Reversible Cellular Automata Based Encryption

Marcin Seredynski[1,3], Krzysztof Pienkosz[1] and Pascal Bouvry[2]

[1] Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
seredynski@acn.waw.pl, K.Pienkosz@ia.pw.edu.pl
[2] Luxembourg University of Applied Sciences
6, rue Coudenhove Kalergi, L-1359, Luxembourg-Kirchberg, Luxembourg
pascal.bouvry@univ.lu
[3] Polish-Japanese Institute of Information Technology, Research Center
Koszykowa 86, 02-008 Warsaw, Poland

Abstract. In this paper cellular automata (CA) are applied to construct a symmetric-key encryption algorithm. A new block cipher based on one dimensional, uniform and reversible CA is proposed. A class of CA with rules specifically constructed to be reversible is used. The algorithm uses 224 bit key. It is shown that the algorithm satisfies safety criterion called Strict Avalanche Criterion. Due to a huge key space a brut-force attack appears practically impossible.

1 Introduction

The increased use of computers resulted in a strong demand for means to protect information and to provide various security services. Encryption is a primary method of protecting valuable electronic information. It transforms the message (plaintext) into the cipher text. The opposite operation is called decryption. Two forms of encryption are in common use. They are called symmetric-key and public-key encryption [3]. If both sender and receiver use the same key, or it is easy to obtain one from another then the system is referred to as symmetric key. If the sender and receiver each uses different key, and it is computationally infeasible to determine one from another without knowing some additional secret information then the system is referred to as a public-key. There are two classes of symmetric-key encryption algorithms. They are called block and stream ciphers. A block cipher breaks up the message into blocks of fixed length and encrypts one block at a time. A stream cipher encrypts a data stream one bit or one byte at a time. This paper deals with symmetric-key block ciphers.

In this paper we apply CAs to construct a new symmetric-key algorithm. CAs are highly parallel and distributed systems, which are able to perform complex computations. They have been used so far in both symmetric-key and public-key cryptography. CA-based public cipher was proposed by Guan [1]. Stream CA-based encryption algorithm was first proposed by Wolfram [9] and later some other algorithms were developed by Tomassini et al. [6], and recently by Seredynski et al. [4]. Block cipher using both reversible and irreversible rules was proposed by Gutowitz [2].

This paper presents a new encryption algorithm based on a class of reversible CA with rules specially designed to be reversible. It is organized as follows. The coming section defines elementary and reversible CA. Section 3 presents the idea on how particular reversible CA class can be used for encryption. Detailed description of the encryption algorithm based on reversible CAs and its analysis can be found in section 4. Section 5 concludes the paper.

2 Cellular Automata

2.1 Elementary Cellular Automata

One-dimensional CA is an array of cells. Each cell is assigned a value over some state alphabet. CA is defined by four parameters: *size*, *initial state*, *neighborhood*, *rule* and *boundary conditions*. Size defines number of cells. Each cell updates its value synchronously in discrete time steps accordingly to some rule. Such rule is defined over the *neighborhood* which is typically composed of the cell itself and its *r* left and right neighbors:

$$s_i^{t+1} = R(s_{i-r}^t ,..., s_{i-1}^t, s_i^t, s_{i+1}^t,..., s_{i+r}^t), \tag{1}$$

where s_i^t is a value (state) of *i-th* cell in step *t* and *r* is radius of the neighborhood. Example of rule definition for radius 1 neighborhood is shown on Fig.1.

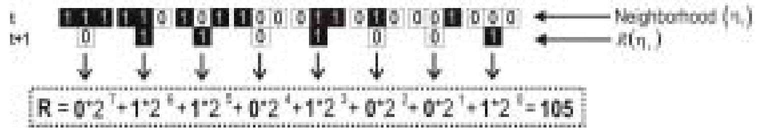

Fig. 1. Elementary rule 105 definition

There are 8 possible neighborhood configurations for radius 1 neighborhood. As shown on Fig. 1 state transition must be defined for each possible case.

