A Fast Digital Chaotic Generator for Secure Communication

Shih-Liang Chen* TingTing Hwang † Shu-Ming Chang‡
Wen-Wei Lin§

Abstract

In this paper, we propose a digitalized chaotic map, Variational Logistic Map (VLM), modified from classical logistic map to be used in secure communication. Compared with classical logistic map, VLM has large parameter space without *windows* and can be implemented at low hardware cost. Referring to statistical testing suites SP800-22 and TestU01, VLM with the proposed scrambling method can significantly improve the output complexity as compared with other logistic-map based generators and piecewise linear chaotic map. Experiments show that the throughput of a 32-bit VLM is up to 3,200 Mbps in $0.18\mu m$ process. Furthermore, a chaotic crypto scheme, Multi-VLM (MVLM), constructed by four 32-bit VLMs can generate an output sequence with a minimal length equal to $2^{128} - 1$ by a 128-bit external key.

*Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan. Email: chensl@cs.nthu.edu.tw

†Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan. Email: tingting@cs.nthu.edu.tw

‡Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan. Email: smchang@math.nctu.edu.tw

§Department of Applied Mathematics, National Chiao Tung University, and Division of Mathematics, National Center for Theoretical Science, National Tsing Hua University, Hsinchu 300, Taiwan. Email: wwlin@math.nctu.edu.tw

Keywords: Chaotic Secure System, Digital Communication, Variational Logistic Map.

1 Introduction

The chaotic orbit generated by a non-linear system is irregular, aperiodic, unpredictable and has sensitive dependence on initial conditions. These characteristics coincide with the confusion and diffusion properties in cryptography. Therefore, in recent years, the chaotic system has been studied for security system in both analog and digital forms [Addabbo *et al.*, 2007; Álvarez & Li, 2006; Cermák, 1996; Chen *et al.*, 2008; Frey, 1993; Götz *et al.*, 1997; Heidari-Bateni & McGillem, 1994; Jakimoski & Kocarev, 2001; Juang *et al.*, 2003; Juang *et al.*, 2000; Li *et al.*, 2006; Li *et al.*, 2001; Matthews, 1989; Wang *et al.*, 2006].

Because of non-linear property and easy implementation, logistic map, defined as $x_{i+1} = \gamma x_i(1 - x_i)$ for $x_i \in (0, 1)$, serves as popular map to generate chaotic sequence for crypto system [Cermák, 1996; Jakimoski & Kocarev, 2001; Li *et al.*, 2006]. Although the trajectory of a logistic map is complex, a crypto system using digital logistic map directly has two problems to solve. First, the output cycle length is short. Second, the parameter space of the map is restricted.

The short output cycle leads to non-uniform output distribution, which is easy to be analyzed and attacked by enumerating all states of output. In [Wheeler, 1989], authors suggest that digital chaotic system implemented with more digits can solve the problem of short output cycle length. In [Li *et al.*, 2001; Wang *et al.*, 2006], authors use coupled map to construct multi-dimensional system to increase the complexity of chaotic dynamics. In [Li *et al.*, 2006], authors propose a timing-based method to reseed the map to increase the output cycle length. Using higher precision and coupled map can increase the output cycle length and complexity, but can not increase the number of usable parameters.

The chaotic behavior of a logistic map is dependent on the parameter, γ. 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. [Strogatz, 1994, 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. Previous systems using logistic maps work only when parameter γ is equal or close to 4 [Li *et al.*, 2006]. This constraint makes the key space of a security system smaller than applications require.

In [Chen *et al.*, 2008], Chen et al. proposed a modified logistic map to extend the parameter space for $\gamma > 4$ and a coupled hyper-chaotic system to generate more complex output sequence. Although Chen's system has large parameter space, it needs up to three multiply operations to compute the next states. 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 paper, 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 [Kocarev *et al.*, 2006] 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 [Rukhin *et al.*, 2001] and the most stringent testing by TestU01 [L'Ecuyer & Simard, 2007] 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 Chen's system [Chen *et al.*, 2008].

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 is a safe cryptology in secure communication.

The rest of the paper is organized as follows. In Section 2, the Variational Logistic Map (VLM) is presented. In Section 3, we propose a scrambling method to scramble the output and parameter of VLM. In Section 4, architecture of MVLM will be shown. In Section 5, cryptanalysis will demonstrate that our system is suitable in security applications. In Section 6, we present hardware implementation of MVLM system. Finally concluding remarks are given in Section 7.

2 Variational Logistic Map (VLM)

A classical logistic map is defined by

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

It is well known that Eq. (1) forms a chaotic map for $3.57 < \gamma \leq 4$. 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 communication. Two facts show that parameters are not equally strong. The first one is *windows* in parameter space. In [Barreto *et al.*, 1997], 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 Eq. (1) and to preserve the advantage

of fully distribution in (0,1) for $\gamma = 4$, in [Chang *et al.*, 2009] and [Chen *et al.*, 2008], the authors proposed a Modified Logistic Map (MLM) to extend the parameter space to $\gamma > 4$. MLM is fully distributed in (0,1) and has no *window* when $\gamma > 4$. MLM is given by

$$x_{i+1} = \begin{cases} \gamma x_i(1-x_i) \pmod 1, & x_i \in I_{\text{ext}}, \\ \frac{\gamma x_i(1-x_i) \pmod 1}{\frac{\gamma}{4} \pmod 1}, & x_i \in I_{\text{int}}, \end{cases} \tag{2}$$

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

Although MLM extends the parameter space and presents chaos when $\gamma > 4$, to compute the next state of MLM, it needs up to three multiplications while only two multiplications are used in 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 communication, we will propose a Variational Logistic Map (VLM) which is based on MLM in digitalized implementation. When compared to MLM, VLM also extends 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,

$$L_p(\alpha, \gamma, x) = \{\alpha \left[(\alpha\gamma x) \pmod 1\right](1-x)\} \pmod 1 \tag{3}$$

which is equivalent to

$$L_p(\alpha, \gamma, x) = \left[\alpha \left(\alpha\gamma x - [\alpha\gamma x]_{\text{g}}\right)(1-x)\right] \pmod 1. \tag{4}$$

In fact, Eq. (4) can be rewritten by

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

where

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

and

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

with $\alpha, \gamma > 0$, 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 Eq. (2), the interval, I_{int}, of function f is equal to

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

which means there is no x in I_{int}. Therefore, when $\hat{\gamma}$ is a multiple of four, function f is a subset of MLM and also a chaotic map. Hence, L_p is constructed by a MLM, 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 Fig. 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 Fig. 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 $c = (\alpha ab) (\text{mod } 1)$ 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 $c = (\alpha ab) (\text{mod } 1)$, 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 $(\alpha ab) (\text{mod } 1)$ 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 Fig. 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, 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 system.

Fig. 1 is near here.

Fig. 2 is near here.

In Eq. (3), to make the output of map uniformly distributed, α should be close to half length of x. Since x is in q-bit precision, α is set for $2^{\lceil \frac{q}{2} \rceil}$, where $\lceil x \rceil$ rounds the element of x to the nearest integer greater than or equal to x. Here, our digitalized VLM in q-bit precision is shown in Eq. (5). Before the presentation of VLM, we define a binary floor function, $\lfloor x \rfloor_q$, which preserves the most significant q bits of x and sets other bits to be zero. The VLM is defined as

$$VLM(\gamma, x) = \lfloor [\alpha \lfloor (\alpha\gamma x) (\text{mod } 1) \rfloor_q (1 - x)] (\text{mod } 1) \rfloor_q, \tag{5}$$

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.

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 Eq. (4) is smaller than 4. However, as studied in [Chang *et al.*, 2009], $\alpha^2\gamma$ is required to be a multiple of 4 for Eq. (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 Fig. 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.

Fig. 3 is near here.

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 Eq. (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* [Kocarev *et al.*, 2006], the *bifurcation graph*, *output spectrum analysis*, and *output cycle length* are conducted.

1). The discrete Lyapunov Exponent and bifurcation graph.

Previous work [Kocarev *et al.*, 2006] was proposed to use numerical experiments to verify chaotic property of digitalized chaotic system by the Discrete Lyapunov Exponent (DLE). Let m_i be the subset of trajectory of a digitalized map F in length M and $d(m_i, m_j)$ be the distance between m_i and m_j. In [Kocarev *et al.*, 2006], the basic expression of DLE is defined as

$$\lambda_F = \frac{1}{M} \sum_{i=0}^{M-1} \ln \frac{d(F(m_{i+1}), F(m_i))}{d(m_{i+1}, m_i)}.$$

Map F is 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 $M = 10000$.

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 Fig. 4. 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 [Kocarev *et al.*, 2006].

Fig. 4 is near here.

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 Fig. 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 chaotic map.

2). The output spectrum analysis.

To analyze the auto-correlation and spectrum of output sequences generated by Eq. (6), we randomly select the values of γ and x_0. Fig. 7 shows the results when $\gamma = 0.609375$ and $x_0 = 0.21875$. First, in Fig. 7(a) we plot 10,000 output data. The result shows that the output sequence is visualized randomly. In Fig. 7(b), the spectrum analysis by FFT signifies that the output sequence is a chaotic sequence [Parker & Chua, 1989]. In

Fig. 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

$$x_{i+1} := VLM(\gamma, x_i), \quad i = 0, 1, \ldots \tag{6}$$

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 Eq. (6) and from interval $3.57 < \gamma \leq 4$ for a classical logistic map.

In Fig. 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.

Fig. 5 is near here.

