# Faster cellular automata cryptosystems with neighbor sequences

Kittiphop Phalakarn and Athasit Surarerks

Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand

**Abstract.** The encryption processes and cryptosystems are very important. We use them to protect our private information over the Internet. Cellular automata are ones of the computational models that can also be used in cryptosystems. The advantage of the cellular automata is their abilities to work in parallel, and thus can reduce the encryption time. Some applications require the encryption time to be small, so this paper aims to reduce the encryption time of the cellular automata cryptosystems. We propose a new technique to permit the cryptosystems to get the avalanche effect faster. This avalanche effect is one of the desired properties for cryptosystems. In the proposed technique, the new type of neighbor is defined, a sequence of neighbor tuples. We apply our technique to Seredynski and Bouvry's work, and the results show that the number of iterations can be reduced up to three times. This makes our cellular automata cryptosystems run faster. The relationship between the size of the neighbor and the size of the cellular automata, and the effect of neighbor sequences to the hardware implementations are left for further studies.

**Keywords:** cryptography, cellular automata, neighbor sequence.

## 1. Introduction

In everyday life, we send and receive a lot of private information, such as, passwords for user identification, financial transactions, and health information. When exchanging these data in a computer network, attackers may steal such information, and can cause damage to the properties and life. To prevent this, many cryptosystems have been proposed.

The cryptosystems were developed since past until now. It started with the simple method like alphabet substitution as shown in ROT13 [1]. Today, there are numbers of standard cryptosystems, for example, 3DES, AES, RSA, elliptic curve cryptography, etc. Each method has its own advantages and disadvantages.

Some applications require less time for encryption and decryption processes. Many studies tried to reduce this processing time, and they found that cellular automata performed well in this purpose. The parallel operations of the cellular automata make them a good choice for cryptosystems. Many papers have proposed ways to use cellular automata in both symmetric key (secret key) [2, 3, 4] and asymmetric key (public key) cryptosystems [5].

In this paper, we try to reduce the processing time of the cellular automata cryptosystems. We define a new type of neighbor of cellular automata, "neighbor sequence", and try to choose the appropriate neighbor sequences that can reduce the processing time.

This paper is organized as follows. In Section 2, we give some background knowledge on related topics, cryptosystems and cellular automata. Section 3 gives some information about previous work, Seredynski and Bouvry's work. Our main contribution is presented in Section 4. We define a new type of neighbor of cellular automata, a neighbor sequence. In section 5, we apply the neighbor sequence technique to the cryptosystems to show that we can reduce the encryption time of the cryptosystems by using the appropriate neighbor sequence. Finally, Section 6 concludes the paper.

## 2. Background knowledge

In this section, we present some background knowledge concerning cryptosystems, the avalanche effect, and cellular automata.

### 2.1. Cryptosystems and avalanche effect

Cryptosystem is a process that secures the communication, or keeps the communication secret. There are two types of cryptosystems, one with symmetric secret key, and the other with asymmetric public key [1]. Some examples of standard symmetric key cryptosystems used today are 3DES and AES [6], and the example of standard asymmetric key cryptosystem is RSA [6].

A desired cryptosystem usually satisfies many good properties. For example, it must have large number of possible keys and possible cipher texts. The time and memory used in encryption and decryption processes should be small. The encryption and decryption processes should be easy if the key is known, but it should be very difficult without the key. The cipher text must have enough randomness.

One important property of a cryptosystem is an avalanche effect. This means that if you encrypt messages $M_1$ and $M_2$ that have only small differences with the same key $K$, the result cipher texts $M_1'$ and $M_2'$ should have much difference. In the same way, if you encrypt a message $M$ with two different keys $K_1$ and $K_2$, which have only small differences, the results should be much different. If the results look nearly the same, the cryptosystem will be easy to be predicted. See Example 1 below.

**Example 1** (Avalanche effect). Give two messages "ABCD" and "ABCE". If "ABCD" is encrypted to "PQRS", and "ABCE" is encrypted to "PXRS", you may easily imply that "D" is encrypted to "Q", and "E" is encrypted to "X". So, the better cryptosystem should encrypt "ABCE" to "ZKMJ" which has much difference to "PQRS".

### 2.2. Cellular automata

Cellular automata (CA) are ones of the computational models that can represent how computers work. CA have many cells that work identically and synchronously. (Some papers may use CA that consisted of different types of cell [7].) The cells change their states depend on the states of their neighbors.

From the past until now, CA have been used to model the behavior of many things, including the biological spreading in biology, and the diffusion of gases in chemistry and physics. In the field of computer science, the behavior of CA is studied, for example, how can we model the CA to represent the languages, and how can the CA accept the words in the languages. CA are Turing complete [8]. CA have also been used in many computer applications, such as, cryptography and data compression [2, 3, 5, 9].