When dealing with finite CA, *cyclic boundary conditions* are usually applied which means that CA can be treated as a ring. Changing values of all cells in step *t* is called CA *iteration*. Before the first iteration can take place some initial values must be assigned to all cells. This is called the *initial state* of CA. By updating values in all cells the initial state is transformed into a new configuration. When each cell updates its state according to the same rule, CA is said to be *uniform*. Otherwise such CA is called *non-uniform*. In this paper one-dimensional, uniform CA defined over binary state alphabet with neighborhood size two and three is used.

2.2 Reversible Cellular Automata

When analyzing elementary CA it turns out that only a small number of rules have the property of being reversible. For example, among all 256 radius 1 CA only six are reversible. This is why class of CA with rules specially created to be reversible is considered. Different reversible CA classes are presented in [5]. In this paper we are using reversible CA class presented by Wolfram [8]**.** In this class rule depends not on one but on two steps back:

$$s_i^{t+1} = R(s_{i-r}^t\ ,..., s_{i-1}^t, s_i^t, s_i^{t+1}, s_{i+1}^t, ..., s_{i+r}^t). \tag{2}$$

In the elementary CA value s_i^{t+1} of *i-th* cell in configuration *t+1* depends on the value of the state of itself and *r* of its neighbors in configuration *t.* In this reversible class additional dependency is added. Now, the value of the central cell s_i^{t-1} in step *t-1* is also considered. Such a rule can be simply constructed by taking elementary CA rule and adding dependency on two steps back (*t-1*). Example of such rule definition is shown on the Fig.2.

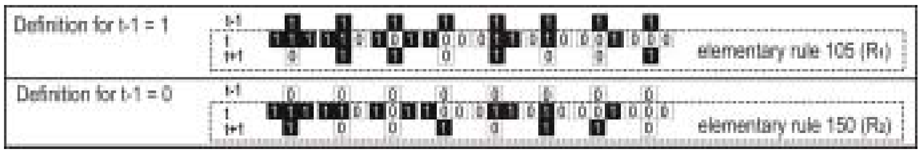

Fig. 2. Reversible rule 105/150 definition

Definition of the rule is now composed of two elementary rules. The first one is defining state transition in case when in step *t-1* central cell was in a state 1, and the second one, when that cell was in the state 0. Fig. 2 gives an example of reversible rule based on two elementary rules: 105 and 150. These two rules are complementary to each other. Knowing one value it is possible to calculate the second one using the following formula:

$$R_2 = 2^d - R_1 - 1\ , \tag{3}$$

where $d = 2^{2*r+1}$. The same rule is used in forward and backward iteration. The total number of radius r rules is 2^d. Since a reversible rule depends now on two steps back, CA initial state must be composed of two successive configurations labeled q_0 and q_1.

3. The idea of using reversible CA class in a block cipher

When using reversible CA described in the previous section, plaintext is encoded as a part of CA initial state - configuration q_1. Configuration q_0 is set up with random data. Encryption is done by forward iteration of the CA by fixed number of steps according to some reversible rule. This rule forms a secret *key*. Encryption algorithm is shown on Fig. 3.

