2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

A New Cryptographic Hash Function Based on Cellular Automata Rules 30, 134 and Omega-Flip Network

Norziana Jamil [1,2], Ramlan Mahmood [2], Muhammad Reza Z'aba [3], Nur Izura Udzir [2] and Zuriati Ahmad Zukarnaen [2]

[1] Universiti Tenaga Nasional, KM7, Jalan IKRAM-UNITEN, 43600 Kajang, Selangor, MALAYSIA
[2] Universiti Putra Malaysia, 43400 Seri Kembangan, Selangor, MALAYSIA
[3] MIMOS Berhad, Technology Park Malaysia, 57000 Bukit Jalil, Kuala Lumpur, MALAYSIA

Abstract. Cryptographic hash function is a combination of several components including a compression function which is made up of a few underlying elements such as Boolean functions and permutation. Some properties of Boolean functions are very helpful for determining sufficient conditions used in modern attack for hash functions. We study these underlying elements by looking at Cellular Automata (CA) rules as Boolean functions and Omega-Flip network as a permutation technique. Two main cryptographic properties are studied for both elements namely Strict Avalanche Criterion (SAC) and randomness. One of the CA rules which was widely studied for cryptography is Rule 30 because it exhibits a desired behavior needed in cryptographic primitives. However Rule 30 alone is not sufficient to provide full avalanche and randomness. Therefore we study a few other non-linear CA rules and combine them with CA rule 30. Our study finds that the hybrid rules do not exhibit a desirable Strict Avalanche Criterion (SAC) for a small iteration. However it achieves full avalanche for a very large iteration. We then introduce a customized omega-flip permutation technique to increase its diffusion level within small iteration. The result shows that some combinations of CA rules and our customized omega-flip permutation technique yields a desirable SAC within small iteration and passes all the statistical tests in a statistical battery package distributed by the National Institute and Standard Technology (NIST).

Keywords: cryptographic hash function, Cellular Automata, Rule 30, Omega-Flip Permutation.

1. Introduction

Cellular automata (CA) were proposed by John von Neumann as a discrete system for self-reproducing organisms. They were used as a toy model for phenomenology associative with macroscopically irreversible process. CA is considered as a new development in modern science. In cryptography, a number of researchers have concentrated on the study of additive CA (a combination of XOR and XNOR operation in its transition function f) due to several properties such as adaptability, reversibility and scalability [15].

Somanath and Sukumar [26] have designed a lightweight CA-based symmetric-key encryption that supports 128-bit block size with 128-, 192- and 256-bit keys. The motivation for embarking on CA for their cipher design is due to the fact that CA provides a high level of parallelism and therefore, able to achieve high speeds. The cipher has also been proved to be resistant against timing analysis attacks.

For cryptographic hash functions, Daemen et. al [13] have successfully attacked one of the three examples of collision free functions given by Damgard [10]. Furthermore, they also proposed a direct design of a framework of collision free hash functions based on CA named Cellhash. They claimed that Cellhash is suitable for hardware implementation, making extremely high speed possible. Mihaljevic, Zheng and Imai [18] proposed a cryptographic hash function based on linear CA over GF (q). The proposal is an extension of bit oriented hashing that was proposed earlier [17] and employs the approved model of iterative hash function with compression and output functions. Their compression function is one of the Davies-Meyer type employing CA and their output function is a key generator which is also based on CA over GF(q).

It has been shown in a number of studies (for eg:[19], [28], [29]) that the 1-dimensional 3-neighborhood CA exhibits excellent performance but some of the objects in the particular system (for example in modeling physical system) do not obey the homogeneity of the rule, i.e the same local rule throughout the system evolution. Then the non-homogenous CA (or also called a hybrid CA) is evolved to describe such a system evolution. A number of researchers have studied on hybrid CA such as [16], [21] and [20] for VLSI implementation and found interesting results out of it.

In this paper, a hybrid CA rule 30 and rule 134, and a customized Omega Flip permutation technique are proposed as underlying elements of compression function in cryptographic hash functions. As far as we know, this is the first time that such underlying elements are employed in compression function of cryptographic hash functions.

This paper is organized as follows. In Sections 2, we discuss some preliminaries about CA followed by the relevant background about one way hash functions in Section 3. In Section 4 the primitives used in our cryptographic hash function is described. Its security analysis is carried out in Section 5 and some concluding remarks are made in Section 6.

2. CA Preliminaries

The simplest CA is a 1-dimensional linearly connected array of n cells, usually referred to a 3-neighborhood CA cells. Each of the cells in the array takes a discrete state s of 0 or 1. A transition function f is used to determine the next state configuration c whereby the value of the ith cell state s_i is updated in parallel using this function in discrete time step t. The next state of a cell at t+1 is influenced by its own state and the states of its left and right neighbours at time t. The state configuration c that is specified by a transition function f can be seen as:

$$s_i(t+1) = f\{s_{i-1}(t), s_i(t), s_{i+1}(t)\}$$

where $s_i(t)$ are the states of the left neighbour, self and right neighbour of the ith cell at time t. For a 2-state of 3-neighborhood CA, there are 2^3=8 possible state configurations. If the configuration of the ith cell is expressed in the form of a truth table, the decimal equivalent of its sequence output- is referred to as 'Rule' R_i [28]. So for a 2-state 1-dimensional 3-neighborhood CA there is a total of 2^8=256 CA local rules [29]. In our paper, we refer this rule as the transition function f.