Fig. 6 is near here.

Fig. 7 is near here.

As to hardware cost, the modular arithmetic and truncation operation in Eq. (5) 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 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 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 Fig. 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. To increase the output cycle length and complexity, in this section, we will introduce our scrambling method and in the next section, coupled multiple system.

A scrambling method is useful and widely used in digital chaotic system. In [Cermák, 1996], scrambling is applied not only to the output but also the parameter to increase the output length. In [Li *et al.*, 2001], linear feed back shift registers (LFSR) which has uniformly distributed output is used as a noise to scramble the output of chaotic system. These methods show that scrambling method can improve the uniformity of output distribution and increase cycle length.

The scrambling strategy for VLM is shown in Fig. 8. A q-bit LFSR, L_x is used to scramble x_i.

Fig. 8 is near here.

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

$$n_i = L_x(n_{i-1}), \quad i = 0, 1, \ldots. \tag{7}$$

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 Eq. (7) the sequence generated by VLM after scrambling is defined as follows.

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

By the proposed scrambling method, a deterministic bound of cycle length is calculated as follows. In [Sang *et al.*, 1998], the low bound of cycle length of a scrambling system is given by

$$\Delta \cdot (2^m - 1),$$

where Δ and m are the scrambling period and the length of LFSR used to scramble the system, respectively.

The main idea to scramble a chaotic system is to scramble the trajectory before it enters a loop. Hence, the output cycle length is increased. For example, Li's [Li *et al.*, 2006] 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 m are equal to 1 and q, respectively. Then, for a q-bit VLM, the low bound of cycle length will be $2^q - 1$. 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 Fig. 9, when the scrambling period is decreased, the probabilities of 1 in outputs are close to 0.5.

Fig. 9 is near here.

We will verify the statistical property of VLM with scrambling method by statistical test suite in Section 5.4. 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.

4 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.

4.1 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

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

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

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

Finally, the output sequence of MVLM is generated by an output function T. Output function will be discussed in Section 4.3. The output sequence Seq is given by

$$Seq_j = T(\bar{x}_m^{(j+1)}), \qquad j = 0, 1, \ldots. \tag{8}$$

For example, a MVLM constructed by 4 VLMs is shown in Fig. 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 generating Seq.

Fig. 10 is near here.

4.2 Key Initialization

In Section 2, 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 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 Fig. 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 Fig. 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

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

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

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

Fig. 11 is near here.

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

$$n^{(0)}[k] = KEY.$$

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

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

where $i = 1$ to 4.

The architecture of cascaded VLMs we use in the first stage is shown in Fig. 12. Let one output be generated in one cycle with initial values described in Fig. 11. The 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.

Fig. 12 is near here.

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 Fig. 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

$$n_{reg}[k-128] = x_4^{(k)}[1], \quad 129 \le k \le 256. \tag{9}$$

Fig. 13 is near here.

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.

4.3 Output Function

The $\bar{x}_m^{(j)}$ generated by MVLM is a good random source for security applications. 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.

5 Cryptanalysis of MVLM

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

5.1 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 4.2, 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 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}.

5.2 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 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 4, the cycle length of the MVLM coupled by four 32-bit VLMs will be at least $2^{128} - 1$.

5.3 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 4, 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 Fig. 14, it shows that the cross-correlations between two sequences generated by two KEYs are very weak.

Fig. 14 is near here.

5.4 Statistical Analysis

The randomness of the our system is tested by two test suites, SP800-22 developed by NIST [Rukhin *et al.*, 2001] and TestU01 proposed by L'Ecuyer [L'Ecuyer & Simard, 2007]. 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.

The parameters for SP800-22 tests include $\alpha = 0.01$, $T = 120$, and others as shown in Table 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.

Table 1 is near here.

As to tests by TestU01, three test suites, *SmallCrush*, *Crush*, and *BigCrush* are applied. Recommended by TestU01, *SmallCrush* is used first to take a fast test to check the basic requirement of randomness and followed by *Crush*, and *BigCrush* which is the most stringent statistical testing suite in TestU01. 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. *Crush* needs 2^{35} output sequences of tested system to perform 144 statistical tests, whereas *BigCrush* needs 2^{38} output sequences to perform 160 statistical tests. 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. A 32-bit VLM with/without scrambling function is tested by SP800-22. The initial values of γ is $0.05079f23$ 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 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.

Table 2 is near here.

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. 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.

Table 3 is near here.

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 [Li *et al.*, 2006] 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 Chen's system [Chen *et al.*, 2008] which is a hyper-chaotic system based on modified logistic map to extend the parameter space and output complexity. The third system is Addabbo's system [Addabbo *et al.*, 2007] 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 *Chen's* system which generates 24-bit output per cycle. All systems are operated in 32-bit precision or the closet precision reported by the literatures. In Table 4, test results for scrambled *VLM* are compared with those for *Classical Logistic Map*, *Li's* [Li *et al.*, 2006] system, *Chen's* [Chen *et al.*, 2008] system and *Addabbo's* [Addabbo *et al.*, 2007] 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 described in Section 3, *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 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, *Chen's* hyper-chaotic system 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.

Table 4 is near here.

4). Scrambling function for different systems.

The total numbers of states in each testing system are different. For example, a 32-bit

VLM with scrambling function has 64-bit (32+32-bit LFSRs) registers and Li's has 42-bit (32+10-bit counter) registers. In order to provide each system has comparable number of registers, the scrambling function is applied to each system. In Table 5, testing results show that *classical logistic map* and *Li's* system have a lot of statistical weak points even when the output and parameter are scrambled. Moreover, *Li's* system is worse than its non-scrambling version. Similar to *VLM*, *Addabbo's* system can improve the quality of randomness by scrambling. The table also shows that scrambled *VLM* performs slightly better than scrambled *Addabbo's* system in terms of failure count.

The proposed scrambling function is to scramble the output as well as the input of the scrambled system. When inputs are uniformly distributed in (0,1), outputs of VLM and Addabbo's system tend to uniformly distribute in (0,1). Hence, the scrambling function we proposed is suitable for VLM and Addabbo's system to increase the output complexity and keep output sequence uniformly distributed. However, the output value distribution of classical logistic map and *Li's* system tend to be non-uniformly distributed. When the scrambling function is used, it results in worse statistical properties.

Table 5 is near here.

5.5 Reconstruction complexity

Since Adddabbo's system shows the best statistical property among all previous work, we will compare VLM and Addabbo's [Addabbo *et al.*, 2007] 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 communication 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 understood by autocorrelation analysis. 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 Fig. 15(a), Addabbo's system has relative high correlations between sequential states when $-25 < Lag < 25$. On the contrary, in Fig. 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. The correlations can be reduced by avoiding using sequential states (skipping several states). Then, the system will suffer from short the length of output cycle.

Fig. 15 is near here.

6 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 .18μm technology library. Area and timing information is obtained in gate-level netlist. Fig. 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 6. 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 7. 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 [Li *et al.*, 2006], 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 proposed by *Chen* [Chen *et al.*, 2008], *VLM* has smaller area cost and higher throughput because one multiplier is removed as described in Section 2. 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, Fig. 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 8 shows the synthesized results including control circuit for $m = 1$ to 4. The area and throughput of *Chen's* system [Chen *et al.*, 2008] coupled by two 32-bit modified-logistic maps are also reported. When compared to *Chen's* system, 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. Moverover, 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 oper-

ating frequency, VLM achieves 3,200 Mbps throughput, which is the best in all systems.

Fig. 16 is near here.

Table 7 is near here.

Fig. 17 is near here.

Table 8 is near here.

7 Conclusion

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.

Acknowledgment

This research is supported in part by National Science Council, National Center for Theoretical Sciences, and the Center of Mathematical Modeling and Scientific Computing (National Chiao Tung University) in Taiwan.

References

[1] Addabbo, T., Alioto, M., Fort, A., Pasini, A., Rocchi, S. & Vignoli, V. [2007] "A class of maximum-period nonlinear congruential generators derived from the Rényi chaotic map," *IEEE Trans. Circuits Syst. I, Regular Papaers* **54**, 816–828.

[2] Álvarez, G. & Li, S. [2006] "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation and Chaos* **16**, 2129–2151.

[3] Barreto, E., Hunt, B. R., Grebogi, C. & Yorke, J. A. [1997] "From high dimensional chaos to stable periodic orbits: the structure of parameter space," *Phys. Rev. Lett.* **78**, 4561–4564.

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

[5] Chang, S. M., Li, M. C. & Lin, W. W. [2009] "Asymptotic synchronization of modified logistic hyper-chaotic systems and its applications," *Nonlinear Analysis: Real World Applications*, **10**, 869–880.

[6] Chen, S. L., Chang, S. M., Hwang, T. T. & Lin, W. W. [2008] "Digital secure-communication using robust hyper-chaotic systems," *Int. J. Bifurc. Chaos* **18**, 3325–3339.

[7] Frey, D. R. [1993] "Chaotic digital encoding: an approach to secure communication," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process* **40**, 660–666.

[8] Götz, M., Kelber, K. & Schwarz, W. [1997] "Discrete-time chaotic encryption systems part I: statistical design approach," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* **44**, 963–970.

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

[10] Jakimoski, G. & Kocarev, L. [2001] "Chaos and cryptography: Block encryption ciphers based on chaotic maps," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* **48**, 163–169.

