without direct relationship to requirements of secure communications. Two facts show that parameters are not equally strong. The first one is *windows* in parameter space. In [32], authors have proved that the parameter space of logistic map has *windows* which is open and dense. Namely, a large number of chaotic orbits are unstable, i.e., settle down to a stable orbit which has

short cycle length and can not be improved even the system precision is increased. Parameters in *window* are obviously not secure. The second one is limited precision of digitalized chaotic map. If the difference of two numbers is smaller than the resolution, two numbers will become identical during computation. Some parameters may generate short length orbit because two close points on the orbit become identical due to truncation. Based on the above two observations, a classical logistic map can not be directly utilized in digital security system.

In order to remove the *window* generated by Equation (3.1) and to preserve the advantage of fully distribution in (0,1) for $\gamma = 4$, in Chapter 2, we proposed a robust logistic map (RLM) to extend the parameter space to $\gamma > 4$. RLM is fully distributed in (0,1) and has no *window* when $\gamma > 4$. RLM is given by

$$x_{i+1} = \begin{cases} \gamma x_i(1 - x_i) \pmod 1, & x_i \in I_{\text{ext}}, \\ \frac{\gamma x_i(1-x_i) \pmod 1}{\frac{\gamma}{4} \pmod 1}, & x_i \in I_{\text{int}}, \end{cases} \tag{3.2}$$

where $I_{\text{ext}} \in (0,1) \setminus I_{\text{int}}$, $I_{\text{int}} = [\eta_1, \eta_2]$, $\eta_1 = \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{[\gamma]_{\text{g}}}{\gamma}}$ and $\eta_2 = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{[\gamma]_{\text{g}}}{\gamma}}$ in which $[w]_{\text{g}}$ is the greatest integer less than or equal w.

Although RLM extends the parameter space and presents chaos when $\gamma > 4$ [33], to compute the next state of RLM, it needs up to three multiplications while only two multiplications are used in a classical logistic map. More precisely, the first multiplication is used to compute (γx_i) and the second multiplication is used to multiply $(1 - x_i)$. The third multiplication is needed to multiply $(\frac{\gamma}{4} \pmod 1)^{-1}$ if x_i is in I_{int}.

Since we focus on digital secure communications, we will proposed a Variational Logistic Map (VLM) which is based on RLM in digitalized implementation. When compared to RLM, VLM also extend parameter space to $\gamma > 4$ and needs only two multiplication operations to compute the next state.

To construct VLM, first, we propose a raw model,

$$L_p(\alpha, \gamma, x) = \{\alpha \left[(\alpha\gamma x) \pmod 1\right](1 - x)\} \pmod 1, \tag{3.3}$$

which is equivalent to

$$L_p(\alpha,\gamma,x) = [\alpha\,(\alpha\gamma x - [\alpha\gamma x]_{\mathrm{g}})\,(1-x)] \pmod 1, \tag{3.4}$$

In fact, Equation (3.4) can be rewritten by

$$L_p(\alpha,\gamma,x) = \begin{cases} f(\alpha,\gamma,x) - g(\alpha,\gamma,x) & \text{if } f \geq g, \\ f(\alpha,\gamma,x) - g(\alpha,\gamma,x) + 1 & \text{if } f < g, \end{cases} \tag{3.5}$$

where

$$f(\alpha,\gamma,x) = [\alpha^2\gamma x(1-x)] \pmod 1, \tag{3.6}$$

and

$$g(\alpha,\gamma,x) = \{\alpha[\alpha\gamma x]_{\mathrm{g}}(1-x)\} \pmod 1. \tag{3.7}$$

where $\alpha,\gamma > 0$, and $x \in (0,1)$.

Let $\hat{\gamma} = \alpha^2\gamma = 4k$, $k \in \mathbb{N}$, function f can be rewritten as $f = [\hat{\gamma}x(1-x)] \pmod 1$. By the definition of Equation (3.2), the interval, I_{int}, of function f is equal to

$$\begin{aligned} I_{\mathrm{int}} &= [\eta_1\ ,\ \eta_2] \\ &= \left[\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{[\frac{4k}{4}]_{\mathrm{g}}}{4k}}\ ,\ \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{[\frac{4k}{4}]_{\mathrm{g}}}{4k}}\ \right] \\ &= [\tfrac{1}{2}\ ,\ \tfrac{1}{2}]. \end{aligned} \tag{3.8}$$

which means there is no x in I_{int}. When $\hat{\gamma}$ is a multiple of four, function f is a subset of RLM. Hence, L_p is constructed by RLM, f, and function g which is a scrambling function with respect to function f.

Then, a q-bit VLM will be defined by a digitalized L_p in q-bit precision. First of all, we define the value of coefficients α because the value of α will affect the truncation result during successive multiplications and modular operations.

To define the value of α, We start from the point of implementing the multiplication in finite-precision arithmetic. Because the multiplication is defined in finite precision, to store product in the same number of bits, truncation is needed after multiplying two numbers.

For example, in Figure 3.1(a), let a, b and c be 4-bit binary numbers and $c = a \times b$. The result of $a \times b$ will be truncated from 8-bit to 4-bit and assigned to c. Suppose the least significant bits be truncated. We find that c_H is directly determined by only 6 partial products, which are a_3b_1, a_3b_2, a_2b_2, a_3b_3, a_2b_3 and a_1b_3, and indirectly determined by the carry-outs generated by other partial products. That means, the change of inputs may not cause the change of outputs. Hence, with different x, a sub-operation $\gamma \times x$ in a logistic map may lead to the same next value during sequence generation because the difference in the least significant bits is eliminated by truncation. That different xs can not generate different orbits result in short length cycles.

On the contrary, in Figure 3.1(b), the value of c_M depends on the largest number of partial products. That means, when any bit of input is changed, c_M has higher probability to change its value than c_H.

The purpose of α in equation $c = (\alpha ab)(\mathrm{mod}\ 1)$ is to make the output of the equation depend as many input bits as possible. We construct an experiment to see the output distribution versus different values of α in equation $c = (\alpha ab)(\mathrm{mod}\ 1)$ where $\alpha = 2^k$ and $k = 0, 4, 8$. Let a, b and c be 8-bit binary numbers in (0,1) and $0.p_1p_2p_3 \cdots p_{16}$ denotes 16-bit product of $a \times b$. For example, when $\alpha = 2^4$, the result of $(\alpha ab)(\mathrm{mod}\ 1)$ is equal to $0.p_5p_6 \cdots p_{16}$. Because c is 8-bit, p_{13}, p_{14}, p_{15}, and p_{16} are dropped. Three results, c_L, c_M, and c_H are computed with α=2^8, α=2^4, and α=2^0 respectively. More specifically, c_L is equal to $0.p_9p_{10} \cdots p_{16}$, c_M is equal to $0.p_5p_6 \cdots p_{12}$, and c_H is equal to $0.p_1p_2 \cdots p_8$.

Let a and b be selected in uniformly distributed. The Probability Mass Function (PMF) of c_L, c_M and c_H are shown in Figure 3.2(a)–(c), respectively, where the x-axis denotes the value of c. The results show that, for truncated result c_H and c_L, we can observe c results in higher probability in some values. On the contrary, c_M has even probability distribution. The values of standard deviation for c_L, c_M, and c_H are 0.26, 0.05 and 0.39, respectively. The uniform distribution is an important property when a function is applied to a crypto

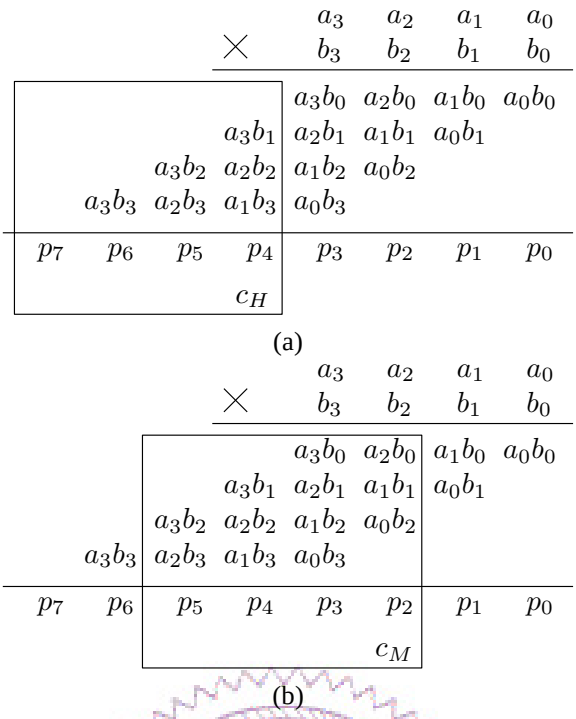


Figure 3.1: (a) The least significant 4 bits truncated. (b) Preserving middle significant bits in truncation operation.

system.

In Equation (3.3), to make the output of map uniformly distributed, α should be equal to half length of x. Since x is in q-bit precision, α is equal to $\alpha = 2^{\lceil \frac{q}{2} \rceil}$. Here, our digitalized VLM in q-bit precision is shown in Equation (3.9). Before the presentation of VLM, we define a binary floor function, $\lfloor x \rfloor_a$, which preserves the most significant a bits of x and sets other bits to be zero. The VLM is defined as

$$VLM(\gamma, x) = \lfloor [\alpha \lfloor (\alpha\gamma x) \,(\mathrm{mod}\, 1) \rfloor_q (1 - x)\,] \,(\mathrm{mod}\, 1) \rfloor_q, \tag{3.9}$$

where x and γ are q-bit binary numbers in (0,1). Let $x[i]$ be the ith bit of variable x. x and γ can be represented by $x = \sum 2^{-i} x[i]$, and $\gamma = \sum 2^{-i} \gamma[i]$ for $i = 1$ to q, respectively. The detail of computation is as follows.

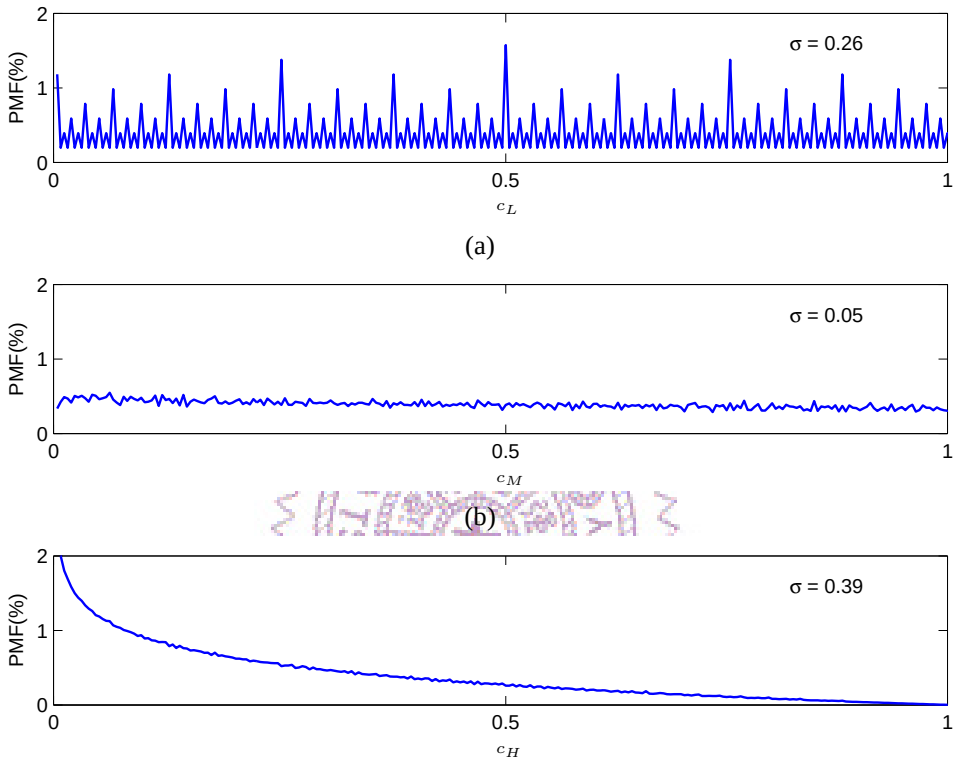


Figure 3.2: Distributions for c_L, c_M, and c_H. (a) $c_L = 0.p_9p_{10}\cdots p_{16}$, (b) $c_M = 0.p_5p_6\cdots p_{12}$, and (c) $c_H = 0.p_1p_2\cdots p_8$.

We take a 32-bit VLM as an example to describe the calculation of $VLM(\gamma, x)$. At the beginning, $(2^{16}\gamma x)(\mathrm{mod}\,1)$ will be computed first. The result of γx will be computed and modulated to keep the value between (0,1), and then truncated to 32-bit by truncation operation. More specifically, the integer part of $2^{16}\gamma x$ is the most significant 16 bits of $2^{16}\gamma x$ since x and γ are between (0,1). The $(\mathrm{mod}\,1)$ operation will drop the most significant 16 bits of $2^{16}\gamma x$, and $\lfloor (2^{16}\gamma x)\,(\mathrm{mod}\,1)\,\rfloor_{32}$ operation will drop the least 16 significant bits of $2^{16}\gamma x$. Hence, the most and least significant 16 bits of γx are both truncated, and then the result is passed to the next step.

Let $\lfloor (2^{16}\gamma x)\,(\mathrm{mod}\,1)\,\rfloor_{32}$ be p. p is a 32-bit binary number and will be multiplied by $1-x$. Since $p(1-x)$ is a 64-bit binary number, $p(1-x)$ will be truncated to 32-bit by the same way we truncate γx. Finally, $VLM(\gamma, x)$ is a 32-bit number and between (0,1).

One last constraint is for the value of γ. γ is in q-bit and between (0,1). When $2^{-q} \leq \gamma \leq 2^{-(q-1)}$, the result of $\alpha^2\gamma$ in Equation (3.4) is smaller then 4. However, as studied in [33], $\alpha^2\gamma$ is required to be a multiple of 4 for Equation (3.4) to generate a chaotic map. In order to keep $\alpha^2\gamma$ a multiple of four, we let the least significant two bits, $\gamma[q-1]$ and $\gamma[q]$, be equal to 0. That means the smallest γ is $2^{-(q-2)}$ and the smallest value of $\alpha^2\gamma$ is $2^{\frac{q}{2}} \times 2^{\frac{q}{2}} \times 2^{-(q-2)}$ which is equal to 2^2. Hence, we guarantee $\alpha^2\gamma$ to be greater than 4 and a multiple of 4.

In Figure 3.3, we plot the return map of VLM with $\gamma = 2^{-16}$. Even with a small value of γ, VLM becomes a nonlinear map with plenty of discontinuity.

The truncation operation during $VLM(\gamma, x)$ computation makes $VLM(\gamma, x)$ not continuously differentiable, but the truncation error can be treated as a scrambling function of Equation (3.3). Because of the discontinuity of the function, it is difficult to theoretically prove the chaotic property of the proposed digitalized VLM. Hence, numerical experiments to verify the chaotic properties of VLM including the *discrete Lyapunov Exponent* [17], the *bifurcation graph*, *output spectrum analysis*, and *output cycle length* are conducted.

