# Hash Functions Based on One- and Multy-Dimensional Cellular Automata

O. Konstantynyuk, Yu. Tanasyuk, S. Ostapov
Yuriy Fedkovych Chernivtsi National University
Chernivtsi, Ukraine
y.tanasyuk@chnu.edu.ua

***Abstract*—Cryptographic hash functions on the basis of one-, two- and three-dimensional cellular automata deploying pseudorandom permutation with the use of various processing rules have been developed. The proposed constructions revealed high-quality scattering properties, strong avalanche effect and sufficient processing rates in producing the message digests of 224, 256, 384, 512 bits.**

***Keywords—cryptographic hash function, cellular automata, cryptographic sponge, Keccak algorithm***

## I. INTRODUCTION

Cellular automata (CA) are well known to be self-organizing statistical systems, providing ample opportunities for simulation of physical systems, image processing, design of computer architectures and cryptography [1]. Due to their ability to generate high-quality pseudorandom patterns, CA have been considered for design of block and stream ciphers, public key cryptography, message authentication and hash function. A number of CA rules and their combinations with bitwise operations exhibit a desired behavior needed in cryptographic primitives.

Cryptographic hash function is primarily used to create a unique representation of an input message by computing its short fixed-length digest, known as a fingerprint of the message. To be secure, a hash function needs to be irreversible and resistant to collisions [2]. CA based hash functions reviewed in [3] are claimed to be collision free and able to achieve high processing speed, resulting from parallelism and homogeneity of the underlying transition rules.

Recently, a large number of hash function construction approaches have been proposed. Among the most promising there is a Keccak algorithm, adopted for the SHA-3 standard, that doesn't rely on a compression approach of its predecessors but is based on a sponge construction, which provides pseudorandom permutation [4].

The main purpose of the paper is to develop and research cryptographic hash functions, based on the sponge construction of the Keccak algorithm and various processing rules of one- two- and three-dimensional CA.

## II. KECCAK FUNDAMENTALS

The Keccak algorithm is reported to possess many attractive features, including its ability to run well on different computing devices, i.e. embedded or smart, and high performance in hardware implementation, comparing to SHA-2 [4]. Keccak is based on the sponge function, which is, in general, a cryptographic hash function with a varying output.

Sponge has its inner state, which is a binary array of the fixed length $b$. The array consists of two parts – $r$ and $c$. Parameter $r$ is called a bit rate. This very part is combined with the equal portions of the input message and is used to produce a resulting hash string. Parameter $c$ is called the capacity, $c=b-r$. This value is not directly affected by input message blocks and is responsible for security level of the hash function. Namely, to derive the hash with defined mathematical stability, the value of capacity must be twice as large as the hash length. For SHA-3 with the state of $b$=1600 bits the parameters of sponge are given in Table I.

Prior to processing a message by the Keccak hash function, the input message has to be padded to the length, which is the multiple of $r$ bits. Then, the padded message is divided into the blocks of $r$-length. The sponge construction operates in two phases: absorbing and squeezing.

At the absorbing stage $r$ portion of the sponge inner state is combined with the message block of the same length by means of XOR operation, and the whole state array is processed by a permutation function for a fixed number of rounds. Squeezing phase starts after all message blocks have been absorbed and is aimed at generation of the message digest of the desired length.

In the original Keccak algorithm the sponge state is presented as a three-dimensional array of 5×5 64 bit words. At heart of the described construction there is the permutation function, which consists of five steps, denoted by Greek letters: $\theta$ (theta), $\rho$ (rho), $\pi$ (pi), $\chi$ (chi) and $\iota$ (iota). The named functions include bitwise operations, and are claimed to be relatively hardware friendly resulting in high performance of the Keccak algorithm [2, 4].

TABLE II. KECCAK PARAMETERS FOR HASH OF VARIOUS LENGTH

| Hash Length, *Z* (bits) | Bit Rate, *r* (bits) | Capacity, *c* (bits) | Security Level, Z/2 |
|---|---|---|---|
| 224 | 1152 | 448 | 112 |
| 256 | 1088 | 512 | 128 |
| 384 | 832 | 768 | 192 |
| 512 | 576 | 1024 | 256 |

In the research conducted we have focused on the design of cryptographic hash functions that are based on sponge construction, implemented in the shape of one-, two- and three-dimensional cellular automata with the use of specific combinations of the CA processing rules and bitwise operations.