[11] Juang, C., Hwang, S. T., Liu, C. Y., Wang, W., Hwang, T. M., Juang, J. & Lin, W. W. [2003] "Subcarrier multiplexing by chaotic multi-tone modulation," *IEEE J. Quantum Electronics* **39**, 1321–1326.

[12] Juang, C., Hwang, T. M., Juang, J. & Lin, W. W. [2000] "A synchronization scheme using self-pulsating laser diodes in optical chaotic communication," *IEEE J. Quantum Electron.* **36**, 300–304.

[13] Kocarev, L., Szczepanski, J., Amigo, J. M & Tomovski, I. [2006] "Discrete chaos–I: Theory," *IEEE Trans. Circuits Syst. I, Regular Papaers* **53**, 1300–1309.

[14] L'Ecuyer, P. & Simard, R. [2007] "Testu01: A C library for empirical testing of random number generators," *ACM Trans. Mathem. Softw.* **33**, 1–40.

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

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

[17] Matthews, R. [1989] "On the derivation of a chaotic encryption algorithm," *CRYPTOLOGIA* **13**, 29–42.

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

[19] Parker, T. S. & Chua, L. O. [1989] *Practical Numerical Algorithms for Chaotic Systems* (Springer-Verlag).

[20] Sang, T., Wang, R. & Yan, Y. [1998] "Perturbance-based algorithm to expand cycle length of chaotic key stream," *Electrno. Lett.* **34**, 873–874.

[21] Strogatz, S. H. [1994] *Nonlinear Dynamics and Chaos* (Springer-Verlag).

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

[23] Wheeler, D. D. [1989] "Problems with chaotic cryptosystems," *CRYPTOLOGIA* **13**, 243–250.

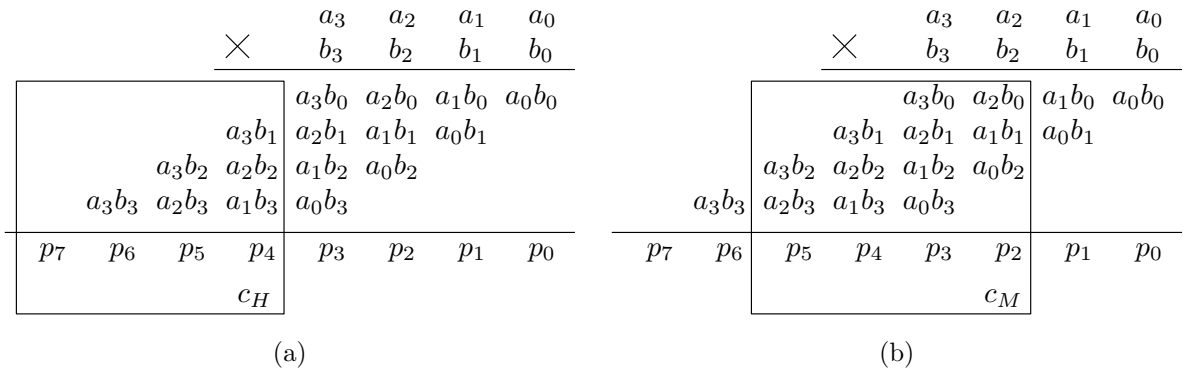
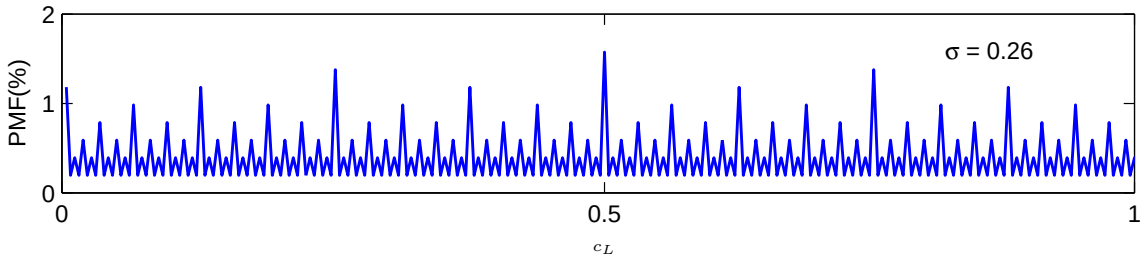
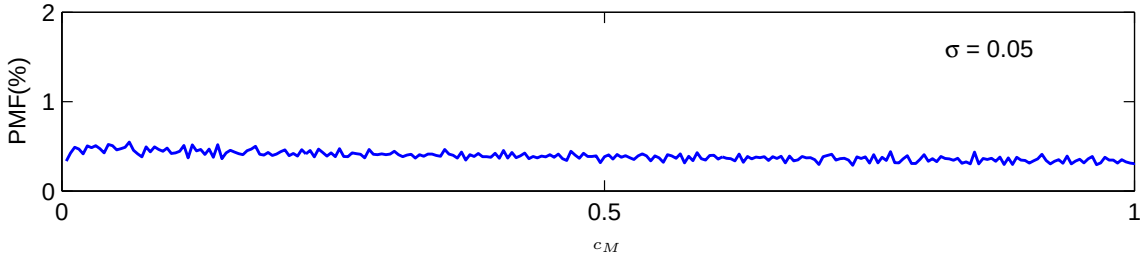


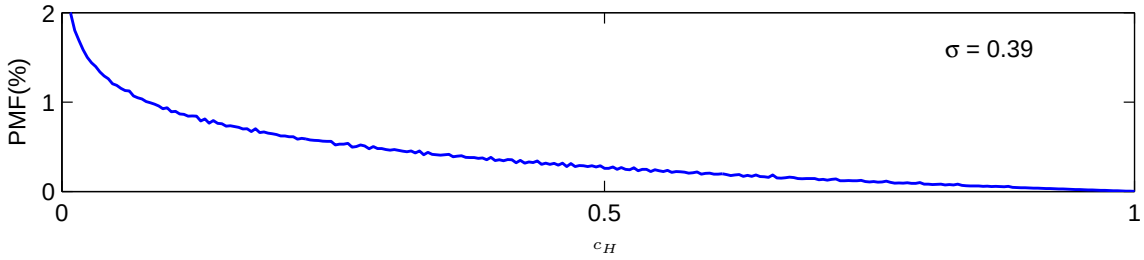
Fig. 1: (a) The least significant 4 bits truncated. (b) Preserving middle significant bits in truncation operation.



(a)



(b)



(c)

Fig. 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) $c_H = 0.p_1p_2\cdots p_8$.

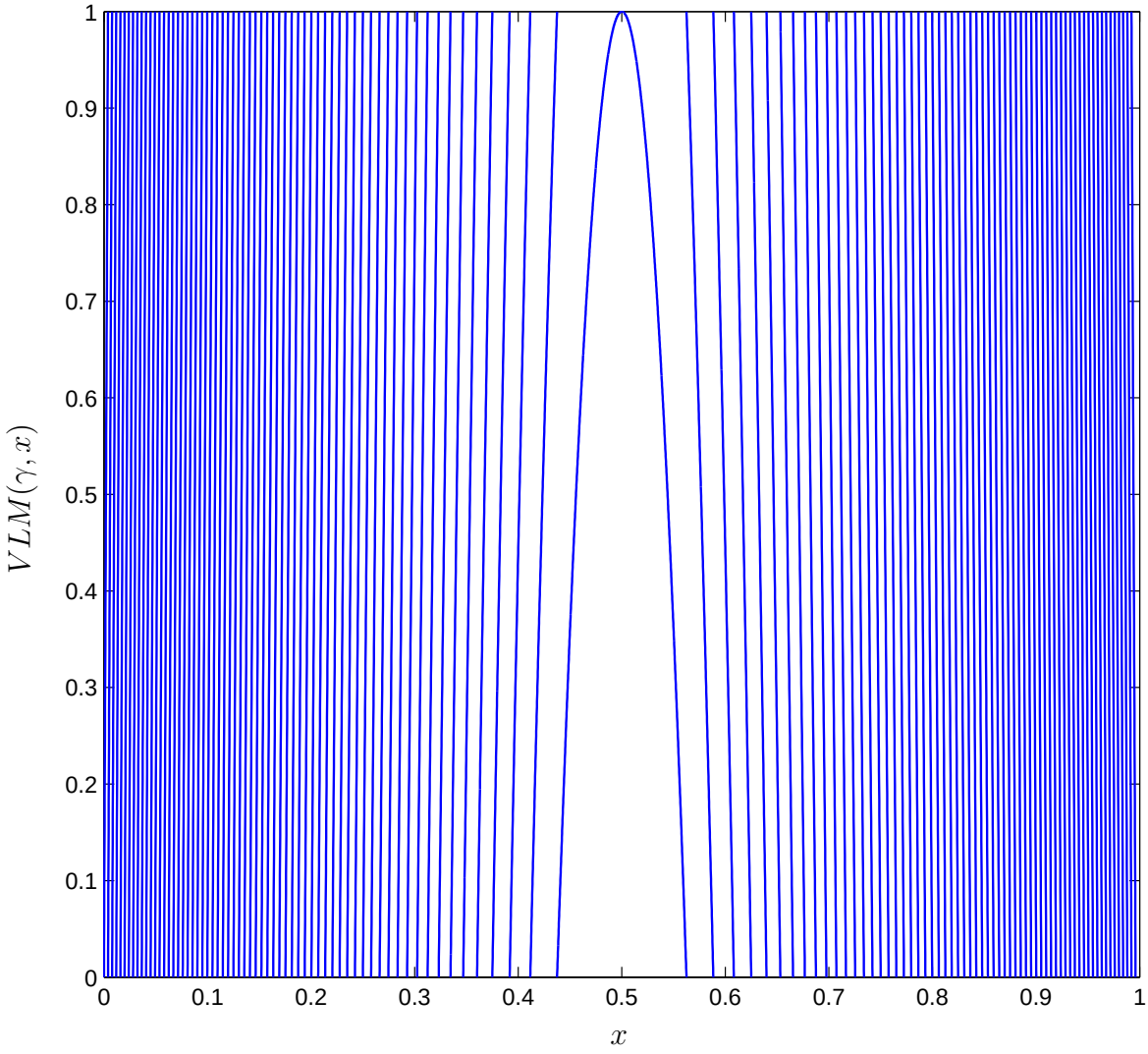


Fig. 3: $VLM(\gamma, x)$ with $\gamma = 2^{-16}$.