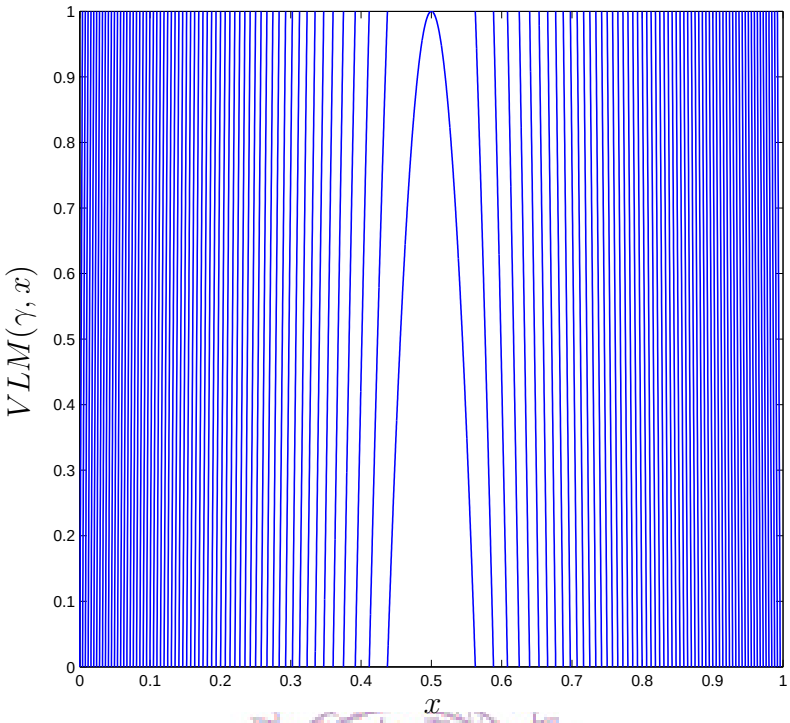


Figure 3.3: $VLM(\gamma, x)$ with $\gamma = 2^{-16}$.

1). The discrete Lyapunov Exponent and bifurcation graph.

The discrete Lyapunov Exponent (dLE) [17] was used to verify discrete chaotic properties of the proposed system. Let $\mathcal{M}$ be a finite subsegment of the trajectory in length m for a digitalized map F and $d(M_\mu, M_\nu)$ be the distance between M_μ and M_ν, where M_μ and M_ν are in $\mathcal{M}$. The basic expression of dLE is defined as

$$\lambda_F = \frac{1}{m} \sum_{\mu=0}^{m-1} \ln \frac{d(F(M_{\mu+1}), F(M_\mu))}{d(M_{\mu+1}, M_\mu)}. \tag{3.10}$$

Map F is considered as a discrete chaotic map if it's dLE (λ_F) tends to a positive number when $m \to \infty$. For example, let $\gamma = 4$, dLE of a 32-bit classical logistic map is 0.69 when $m = 10000$.

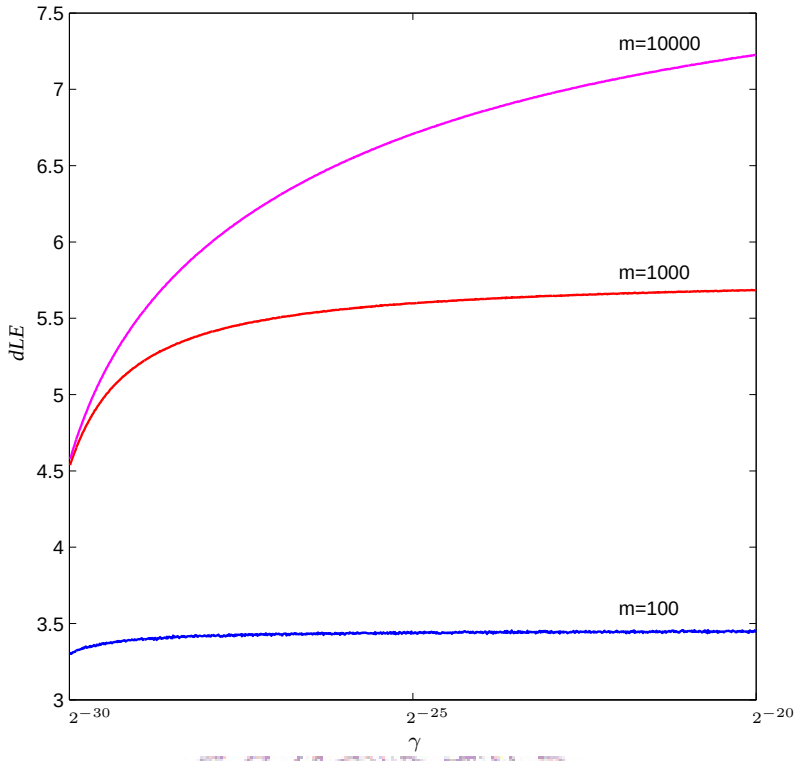


Figure 3.4: dLEs for 32-bit VLM when $2^{-30} \leq \gamma \leq 2^{-20}$, where $m = 100$, 1000, and 10000.

We compute dLEs for 32-bit VLM when $2^{-30} \leq \gamma \leq 2^{-20}$. For each γ, dLEs are calculated with three different ms, 100, 1000, and 10000 to understand the trends of dLEs when m is increased. For each m, dLEs for 1000 different trajectories are computed and the average value is shown in Figure 4.1. The results show that dLEs are all positive when $\gamma > 0$. Moreover, dLEs are increased when m is increased. Hence, we know VLM is discrete chaos as defined in [17].

Moreover, to understand if there are *windows* in VLM. We compute the bifurcation diagram of VLM for $\gamma = 2^{-30}$ to 2^{-1}. The result is shown in Figure 3.6. It reveals that the output data of VLM has no *window* for $\gamma = 2^{-30}$ to 2^{-1}. As a result, from theoretical and numerical point of view, we know that VLM is a pseudo-chaotic map.

2). The output spectrum analysis.

To analyze the auto-correlation and spectrum of output sequences, we randomly select the values of γ and x_0. Figure 3.7 shows the results when $\gamma = 0.609375$ and $x_0 = 0.21875$. First, in Figure 3.7(a) we plot 10,000 output data. The result shows that the output sequence is visualized randomly. In Figure 3.7(b), the spectrum analysis by FFT signifies that the output sequence is a chaotic sequence [25]. In Figure 3.7(c), the auto-correlation of the output sequence indicates that the output data are quite independent.

3). The output cycle length.

This experiment is conducted to compare the cycle number of an output sequence generated by

$$x_{i+1} := VLM(\gamma, x_i), \quad i = 0, 1, \ldots. \tag{3.11}$$

with that by a classical logistic map in 32-bit precision.

With random initial values, the cycle lengths of 10,000 output sequences are generated by 10,000 γs chosen evenly from interval $2^{-30} \leq \gamma \leq 2^{-1}$ for our VLM shown in Equation (3.11) and from interval $3.57 < \gamma \leq 4$ for a classical logistic map.

In Figure 3.5, we observe that in 10,000 output sequences generated by logistic map, over 10% of the output sequences form periodic orbits with a period less than 100, and only 9% of the output sequences form chaotic orbits with cycle lengths more than $50,000$. On the contrary, the result by VLM shows that there is only 0.2% of output sequences with cycle lengths smaller than 100 and 31.8% larger than 50,000.

As to hardware cost, the modular arithmetic and truncation operation in Equation (3.9) can be easily implemented by bit-selection, i.e., by signals routing. Implementation of VLM will not increase circuit area as compared with classical logistic map. From application point of view, our VLM is more efficient and reliable than the classical logistic map under the same implementation hardware cost. Thus, based on the above properties of

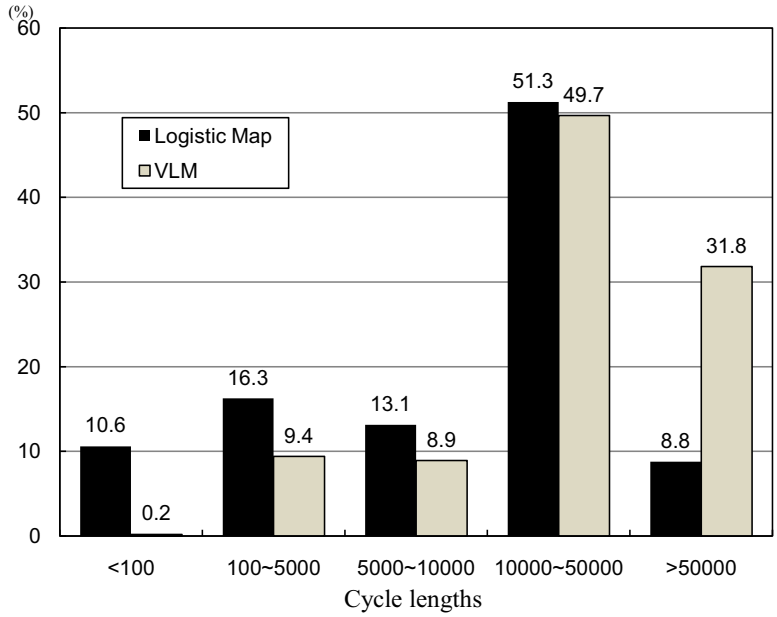


Figure 3.5: The histogram of cycle length for VLM and classical logistic map.

VLM, in the next two sections, we will develop a scalable Multi-VLM (MVLM) system to increase key space and complexity by scrambling and coupling methods.

3.2 Scrambling Method for VLM

Although VLM has no *window* in parameter space, similar to other digitalized chaotic map, the output cycle length is far below the number of states. As shown in Figure 3.5, 68.2% of 10,000 γs generate an output cycle with length small than 50,000. The cycle length is relatively smaller than the number of states of a 32-bit VLM.

Scrambling methods are useful and widely used in digital chaotic system. In [5], an external uniformly distributed pseudo random number sequence is used as a noise to scramble not only the output but also the parameter to increase the cycle length. In [34], a linear-feedback-shift-register (LFSR) which has uniformly distributed outputs is used to scramble the output of a digital chaotic system. The scrambling strategy for VLM is shown in Fig-

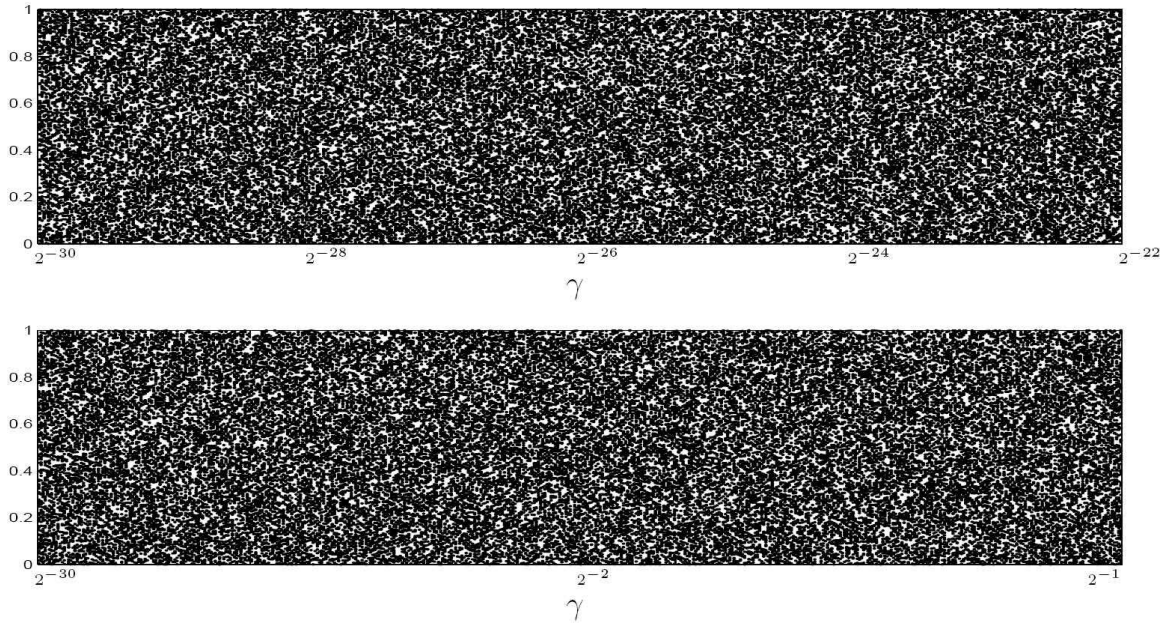


Figure 3.6: The bifurcation diagram of VLM for $2^{-30} \leq \gamma \leq 2^{-1}$.

ure 3.8, where L_x is a q-bit LFSR and δ is a one-step delay block.

The LFSR, L_x, generates a pseudo random number sequence n_i which is defined as

$$n_i = L_x(n_{i-1}), \quad i = 0, 1, \ldots. \tag{3.12}$$

Then, n_i is xor-ed with x_i to scramble output. L_x should be primitive to have a maximum cycle length output and uniform scrambled outputs. From Equation (3.12) the sequence generated by VLM after scrambling is defined as follows.

$$\bar{x}_{i+1} = VLM(\gamma, \bar{x}_i) \oplus n_i, \quad i = 0, 1, \ldots. \tag{3.13}$$

By the proposed scrambling method, a deterministic bound of cycle length is calculated as follows. In [34], the low bound of cycle length of a scrambling system is given by

$$\Delta \cdot (2^l - 1), \tag{3.14}$$

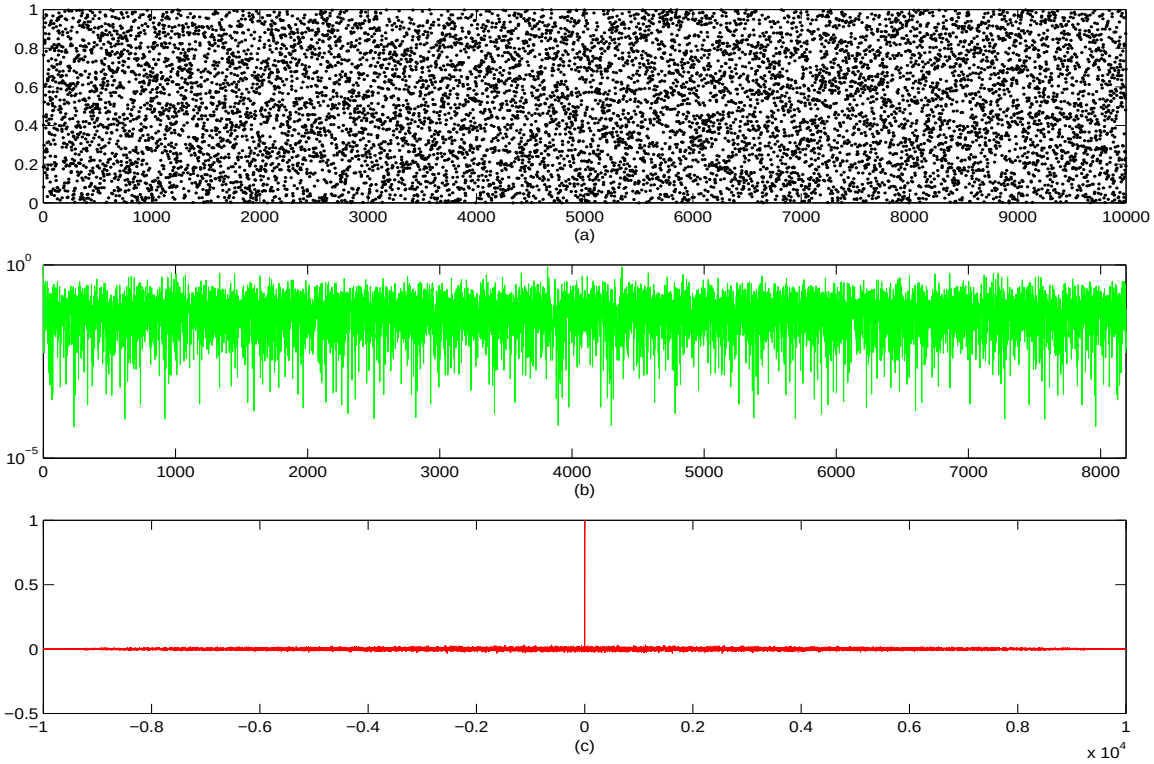


Figure 3.7: (a) output value plotting, (b) spectrum analysis, and (c) auto-correlation of a trajectory generated by VLM with $\gamma = 0.609375$ and $x_0 = 0.21875$.