## III. DESIGNING HASH FUNCTIONS ON THE BASIS OF CA

A CA is a collection of simple cells connected in a regular manner. Each cell can assume the value of binary 0 or 1. The cells evolve simultaneously in discrete time steps according to some deterministic rule. The next state of the cell depends on itself and on its neighbors. Our investigations considered the following CA processing rules:

$$\text{rule 30: } b' = a \oplus (b \vee c), \quad (1)$$

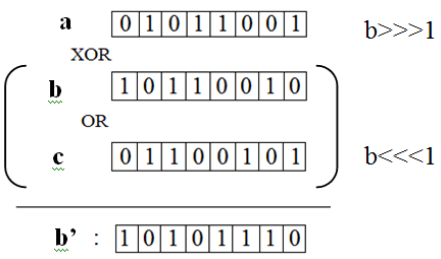
$$\text{rule 86: } b' = (a \vee b) \oplus c, \quad (2)$$

$$\text{rule 150: } b' = a \oplus b \oplus c, \quad (3)$$

where $b$ is the current cell, $b'$ is its new value after the rule application, $a$ is the previous cell, $c$ is the next cell, and $\oplus$, $\wedge$, $\vee$ denote the bitwise XOR, AND, and OR operations, respectively. According to [5] the rule 150 (3) is called linear, since it involves only XOR logic. The rules (1) and (2) containing XNOR logic are nonlinear. As recommended in [1], in order to design a reliable hash function a combination of linear and nonlinear CA rules is to be used. Linear rules provide collision resistance, while nonlinear ones bring about one-way property and nonlinearity.

### A. One-dimensional CA

One-dimensional sponge state is implemented as a 1600-bit long binary array, which is a three neighborhood CA, with extreme cells adjacent to each other. The round permutation is performed through the multiple use of one of the rules of 30, 86, or 150, their joined sequential application, or a combination of the rules with bitwise operations of cyclic shift and negation, depending on the number of iteration [6]. It's noteworthy, that cell processing was implemented not in a bit-to-bit manner, but in parallel. For this purpose at each round two instances of the current state array were created: one-bit cyclically shifted to the right copy represented all previous cells, while one-bit cyclically left-shifted one contained all next cells. This approach enabled us to apply corresponding bitwise operations to the obtained bit sets. Fig. 1 shows schematically application of rule 30 (1) to the 8-bit long string.




Concurrent application of rule 30 (1) to the entire bit string

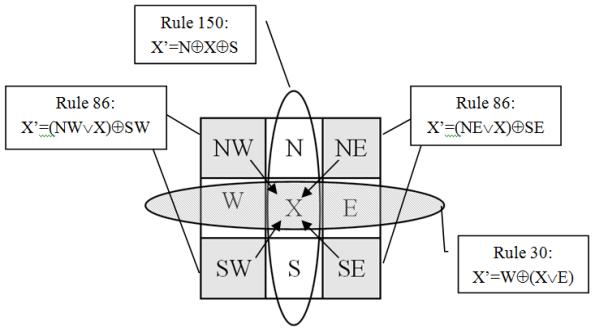
### B. Two-dimentional CA

In two-dimensional CA representation the sponge state is arranged as an array of 25 64-bit long strings, making 1600 bits in total. The cells are localized according to the Moore neighborhood [5], when two cells are considered adjacent if they have either a common edge or a vertex. Therefore, each cell interacts with its eight direct neighbors, denoted as parts of the world (Fig. 2). Extreme cells are connected in tor with their counterparts on the opposite edge (row/column) of the array.

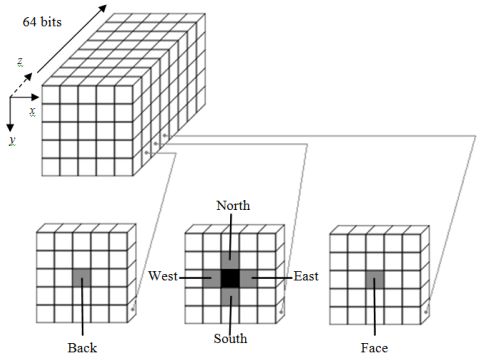
To ensure effective permutation, combinations of adjacent cells were processed with different CA transformation rules, as shown in Fig.2. The rule deals with the entire rows concurrently, according to the technique, described above.

### C. Three-dimensional CA

The sponge state is arranged as a two-dimensional array (5x5) of 64-bit vectors ($b$=5×5×64=1600 bits). As in the constructions, described above, $r$ and $c$ portions were defined by the desired hash length (see Table I). First $r$/64 vectors were initialized with binary 1s, while a bit corresponding to a vector's index was inverted. This procedure is chosen to improve scattering properties of the developed construction. Each cell of the proposed three-dimensional CA possesses 6 neighbors with common edges (Fig. 3), denoted as North, South, East and West within the same plate, while Face and Back are shown as neighboring cells of the adjacent plates.



Interaction rules for a cell X with its neighbors in two-dimensional CA, where N, W, NE, NW are treated as previous cells, while S, E, SE, SW are the next ones.




A group of cells, participating in interaction in three-dimensional CA. With respect to the central cell, North, West and Back are considered as previous neighbors, and South, East and Face are the next ones.