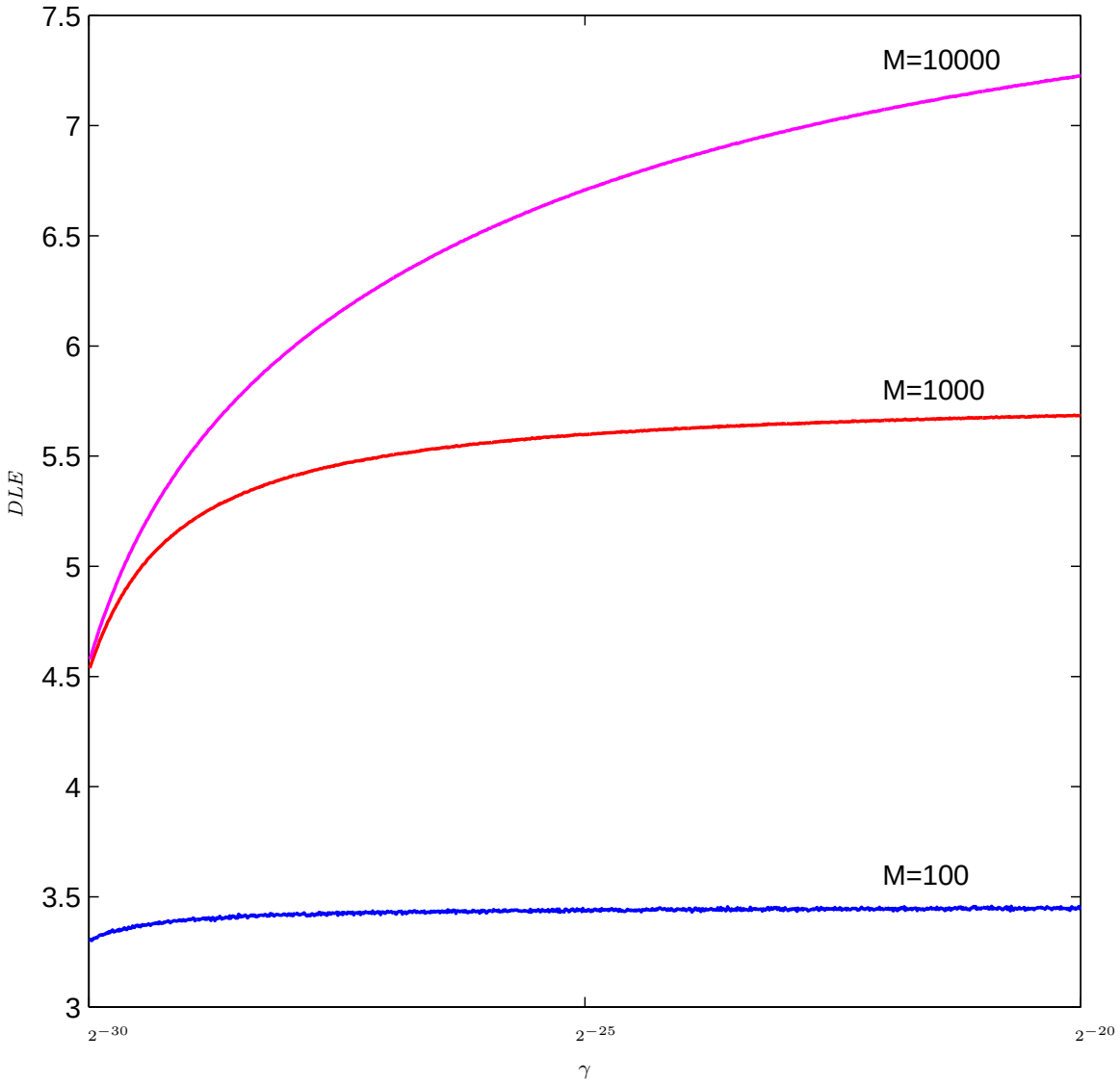


Fig. 4: DLEs for 32-bit VLM when $2^{-30} \leq \gamma \leq 2^{-20}$, where $M = 100$, 1000, and 10000.

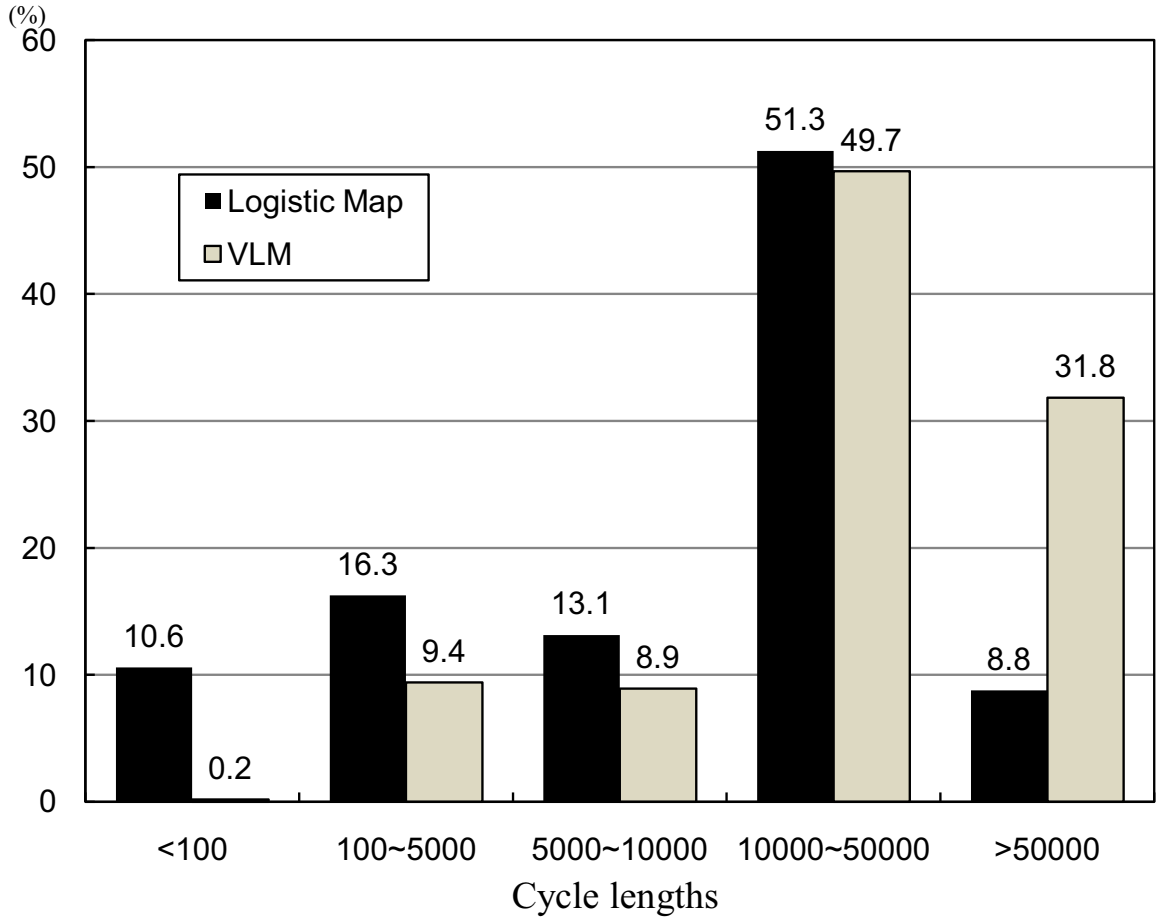


Fig. 5: The histogram of cycle length for VLM and classical logistic map.