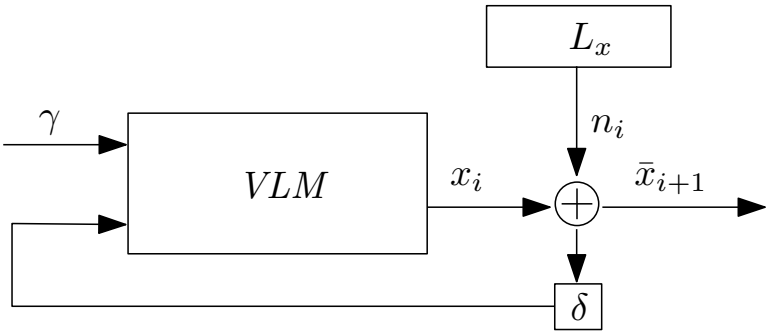


Figure 3.8: The scrambling strategy for VLM.

where Δ is the scrambling period and l is the register length of the LFSR.

Li's [12] scrambles outputs of a logistic map by a fixed pattern with $\Delta = 700$. By this method, the cycle extension is small because the Δ and m are small. Moveover, the method needs a counter to count the scrambling period. In our proposed scrambling method, Δ and l are equal to 1 and q, respectively. For a q-bit VLM, the low bound of cycle length is $2^q - 1$. The hardware cost is a q-bit LFSR which is smaller than a counter used to count the period.

In order to generate uniformly distributed outputs with maximum cycle length, a primitive LFSR is used in our scrambling system. After scrambling, the output as well as the next input of VLM tends to be more uniformly distributed. An experiment is conducted to show that the proposed scrambling can improve the output distribution by measuring the 1's probability of the output sequence generated by a scrambled 32-bit VLM. In this experiment, γ is equal to $(0.10001000)_H$ and L_x is defined by $L_x(x) = x^{32}+x^{31}+x^{30}+x^{29}+x^{28}+x^{22}+1$. For each scrambling period, 10^6 outputs are generated. In Figure 3.9, when the scrambling period is decreased, the probabilities of 1 in outputs are close to 0.5.

We will verify the statistical property of VLM with scrambling method by statistical test suite in Section 3.4.4. It will show that VLM with scrambling can significantly increase output complexity when compared with classical logistic map. In next section, we will construct MVLM by coupling VLMs to improve output complexity and enlarge key space.

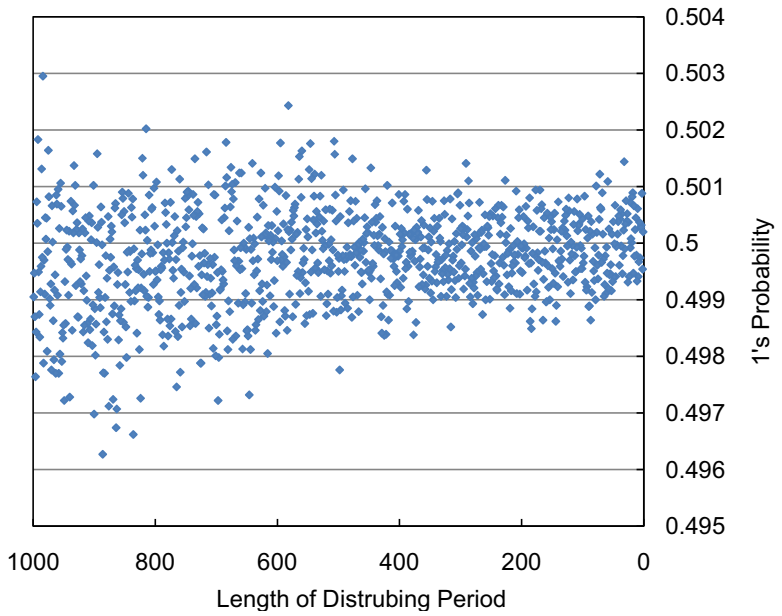


Figure 3.9: The 1's probability when scrambling the sequence with different period.

3.3 Coupling Multi-VLM

Based on VLM and scrambling functions, a scalable Multi-VLM (MVLM), is proposed to increase the number of keys and complexity of output sequence.

3.3.1 Structure of Multi-VLM

An MVLM is constructed by m VLMs denoted by VLM_i for $i = 1$ to m. For each VLM_i, the output is scrambled by a noise sequence, $n^{(j)}$, generated by a global $(q \times m)$-bit LFSR, L_x.

Let the output of VLM_i at jth iteration be $x_i^{(j)}$ and $x_i^{(j)}$ be scrambled by a segment of $(q \times m)$-bit noise sequence, $n^{(j)}[1 : qm]$. The scrambling function is defined by

$$\bar{x}_i^{(j+1)} = x_i^{(j)} \oplus n^{(j)}[q(i-1)+1 : qi], \qquad j = 0, 1, \ldots, \tag{3.15}$$

where $n^{(j)}$ is generated by L_x, and $n^{(j+1)} = L_x(n^{(j)})$.

The scrambled result of VLM_i, $\bar{x}_i^{(j+1)}$, is fed to $VLM_{(i+1)}$, except $\bar{x}_m^{(j+1)}$ which is generated by the last VLM and fed to VLM_0. The whole system forms a cascaded chain and can be defined by

$$x_i^{(j)} = \begin{cases} VLM(\gamma_1^{(j)}, \bar{x}_m^{(j)}), & i = 1; \\ VLM(\gamma_i^{(j)}, \bar{x}_{i-1}^{(j)}), & 1 < i \leq m. \end{cases} \tag{3.16}$$

Finally, the output sequence of MVLM is generated by an output function T. Output function will be discussed in Section 3.3.3. The output sequence Seq is given by

$$Seq_j = T(\bar{x}_m^{(j+1)}), \qquad j = 0, 1, \ldots. \tag{3.17}$$

For example, a MVLM constructed by 4 VLMs is shown in Figure 3.10. In this system, m is equal to 4 and all VLMs are in 32-bit precision. The 32-bit output $x_1^{(j)}$ of *VLM*$_1$ will be xor-ed by $n^{(j)}[1:32]$ to generate $\bar{x}_1^{(j)}$, which is fed to *VLM*$_2$. Similar connections are constructed for *VLM*$_2$ and *VLM*$_3$. Finally, the last $x_4^{(j)}$ is xor-ed by $n^{(j)}[97:128]$ to produce $\bar{x}_4^{(j)}$ which is fed to *VLM*$_1$.

The last problem to be solved is the initial states, $x_i^{(0)}$ for $i = 1$ to 4, and n_0 for L_x. With different initial states, outputs of MVLM will be different. In next section, key initialization process is used to generate initial states. After key initialization, MVLM is ready for output Seq.

3.3.2 Key Initialization

In Section 3.1, we have shown the chaotic properties of the VLM, where with small differences in x and in γ, the generated output sequences will be very different. In MVLM, this chaotic properties are not only used in generating the output sequence but also used in key initialization. Key initialization generates values of γ_i, $x_i^{(0)}$, and $n^{(0)}$ for $i = 1$ to m, called *internal keys*. The purpose of key initialization is to generate *internal keys* that have

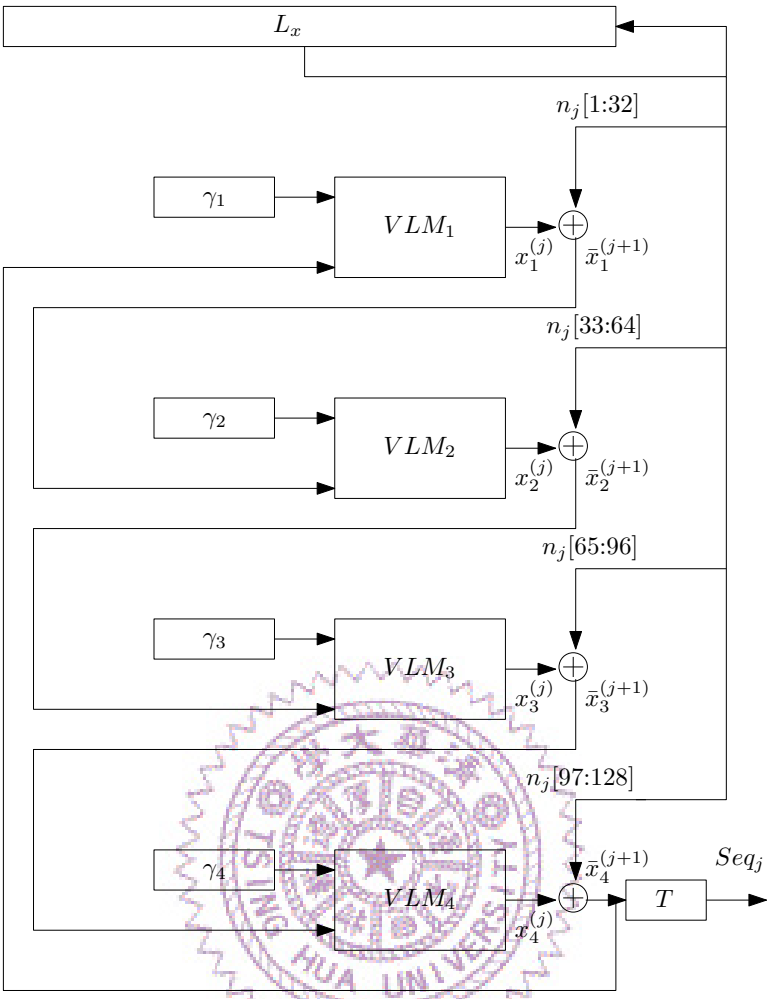


Figure 3.10: The top view of a MVLM coupled by 4 VLMs.

minimal relations to the user specified *KEY*. With different *internal keys*, each MVLM can generate different output sequence.

Without loss of generality, we take the MVLM shown in Figure 3.10 as an example of key initialization process. The process can be easily extended to a MVLM constructed by m q-bit VLMs.

The input of key initialization procedure is an 128-bit *KEY* and the outputs are *internal keys*. There are two steps in key initialization. The first step uses *KEY* and default value

to generate *intermediate internal keys*. Then, the second step will use the *intermediate internal keys* to create *internal keys*.

The purpose of the first step is to allow bit changes in *KEY* to have influence on *internal keys*. In the first step, the configuration is shown in Figure 3.11. The initial value of each *VLM*$_i$ for $i = 1$ to 4 will be given by following equations. First, each γ_i is assigned by KEY[1:64] with

$$\gamma_i[k] = \begin{cases} 1, & k = i; \\ KEY[16i + k - 32], & 17 \leq k \leq 32 \\ 0, & \text{others}, \end{cases} \tag{3.18}$$

where $1 \leq k \leq 32$. Then, KEY[65:128] is loaded to $x_i^{(0)}$ by

$$x_i^{(0)}[k] = \begin{cases} KEY[16i + k + 48], & 1 \leq k \leq 16; \\ 0, & 17 \leq k \leq 32. \end{cases} \tag{3.19}$$

Finally, KEY also becomes the initial value of noise $n^{(0)}$ by equation defined as follows.

$$n^{(0)}[k] = KEY \tag{3.20}$$

Moveover, in order to reduce the correlation between KEY and γ_i, the least significant bit of $x_i^{(j)}$ is fed back to generate γ_i in the first step of key initialization. We will shift γ_i right one bit per cycle and replace the most significant bit of γ_i by the last significant bit of $x_i^{(j)}$ where the updating function of γ_i is given by

$$\gamma_i = (\gamma_i \gg 1) \oplus (\text{0x00}||x_i^{(j)}[32] \,\&\, \text{0x01})\,), \tag{3.21}$$

where $i = 1$ to 4.

The architecture of cascaded VLMs we use in the first stage is shown in Figure 3.12. Let one output be generated in one cycle with initial values described in Figure 3.11. The

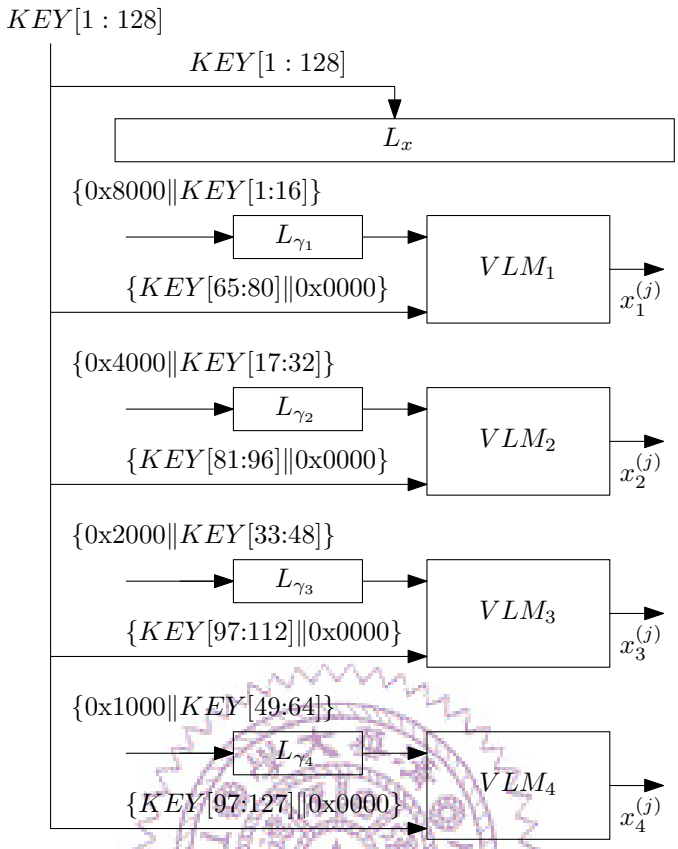


Figure 3.11: The initial values to VLM_i from KEY.

system runs 128 cycles to generate $x_i^{(128)}$ and γ_i, where $x_i^{(128)}$ and γ_i are called the *intermediate internal keys* and will be used to generate *internal keys*. The reason why 128 cycles are required is that LFSR will shift one bit to left each cycle and the length of L_x is 128 bits. It needs 128 cycles to shift the first bit to the last bit.

In the first step, value of $n^{(0)}$ is directly assigned from KEY and generated by L_x which is linear and predictable. In the second step, we use the *intermediate internal keys* to generate $n^{(0)}$ non-linearly and chaotically.

In the second step, $x_i^{(j)}$ is fed to VLM_{i+1} for $i = 1$ to 3 without scrambling and the last $x_4^{(j)}$ is fed to VLM_0. The configuration is shown in Figure 3.13. Moreover, the first bit of $x_4^{(j)}$, i.e., $x_4^{(j)}[1]$, is fed to a 128-bit register, n_{reg}. With $x_i^{(128)}$ and γ_i generated in the first

step, the system will run the next 128 cycles to generate n_{reg} which can be defined by

$$n_{reg}[k-128] = x_4^{(k)}[1], \quad 129 \leq k \leq 256. \tag{3.22}$$

One bit of n_{reg} is generated one cycle by cascaded VLMs. After that, the value of n_{reg} will be used as initial values of L_x, $n^{(0)}$.