In this subsection, we will give the formal definition of CA. The definitions of CA in each study are very similar; they have only small differences in detail. Here, we refer to the definition in [10], because it covers the scope of the definitions from most studies.

**Definition 2** (Cellular automata). Cellular automata $C$ can be defined using 4-tuple, $C = (d, S, N, f)$. The meaning of each member is described as follows.

Let $d \in Z^+$ be the dimension of the CA. Each cell in the CA can be referenced with a vector in $Z^d$.

Let $S$ be a finite set of states of the CA. We define function $\alpha : Z^d \rightarrow S$ that tells the state of the given cell.

Let $N = [n_1, n_2, ..., n_k]$ be a neighbor tuple. Each $n_i \in N$ is a member of $Z^d$. This tuple is used to describe which cells will be considered in the state transition phase (calculated relatively to the given cell). The neighbor tuple may include (0, ..., 0) which means the state of the cell itself can be considered for state transition.

Let $f : S^{|N|} \rightarrow S$ be the local state transition function. All cells will change their states synchronously. The new state of the cell at position $x \in Z^d$ is equal to $f(\alpha(x+n_1), \alpha(x+n_2), ..., \alpha(x+n_{|N|}))$ for each $n_i \in N$.

## 3. Previous work: Seredynski and Bouvry's

In this section, we refer to Seredynski and Bouvry's work in [11] where one dimension (1D) CA was used. The study used 32 cells and 64 cells cyclic 1D CA with radius 2 and radius 3 neighbor tuples,

[-2, -1, 0, 1, 2], and [-3, -2, -1, 0, 1, 2, 3]. Each cell can contain only value 0 or 1, or $S = \{0, 1\}$. The CA is then used as a part of the cryptosystems in [3].

The study in [11] conducted some experiments to show how fast the avalanche effect can propagate to the whole data. In each experiment (for 32 and 64 cells, radius 2 and radius 3 neighbor tuples), 10,000 random second order rules for CA were used, and 10,000 random bit-strings were generated as plain texts. To see the avalanche effect, one random bit from each plain text was changed, and then the new plain text was encrypted. The study compared the results from both set of plain texts, and plotted the average percentage of the number of different cells from 10,000 experiments. The paper concluded that the lowest numbers of iterations used to get more than 49% of cells changed as in Table 1. These numbers of iterations were then used in [3] as the numbers of iterations in their cryptosystems.

Table 1: Lowest number of iterations to get more than 49% of cells changed. [11]

| | Radius 2 | Radius 3 |
|---|---|---|
| 32 cells CA | 19 | 8 |
| 64 cells CA | 38 | 17 |

## 4. New type of neighbor: the neighbor sequence

From Table 1, the CA cryptosystems in Section 3 have to take some number of steps to make the avalanche effect to affect the whole data. We hypothesized that this is because the cells only use the states of their adjacent cells to determine their next states. So we have an idea on the far-apart non-adjacent neighbor.

We define a new type of neighbor tuple, and make a hypothesis that this new neighbor tuple can propagate avalanche effect in fewer steps. We call the new type of neighbor tuple the "extended radius $r$ neighbor tuple". The cardinality of this neighbor tuple is same as the original radius $r$ neighborhood, but the distance between the cell and its neighbors are increased.

**Definition 3** (Extended radius $r$ neighbor tuple). The $i$-th extended radius $r$ neighbor tuple is defined as $R_r(i) = [-ir, -i(r\text{-}1), \ldots, -i, 0, i, \ldots, i(r\text{-}1), ir]$.

In addition, using the same neighbor tuple every time will only propagate the avalanche effect to the same cells repeatedly. We think that changing the distance between neighbors during the encryption process can help propagating avalanche effect. Thus, we also define a new type of cellular automata that can have more than one neighbor tuples. We call it the "cellular automata with a sequence of neighbor tuples". An example of how this type of CA works is given as follows.

**Definition 4** (Cellular automata with a sequence of neighbor tuples). A "cellular automata with a sequence of neighbor tuples" is defined as $C = (d, S, \langle N_0, \ldots, N_{k-1}\rangle, f)$ where $d$, $S$, and $f$ are the same as defined in Definition 2. $\langle N_0, \ldots, N_{k-1}\rangle$ is called a neighbor sequence. Each $N_i$ in the neighbor sequence is the same as the original neighbor tuple, $N$. Suppose that the current step is $t$, in order to determine the next state, the considered neighbor tuple for the next step is $N_{(t \bmod k)}$. In other words, the neighbor tuples used in steps 0, 1, … are $N_0, N_1, \ldots, N_{k-1}, N_0, N_1, \ldots$ and loop in this manner.

