UDC 004.056.55

S.E. Ostapov, Professor, Yu.V. Tanasyuk, Associate Professor (Yuriy Fedkovych Chernivtsi National University, Ukraine),
A.Ye. Perepelitsyn, Senior Lecturer (National Airspace University "KhAI", Ukraine)

Cryptographic Application of Cellular Automata

Cellular automata (CA) are well-known multipurpose computational environment, possessing characteristics of self-replication and parallelism according to a predefined set of rules. The paper is dedicated to investigation of the ability of CA to produce pseudorandom patterns, and its application for the purpose of designing block ciphers, public key generation and hash function development.

Living and working at the age of Internet of Everything stipulates new challenges for making data communication safe and reliable. There is an urgent need for discovering novel cryptographic techniques, which would ensure confidentiality and integrity of information, without degrading connection quality or speed.

Cellular automata, due to their chaotic, complex and unpredictable behavior for some types of transformation rules, exhibit properties that favor their use as a basis for the design of cryptographic primitives [1].

The purpose of the paper is to develop binary stream generators, block ciphers and hash functions on the basis of one- or multi-dimensional CA and to investigate the effectiveness of combining CA rules with binary operations.

In total, there are 256 CA rules divided into several classes. Each class is formed of so-called statistically equivalent rules, i.e. those that can be derived from each other. As it is mentioned in [1] the rule 30 possesses statistical characteristics appropriate for creating qualitative generator of pseudo-random binary sequences. For this purpose, in our research we selected all rules belonging to the same class of the rule 30 (86, 135, 149) and a few others for comparison.

To develop key stream generators on the basis of CA rules we have considered several approaches. In one case cells were combined using simple XOR logical operation through the following formula to define a value of the next bit of the key sequence:

$$M_{\mathrm{out}}=\mathrm{M}[w] \oplus M[v] \oplus M[t] \quad (1)$$

Here *w, v, t* are addresses of the automaton cells. In our research w*, v, t* parameters were used as predefined constant values or could be selected through the formula:

$$\mathrm{w}=\left\{\mathrm{i}+3+\sum_{\mathrm{k}=0}^{\log_2 n-1} c_{(\mathrm{i}+3+\mathrm{k})}\, 2^k\right\} \bmod \mathrm{n}\ , \quad (2)$$

were c is a value of the (i+3+k) cell, n is a total number of cells in the automaton, w is an address of the current cell. The interaction may encompass both closest and long-range neighbors.

Another way to study the suitability of CA for generating high-quality pseudo-random key sequences consisted in designing alternative rules for intercellular interaction. We use the address array $A[i]$ to select the interacting cells. This array contains decimal numbers (cell's addresses) from 1 to n, pseudo-randomly filled and restored into the $B[i]$ buffer array. The interacting cells are selected as follows:

$$C'_{A[i]} = C_{A[a]} \oplus C_{A[i+1]} \oplus C_{A[i]} \tag{3}$$

$$\mathrm{a}= \left(\mathrm{i}+1 + \sum_{\mathrm{k}=0}^{(\log_2 n)-1} c_{\mathrm{i}+1+\mathrm{k}} \cdot 2^{(\log_2 n)-1-k}\right) \bmod n. \tag{4}$$

Here $A[i+1]$ is an address of the first interacting cell. The address of the next one ($A[a]$) is given by (4). The interaction result is contributed as the next key bit. After all the cells have been processed, the address array of $A[i]$ is swapped with the $B[i]$ buffer array:

$$\sum_{i=1}^{n} A\,[i] \Leftrightarrow \sum_{i=1}^{n} B\,[i].$$

The results of NIST testing of the developed key stream generators on basis of a single CA rule 30 showed that up to 95-96 % of all generated sequences passed the tests, indicating their satisfactory statistical properties. In order to improve the achieved characteristics, the combinations of cell interaction rules of 30, 86, 135, 149 belonging to the same class were applied.

The output bits were combined with each other in different ways. However, the best statistical characteristics with at least 97% of generated binary sequences, successfully passing the NIST STS tests, were demonstrated by the generators, organized by the rule (5):

$$C_{out}[w] = (C_{30}[w] \oplus C_{86}[w] \oplus 1) \oplus (C_{30}[w] \oplus C_{135}[w] \oplus 1) \oplus \oplus (C_{30}[w] \oplus C_{149}[w] \oplus 1). \tag{5}$$

All the generators under investigations were implemented in software and their performance was assessed. Absolute values of generation rates for different generators fell within 48 ÷250 Mbps.

Due to the involved logical operations, parallelism and extreme sensitivity to initial conditions CA can be effectively used to design robust cryptographic hash functions with low hardware and software complexity [2]. In our research we have proposed special framework of one- two- and three-dimensional CA to develop hash functions. The designed constructions rely on the cryptographic sponge, deployed in Keccak algorithm of the SHA-3 standard [4]. The sponge operates in two phases of absorbing and squeezing and provides pseudorandom permutation, aimed at preventing collision occurrence. In order to implement high-quality permutation functions, ensure collision resistance and nonlinearity we have proposed joined application of both linear (rule 150) and nonlinear (rules 30, 54, 86, 158) [3] CA processing rules.

The sponge construction possesses 1600-bit long state, which in our research is implemented as a CA in one- and multi-dimensional representation. To ensure effective permutation each cell communicated with a specific pair of neighbors through application of various combinations of the 30, 54, 86, 150 and 158 CA rules together with bitwise operations of XOR and cyclic shift. The interaction was conducted in parallel between binary vectors, rather than individual cells, which significantly accelerated the processing rate.