Key initialization procedure totally needs 256 cycles. The first 128 cycles is used to propagate the influence of each bit in KEY to *intermediate internal keys*. In the second 128 cycles, *intermediate internal keys* are used to generate n_{reg} (i.e. $n^{(0)}$) and reduce the correlation between $n^{(0)}$ and KEY. After 256 cycles, the values of $x_i^{(256)}$ become $x_i^{(0)}$. Then, $x_i^{(0)}$, γ_i and $n^{(0)}$ which are *internal keys* are ready for MVLM to generate Seq.

3.3.3 Output Function

The $\bar{x}_m^{(j)}$ generated by MVLM is a good random source for security application. The output function is used to further increase the complexity and prevent the whole trajectory from attacking. With small amount of implementation cost, bit-selection is the most common method to perform output function where several bits of trajectory are selected to be the output. At one extreme, only one bit is selected. In this case, reconstructing the trajectory from the output is impossible but the system is not efficient since only one bit is generated in one cycle. The number of selected bits can be decided by the secure level that application needs. For example, if 8-bit output data for application usage is required, the output function can be defined by selecting the middle 8 bits from a 32-bit $\bar{x}_m^{(j)}[1:32]$, and $Seq[1:8]$ will be equal to $\bar{x}_m^{(j)}[13:20]$. The selection provides a trade-off between output efficiency and information security.

3.4 Cryptanalysis of MVLM

In this section, we consider some security properties and general attacks against a MVLM.

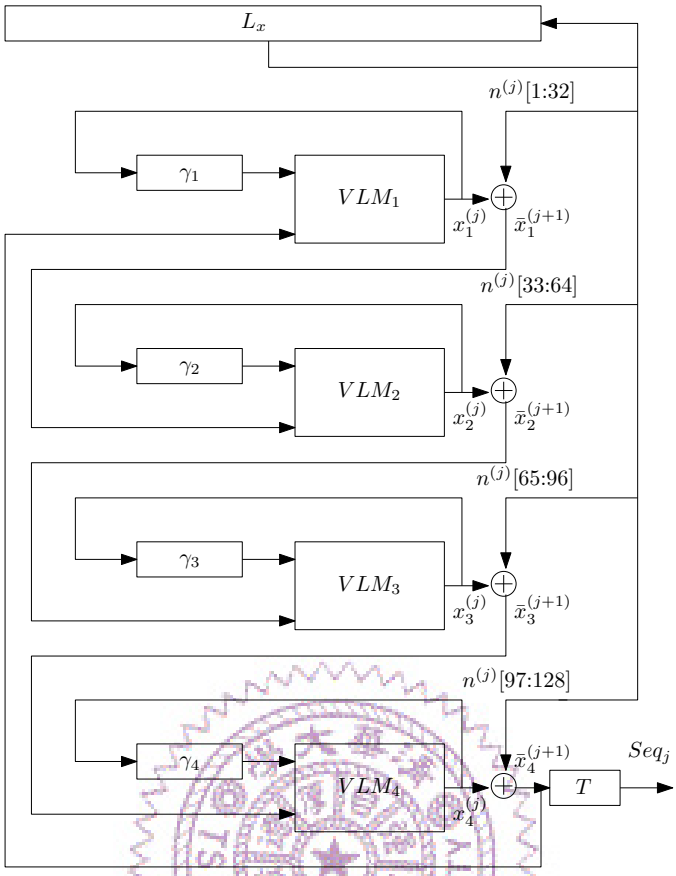


Figure 3.12: In the first step of key initialization, γ_i will be shifted to right one bit per cycle and the most significant bit of γ_i will be replaced by $x_i^{(j)}[0]$.

3.4.1 Key Space

The key length of MVLM is $(q \times m)$ bits where m is the number of coupled VLMs in MVLM and q is the precision of VLM. In Section 3.3.2, 128-bit KEY is used to generate *internal keys* which are values of four 32-bit $x_i^{(0)}$, four 32-bit γ_i and one 128-bit n^0. There are two properties we want to ensure in key initialization stage when we map KEY to *internal keys*. One is being a one-to-one mapping and the other is to reduce the correlations between both.

Since segments of KEY are separated and assigned as initial states of primitive LFSR

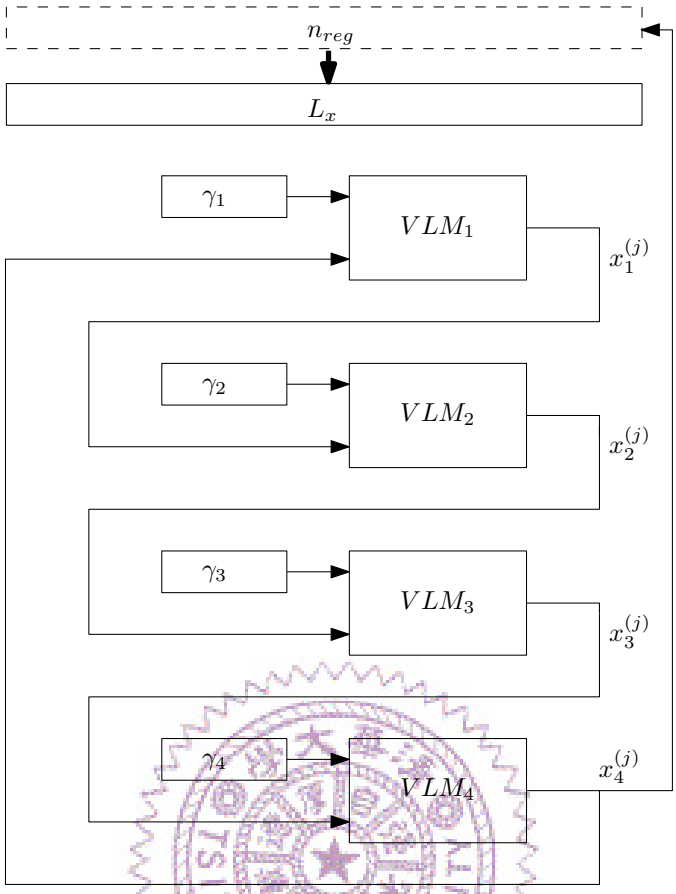


Figure 3.13: Using cascaded VLMs to generate $n^{(0)}$ in the second step of key initialization.

which are L_x, the LFSR will generate different sequence with different KEYs. That means the map is a one-to-one mapping. Moreover, the second property that the correlations between KEY and *internal keys* should be reduced is also achieved because *internal keys* is generated after 256 chaotic-system iterations staring with KEY. The key space of the MVLM is 2^{128}.

3.4.2 Cycle Length

To avoid short length of a single VLM, we use a q-bit noise n_i to scramble each output, x_i, periodically. The cycle length of x_i is not predictable but the cycle length of n_i depends

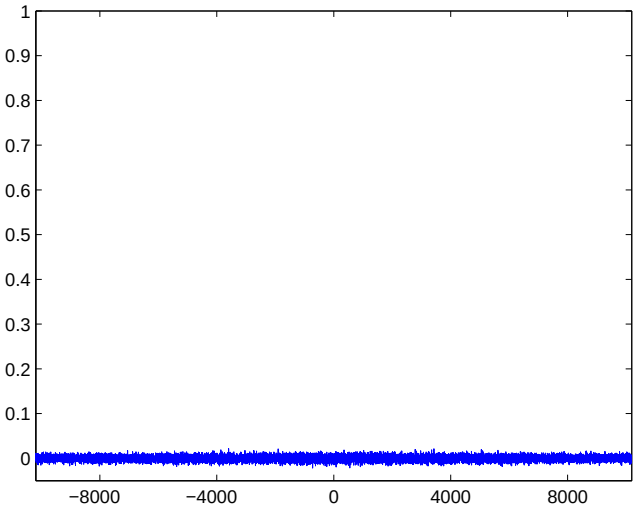


Figure 3.14: The cross-correlations between $KEY = 0$ and $KEY = 1$.

on the length of LFSR, L_x, which is $2^q - 1$. Since the scrambled output is computed by $x_i \oplus n_i$, $2^q - 1$ is the low bound of the cycle length of scrambled output. In a MVLM which is constructed by m VLMs, the scrambling noise is a $(q \times m)$-bit n_i. The minimal cycle length of Seq will be $2^{qm} - 1$. In Section 3.3, the cycle length of the MVLM coupled by four 32-bit VLMs will be at least $2^{128} - 1$.

3.4.3 Correlation

The cross-correlations between the output sequences generated by different KEYs is considered. To check this properties, two Seqs are generated by selecting all 32-bit of MVLM's output described in Section 3.3, i.e., $Seq_j = x_m^{(j)}$ when $KEY = 0$ and $KEY = 1$. Note that, only one bit is different between these two input KEYs. In Figure 3.14, it shows that the cross-correlations between two sequences generated by two KEYs are very weak.

Table 3.1: Parameters for SP800-22.

Block freq.	m=128	Serial	m=16
Longest run	M=10000	Apen	m=10
Nonoverlap.	m=9	Linear Comp.	m=500
Overlap.	m=9	Universal	L=7,Q=1280

3.4.4 Statistical Analysis

The randomness of the our system is tested by two test suites, SP800-22 developed by NIST [30] and TestU01 proposed by L'Ecuyer [31]. The SP800-22 test suite has been the standard reference for randomness testing. Hence, we use SP800-22 as our first randomness test. Then, to further compare the randomness of the proposed system and other digital chaotic generator, TestU01 is used.

We summarize the configuration for SP800-22 tests as follows. We let $\alpha = 0.01$, $T = 120$, and others be the values as shown in Table 3.1. For each test, 120 sequences will be generated by systems with the length of 10^6. For each sequence, each test produces a *P-value*, where *P-value* should be in range, [0.01, 1.00], to pass the test. For each test, the minimum passing rate of a well random source is 0.9627 out of 120 binary sequences. The distribution measurement of collected *P-values* denoted by *U-value* are also reported. If *U-value* is greater than 10^{-4}, the sequence can be considered to be a good random-sequence.

To test by TestU01, three test suites, *SmallCrush*, *Crush*, and *BigCrush* are applied. Recommended by TestU01, *SmallCrush* including 15 sub-tests is taken as a fast check for the basic randomness requirement. Next, $Crush$ needs 2^{35} output sequences to perform further 144 tests. Finally, *BigCrush* is the most stringent statistical testing suite in TestU01. It needs 2^{38} output sequences to perform 160 statistical tests. For each test, a p-value is calculated. If p-value is out of the range, $[0.001, 0.9990]$, the system fails the test. In the following, we will illustrate the quality of randomness of the proposed system in terms of statistical testing results.

1). Randomness improvement by the scrambling function.

The first experiment is conducted to understand the efficiency of the scrambling function described in Section 3.2. A 32-bit VLM with/without scrambling function is tested by SP800-22. The initial values of γ is $(0.05079f23)_H$ and polynomials of L_x are chosen as $L_x(x) = x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{22} + 1$. All 32 bits of system output are selected and fed to testing package directly. As shown in Table 3.2, without the scrambling function, the *VLM* fails some tests because the short output length. On the contrary, with scrambling function, the *scrambled VLM* passes all tests and has uniformly distributed p-values for all tests in SP800-22 test suite.

2). Quality of randomness versus system precisions.

To understand quality of randomness of a scrambled VLM when system precision is increased, testing results of systems in 16-bit, 20-bit, and 24-bit are shown in Table 3.3. The results show that when system precision is 16-bit, the scrambled VLM fails 7 tests because of the non-normally distributed P-values shown in column 3. However, when system precision is larger than 24, the scrambled VLM passes all tests in SP800-22 and has uniformly distributed P-values.

3). Comparison with previous work.

We will compare the proposed system to several digitalized chaotic map based generators with respect to quality of randomness. The first system is Li's system [12] based on classical logistic map. To increase the output cycle of digitalized logistic map, Li used a timing-based reseeding method which disturbed the last five least significant bits of the output sequence by a fixed pattern when a period of time is reached. The second system is Robust Hyper-Chaotic System (*RHCS*) coupled by two robust logistic maps (RLM) [15].

Table 3.2: The statistical test results of the VLM with/without scrambling function by SP800-22

Tests	VLM		scrambled VLM	
	Yield	U-value	Yield	U-value
Frequency	0.9917	0.04374	0.9917	0.77276
Block freq.	0.9917	0	0.9833	0.99146
Cumulative*	0.9917	0.00347	0.9875	0.72303
Runs	0.9917	0.00038	0.9917	0.42203
Longest run	0.9833	0	0.9917	0.39245
Rank	0.9583	0.40709	0.9917	0.07044
FFT	0.9917	0.96429	0.9833	0.29925
Nonoverlap.*	0.9821	0.02469	0.9812	0.51330
Overlap.	0.9917	0.01596	0.9917	0.26445
Universal	0.9417	0	0.9583	0.78872
Apen	0.9917	0.00516	0.9833	0.35048
Random e.*	0.9966	0.67224	0.9895	0.69997
Random e.v.*	0.9918	0.08338	0.9899	0.09493
Serial*	0.9750	0.00054	0.9833	0.81194
Linear Comp.	0.9750	0.23276	0.9750	0.58520
Fail Count	2	3	0	0

*average result of multiple tests is shown.

The third system is Addabbo's system [14] based on piecewise-linear chaotic map. By utilizing the nonlinear property during truncation, Addabbo's system extended the period of Rényi chaotic map with length up to $2^n - 1$. Authors also provided a method to combine two subsystems to form a system that has maximum global cycle length and well quality of randomness.

The comparison results are divided into two groups according to bits of output per cycle. The first group contains *classical logistic map*, *Li's* and *Addabbo's* systems, where one bit is generated per cycle. In order to compare ours to systems in group one, the 16th bit of $\bar{x}_i$ is selected as the output of *VLM*. The second group contains *RHCS* which generates 24-bit output per cycle. All systems are operated in 32-bit precision or the closet precision

Table 3.3: The statistical test results of a scrambled VLM in different precisions by SP800-22

Tests	16-bit		20-bit		24-bit	
	Yield	U-value	Yield	U-value	Yield	U-value
Frequency	0.9917	0	0.9917	0.00001	0.9917	0.87553
Block freq.	0.9917	0	0.9917	0.07808	0.9833	0.32418
Cumulative*	0.9917	0	0.9833	0.00002	0.9917	0.60582
Runs	0.9917	0	0.9833	0.22286	0.9917	0.06688
Longest run	0.9917	0	0.9833	0.11651	0.9750	0.26445
Rank	0.9667	0.15520	0.9917	0.46859	0.9833	0.84858
FFT	0.9667	0.94960	0.9750	0.08217	0.9750	0.88813
Nonoverlap.*	0.9827	0.03304	0.9813	0.43792	0.9827	0.50418
Overlap.	0.9917	0	0.9750	0.37813	0.9750	0.46859
Universal	0.9917	0.01791	0.9833	0.45279	0.9833	0.87553
Apen	0.9917	0	0.9917	0.80433	0.9750	0.99820
Random e.*	0.9981	0.23156	0.9983	0.31502	0.9834	0.37574
Random e.v.*	0.9924	0.06688	0.9919	0.00348	0.9919	0.03978
Serial*	0.9917	0.00232	0.9875	0.43128	0.9833	0.83925
Linear Comp.	0.9833	0.33716	0.9917	0.75647	0.9917	0.98503
Fail Count	0	7	0	2	0	0

*average result of multiple tests is shown.

reported by the literatures. In Table 3.4, test results for scrambled *VLM* are compared with those for *Classical Logistic Map*, *Li's* [12] system, *RHCS* [15] and *Addabbo's* [14] system in terms of the number of failed tests. The test suites and the number of tests are shown in the first row. The columns, *Precision* and *Output Width* shows the precision of system and number of bits of one output, respectively.