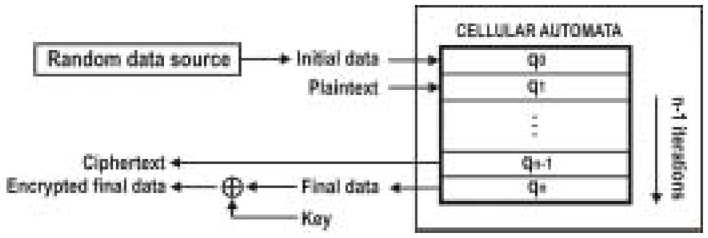

Fig. 3. Encryption using reversible cellular automata

Configuration q_{n-1} is a ciphertext. There are two options concerning configuration q_n (called *final data*) generated by the encryption. The most secure one assumes that this information is kept secret, which means that configuration q_n together with rule forms the *key*. The disadvantage of this option is that the *key* changes with each encryption. This is because the *key* is now a function of the rule, plaintext and random initial data. In the second option, the final configuration q_n is encrypted using *Vernam* cipher [3]. It is done by applying exclusive-OR operation (XOR, $\oplus$) on the final configuration q_n and the *key*. Now encrypted final configuration no longer has to be kept secret, and can be added to the ciphertext. To obtain final data for decryption, XOR operation must be applied to encrypted final configuration and the *key*.

For the decryption, the same operations are applied in reverse order. CA initial state is composed of the *final data* and the ciphertext. CA is then iterated for the same number of steps as used for encryption. The same secret rule taken from the *key* is used.

Good encryption algorithm should satisfy the *Strict Avalanche Criterion* (SAC) [7]. This means that each output bit should change with a probability of one half whenever a single input bit is complemented. In this paper, we are using 32 cells radius 2 CAs and 64 cells radius 3 CAs. In order to satisfy SAC, 32 cells radius 2 CAs should be iterated for at least 22 iterations, while 64 cells radius 3 CAs should be iterated for at least 19 iterations. These results are based on 10000 experiments for each parameters set (CA size/rule size/iteration number). For each experiment randomly generated rule and initial configuration were used.

4 Cipher based on reversible cellular automata

Our cipher is composed of four one-dimensional CA labeled CA32L, CA32R, CA64, and CAS. Both CA32L and CA32R are composed of 32 cells. CA64 is composed of 64 cells and CAS is composed of 16 cells. There are two inputs to the encryption function: plaintext and the key. The plaintext is divided into 64 bit blocks. The key is 224 bits in length (see Sec. 4.2). Automata CA32L, CA32R, CAS use radius 2 reversible rules and CA64 uses radius 3 reversible rule. Encryption process of a single plaintext block consists of 16 rounds (typical number for block ciphers). Each round is composed of 4 operations: iterations in CA64, CA32L, CA32R and *Shift* transformation.

4.1 Details of a single round

Fig. 4 presents details of a single round of encryption.