In order to apply CA processing rules to each cell and explain the interaction of the current cell with its neighbors, we've introduced the following notations: along X axis West is a previous cell and East is a next one. In the direction of Y axis (as shown in Fig. 3) North and South are previous and next cells, respectively. And along Z axis Back is a previous neighbor and Face is the following one.

To implement any rule of CA all previous cells and all next cells are combined by XOR operation with each other. Namely, rule 86 (2) is performed as follows:

b' = (([North] xor [West] xor [Back]) **OR** [b]) **XOR** ([South] xor [East] xor [Face])

Interaction with compliance to the described rules is carried out between the entire binary vectors, rather than individual cells, which can significantly accelerate the processing rate.

With regard to transformations between the cells of the face and back within the current vector, two copies of it are created: one bit cyclically shifted to the left, and to the right, denoting the Back and the Face, respectively.

A permutation function on the basis of the designed three-dimensional CA includes a combination of CA processing rules and binary functions, applied at each round of absorbing and squeezing. The processing involves two empty two-dimensional (5x5) arrays of 64-bit vectors newArrayRC and tempArray. Each 64-bit vector of the original array (ArrayRC) is processed with the use of rule 30, followed by 23-bit cyclic shift to the right, and the resulting vectors are consistently written into tempArray.

As the transformation is complete, ArrayRC is combined with tempArray as its shifted copy through XOR operation. Then, similarly, the basic array is updated by XOR with its copy tempArray, obtained through application of rule 86 with further 3-bit cyclic shift to the left. After that the vectors of the basic array undergo processing by rule 150. When the manipulation of the main array is over, the content of its first column of 64-bit vectors is copied to the last column of the newArrayRC, followed by one-position horizontal and vertical shift of the vectors to the left and down, respectively. The whole procedure is accomplished in 5 steps. On completion of the transformations, the newArrayRC becomes a main array. Its final processing is performed with the use of rule 86, 3-bit left cyclic shift, XOR and rule 150 operations, in the manner described above.

## IV. Results and Discussion

Computer program to implement the proposed permutation functions has been developed. The parameters of the inner state of cryptographic sponge comply with those, proposed by the Keccak algorithm. Although, the created software enables generation of the hash strings of 224, 256, 384 and 512 bits, message digest of any other desired length may be calculated, if corresponding ratio between *r* and *c* parameters is preserved. In order to provide a sufficient level of security, value of *c* must be twice as large as the hash length.

Scattering properties of the developed hash functions were studied using the NIST STS technique. The binary sequences of $10^8$ bits were generated by the proposed one-, two- and three-dimensional constructions with such parameters (bits): *b*=1600, hash length *Z*= 512, *r*=576, *c*=1024.

According to the obtained statistical data, at least 96 % of the sequences 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, by their properties approach the pseudorandom ones.

Fig. 4 and 5 shows typical statistical portraits of the cryptographic hash functions, built on the proposed construction of multi-dimensional CA. The generalized results of the conducted statistical investigation are given in Table II.

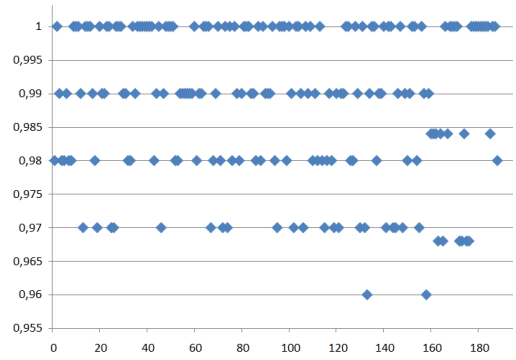



Fig. 4. Statistical portrait of the cryptographic hash function on the basis of two-dimensional CA after 5 rounds of permutation, where N is a number of a test, P is the portion of test sequences, which passed the test

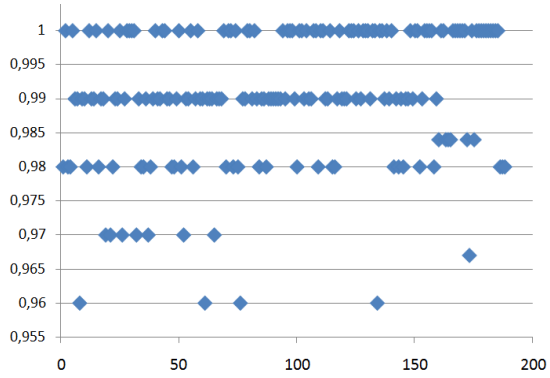



Fig. 5. Statistical portrait of the cryptographic hash function on the basis of three-dimensional CA after 2 rounds of permutation, where N is a number of a test, P is the portion of test sequences, which passed the test