With the scrambling function, *VLM* has least number of failure counts both in single bit output and multiple bits output. It shows that the scrambled *VLM* has good quality of randomness. In row 4, *Li's* system can improve the randomness when compared with *classical logistic map* showed in row 3 but still fail the *Crush* and *BigCrush* tests. The results of *Addabbo's* system are shown in row 5. Although a single *Addabbo's* system has

Table 3.4: Failure counts in statistical tests for different systems.

Systems	Prec-ision	Out-put Width	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)	*Big-Crush* (160)
scrambled VLM	32	1	0	0	3	8
classical logistic map	32	1	9	15	140	155
Li's [12]	32	1	1	15	144	156
Addabbo's [14]	31	1	2	14	122	141
Addabbo's [14]*	32	1	0	0	3	15
scrambled VLM	32	32	0	0	1	5
RHCS [15]†	32	24	0	0	25	59
MVLM(2)	32	32	0	0	0	0
MVLM(3)	32	32	0	0	0	0

*A combined 32-bit system by a 17-bit and a 15-bit subsystems.
†A hyper-chaotic system coupled by two 32-bit RLMs.

less implementation cost and passes most tests in SP800-22, it fails lots of tests in TestU01 testing suites. In row 6, the results for combined Addabbo's system of 17-bit and 15-bit sub-systems are also included. In row 7, *RHCS* passes tests in SP800-22 but fails several tests in Testu01. Finally, to verify the statistical properties of MVLM, results for MVLMs coupled by two 32-bit VLMs (labeled *MVLM(2)*) and three (labeled *MVLM(3)*) are presented. As presented in rows 8 and 9, MVLMs can pass all tests in all test suites when the number of coupled systems is more than 2. This statistical testing result shows that both scrambled VLM and combined Addabbo's systems have well statistical properties.

3.4.5 Reconstruction complexity

Since Adddabbo's system shows the best statistical property among all previous work, we will compare VLM and Addabbo's [14] system in reconstruction complexity. Addabbo's system is based on Rényi map which is a linear chaotic map. Addabbo's system is a good

pseudo random number generator because the system generates output sequence with well quality of randomness and maximum cycle length at low hardware cost. However, the system may be not suitable to apply in secure communications directly.

The first reason is a small set of parameter space. With particular parameters, Addabbo's system generates output sequence with maximum cycle length. The restricted parameter space reduce the complexity of cryptoanalysis. The second reason is the linearity of Rényi map. Since the piecewise-linear map has the same slope everywhere in each subinterval, the Lyapunov Exponent, topological, metric, and Rényi specific entropies are all equal. On the contrary, VLM is based on a piecewise and non-linear map which is different from piecewise-linear map in non-linear senses. These linearity properties can be further analyzed. For example, the autocorrelation function is calculated for 10,000 sequential states on trajectories of a 31-bit Addabbo's system and a 32-bit VLM, respectively. As shown in Figure 3.15(a), Addabbo's system has relative high correlations between sequential states when $-25 < Lag < 25$. On the contrary, in Figure 3.15(b), VLM has no peak value except $Lag = 0$. The high correlation among sequential states will become a flaw which can be utilized to reconstruct the system when sequential states are used as system outputs directly. Although the correlations can be reduced by avoiding using sequential states (skipping several states), the system will suffer from short the length of output cycle.

3.5 Hardware Architecture of MVLM

In this section, we show implementation of MVLM in hardware. We describe our designs in hardware description language (HDL), and then synthesize them by commercial tools. To be more specific, designs are written in Verilog and synthesised by Synopsys Design Compiler (Version X-2005.09-SP4) with TSMC $.18\mu m$ technology library. Area and timing information is obtained in gate-level netlist. Figure 3.16 shows the block diagram of the core to compute a single $VLM(\gamma, x)$. The block, *zero detector* computes two functions.

The first is to set $\gamma[31] = 0$ and $\gamma[32] = 0$ to satisfy the constraint that $\alpha^2\gamma$ should be a multiple of four. The second is to prevent VLM from generating all zero output sequence by assigning r[30]=1 when $\gamma = 0$. Afterwards, the blocks denoted by $truncation$, are used to implement the modular and truncation operations to keep the product in 32-bit precision. Since the truncation operation of *VLM* drops the most significant 16 bits during multiplication, (i.e., the logic circuit for these bits are no longer needed and can be removed), only 24×32 multiplier is required as compared to 32×32 multiplier needed by classical logistic map. Moreover, a 32-bit subtractor is used to compute $(1 - x_i)$.

The components for data-path in a scrambled *VLM* and other systems are compared in Table 3.5. In our VLM, two 24×32 multipliers are required. One comparator is used to check the input x and γ are zero or not, and one LFSR is used for the scrambling function. After synthesizing logic equation to gate-level netlist, the comparison of area cost in terms of gate counts for a scrambled *VLM* and other systems are shown in Table 3.6. The area cost of *VLM* is smaller than *classical logistic map* because the multipliers used in *VLM* is smaller than that used in *classical logistic map*. From [12], the gate-count for *Li's* system is calculated by total gate area divided by a two-input NAND gate which is equal to 9.98 μm^2(The same implementation technology as ours). Compared with *Li's* system, VLM is operating at lower frequency, but has smaller area and more complex output sequence. When compared to a single modified logistic map (*RLM*) [15], *VLM* has smaller area cost and higher throughput because one multiplier is removed as described in Section 3.1. The hardware cost of *Addabbo's* system is not available, but we believed that *Addabbo's* system has the smallest area cost since only one multiplier is required. Compared with *Addabbo's* system, *VLM* has more complex statistical properties with reasonable area overhead.

Furthermore, for MVLM implementation, Figure 3.17 shows the data-path architecture of a MVLM with $m = 4$. The data flow of the system is partitioned into 4 stages separated by registers denoted by black blocks. Table 3.7 shows the synthesized results including

Table 3.5: The components for data-path in VLM and other systems.

	VLM	classical logistic map	Li's [12]	Addabbo's [14]	RLM [15]
multiplier	2(24x32)	2(32x32)	1(32x32)	1(31x31)	3(32x32)
counter	0	0	1(10-bit)	0	0
comparator	1(32-bit)	0	1(10-bit)+1(32-bit)	0	2(32-bit)
LFSR	1(32-bit)	0	0	0	0

Table 3.6: The synthesis result for VLM and other systems.

	VLM	classical logistic map	Li [12]
Technology(μm)	.18	.18	.18
Area(#gate-count)	15697	20167	20075
Clock Frequency(Mhz)	100	100	200
Bits/Cycle	32	32	1
Bits/Second(Mbps)	3200	3200	200
Area Ratio	1	1.28	1.27
Throughput Ratio	1	1	0.06

control circuit for $m = 1$ to 4. The area and throughput of *RHCS* [15] coupled by two 32-bit RLMs are also reported. When compared to *RHCS* which is coupled by 2 RLMs, a MVLM with $m = 2$ has smaller area and higher throughput because of the proposed VLM reduces the number of multipliers in the data-path. Moveover, MVLM with $m = 2$ has better quality of randomness. The experimental results also show that the area of a MVLM is increased linearly with the number of VLMs. The system can be easily scaled up to higher degree.

To end this, we know that the statistical test results show that the output sequence generated by *VLM* in 32 bits per cycle can pass all tests. Moveover, under 100 Mhz operating frequency, VLM achieves 3,200 Mbps throughput, which is the best in all systems.

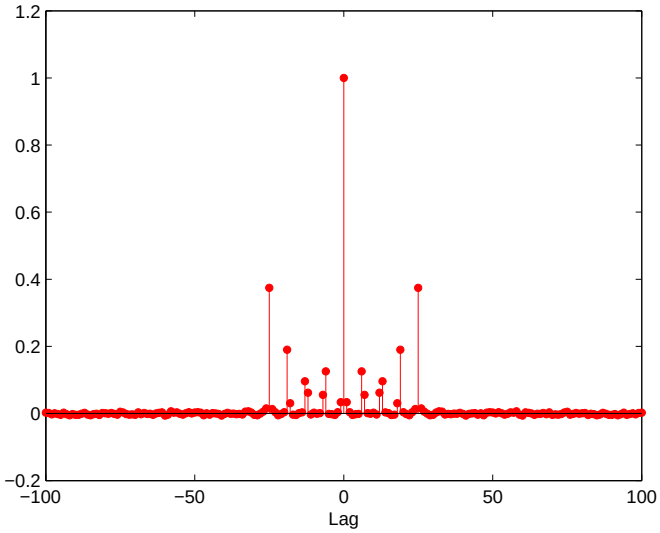
Table 3.7: The synthesized result of MVLM, and RHCS for $m = 1$ to 4

Number of VLMs(m)	1	2	3	4	RHCS [15]*
Technology(μm)	.18	.18	.18	.18	.18
Area(#gate-count)	15732	31655	46910	62223	37741
Clock Frequency(Mhz)	100	100	100	100	100
Bits/Cycle	32	32	32	32	24
Bits/Seconds(Mbps)	3200	3200	3200	3200	2400
Area Ratio	1	2.01	2.98	3.96	2.39

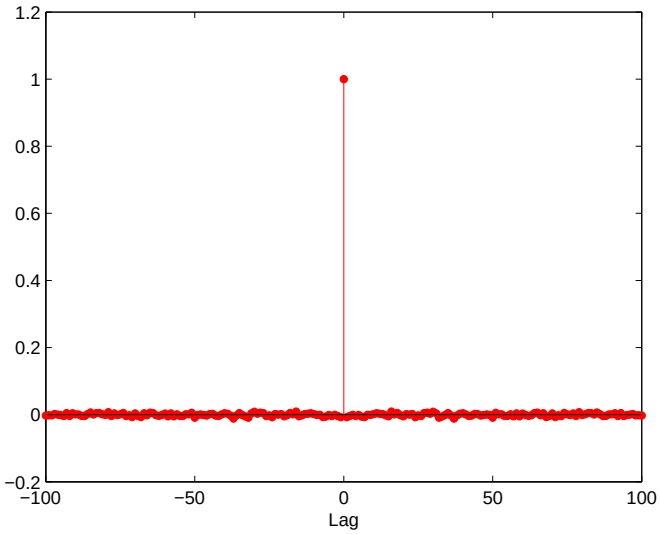
* A system coupled by two 32-bit RLMs [15].

3.6 Summary

A new chaotic map, VLM, has been proposed to have large parameter space without $windows$ and high throughput in low hardware cost. A 32-bit VLM with the proposed scrambling method can pass all tests in SP800-22 and the most stringent statistical testing suite in TestU01. With up to 3,200 Mbps throughput and complex output properties, VLM is suitable for security applications. We also showed a chaotic cryptographical scheme, MVLM, constructed by multiple VLMs. In an embodiment, by coupling four 32-bit VLMs, the MVLM generates the output sequence with a minimal length equal to $2^{128} - 1$ by a 128-bit external key.



(a)



(b)

Figure 3.15: Autocorleation functions for (a) a 31-bit Addabbo's system, and (b) a 32-bit VLM.

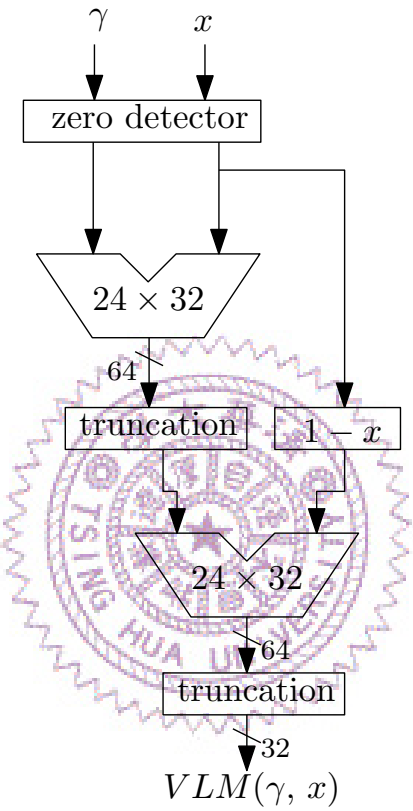


Figure 3.16: The architecture of the VLM.

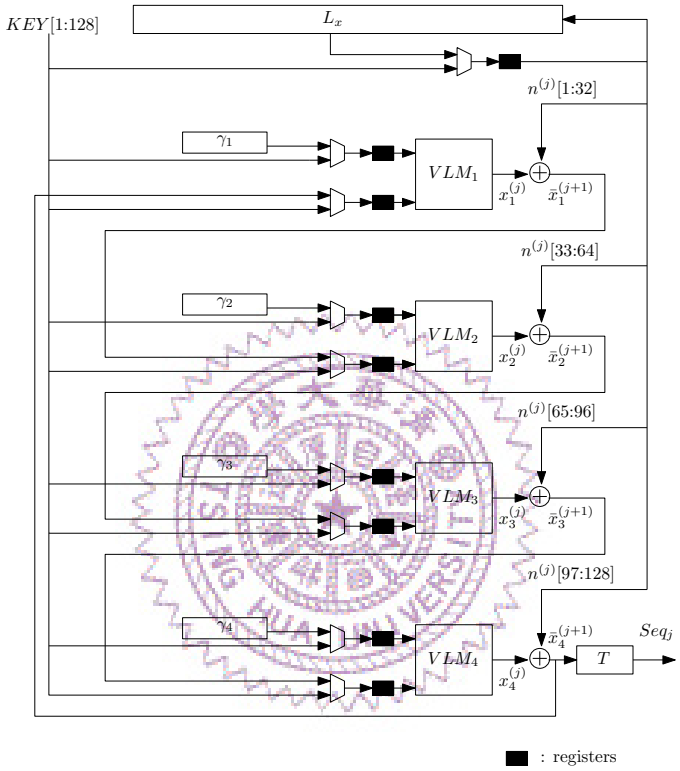


Figure 3.17: The data-path architecture of the MVLM with $m = 4$.

Chapter 4

Randomness Enhancement Using Digitalized Modified-Logistic Map

Pseudo Random Numbers Generators (PRNGs) are widely used in many applications, such as numerical analysis, integrated circuit testing, computer games and cryptography. The quality of randomness is usually the main criteria to distinguish different PRNGs. Besides the quality of randomness, implementation cost and throughput are also important factors to evaluate the effectiveness of PRNGs in applications such as modern-communication protocols.

Previous research on PRNGs can be classified into two approaches: linear and nonlinear. While linear approaches have the advantage of high throughput and low implementation cost, they suffer a low quality of randomness. Nonlinear approaches can be used to enhance the randomness properties but some of them require more hardware or more processing time.

Among other nonliner approaches, chaotic map based PRNGs have been proposed [12–14]. Moreover, in recent years non-quantized chaotic maps have been used for generating true random numbers [35].

In [12], Li *et al.* proposed a logistic-map based PRNG with timing-based reseeding method which replaced the last five least significant bits of the output sequence by a fixed

pattern when a period of time is reached. Although Li's system [12] improves the randomness quality of the output sequence when compared with a classical logistic map, Li's system still has some statistical weak points in testing results of SP800-22 [30] and TestU01 [31].

In [14], Addabbo *et al*. proposed a low-hardware complexity PRNGs based on a piecewise-linear Rényi map. By utilizing the nonlinear property during truncation, Addabbo's system extended the period of Rényi map with the length up to $2^n - 1$. The authors also provided a method to combine two subsystems to form a system that has the maximum global cycle length and good quality of randomness. One disadvantage of the system is that it is not easy to scale the system to high precision. The first reason is that the maximum cycle length of output sequence depends on values of parameters which are not easy to find. Second, the quality of randomness is not predictable even when system precision is increased. For example, in [14], a 24-bit system can pass statistical tests in SP800-22 but a 31-bit system which is the largest precision reported in [14] fails 2 of 15 tests in the same test suite.