We expect that using the neighbor tuple from Definition 3 together with the CA from Definition 4 can improve the avalanche effect propagation. In the next section, we will show experimental results that support our idea.

## 5. Using appropriate neighbor sequences in cryptosystems

We now apply our proposed neighbor sequence technique to the cryptosystems. The original experiments in [11] used the neighbor tuples $R_2(1)$ and $R_3(1)$. We will compare the results from $R_2(1)$ to the CA with neighbor sequences $\langle R_2(3)\rangle$, $\langle R_2(5)\rangle$, $\langle R_2(1), R_2(2)\rangle$, $\langle R_2(1), R_2(3)\rangle$, and $\langle R_2(1), R_2(5)\rangle$. The process for $R_3(1)$ is similar to $R_2(1)$. We conduct 10,000 experiments, same as [11] which were explained in Section 3, on each number of cells and the neighbor sequences, and then plot the average percentage of the number of different cells. Our simulations were done using Python. Fig. 1 is the plot for 32 cells CA with radius 2 neighbor sequences, and Fig. 2 is the plot for 64 cells CA with radius 3 neighbor sequences.

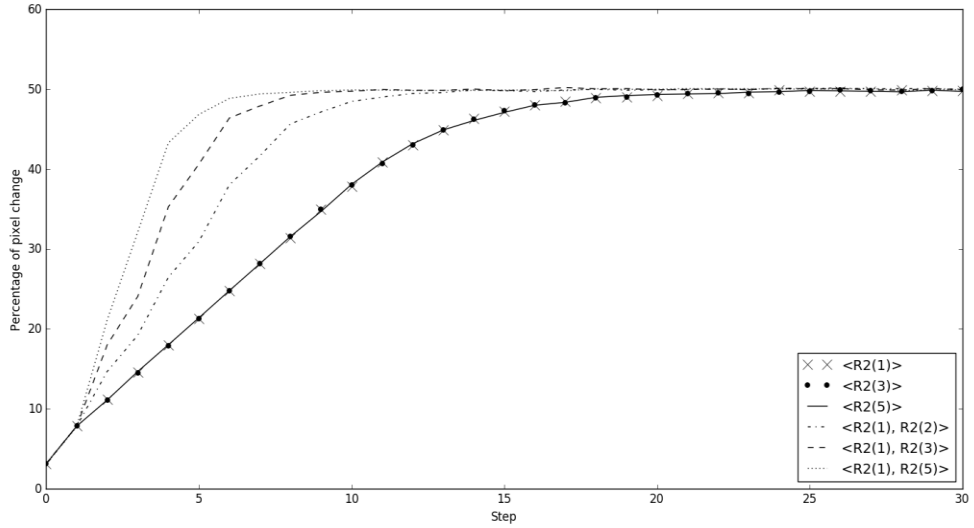


Fig. 1: Average percentage of the number of different cells for 32 cells CA with radius 2 neighbor sequences. The axis of the figure is step-percentage of pixel change.

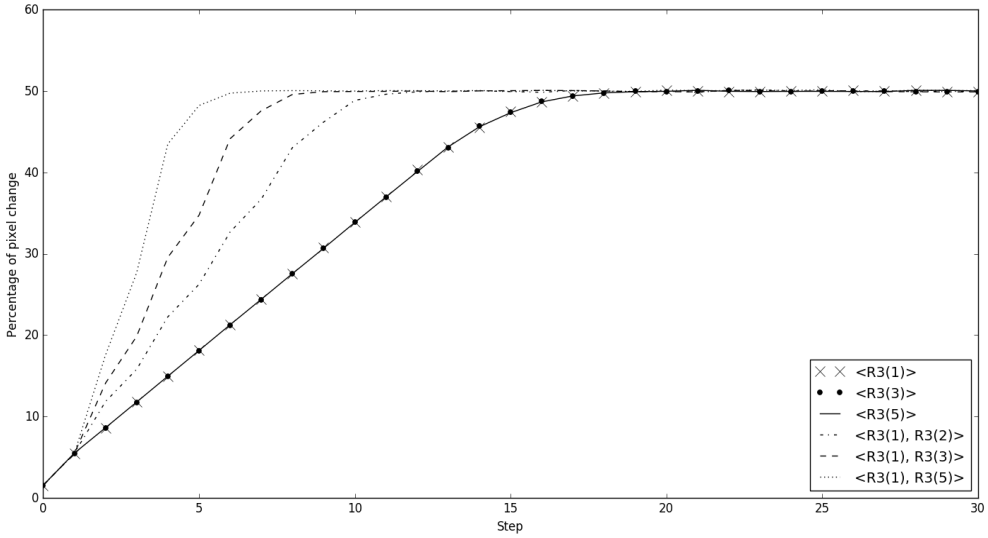


Fig. 2: Average percentage of the number of different cells for 64 cells CA with radius 3 neighbor sequences. The axis of the figure is step-percentage of pixel change.