TABLE III. GENERALIZED RESULTS OF STATISTICAL TESTIG OF VARIOUS PERMUTATION FUNCTIONS ON THE BASIS OF CA

| **Hash algorithm version**[a] | **NIST STS testing results** | | | | |
|---|---|---|---|---|---|
| | ***<0.96*** | ***0.96*** | ***0.98-0.97*** | ***1-0.99*** | ***Average*** |
| 1. | 0 | 2 | 56 | 130 | 0.9897 |
| 2. | 0 | 2 | 65 | 121 | 0.9889 |
| 3. | 2 | 6 | 41 | 139 | 0.99 |
| 4. | 4 | 4 | 55 | 125 | 0.9883 |
| 5. | 0 | 4 | 46 | 140 | 0.9907 |

[a] where 1 - one-dimensional CA, 25 rounds; 2, 3 - two-dimensional CA, 5 rounds and 10 rounds, respectively; 4, 5 – three-dimensional CA, 1 round and 2 rounds, respectively.

The performance of the cryptographic hash functions, given in Table II, was estimated on the computer with CPU Intel Core i5 4200U, 1.5 GHz and RAM 4GB by processing rates of forming a 100 MB text file of 512-bit binary hash strings, used in the NIST STS statistical tests. As Table III shows, the highest processing rates were achieved for two-dimensional CA construction at 5 rounds of permutation (hash algorithm of version 2). The second best result applies to the functions built on one-dimensional CA that underwent processing by the set of rules and binary operations for at least 25 rounds (version 1).Keccak Parameters For Hash of Various Length

| **Hash algorithm version**[a] | **Number of rounds** | **Time to form a 100 MB text file of 512-bit hash strings (seconds)** | **Processing rate (KB/s)** |
|---|---|---|---|
| 1. | 25 | 213 | 492.3 |
| 2. | 5 | 136 | 771 |
| 3. | 10 | 288 | 364.1 |
| 4. | 1 | 408 | 257 |
| 5. | 2 | 834 | 125.7 |

[a] where 1 - one-dimensional CA, 25 rounds; 2, 3 - two-dimensional CA, 5 rounds and 10 rounds, respectively; 4, 5 – three-dimensional CA, 1 round and 2 rounds, respectively.

The application of the developed permutation functions for obtaining a hash image revealed the dependence of the scattering properties on the length of the hash, the type of the function, and the number of processing rounds. For all proposed hash functions, strong avalanche effect was observed, i.e. the digest of the incoming message was completely updated when changing the hash length (224, 256, 384, 512 bits) or at the smallest changes in the message. It should be noted, that for various hash functions the avalanche effect was observed at different number of the processing rounds. Namely, the one-dimensional permutation functions based on the use of one CA transformation rule need up to 100 rounds, while application of several rules brings about satisfactory outcomes after 50 iterations [6]. A full change in the resulting hash occurs for two-dimensional hash functions starting from 5 processing rounds, while three-dimensional constructions produce a completely different hash string after 1 round of permutation, without significant degrading a processing rate.

## V. CONCLUSION

Summarizing the conducted investigations the following conclusions can be made:

1. The permutation functions, based on one-, two- and three-dimensional CA, with the use of various rules of CA interactions have been developed.

2. Joint application of both linear and nonlinear CA processing rules together with bitwise operations enabled us to achieve high-quality scattering properties and provide satisfactory level of security in the designed constructions.

3. Concurrent manipulation of the inner state's vectors of the cryptographic sponge ensured reasonable processing rates, while deployment of multi-dimensional CA significantly reduces the number of iterations.

4. All the designed transformation functions under investigation revealed the appearance of the avalanche effect, considered to be a desirable characteristic of cryptographic hash function.

## REFERENCES

[1] J.-Ch. Jeon Analysis of hash functions and cellular automata based schemes. International Journal of Security and Applications, 2013. – Vol. 7, No. 3, pp.303–316.

[2] Ch. Paar, J. Peltz. Understanding cryptography. – Springer-Verlag Berlin Heidelberg, 2010. – 372 p.

[3] N. Jamil A new cryptographic hash function based on cellular automata rules 30, 134 and omega-flip network. ICICN 2012, 2012. – Vol. 27, pp. 163 – 169.

[4] G. Bertoni [Electronic resource]. – The Keccak sponge function family. – Access mode : http://keccak.noekeon.org/.

[5] S. Wolfram S. A New Kind of Science Wolfram Media, Inc. – 2002. 1197 p. – [Electronic resource]. – Access mode: http://www.wolframscience.com/nksonline/toc.html.

[6] Yu. Tanasyuk, Kh. Melnychuk, S. Ostapov. Development and research of cryptographic hash funcions on the basis of cellular automata. – Information Processing Systems, 2017. – Vol. 4(150), pp. 122 – 127