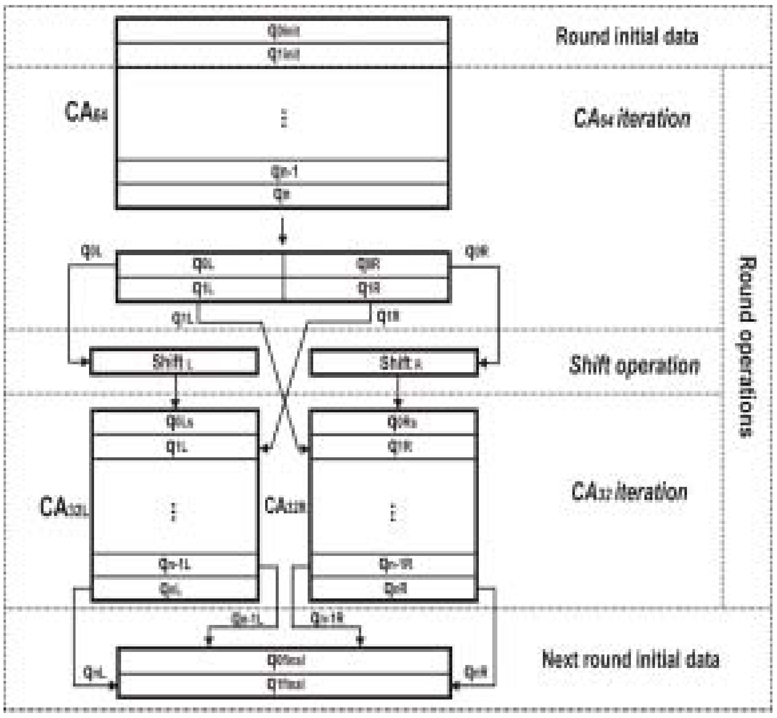

Fig. 4. Single round of the algorithm

Each round starts with two 64-bit values labeled q_{0init} and q_{1init} that form initial state of CA64. In the first round of encryption, q_{1init} is a plaintext to be encrypted. Configuration q_{0init} called *initial data* is described latter. In case of rounds 2-16, both configurations get their values from the result of the preceding round. After being initiated, CA64 is iterated for 19 steps. The result is divided into four 32 bit values labeled q_{0L} , q_{0R} , q_{1L} and q_{1R} . Then configurations q_{0L} and q_{0R} are shifted respectively left and right by *ns* cell positions (described latter). Shifted values together with q_{1L} and q_{1R} form initial state of CA32L, CA32R. Next, both CA are iterated for 22 steps. The result forms two 64 bit values labeled $q_{0\,final}$ and $q_{1\,final}$. This values become initial data for the next round: $q_{0\,final}$ becomes q_{0init} and $q_{1\,final}$ becomes q_{1init} . After the last round $q_{0\,final}$ is a 64 bit ciphertext block.

Successive blocks encryption is shown on the Fig.5.

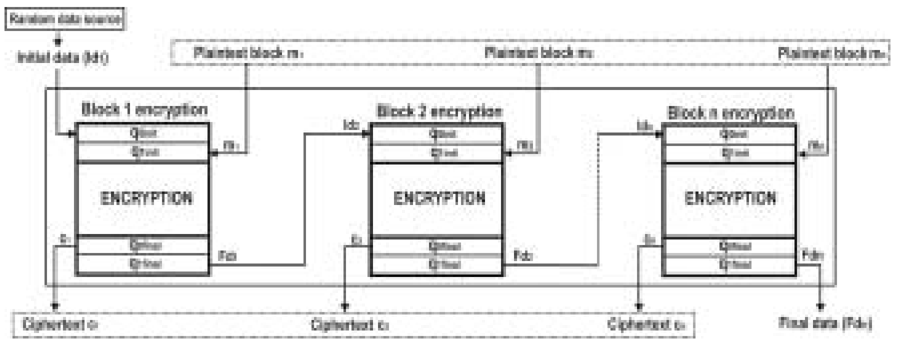

Fig. 5. Successive blocks encryption scheme

For the encryption of the first plaintext block, configuration q_{0init} (*initial data*) is generated randomly. Otherwise, $q_{1\,final}$ from the encryption of the previous block is used as q_{0init} . Configuration $q_{1\,final}$ (called *final data*) obtained from the encryption of the last plaintext block encryption is encrypted (see Sec. 4.2) and then added to the ciphertext.

Values used for *Shift* operation are generated by CAS. Before first block is being encrypted, CAS is initiated with random initial values. Then whenever new *ns* value is needed, CAS is iterated for 5 steps. The consecutive values of the central cell form 5-bit *ns* value. For each round new *ns* is generated. Final configuration of CAS is encrypted (see Sec. 4.2) and added to the ciphertext.

Decryption is using the same operations in reverse order. The only difference is that the selected output from CA32L is shifted right and output from CA32R is shifted left.

4.2 Key structure

Each automata runs with its own reversible rule. The key is composed of 4 rules corresponding to each automaton in the following order: CA32L, CA32R, CA64, CAS. There are three radius 2 rules (CA32L, CA32R, CAS) and one radius 3 rule (CA64). Each radius 2 rule is 32 bits in length while radius 3 rule is 128 bits in length. This makes the key size of 224 bits. *Final data* configuration ($q_{1\,final}$) is encrypted using XOR operation and bits 0-63 of the key and final configuration of CAS is encrypted using the same operation and bits 192-223. The key should be generated using some high quality random number generator.