From Fig. 1 and Fig. 2, the neighbor sequences $\langle R_2(1), R_2(5)\rangle$ and $\langle R_3(1), R_3(5)\rangle$ perform best. From the experiments, we can see that more distance between neighbors results in less steps to propagate the change, but it is not the only factor. Changing the distance between neighbors during the process is important. Here, you have to swap between a small distance and a large distance neighbors to make the change propagation effective. This is because using the same neighbor tuple will propagate the change to the same cells every time, while changing the distance between neighbors back and forth prevents this.

We also conclude our lowest numbers of iterations used to get more than 49% of cells changed as in Table 2. Comparing to Table 1, the numbers in Table 2 are approximately three times smaller. Thus, applying our neighbor sequence technique to CA cryptosystems can reduce the number of iterations.

Table 2: Lowest number of iterations to get more than 49% of cells changed when using neighbor sequence $\langle R_r(1), R_r(5)\rangle$.

| | Radius 2 | Radius 3 |
|---|---|---|
| 32 cells CA | 7 | 4 |
| 64 cells CA | 10 | 6 |

## 6. Conclusion and future works

Our paper proposes a new technique to improve cellular automata cryptosystems. We define a new type of neighbor, *i*-extended radius *r* neighborhood, $R_r$ (*i*). We also combine neighbor tuples into "neighbor sequence", and adjust the definition of the cellular automata to use the sequence of neighbor tuples.

We have done some experiments to show that our neighbor sequence technique can improve the CA cryptosystems. We apply our technique to the cryptosystems in [11], compare the results between different neighbor sequences, and conclude that swapping between the small and large neighbors (the value *i* in $R_r$ (*i*)) in the neighbor sequence improves the propagation of the avalanche effect. As final results, the number of steps used in encryption processes can be reduced up to three times in 1D CA cryptosystems.

For future works, the relationships between the size of the neighbor and the size of the cellular automata need to be investigated. We only know that the bigger neighbors are better, but there may be some issues on the cyclic boundary condition that wrap around the avalanche effect propagation. So, too big neighbors may not be good. The effects of neighbor sequences to the hardware implementations are also left for further studies. We have to trade-off between the cost of hardware connecting big neighbors and the time of the encryption processes.

To sum up, our neighbor sequence technique can be used to improve the avalanche effect propagation of the CA cryptosystems, which results in reduction of the encryption time. The technique can be very useful in cryptosystems with real time applications.

## 7. References


[1] I. V. S. Manoj. Cryptography and steganography. *International Journal of Computer Applications.* 2010, **1** (12): 63–68.

[2] J. Jin. An image encryption based on elementary cellular automata. *Optics and Lasers in Engineering.* 2012, **50** (12): 1836–1843.

[3] M. Seredynski, K. Pienkosz, and P. Bouvry. Reversible cellular automata based encryption. In: *IFIP International Conference on Network and Parallel Computing*. Springer. 2004, pp. 411–418.

[4] L. H. Encinas, A. M. Rey, and A. H. Encinas. Encryption of images with 2-dimensional cellular automata. In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics and 8th International Conference on Information System Analysis and Synthesis (SCI/ISAS 2002). International Institute of Informatics and Systemics (IIIS)*. Orlando: 2002.

[5] J. Kari. *Cryptosystems based on reversible cellular automata*. Manuscript, 1992.

[6] G. Singh. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*. 2013, **67** (19).

[7] A. Jaberi, R. Ayanzadeh, and A. S. Z. Mousavi. Two-layer cellular automata based cryptography. *Trends in applied sciences research*. 2012, **7** (1): 68.

[8] M. Mitchell et al. Computation in cellular automata: A selected review. *Nonstandard Computation*. 1996, 95–140.

[9] O. Lafe. Data compression and encryption using cellular automata transforms. *Engineering Applications of Artificial Intelligence*. 1997, **10** (6): 581–591.

[10] K. Morita. *Reversible cellular automata*. Handbook of Natural Computing. Springer, 2012.

[11] M. Seredynski, and P. Bouvry. Block encryption using reversible cellular automata. In: *International Conference on Cellular Automata*. Springer. 2004, pp. 785–792.