In this chapter, we propose a PRNG which is based on a Digitalized Modified-Logistic Map (DMLM). Similar to [23] and [15], DMLM is defined in $\gamma \geq 4$ to extend the parameter space. It is shown that the modified logistic map is a pseudo-chaotic map with larger parameter space as compared with a classical logistic map [23] and is suitable for security communications [15]. However, high implementation cost renders it an un-suitable PRNG. We will propose two techniques, constant parameter selection and output sequence scrambling, to reduce computation cost without sacrificing the complexity of output sequence. Moveover, we show that it is easy to extend the precision of DMLM-PRNG.

The statistical test results show that with only one multiplication, our system passes all cases in SP800-22 [30]. Moreover, it passes most of cases in *Crush*, one of test suites of TestU01 [31], while Li's [12] and single Addabbo's [14] systems fail almost all cases.

The rest of this chapter is organized as follows. In Section 4.1, the Digitalized Modified-Logistic Map based PRNG (DMLM-PRNG) is presented. The parameter selection, scrambling method and implementation issues will also be discussed. In Section 4.2, we will compare DMLM-PRGN to other pseudo-chaotic map based PRNGs with respect to statistical properties, implementation cost and throughput. Finally, concluding remarks are given in Section 4.3.

4.1 Modified Logistic Map based PRNG

4.1.1 Digitalized Modified-Logistic Map

We know that a classical logistic map, $x_{i+1} = \gamma x_i(1 - x_i)$, where $x \in [0, 1]$, presents chaos when $3.57 < \gamma \leq 4$. To use $\gamma \geq 4$ and to digitalize a logistic map, we define a q-bit Digitalized Modified Logistic Map (DMLM) by

$$x_{i+1} = \lfloor [\gamma x_i(1 - x_i)] (\mathrm{mod}\, 1) \rfloor_q, \tag{4.1}$$

where the operation, x(mod 1), is used to keep x between [0,1) by dropping the integer part of x, and $\lfloor x \rfloor_q$ is a truncation function to preserve the most significant q bits of x and drop others. Furthermore, a q-bit binary number x in [0,1) can be presented by $x = \sum 2^{-j} x[j]$, where $x[j]$ is the j-th bit of x. It holds that x_i in Equation (4.1) is a q-bit binary number.

We verify pseudo-chaotic properties of DMLM by numerical experiments including *bifurcation analysis* and *discrete Lyapunov Exponent* (*dLE*) [17].

First, the bifurcation analysis is performed on 1000 γs evenly selected in $4 \leq \gamma \leq 2^{16}$ for a 32-bit DMLM. In our experiment, we do not detect any short-periodic *windows* in 1000 γs.

Second, the *discrete Lyapunov Exponents* (*dLEs*) [17] for DMLM is calculated, where the basic expression *dLEs* is defined as

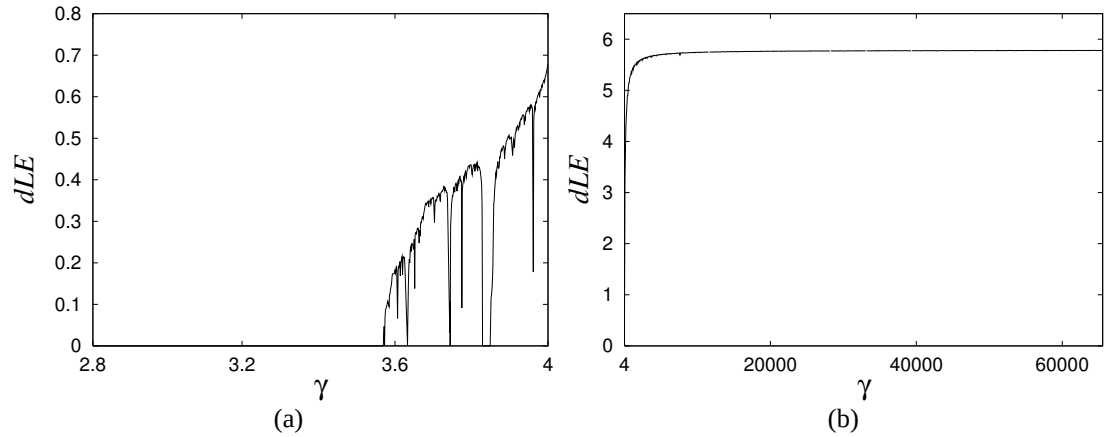


Figure 4.1: The discrete Lyapunov Exponents (*dLEs*) of a 32-bit DMLM for (a) $2.8 < \gamma < 4$, and (b) $4 \leq \gamma \leq 2^{16}$

$$\lambda_F = \frac{1}{m} \sum_{\mu=0}^{m-1} \ln \frac{d(F(M_{\mu+1}), F(M_\mu))}{d(M_{\mu+1}, M_\mu)}. \tag{4.2}$$

$\mathcal{M}$ is a finite subsegment of the trajectory in length m for a digitalized map F, and $d(M_\mu, M_\nu)$ is the distance between M_μ and M_ν, where M_μ and M_ν are in $\mathcal{M}$.

Figure 4.1 shows *dLEs* for two different ranges of γ. For each range, 1000 γs are selected evenly. For each γ, 1000 different initial conditions of x_0 are given to generate 1000 different subsegments, and in each segment, 1000 M_μs are computed (i.e. $m = 1000$). Then, the average *dLE* is reported.

As shown in Figure 4.1(a), when $2.8 < \gamma < 4$, *dLEs* for DMLM are non-positive in some γs. These non-positive *dLEs* result from the use of parameters in *windows*. However, as shown in Figure 4.1(b), when $\gamma \geq 4$, all *dLEs* are positive. These numerical results indicate that DMLM presents pseudo-chaos with large parameter space when $\gamma \geq 4$.

To apply DMLM in a PRNG, we will further define the value of γ. The objectives are twofold: low-cost computation and randomness quality. To reduce the computation cost, we focus on the subset of γ where $\gamma = 2^k$. In this case, only one multiplication is needed to compute the orbit because the multiplication of $\gamma = 2^k$ only requires a shifting operation.

Next, to increase randomness quality, we start from the point of implementing the multiplication in finite-precision arithmetic. Because the multiplication is defined in finite precision, truncation is needed after multiplying two numbers to store a product in the same number of bits. Let c be the $2q$-bit result of $x_i(1-x_i)$. Equation (4.1) can be rewritten by $x_{i+1} = c[k+1 : k+q]$ for $0 \leq k \leq q$. To have a uniformly distributed x_{i+1}, k should be close to $\frac{q}{2}$ so that middle bits of c are preserved after truncation. (The reason will be explained in the next subsection) Hence, our q-bit DMLM is reduced to

$$DMLM(x_i) = \lfloor [2^{\lceil \frac{q}{2} \rceil} x_i(1-x_i)] (\mathrm{mod}\, 1) \rfloor_q. \tag{4.3}$$

The sequence generated by DMLM is defined by $x_{i+1} = DMLM(x_i)$.

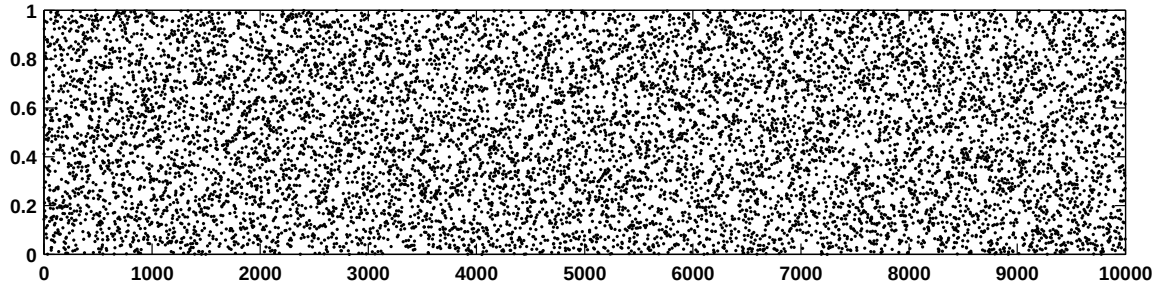
In hardware implementation, multiplying $2^{\lceil \frac{q}{2} \rceil}$ can be implemented by a shift operation. Now, DMLM requires only one multiplication. Moreover, in Section 4.1.4, we will show that DMLM can be implemented at lower cost as compared with a classical logistic map.

We analyze the spectrum of the trajectory generated by a 32-bit DMLM. First, in Figure 4.2(a) we plot the trajectory with 10,000 outputs. The result shows that the trajectory is visualized randomly. In Figure 4.2(b), the spectrum analyzed by FFT shows that the trajectory is broad-band or pseudo-random.

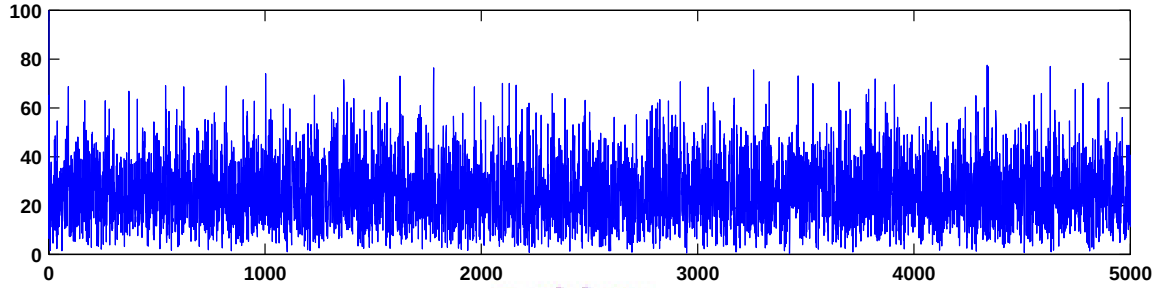
4.1.2 Scrambling Method

It is known that the length of an orbit generated by a digitalized pseudo-chaotic map is far below the total number of states. For example, let x_0=$(0.10001000)_H$, the length of the orbit generated by a 32-bit DMLM is 21,998 and by a 32-bit classical logistic map (γ=4) is 29,551. It is obviously insufficient if a long random number sequence is required. As discussed in Section 3.2, to increase the cycle length, the method to scramble several least significant bits is useful and widely used [5, 12, 34].

Similar to [36], we use a LFSR to scramble the output of DMLM. Figure 4.3 shows the proposed scrambling strategy, where L is a q-bit primitive LFSR and δ is a one-step delay



(a)



(b)

Figure 4.2: (a) Output value plotting, and (b) Spectrum analysis of a trajectory generated by DMLM with x_0=$(0.0f5a5a00)_H$.

block. The noise sequence generated by $n_{i+1} = L(n_i)$ is xor-ed with x_{i+1} to produce $\bar{x}_{i+1}$ before $\bar{x}_{i+1}$ is fed back to DMLM. The low bound of cycle length of the scrambled system is analyzed as follows. The low bound of the cycle length of output sequence is specified by $\Delta \cdot (2^l - 1)$, where Δ and l are the scrambling period and the register length of a LFSR, respectively. As discussed in Section 3.2, to have the maximum output cycle length, we let $\Delta = 1$ and $l = q$. Hence, the low bound of cycle length will be $2^q - 1$. The overhead of the scrambling function is a q-bit LFSR.

An experiment is conducted to show that a 32-bit scrambled DMLM has uniformly distributed outputs by measuring the 1's probability of the most significant bit of $\bar{x}_i$. Here, the length of each sequence is 10^6 bits. Figure 4.4 shows that the 1's probability of the output sequence is close to $\frac{1}{2}$ when the period, Δ, is decreasing.

Since the LFSR and DMLM are not functionally independent, the transition behavior

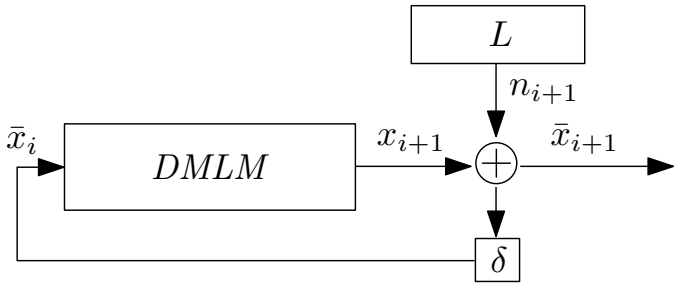


Figure 4.3: Scrambling strategy for DMLM.

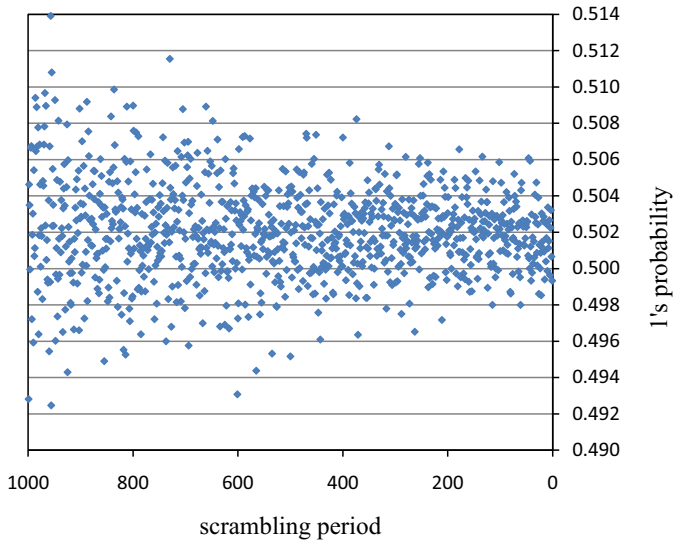


Figure 4.4: 1's probability with different scrambling periods.

of the LFSR and DMLM will affect the distribution of $\bar{x}_i$. The transition behavior of the scrambled system will be analyzed in the next subsection.

4.1.3 Property of the System

In this section, we will show that our constant parameter selection together with the scrambling method can produce uniformly distributed outputs, which is an important property of random number generators.

Our scrambled system can be defined as follows.

Definition 1. *Let $\bar{x}_i$, x_i, and n_i be q-bit binary numbers. Let F and L be a pseudo-chaotic map and a primitive LFSR, respectively. Let $x_{i+1} = F(\bar{x}_i)$ and $n_{i+1} = L(n_i)$. The scrambled system is defined by $\bar{x}_{i+1} = D(\bar{x}_i, n_i) = F(\bar{x}_i) \oplus L(n_i)$.*

Notice that in Definition 1, $F(\bar{x}_i)$ and $L(n_i)$ are functionally dependent and potentially statistically dependent. Although $L(n_i)$ is uniformly distributed, $F(\bar{x}_i) \oplus L(n_i)$ is not always uniformly distributed.

Assume that $\{\bar{x}_i[j]\}_{i>0}$ is selected as the binary output sequence. The objective is to prove that a uniformly distributed sequence, $\{\bar{x}_i[j]\}_{i>0}$, is generated by the given system.

Let the state transition probability matrix of $D : \bar{x}_i[j] \rightarrow \bar{x}_{i+1}[j]$ be

$$T_D = \begin{bmatrix} d_{0,0} & d_{1,0} \\ d_{0,1} & d_{1,1} \end{bmatrix}, \tag{4.4}$$

where $d_{\mu,\nu} = P(\bar{x}_{i+1}[j] = \nu | \bar{x}_i[j] = \mu)$, for $\mu, \nu = 0, 1$. We define the fix-point condition for the discrete Frobenius-Perron equation for the transition D as equation,

$$p = T_D \times p = [p_0\, p_1]^T, \tag{4.5}$$

where $p_\mu = P(\bar{x}_i[j] = \mu)$ is the state probability for $\mu = 0, 1$.