Software solution to implement the proposed permutation functions has been developed. The parameters of the inner state of cryptographic sponge comply with those of Keccak algorithm [4]. The created software enables production of the message digests of 224, 256, 384 and 512 bits. According to the obtained statistical data, at least 96 % of the bit sequences under investigation have successfully passed all the NIST STS tests. It points out, that binary strings generated by the constructed hash functions on the basis of both one- and multi-dimensional CA approach the pseudorandom ones.

All developed hash functions revealed strong avalanche effect, i.e. the digest of the incoming message updates completely, when changing the hash length or the input message content [5]. The processing rates of the designed hash functions ranged within 54 – 700 kBps, with highest values observed for the proposed two-dimensional CA permutation functions.

In pursuit to increase the processing rates, an approach of parallelized implementation of CA in FPGA has been proposed. It allows one to generate the new generation of CA during a single clock cycle. It was shown that only 2 logical elements are required for 1 cell of such implementation. The maximum clock rate for the circuit in FPGA depends on internal length of connections and on the chip family. For example, if the maximum clock frequency for the specific chip of Altera Cyclone II family is 500 MHz, the possible hashrate for a short message will be up to 1 Megahash per second, i.e. more than 512 MBps [6].

Two constructions of the CA-based block cipher have been considered. The first one operates as three-dimensional byte-oriented CA container of 64 bytes (512 bits), also used for encryption key. The containers are divided into 4 layers, closed into 3d-torus. For better diffusion, cells may interact either within a layer, or with the cells of adjacent layers. Every cell communicates both with its container close neighbors, as well as with the bytes of the key container through the operations of XOR and cyclic shift. Substitution blocks (s-blocks), randomly filled with the numbers from 0 to 255, have been used. Prior to the encryption itself, round sub-keys have been generated by the same operations deployed in the encryption process. The proposed algorithm has no limitations for the block and key length and uses special layer cards to bypass the layers.

The second version of a block cipher algorithm is based on reverse CA. A cell composition is formed by the cell itself and its left and right neighbours and is represented through a parameter of radius *r*. While the radius is 1, the composition 'm' includes (2*r) + 1 cells. This makes $n = 2^m$ possible patterns and 2^n possible rules, i.e. for $r=1$, $m = 2*1 + 1 = 3$, $n = 2^3 = 8$ possible patterns, 2^8 implies 256 rules [7, 8].

The key contains the rule for the conversion of CA and special bits to conceal its data. The rule can be applied to cells at different radii (1, 2, 3). The larger the radius of the rules, the more time calculation takes, though resulting in better permutation. Each bit of the rule corresponds to a combination of bits of neighboring cells in the CA. To define a reverse rule, to that given by the key (r1), the formula $r2 = 2^n - r1 - 1$ is used [8]. One rule (r1) will work when the CA at step t-1 is zero, and the other (r2) is applied when it is equal to one. These rules are used for encryption and decryption, respectively. Initially, one-dimensional CA is initialized with a random variable and a block of information that needs to be encrypted. Through a number of iterations, a portion of encrypted information and data for encrypting the next block are obtained. The blocks of 128, 256 and 512 bits have been considered while the number of iterations ranged from 5 to 20, yielding binary patterns with rather promising statistical properties. When the last block of information is encrypted, the final data are concealed with a specially created part of the key, to prevent unwanted deciphering the information.

The investigation of statistical characteristics of the developed symmetric key constructions showed that at least 95 % of all generated binary strings successfully passed 189 NIST STS tests.

Conclusions

Summarizing all mentioned above, the following conclusions can be made:

- The developed permutation approach built on one- and multi-dimensional CA with application of interaction rules of 30, 54, 86, 150 and 158 ensures the compliance of at least 96% of the generated hash strings with NIST conditions, pointing out good statistical properties of the proposed cryptographic hash functions.
- The processing rates of the software implementation of the designed hash functions within the range of 54 – 700 kBps have been achieved, with highest values observed for the two-dimensional constructions.
- The hardware implementation of CA in FPGA with moderate resourced needed is expected to entail considerable increase in the processing rates of up to 512 MBps.
- The usage of one- and three-dimensional cellular automata along with Boolean operations enabled us to develop the constructions of block ciphers, which demonstrate good statistical and diffusion characteristics together with satisfactory performance.

References

[1] Wolfram S., "Random sequence generation by cellular automata", *Advances in Applied Mathematics*, Vol. 7, 1986, 123 – 164.

[2] Allaa Eddine Belfedhal, Kamel Mohamed Faraoun. "Building secure and fast cryptographic hash functions using programmable cellular automata", *J. of Computer and Information Technology, CIT*, Vol. 4, 2015, 317–328.

[3] Jeon J.-Ch., „Analysis of hash functions and cellular automata based schemes*", International Journal of Security and Applications*, Vol. 7, 2013, 303–316.

[4] Bertoni G., Daemen J., Peeters M., Van Assche G., "Keccak sponge function family main document", http://keccak.noekeon.org/Keccak-main-2.1.pdf.

[5] Tanasyuk Yu., Ostapov S., "Development and research of cryptographic hash functions based on two-dimensional cellular automata", *IAPGOS,* 1, 2018, 24 – 27.

[6] Tanasyuk Yu., Perepelitsyn A., Ostapov S., "Parametrized FPGA-based implementation of cryptographic hash functions using cellular automata", *The 9th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2018, 24-27 May, 2018, Kyiv, Ukraine, 238 – 241.*

[7] Seredynski F., Bouvry P., Zomaya A.V., "Cellular automata computations and secret key cryptography", *Parallel Computing*, 30, 2004, 753-766.

[8] Debasis Das, Abhishek Ray, "A parallel encryption algorithm for block ciphers based on reversible programmable cellular automata", *J. Computer Science and Engineering*, Vol. 1, No. 1, 2010, 82 – 89.