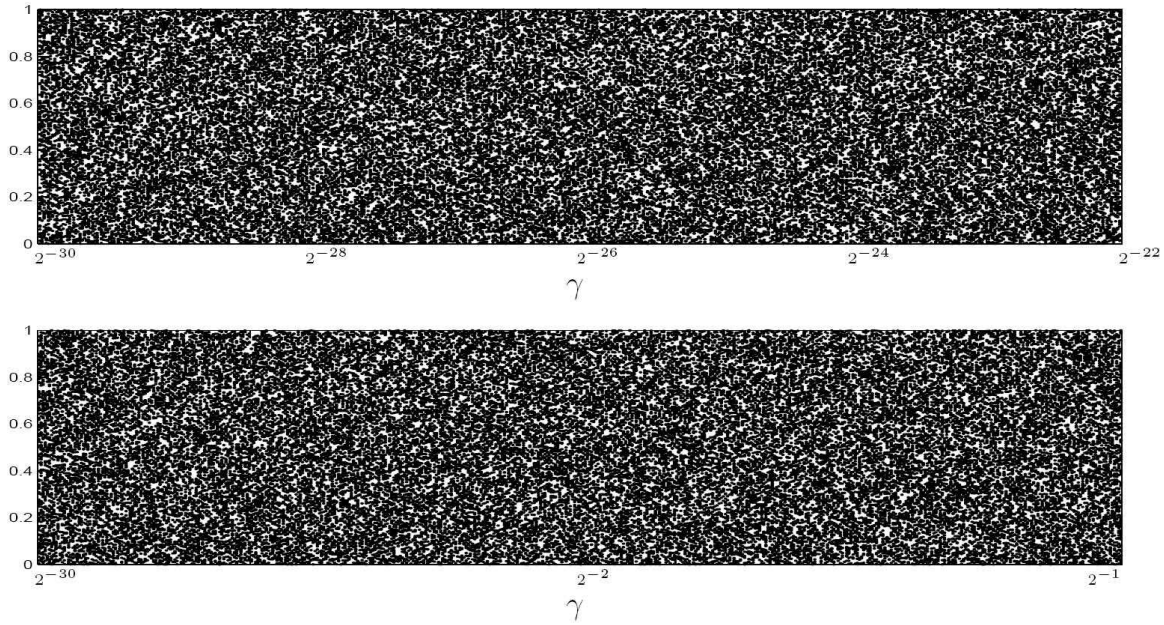


Fig. 6: The bifurcation diagram of VLM for $2^{-30} \leq \gamma \leq 2^{-1}$.

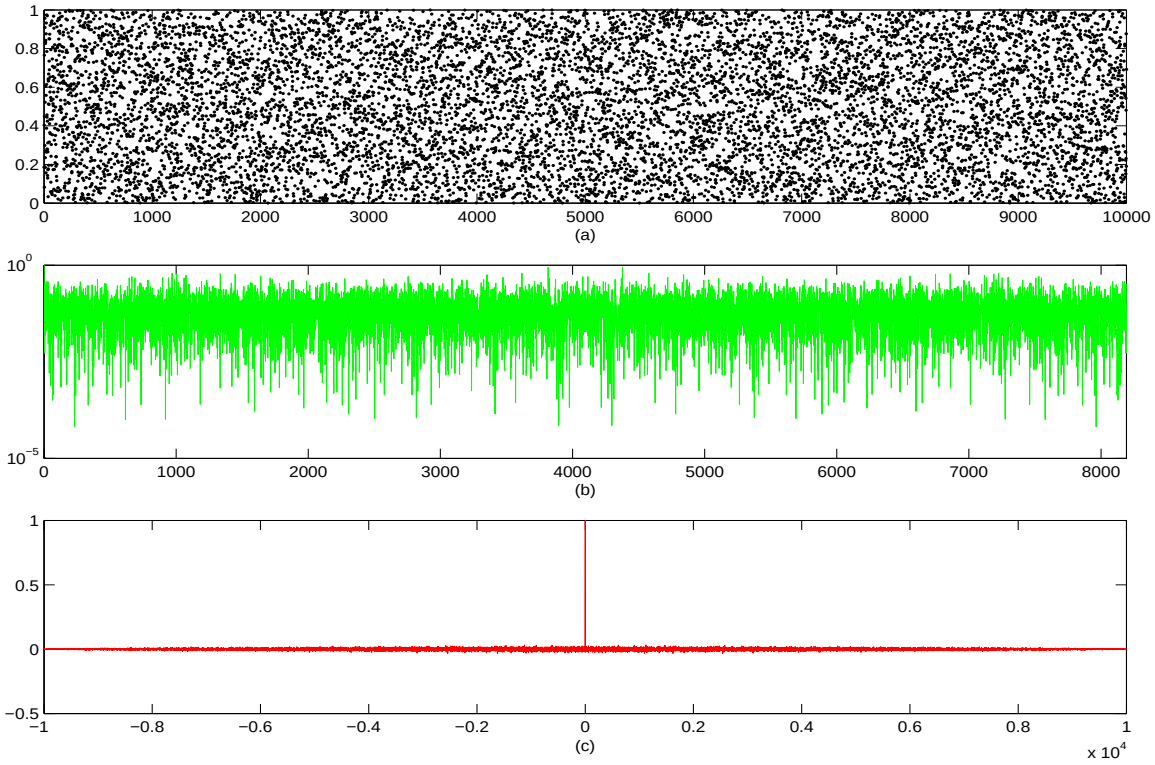


Fig. 7: A trajectory generated by VLM with $\gamma = 0.609375$ and $x_0 = 0.21875$: (a) output value plotting, (b) spectrum analysis, and (c) auto-correlation.

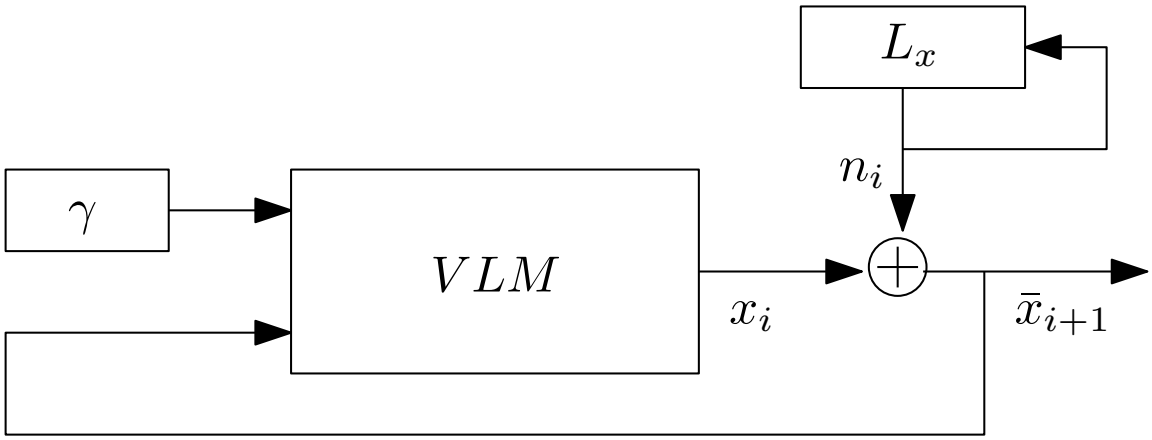


Fig. 8: The scrambling strategy for VLM.

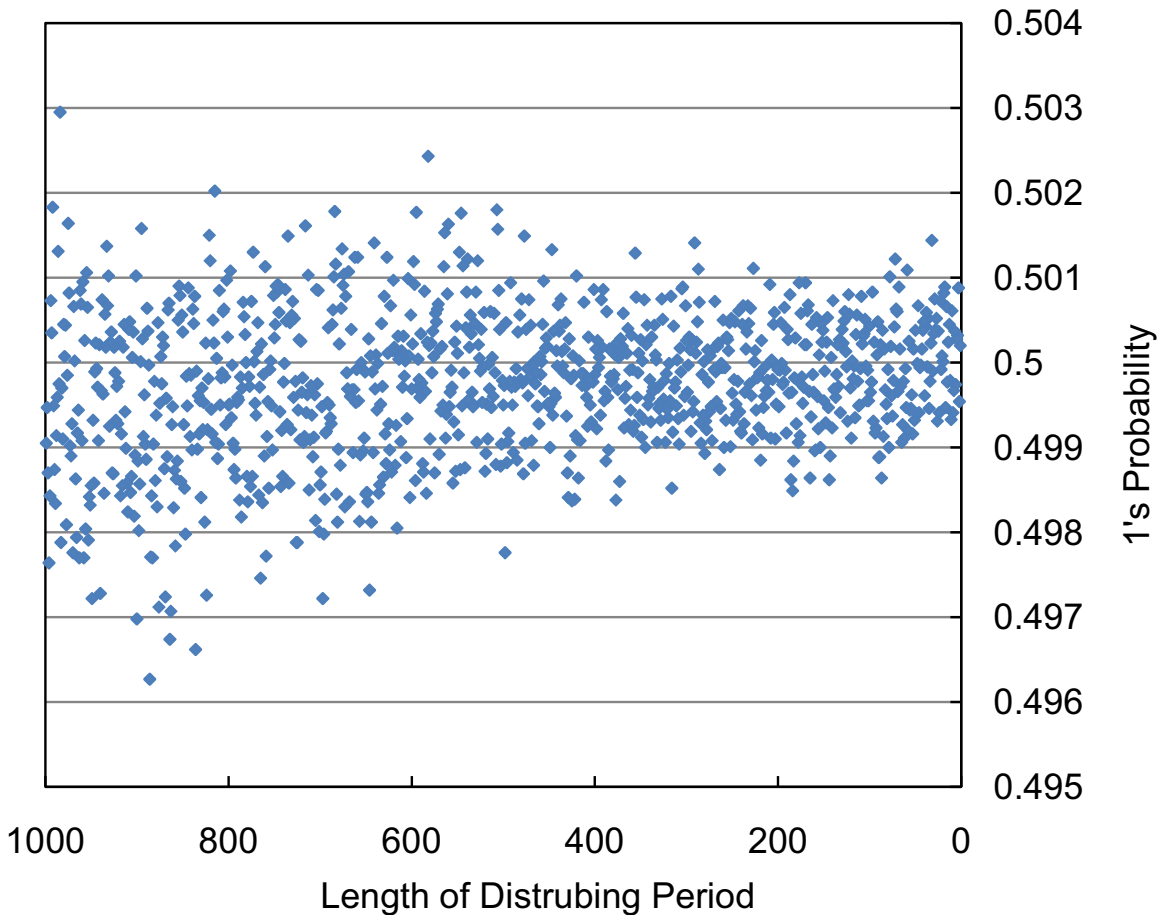


Fig. 9: The 1's probability when scrambling the sequence with different period.