For a uniformly distributed sequence ($\{\bar{x}_i[j]\}_{i>0}$), p in Equation (4.5) should be equal to $[\frac{1}{2}\ \frac{1}{2}]^T$.

The sufficient conditions for Equation (4.5) with $p = [\frac{1}{2}\ \frac{1}{2}]^T$ are discussed in the following properties.

Property 1. *If $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent, then Equation (4.5) holds with $p = [\frac{1}{2}\ \frac{1}{2}]^T$. Also, T_D is a matrix with all entries equal to $\frac{1}{2}$.*

Proof. Let T_F and T_N are the state transition probability matrices of $F : \bar{x}_i[j] \rightarrow x_{i+1}[j]$ and $N : x_{i+1}[j] \rightarrow \bar{x}_{i+1}[j]$, respectively. T_F and T_N can be defined as follows.

$$T_F = \begin{bmatrix} f_{0,0} & f_{1,0} \\ f_{0,1} & f_{1,1} \end{bmatrix} and \quad T_N = \begin{bmatrix} n_{0,0} & n_{1,0} \\ n_{0,1} & n_{1,1} \end{bmatrix}, \tag{4.6}$$

where $f_{\mu,\nu} = P(x_{i+1}[j] = \nu | \bar{x}_i[j] = \mu)$ and $n_{\mu,\nu} = P(\bar{x}_{i+1}[j] = \nu | x_{i+1}[j] = \mu)$ for $\mu, \nu = 0, 1$.

We can rewrite Equation (4.5) by $p = (T_N \times T_F) \times p$. Because $L(n_i)$ is uniformly distributed, $n_{\mu,\nu} = \frac{1}{2}$ for $\mu, \nu = 0, 1$. Moreover, we know that $f_{0,0} + f_{0,1} = 1$, and $f_{1,0} + f_{1,1} = 1$. Thus, T_D is a matrix with all entries equal to $\frac{1}{2}$. This implies $p = [\frac{1}{2}\ \frac{1}{2}]^T$ is the stable probability of T_D. □

Property 2. *If $\bar{x}_i[j]$ and $x_i[j]$ are statistically independent, then $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent. Therefore, the assertion of Property 1 holds.*

Proof. From the assumption, we see that $F(\bar{x}_i)$ is a fully disturbing channel which can remove any statistical dependence between $\bar{x}_i[j]$ and $x_i[j]$. Furthermore, $n_i[j]$ and $x_i[j]$ are statistically independent. If $x_i[j]$ is uniformly distributed, then $n_i[j]$ and $\bar{x}_i[j]$ are statistically independent, which follows from the balanced property of the XOR operation. Thus, the assertion of Property 1 also holds. □

According to Property 1 and Property 2, to have uniformly distributed outputs for the scrambled system, DMLM ($F(\bar{x}_i)$) should be a fully disturbing channel to remove any correlation between $\bar{x}_i[j]$ and $x_i[j]$. That is, transition probability of T_F should be uniform.

In Equation (4.1), γ is equal to 2^k. The objective of our constant parameter selection is to find a k so that the transition probability of Equation (4.1) is close to uniform. Let $x'_i = (1 - x_i)$ and c be the $2q$-bit result of $x_i \times x'_i$. Equation (4.1) can be rewritten by $x_{i+1} = c[k+1 : k+q]$ for $0 \leq k \leq q$. Considering a case where $c[2q]$ is selected as the output sequence. Since $c[2q] = x_i[q] \times x'_i[q]$, $c[2q]$ is equal to 0 when $x_i[q]$ is 0. The high correlation between $c[2q]$ and $x_i[q]$ results in a non-uniform transition probability and makes $c[2q]$ a un-suitable output candidate. Similarly, high correlation exists between the most significant bits of c ($c[1]$) and x_i ($x_i[1]$).

On the contrary, middle bits of c computed by more number of partial products (as

compared to the most/least significant bits of c) and carry-in bits depend on each bit of x_i more uniformly.

An experiment is conducted to understand correlations between bits of c and x_i with $q = 32$. The state transition probability matrix going from the t-th bit of x_i ($x_i[t]$) to the j-th bit of c ($c[j]$) can be defined by a 2-by-2 matrix T_F, where the entry at μ-th column and ν-th row shows the probability $f_{\mu,\nu} = P(c[j] = \nu | x_i[t] = \mu)$, where $\mu, \nu = 0, 1$.

The following equation is used to measure the maximum distance between $f_{\mu,\nu}$ and $\frac{1}{2}$ for each T_F for different values of t and j.

$$\rho_{max} = max(|f_{\mu,\nu} - \frac{1}{2}|), \tag{4.7}$$

where $\mu, \nu = 0, 1$.

The smaller ρ_{max} is, the more uniform transition probability T_F has. Two sets of ρ_{max} are calculated. The first one is ρ^H_{max} which is the transition probability from the most significant bit $x_i[1]$ to $c[j]$ for $1 \leq j \leq 32$. The second one is ρ^L_{max} which is the transition probability from the least significant bit $x_i[32]$ to $c[j]$ for $31 \leq j \leq 63$. As shown in Figure 4.5, ρ^H_{max} and ρ^L_{max} are decreasing when j is close to 32. It shows that $f_{\mu,\nu}$ is close to $\frac{1}{2}$ when middle bits of c are selected. Please be noticed that ρ_{max}s smaller than 10^{-6} are not shown in the graph.

Note that, the above discussion is still empirical because the result shown in Figure 4.5 is not a rigid statistical analysis. However, the discussion could provide some insights into this special case of pseudo-random number generation.

Hence, to have uniform transition probabilities from x_i to x_{i+1} for Equation (4.1), k should be close to $\frac{q}{2}$ to preserve middle bits of the product after truncation. We know that our constant parameter selection for DMLM will produce more uniformly transition probabilities. With our constant parameter selection and the scrambling method, the DMLM-based PRNG, DMLM-PRNG, can generate uniformly distributed outputs.

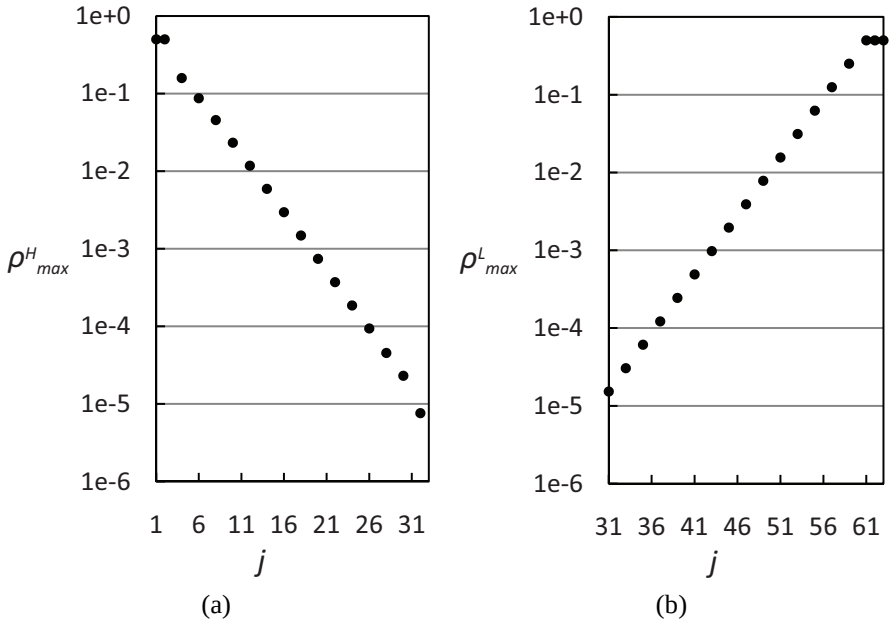


Figure 4.5: (a)ρ^H_{max} for $1 \leq j \leq 32$, and (b) ρ^L_{max} for $31 \leq j \leq 63$

4.1.4 Implementation

We take a 32-bit DMLM-PRNG as an example to show the efficiency of our DMLM-PRNG implementation. The structure of DMLM-PRNG is shown in Figure 4.6. The DMLM-PRNG is divided into two modules, *DMLMCore* and *ScraFunc*. The first module, *DMLM-Core*, generates state value of DMLM. The second module, *ScraFunc*, scrambles the state value by a noise generated by *LFSR*(L). The scrambled state value will be fed back to *DMLMCore*. Finally, *OUT* is the output generator which selects the most significant bit of $\bar{x}_{i+1}$ to be the random number sequence.

The current state value of DMLM-PRNG is stored in a 32-bit register, *StateREG*. A 32-bit subtractor is used to compute $(1 - x_i)$. To compute the next state, only one multiplier is needed for $x_i(1 - x_i)$. Operations to multiply 2^{16} and truncate the result of $x_i(1 - x_i)$ to 32-bit are implemented by signals selection in *TrunOP*. Since the truncation operation drops the most significant 16 bits during multiplication (i.e., the logic circuit for these bits

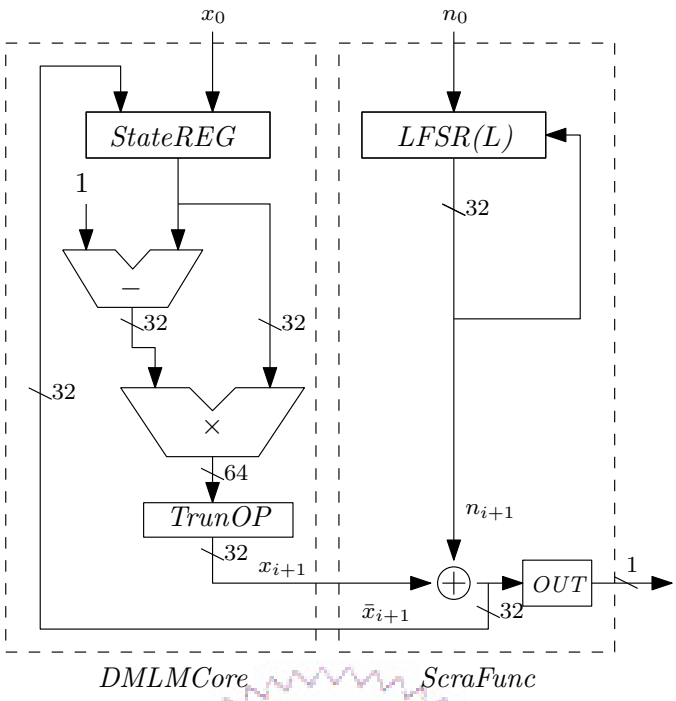


Figure 4.6: Architecture of DMLM-PRNG.

are no longer needed and can be removed), the required area is only a 24×32 multiplier as compared to the area of a 32×32 multiplier needed by classical logistic map. To compare the area cost of DMLM used in a 32-bit DMLM-PRNG and a classical logistic map, two maps are implemented and synthesized with TSMC $.18\mu m$ technology library. Area and timing information are obtained in gate-level netlist. The area cost in terms of number of 2-input-NAND gate is shown in Table 4.1. Under the same timing constraint, the area cost of *DMLM* is 85.6% of that of *Classical Logistic Map* with γ=4. The detailed performance evaluation of DMLM-PRNG will be discussed in the next section.

4.2 Performance Evaluation

In this section, we will compare DMLM-PRGN to other pseudo-chaotic map based PRNGs with respect to statistical properties, implementation cost and throughput.

Table 4.1: Comparisons of area and timing for 32-bit DMLM and classical logistic map in hardware.

Maps	Area (2-NAND)	Data Arrival Time (ns)
Classical Logistic Map (γ=4).	10563	5.00
DMLM (γ=2^{16})	9046	4.83
DMLM/Classical Logistic Map	85.6%	96.6%

Similar to Section 3.4.4, the same tools are used for statistical analysis. To evaluate the randomness of our system, two test suites, NIST SP800-22 [30] and TestU01 [31] are used. SP800-22 test suite has been the standard reference for PRNGs. We use SP800-22 as our first randomness test. Then, to further compare the randomness among DMLM-PRNG and other PRNGs, a more complex test suite, TestU01, is used for testing.

The configuration for SP800-22 test is as follows. α is equal to 0.01 and T=120. Hence, 120 sequences will be generated by DMLM-PRNG with the length of 10^6 bits. Finally, generated sequences are fed to the test suite. Each test will produce a *P-value* from SP800-22. *P-value* should be in range, [0.01, 1.00], to pass the test. As suggested in SP800-22, for each test, the minimum pass rate of a pseudo random source is 0.9627 out of 120 binary sequences. The *U-value* is also reported for the distribution measurement of collected *P-values*. If *U-value* is greater than 10^{-4}, then the sequences can be considered to be a pseudo random sequence with acceptable quality of randomness. Table 4.2 shows the parameter configuration used in the following SP800-22 tests.

We first conduct an experiment to understand the quality of randomness of DMLM-PRNG when system precision is increased. As shown in Table 4.3, systems in 20-bit, 24-bit and 32-bit are tested. When system precision is 20-bit, the proposed PRNG passes all tests but has 4 failure U-values. However, when the system precision is larger than 24-bit, DMLM-PRNG passes all tests in SP800-22 and has uniformly distributed *P-values* for each test. The experimental results show that the statistical properties of DMLM-PRNG is

Table 4.2: Parameters for SP800-22.

Block freq.	m=128	Serial	m=16
Longest run	M=10000	Apen	m=10
Nonoverlap.	m=9	Linear Comp.	m=500
Overlap.	m=9	Universal	L=7,Q=1280

becoming better when we increase the system precision. The LFSR used for scrambling function is the only parameter needed to change. In this experiment, LFSRs for 20-bit, 24-bit and 32-bit are $x^{20}+x^{19}+x^{18}+x^{15}+x^{10}+x^3+1$, $x^{24}+x^{22}+x^{21}+x^{19}+x^{18}+x^{16}+x^{15}+x^{11}+x^9+x^8+1$, and $x^{32}+x^{22}+x^{20}+x^{14}+x^{13}+x^{11}+1$, respectively.

The discussion in Section 4.1.3 has shown that the middle bits of $\bar{x}_i$ are those with the best randomness properties. We compare the randomness properties of two cases, where the first bit and the 16th bit of $\bar{x}_i$ are respectively selected as output sequences. When the precision is 20-bit, the failure-count of U-value is indeed reduced from 4 to 2 for the middle bit case.

To show the improvement of randomness quality by DMLM-PRNG as compared to other systems, three testing suites, SP800-22, and *SmallCrush*, *Crush* in TestU01 [31] are applied. Recommended by TestU01, *SmallCrush* including 15 sub-tests is taken as a fast check for the basic randomness requirement. Next, $Crush$ needs 2^{35} output sequences to perform further 144 tests. For each test, a *P-value* is calculated. If *P-value* is out of the range, [0.001, 0.9990], the sequence fails the test.

1). Randomness improvement by scrambling function.

To understand the efficiency of the scrambling function in DMLM, tests with/without scrambling function are performed. The failure counts of tests in different testing suites are shown in Table 4.4. Without scrambling function, DMLM fails some tests because the short output length. On the contrary, with scrambling function, 32-bit DMLM pass all tests in SP800-22, *SmallCrush* and almost all tests in *Crush* test suites.

2). Comparison with previous work.

Table 4.3: Testing results of DMLM-PRNG in different precisions by SP800-22.