4.3 Cryptanalysis

There are 2^{224} possible keys, which means that a brute-force attack appears practically impossible.
Another attack consists in assuming to find final states of CA32L, CA32R, CAS. Indeed the knowledge of the two successive configurations of the CA makes finding a rule much easier. For CA32L and CA32R the final state is composed of the last ciphertext block and *final data* configuration. Last ciphertext block is known so we only need to find out two 32 bits *final data* configurations. For CAS final state is composed of two last 16 bit configurations. Knowledge of the final state of CAS enables finding values used in *Shift* transformation. Since final states of CA32L, CA32R and CAS were encrypted using XOR operation and some random data (key) it can only be found by enumeration. There are 2^{32} possible values of *final data* configurations of CA32L, CA32R and final state of CAS. We also need to know the rule used for CA64. There no information on any configuration used in CAS, which means that there is no way of telling which rule was used. Described attack assumes verifying 2^{224} possible combinations of "final states of CA32L/CA32R/CAS / CA64 rule" ($2^{32} * 2^{32} * 2^{32} * 2^{128}$). Complexity of this task is just the same as the one for brute-force attack.

Greater security can be achieved by using larger block size but it reduces encryption/decryption speed.

4.4 Cipher properties

Our reversible CA-based cipher works in a mode that is similar to CBC mode [3] in terms of achieved result. The same plaintext block that appears in the whole plaintext more than once produces different blocks of ciphertext. This is because encryption of each plaintext block starts with some initial data taken from the encryption of the previous block (each time a different value). In DES like ciphers there is still problem with encryption of the same plaintext more than once, or when two encrypted plaintext begin with the same information (in both cases the same key is used). In the first case the same ciphertext will be produced, while in the second case both plaintexts will be encrypted the same way until the first difference is reached. It is possible to overcome this problem by encrypting some random data block first. In the proposed cipher encrypting the same plaintext with the same key will always result in a different ciphertext. This is achieved by using randomly generated data and new initial configuration of automaton CAS for encryption of each plaintext.

5 Conclusions

In this paper we have presented an idea on how a particular class of reversible CAs can be used in cryptography. We have given an example of a new block encryption algorithm based on that idea. One dimensional, radius 2 and radius 3 CAs are used. The algorithm uses 64-bit blocks as an input. The same operations are used for both encryption and decryption. Strict Avalanche Criterion is satisfied. Due to a huge key space a brute-force attack appears practically impossible.

References

1. Guan, P.: Cellular Automaton Public-Key Cryptosystem. Complex Systems 1 (1987) 51-56
2. Gutowitz, H.: Cryptography with Dynamical Systems, manuscript
3. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, CRC Press (1996)
4. Seredynski, F., Bouvry, P., Zomaya, A.Y.: Cellular Programming and Symmetric Key Cryptography Systems. In: E.Cantú-Paz et al. (eds.): Genetic and Evolutionary Computation – GECCO 2003. LNCS 2724. Part II. Springer (2003) 1369-1381
5. Toffoli, T., Margolus, N.: Invertible cellular automata: a review. Physica D 666. North-Holland, Amsterdam (1997)
6. Tomassini, M., Perrenoud, M.: Stream Ciphers with One and Two-Dimensional Cellular Automata. In: M. Schoenauer et al. (eds.): Parallel Problem Solving from Nature – PPSN VI. LNCS 1917. Springer (2000) 722-731
7. Webster, A.F., Tavares, S.E.: On the Design of S-Boxes, Advances in Cryptology : Crypto '85 Proceedings. Springer. LNCS 218. Springer (1985) 523-534
8. Wolfram, S.: A New Kind of Science, Wolfram Media (2002) 435-441
9. Wolfram, S.: Cryptography with Cellular Automata in Advances in Cryptology : Crypto '85Proceedings. LNCS 218. Springer (1985) 429-432