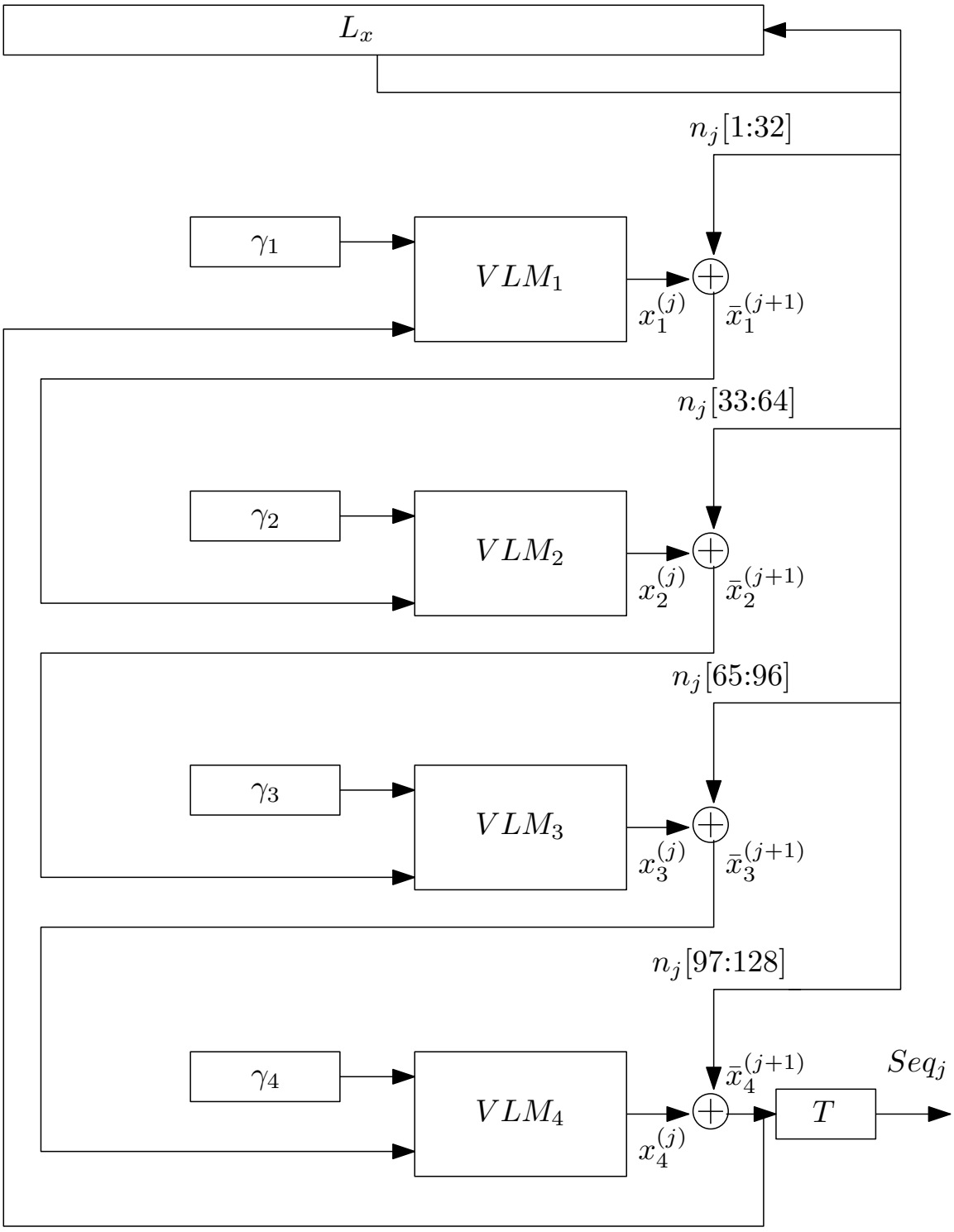


Fig. 10: The top view of a MVLM coupled by 4 VLMs.

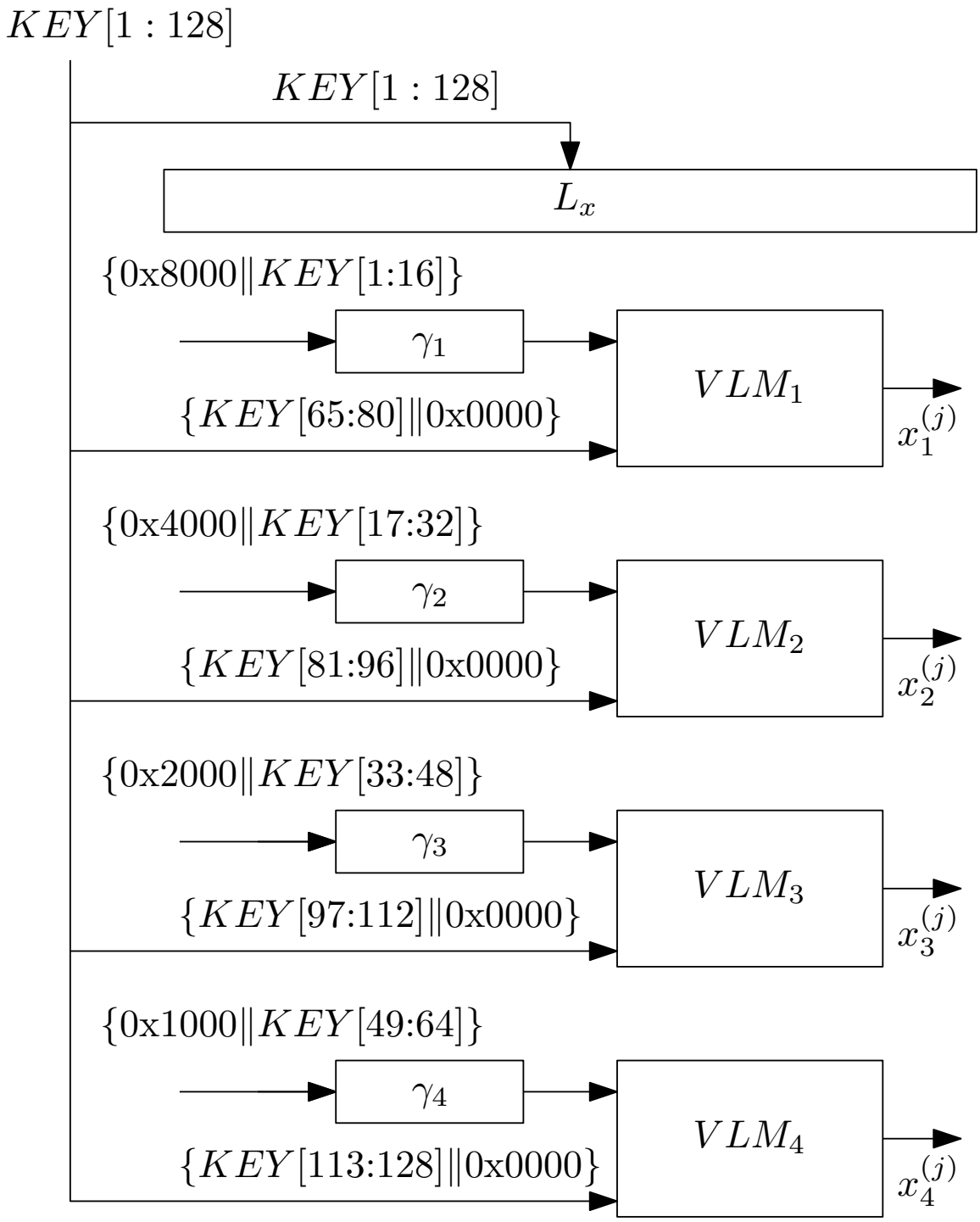


Fig. 11: The initial values to VLM_i from KEY.