Tests	20-bit		24-bit		32-bit	
	Yield	U-value	Yield	U-value	Yield	U-value
Frequency	0.9917	0	0.9917	0.15520	0.9917	0.00619
Block freq.	0.9917	0	0.9750	0.02381	0.9750	0.33716
Cumulative*	0.9917	0	0.9791	0.31171	0.9917	0.46264
Runs	0.9833	0	0.9917	0.28730	0.9917	0.12837
Longest run	0.9833	0.09561	0.9833	0.50093	0.9917	0.88813
Rank	0.9833	0.00699	0.9917	0.42203	0.9833	0.46859
FFT	0.9750	0.25355	0.9667	0.50093	0.9750	0.16260
Nonoverlap.*	0.9791	0.23426	0.9805	0.48152	0.9816	0.49706
Overlap.	0.9833	0.00095	0.9667	0.98088	0.9917	0.80433
Universal	0.9667	0.01341	0.9833	0.23276	0.9833	0.32418
Apen	0.9833	0.36414	0.9833	0.00887	0.9833	0.48464
Random e.*	0.9928	0.41009	0.9834	0.29321	0.9945	0.14739
Random e.v.*	0.9968	0.08301	0.9877	0.00788	0.9983	0.00131
Serial*	0.9875	0.63531	0.9833	0.39006	0.9833	0.28810
Linear Comp.	0.9917	0.60245	0.9833	0.78872	0.9667	0.00836
Failure Counts	0	4	0	0	0	0

*average result of multiple tests is shown.

In Table 4.5, test results for 32-bit *DMLM-PRNG* are compared with those for *Classical Logistic Map*, *Addabbo's* [14] system, and *Li's* [12] system. For Addabbo's system, the precision for the experiment is 31-bit because it is the largest precision reported in [14]. In the last row, the results for combined Addabbo's system of 17-bit and 15-bit sub-systems are also included. This table shows that both DMLM-PRNG and combined Addabbo's systems have good statistical properties.

3). Adding Scrambling function to other systems.

The total numbers of states in each testing system are different. For example, DMLM-PRNG has 64-bit (32+32-bit LFSR) states and Li's has 42-bit (32+10-bit counter) states. In order to provide each system comparable number of states, the scrambling function is applied to other systems in the same way it is applied to DMLM-PRNG. In Table 4.6, testing

Table 4.4: Randomness improvement by scrambling function in terms of failure count in statistical tests.

System	Scrambling Function	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)
DMLM	W/O	9	14	139
	W/T	0	0	3

Table 4.5: Failure counts in statistical tests for different systems.

System	Prec-ision	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)
DMLM-PRNG	32	0	0	3
Classical Logistic Map	32	9	15	140
Addabbo's [14]	31	2	14	122
Li's [12]	32	1	15	144
Addabbo's [14](combined)	32	0	0	3

results show that *Classical Logistic Map* and *Li's* systems still fail lots of tests even when the number of registers is doubled. Moreover, *Li's* system is worse than its non-scrambling version. Similar to *DMLM-PRNG, Addabbo's* system can improve the quality of randomness by scrambling. The table also shows that *DMLM-PRNG* performs slightly better than *Addabbo's* system in terms of failure count. The row labeled Addabbo's [14](combined) shows the results of the system which is also a kind of scrambled system combined with a 15-bit and a 17-bit Addabbo's systems. It shows that the randomness is the same with ours system. Nerveless, for the combined system, the system precision (cycle length) is not easy to extend and also the quality of randomness is not predictable. Moveover, results for a non-scrambled 64-bit classical logistic map is also reported in the row labeled *Classical Logistic Map (no-scr.)*. It shows that the quality of randomness can be improved by precision increase (see a 32-bit version in Table 4.5), but the trajectory of a digitalized logistic map eventually enters a loop with unknown length. The quality of randomness of a 64-bit classical logistic map is worse than that of DMLM-PRNG in terms of failure count.

Table 4.6: Failure counts in statistical tests with scrambling function.

System	Number of Registers	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)
DMLM-PRNG	32+32	0	0	3
Classical Logistic Map	32+32	8	14	124
Li's [12]	42+32	12	13	123
Addabbo's [14]	31+31	0	1	14
Addabbo's [14](combined)	15+17	0	0	3
Classical Logistic Map (no-scr.)	64	0	0	27

The transition probabilities of Addabbo's system are close to uniform while those of classical logistic map and Li's systems are not. Hence, when the scrambling function is used, it results in good result for Addabbo's system but worse statistic properties in classical logistic map and Li's systems.

The last experiment is to compare the components with respect to data-path for DMLM-PRNG and other systems. In order to compare systems in the same process technology, our system is synthesized with UMC .18μm technology. The timing and area information are reported with gate-level netlist. As shown in Table 4.7, in a 32-bit *DMLM-PRNG*, one 24×32 multipliers and a 32-bit LFSR are required. From [12], the gate-count for *Li's* system is calculated by total gate area divided by a 2-input-NAND gate which is equal to 9.98 μm^2. The comparison of area cost in terms of gate counts for *DMLM-PRNG* and other systems are shown in the column denoted by *Area*. The last column, *ThroughPut/Area*, shows that the area efficiency of *DMLM-PRNG* is 200% of that of *Li's* system.

Compared with *Li's* system, *DMLM-PRNG* has smaller area and more complex output sequence with the same throughput. As compared to *Addabbo's* system, *DMLM-PRNG* is easy to scale to large precision with reasonable area overhead.

Table 4.7: Comparisons of data-path components, area, and throughput.

PRNGs	Multi-pliers	Count-ers	LFSRs	Area (2-NAND)	Throu-ghPut (bits/sec)	Throu-ghPut/ Area
DMLM-PRNG	1(24x32*)	0	1(32-bit)	9517	180M	0.018
Li [12]	1(32x32)	1(10-bit)	0	20075	200M	0.009
Addabbo [14]	1(31x31)	0	0	N.A.	N.A.	N.A.
Addabbo [14] (combined)	1(15x15), 1(17x17)	0	0	N.A.	N.A.	N.A.

*multiplier with the same area cost.

4.3 Summary

In this chapter, we have proposed a nonlinear, Digitalized Modified-Logistic Map based Pseudo Random Number Generator (DMLM-PRNG). With our constant parameter selection and scrambling method, DMLM-PRNG has output sequence with good randomness quality at low implementation cost. Statistical test results have shown that the randomness quality of DMLM-PRNG is as good as Addabbo's [14] combined system and better than Li's [12] system. Moreover, our system has shown better scalability than Addabbo's system.

Chapter 5

Conclusions

In this dissertation, we have proposed several modified logistic maps for secure communications and pseudo random number generation. To increase the number of parameter, a Robust Logistic Map (RLM) is proposed for $\gamma > 4$. As compared to a classical logistic map, the parameter space is large enough for the security applications. Based on RLM, a Robust Hyper-Chaotic Encryption-Decryption System (RHCS) is proposed for digital secure-communications. By coupling n RLMs, a RHCS has large parameter space and high complexity output. Moreover, RHCS is difficult to re-construct. We have shown that the output sequence of RHCS has good quality of randomness for secure communications. The example of implementation demonstrates that the proposed multiple-cycle and pipelined architectures are effective for area and performance optimization, respectively.

Second, we proposed a Variational Logistic Map (VLM) to reduce the computation cost and improve the quality of randomness of RLM. VLM has large parameter space without $windows$. Moreover, it has high throughput with low hardware cost and good quality of randomness. A 32-bit VLM passed all tests in SP800-22 and the most of tests in the stringent statistical testing suite in TestU01. With up to 3,200 Mbps throughput and complex output properties, VLM is suitable for high throughput secure communications. Furthermore, we proposed a chaotic cryptographical scheme, MVLM, constructed by multiple VLMs. In an embodiment using four 32-bit VLMs, the MVLM generates the output se-

quence with a minimal length equal to $2^{128} - 1$ by a 128-bit external key.

Finally, for pseudo random number generation, we have proposed a nonlinear, Digitalized Modified-Logistic Map based Pseudo Random Number Generator (DMLM-PRNG). To reduce the computation cost without reducing the quality of randomness, we proposed two techniques, constant parameter selection and output scrambling, for DMLM-PRNG. The properties of the scrambled system indicated that our DMLM-PRNG can generate uniformly distributed outputs. Statistical test results have shown that randomness quality of DMLM-PRNG is as good as Addabbo's [14] combined system and significantly better than Li's [12] system with smaller area cost. Moreover, our PRNG has better scalability than Addabbo's system.

5.1 Future Work

In this dissertation, our proposed systems have large parameter space and good quality of randomness with low computation cost. The proposed scrambling method is suitable for our systems to increase the output cycle length and the quality of randomness. The further cryptoanalysis on our system will be conducted.

We also shown the property that the transition probabilities of the chaotic map should be uniformly distributed to have uniformly distributed outputs for the scrambled system. Based on the property, we can further analyze the transition behavior of different chaotic maps and the scrambling function.

Moreover, optimization of the hardware architecture for our system will be studied.

Bibliography

[1] S. H. Strogatz, *Nonlinear dynamics and chaos*. Boston: Addison-Wesley, 1994.

[2] R. Matthews, "On the derivation of a chaotic encryption algorithm," *Cryptologia*, vol. 13, no. 1, pp. 29–42, 1989.

[3] D. R. Frey, "Chaotic digital encoding: an approach to secure communication," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process*, vol. 40, no. 10, pp. 660–666, 1993.

[4] G. Heidari-Bateni and C. D. McGillem, "A chaotic direct-sequence spread-spectrum communication system," *IEEE Trans. Comm.*, vol. 42, pp. 1524–1527, 1994.

[5] J. Cermák, "Digital generators of chaos," *Phys. Lett. A*, vol. 214, pp. 151–160, 1996.

[6] M. Götz, K. Kelber, and W. Schwarz, "Discrete-time chaotic encryption systems – part I: statistical design approach," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 44, no. 10, pp. 963–970, 1997.

[7] C. Juang, T. M. Hwang, J. Juang, and W. W. Lin, "A synchronization scheme using self-pulsating laser diodes in optical chaotic communication," *IEEE J. Quantum Electron.*, vol. 36, no. 3, pp. 300–304, 2000.

[8] S. Li, X. Mou, and Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography," in *Proc. Progress in Cryptology-IndoCrypt, LNCS*, vol. 2247. Springer-Verlag, 2001, pp. 316–329.

[9] G. Jakimoski and L. Kocarev, "Chaos and cryptography: Block encryption ciphers based on chaotic maps," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 2, pp. 163–169, 2001.

[10] C. Juang, S. T. Hwang, C. Y. Liu, W. Wang, T. M. Hwang, J. Juang, and W. W. Lin, "Subcarrier multiplexing by chaotic multi-tone modulation," *IEEE J. Quantum Electronics*, vol. 39, no. 10, pp. 1321–1326, 2003.

[11] G. Álvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurc. Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.

[12] C. Y. Li, J. S. Chen, and T. Y. Chang, "A chaos-based pseudo random number generator using timing-based reseeding method," in *IEEE Proc. Int. Symp. on Circuits and Systems*, 2006, pp. 21–24.

[13] X. Wang, J. Zhang, and W. Zhang, "Chaotic keystream generator using coupled NDFs with parameter perturbing," in *Proc. 5th Int. Conf. on Cryptology and Network Security, LCNS*, vol. 4301. Springer-Verlag, 2006, pp. 270–285.

[14] T. Addabbo, M. Alioto, A. Fort, A. Pasini, S. Rocchi, and V. Vignoli, "A class of maximum-period nonlinear congruential generators derived from the Rényi chaotic map," *IEEE Trans. Circuits Syst. I, Regular Papaers*, vol. 54, no. 4, pp. 816–828, 2007.

[15] S. L. Chen, S. M. Chang, T. T. Hwang, and W. W. Lin, "Digital secure-communication using robust hyper-chaotic systems," *Int. J. Bifurc. Chaos*, vol. 18, no. 11, pp. 3325–3339, 2008.

[16] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Phys. Rev. Lett.*, vol. 64, pp. 821–824, 1990.

[17] L. Kocarev, J. Szczepanski, J. M. Amigo, and I. Tomovski, "Discrete chaos–I: Theory," *IEEE Trans. Circuits Syst. I, Regular Papaers*, vol. 53, no. 6, pp. 1300–1309, 2006.

[18] W. Stallings, *Cryptography and Network Security: Principles and Practices*. New-Jersey: Pearson Education, 2003.

[19] Y. Tao, "A survey of chaotic secure communication systems," *International Journal of Computational Cognition*, vol. 2, no. 2, pp. 81–130, 2004.

[20] P. Li, Z. Li, W. A. Halang, and G. Chen, "A stream cipher based on a spatiotemporal chaotic system," *Chaos, Solitons & Fractals*, vol. 32, pp. 1867–1876, 2007.

[21] D. D. Wheeler, "Problems with chaotic cryptosystems," *Cryptologia*, vol. 13, no. 3, pp. 243–250, 1989.

[22] G. Álvarez, F. Montoya, M. Romera, and G. Pastor, "Cryptanalyzing a discrete-time chaos synchronization secure communication system, Tech. Rep. 3, 2004.

[23] S. M. Chang, M. C. Li, and W. W. Lin, "Asymptotic synchronization of robust hyperchaotic systems and its applications," *Nonlinear Anal. Real World Appl.*, vol. 10, no. 2, pp. 869–880, 2009.

[24] M. I. Sobhy and A. e. R. Shehata, "Methods of attacking chaotic encryption and countermeasures," *IEEE International Conf. on Acoustics, Speech, and Signal Processing*, vol. 12, pp. 1001–1004, 2001.

[25] T. S. Parker and L. O. Chua, *Practical numerical algorithms for chaotic systems*. New-York: Springer-Verlag, 1989.

[26] S. Hu, Y. Zou, J. Hu, and L. Bao, "A synchronous CDMA system using discrete coupled-chaotic sequence," in *IEEE Proc. of Southeastcon '96: Bringing Together Education, Science and Technology*, 1996, pp. 484–487.

[27] H. Lu, S. Wang, X. Li, G. Tang, J. Kuang, W. Ye, and G. Hu, "A new spatiotemporally chaotic cryptosystem and its security and performance analyses," *Chaos*, vol. 14, no. 3, pp. 617–629, 2004.

[28] P. Li, Z. Li, W. A. Halang, and G. Chen, "Analysis of a multiple output pseudorandom-bit generator based on a spatiotemporal chaotic system," *Int. J. Bifurcation Chaos*, vol. 16, pp. 2949–2963, 2006.

[29] ——, "A multiple pseudorandom-bit generator based on a spatiotemporal chaotic map," *Phys. Lett. A*, vol. 349, pp. 467–473, 2006.

[30] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," National Inst. of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST Special Publication 800-22, 2001.

[31] P. L'Ecuyer and R. Simard, "Testu01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 1–22, 2007.

[32] E. Barreto, B. R. Hunt, C. Grebogi, and J. A. Yorke, "From high dimensional chaos to stable periodic orbits: the structure of parameter space," *Phys. Rev. Lett.*, vol. 78, no. 24, pp. 4561–4564, 1997.

[33] S. M. Chang, M. C. Li, and W. W. Lin, "Asymptotic synchronization of robust hyperchaotic systems and its applications," *Nonlinear Anal. Real World Appl.*, vol. 10, no. 2, pp. 869–880, 2009.

[34] T. Sang, R. Wang, and Y. Yan, “Perturbance-based algorithm to expand cycle length of chaotic key stream,” *Electrno. Lett.*, vol. 34, pp. 873–874, 1998.

[35] S. Callegari, R. Rovatti, and G. Setti, “Embeddable adc-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos,” *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 793–805, 2005.

[36] S. L. Chen, T. T. Hwang, S. M. Chang, and W. W. Lin, “A fast digital chaotic generator for secure communication,” *Int. J. Bifurc. Chaos*, p. to appear, 2010.