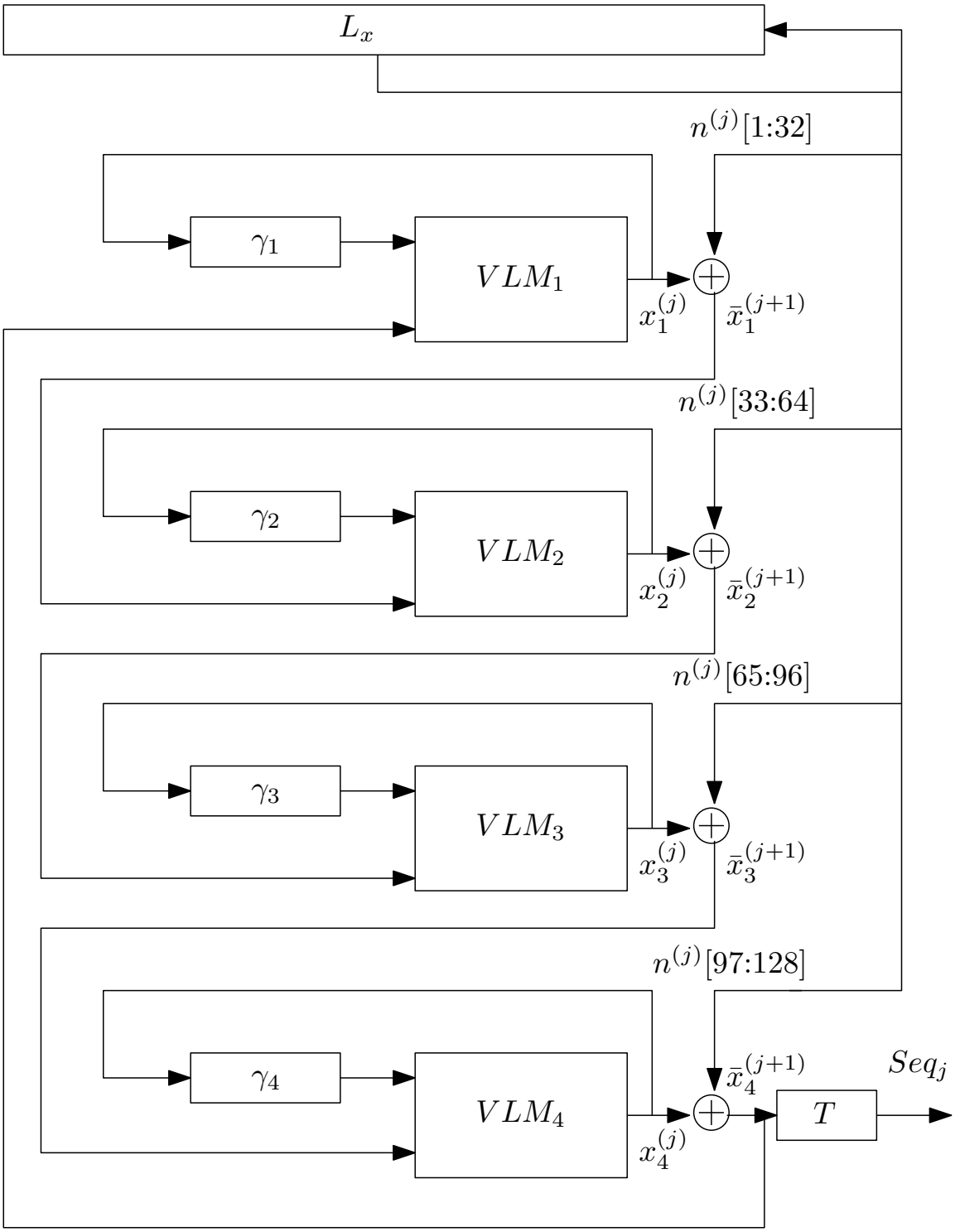


Fig. 12: In the first step of key initialization, γ_i will be shifted to right one bit per cycle and the most significant bit of γ_i will be replaced by $x_i^{(j)}[0]$.

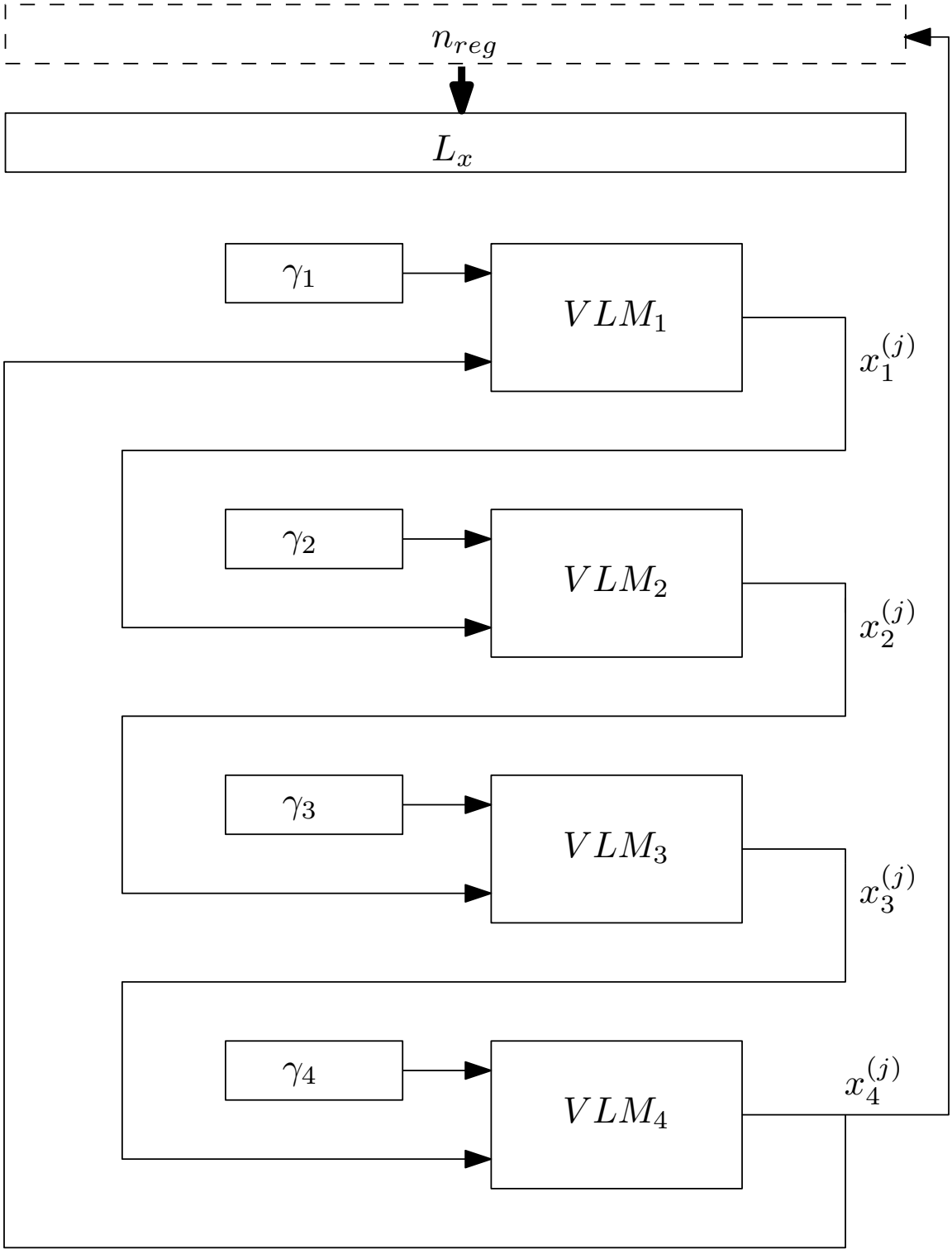


Fig. 13: Using cascaded VLMs to generate $n^{(0)}$ in the second step of key initialization.

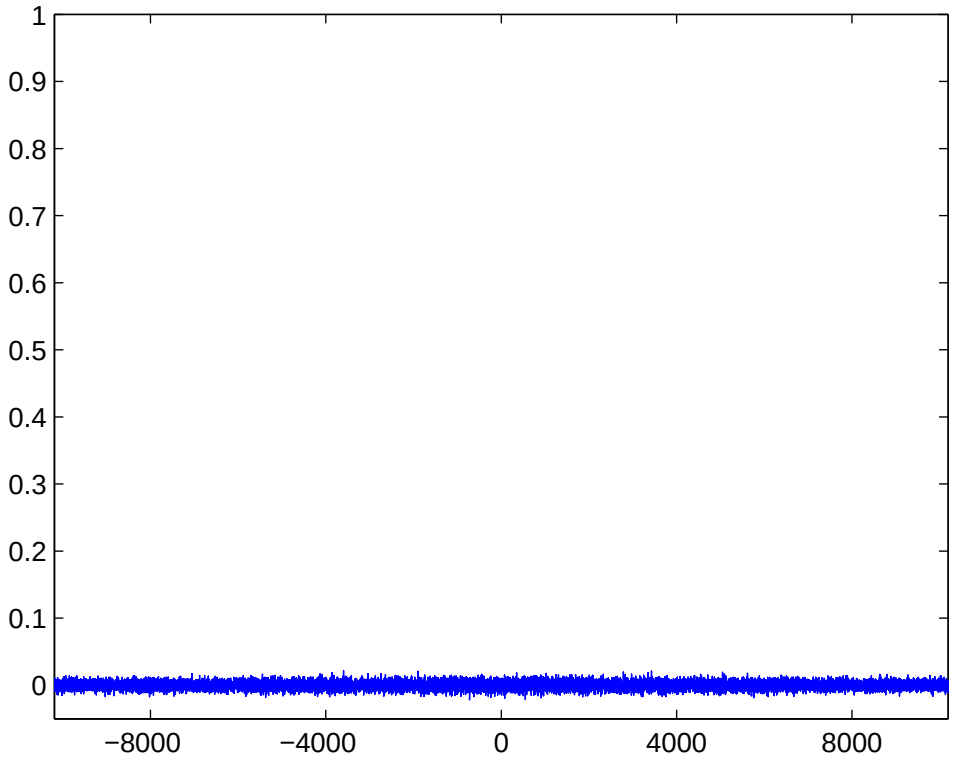
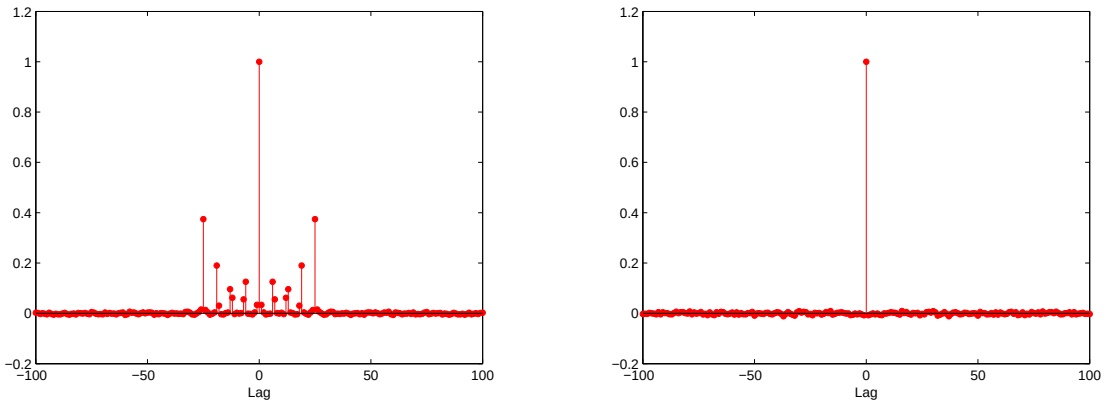


Fig. 14: The cross-correlations between $KEY = 0$ and $KEY = 1$.



(a) (b)

Fig. 15: Autocorleation functions for (a) a 31-bit Addabbo's system, and (b) a 32-bit VLM.

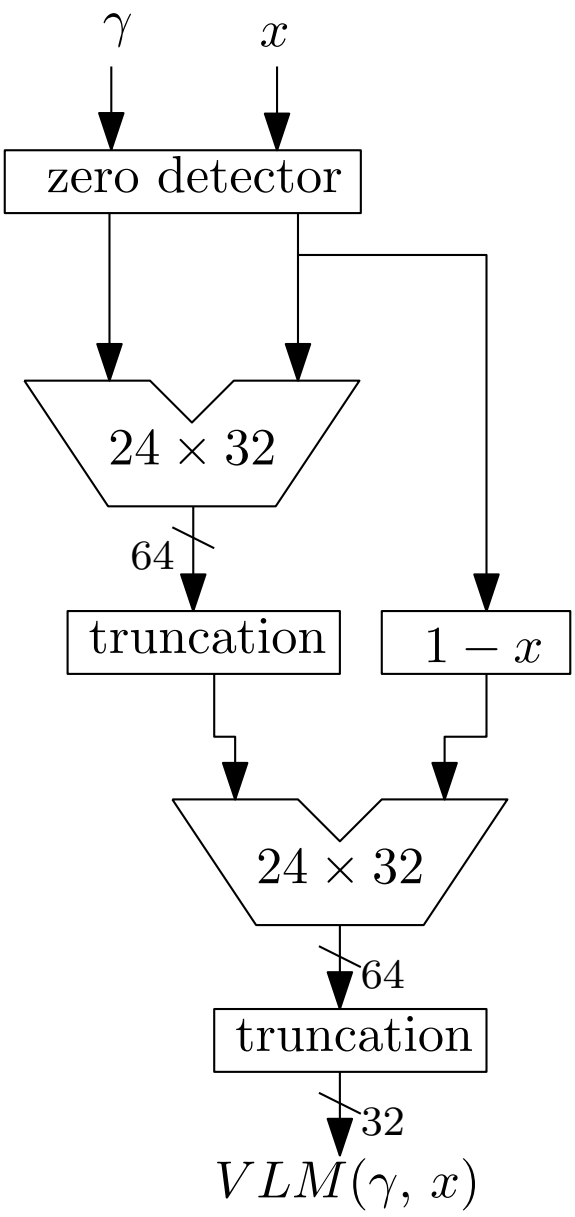


Fig. 16: The architecture of the VLM.

Table 1: Parameters in 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

Table 2: The statistical test results of the VLM with/without scrambling function by SP800-22.

	VLM		scrambled VLM	
Tests	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.

Table 3: The statistical test results of a scrambled VLM in different precisions by SP800-22.

	16-bit		20-bit		24-bit	
Tests	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.

Table 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 [Li *et al.*, 2006]	32	1	1	15	144	156
Addabbo's [Addabbo *et al.*, 2007]	31	1	2	14	122	141
Addabbo's [Addabbo *et al.*, 2007]*	32	1	0	0	3	15
scrambled VLM	32	32	0	0	1	5
Chen's [Chen *et al.*, 2008]†	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 modified logistic maps.

Table 5: Failure counts in statistical tests with scrambling function.

System	Num. of Reg.	SP800-22 (15)	*Small-Crush* (15)	*Crush* (144)	*Big-Crush* (160)
scrambled	32+32	0	0	1	5
classical logistic Map	32+32	8	14	124	142
Li's [Li *et al.*, 2006]	42+32	12	23	123	143
Addabbo's [Addabbo *et al.*, 2007]	31+31	0	1	14	18

Table 6: The components for data-path in VLM and other systems.

	VLM	classical logistic map	Li's [Li *et al.*, 2006]	Addabbo's [Addabbo *et al.*, 2007]	modified logistic map*
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

*A single modified-logistic map proposed by Chen [Chen *et al.*, 2008].

Table 7: The synthesis result for VLM and other systems.

	VLM	classical logistic map	Li [Li *et al.*, 2006]
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

Table 8: The synthesized result of MVLM, and Chen's system for $m = 1$ to 4

Number of VLMs(m)	1	2	3	4	Chen's*
Techonlogy(μm)	.18	.18	.18	.18	.13
Area(#gate-count)	15732	31655	46910	62223	∼57000
Clock Frequency(Mhz)	100	100	100	100	110
Bits/Cycle	32	32	32	32	24
Bits/Seconds(Mbps)	3200	3200	3200	3200	2840
Area Ratio	1	2.01	2.98	3.96	∼3.62

* A system coupled by two 32-bit modified-logistic maps [Chen *et al.*, 2008].

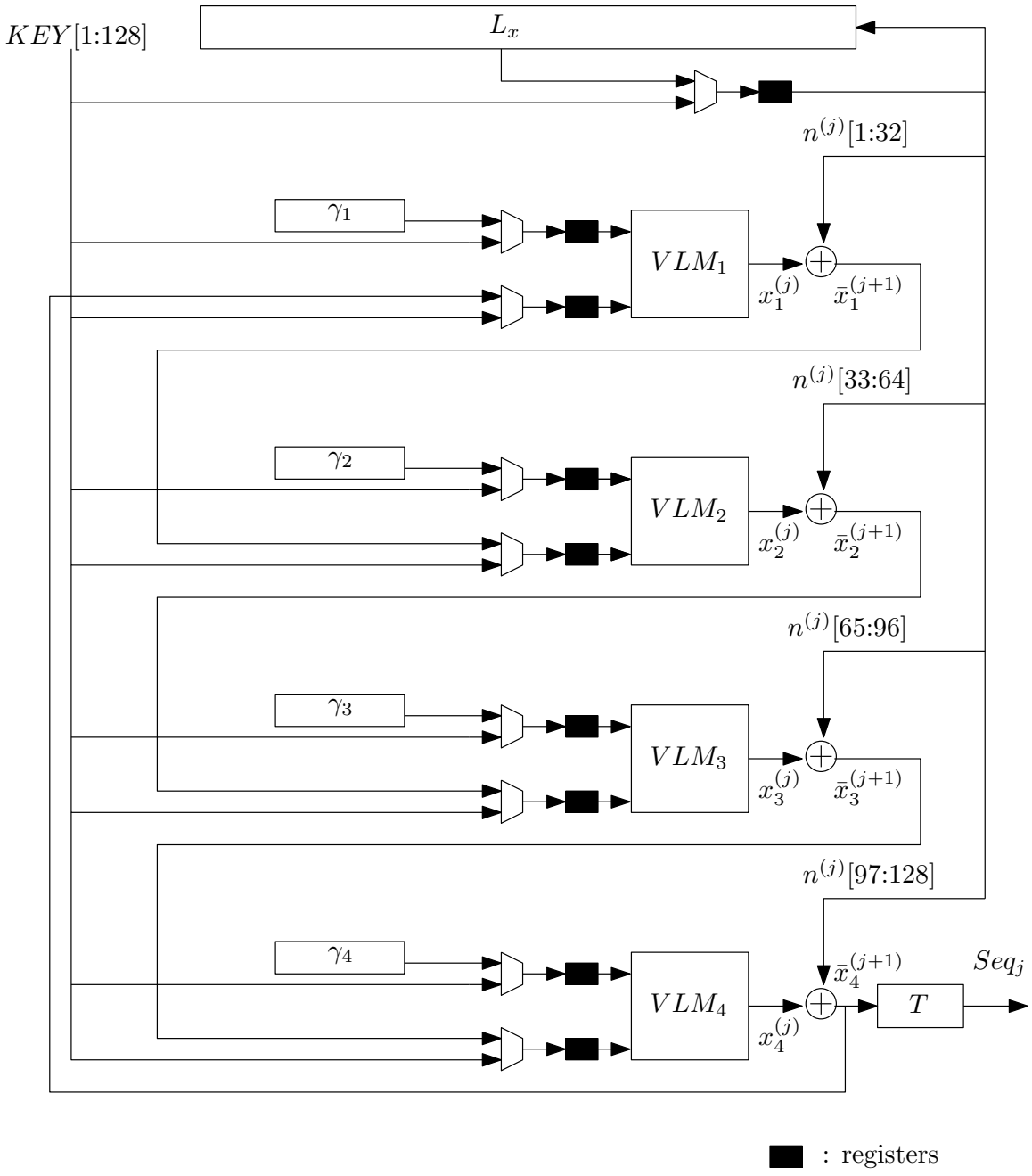


Fig. 17: The data-path architecture of the MVLM with $m = 4$.