Definition 1. If f of CA involves only XOR logic then the rule is called a linear CA rule whereas a non-linear CA rule refers to f of CA which involves the combination of any of AND, OR and XOR logic.

Definition 2. If the output sequence of the CA is periodic then the CA is called a group CA. Otherwise it is called a non-group CA.

Definition 3. If the extreme cells are adjacent to each other, a CA is called a periodic boundary CA.

In our work, we concentrate on non-linear and non-group CA whose cells are of type periodic boundary.

3. Cryptographic Hash Function

Cryptographic hash functions are widely used in digital signatures, e-cash, signature verification and many more applications. In general, a hash function H accepts an arbitrary length message as input and produces a fixed length output. It is described as follows:

$$H : [0,1]^* \rightarrow [0,1]^n.$$

A cryptographic hash function principally has three security properties namely:

1. Pre-image resistance: H is said to be pre-image resistant if given a hash value h_{final} it is infeasible to find its associative message m such that $h_{final} = H(m)$.

2. 2nd pre-image resistance: H is said to be 2nd preimage resistance if given a message m and its hash value h_m, it is infeasible to find another message n such that $m \neq n$ and $H(m)=H(n)$

3. Collision resistance: H is said to be collision resistance if it is infeasible to find any two messages m,n where $m \neq n$ such that $H(m)=H(n)$.

Cryptographic hash functions consist of two independent components, namely the mode of operation and the compression function. However, this paper focuses only on the compression function, or to be exact the underlying elements in compression function namely Boolean functions and permutation.

A compression function fc is a function that provides collision resistance property. It iterates itself to achieve one-wayness and to guarantee avalanche effect. Iterated hash functions can be classified into several categories: hash functions based on block ciphers (eg: [17]), hash functions based on modular arithmetic (eg:[25]), hash functions based on knapsack problem, dedicated or customized hash functions and hash functions based on cellular automata [13]. Handbook of Applied Cryptography [4] defines dedicated hash functions as "those which are specifically designed 'from scratch' for the explicit purpose of hashing, with optimized performance in mind, and without being constrained to reusing existing system components such as block ciphers or modular arithmetic". The popular cryptographic hash functions such as MD5 [24] and SHA1[15] are based from block cipher with feedback mode to ensure that the resulting function is not bijective. Since very little research is done on hash functions based on CA, we take the effort to further investigate the possibility of embarking 'good' rules in CA in cryptographic hash functions. Furthermore, CA exhibit interesting behaviours which are desirable in cryptographic primitives.

4. The Building Block of A Compression Function

The building block of a compression function is permutation and Boolean functions. For permutation, we investigate and customize a permutation instruction called an omega-flip permutation, which was proposed by Lee, Shi and Yang[22], and a few non-group and non-linear CA rules to achieve both basic cryptographic properties. We will explain the justification of choosing omega-flip permutation for the building block of our cryptographic hash function primitive in the following section. A high level implementation of the building blocks is given by Algorithm 1.

Algorithm 1. Compression function

```
(1) INPUT: 16 message blocks of 32-bit words, $m_i = m_1, m_2, m_3, \ldots m_{16}$ , IV as four 32-bit registers, IV = $r_1, r_2, r_3, r_4$
(2) Length: Message block = 512 bits, message word = 32 bits, output block = 128 bits
(3) For $j$=0 to $k$ ($k$ = number of rounds)
(4) Compute $CA_1(m_i, r_i+1, r_i+2, r_i+3)$
(5) Compute $CA_2$(output from (4))
(6) Permute the new configuration $n_i$ after performing $CA_1$ and $CA_2$.
(7) Perform addition modulo $2^{32}$ operation for $n_i$ and IV to form a final hash value $h_i$
(8) OUTPUT: $h$
```

4.1. Omega Flip Permutation

We explore a few candidates of permutation methods which can yield efficient hardware implementations. The best candidate is the omega-flip network [22], which consists of an omega network followed by a flip network. The following steps as described in [22] are used to generate the original omega-flip instruction sequence for a specific permutation:

i. A valid configuration for the desired permutation is obtained.

ii. The configuration is mapped to the omega-flip network based on the isomorphism between them.

iii. The configured omega-flip network is broken into pairs of stages.

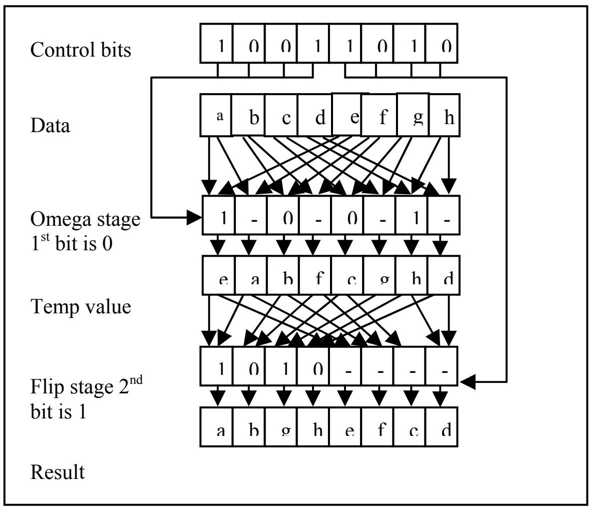
iv. Each pair of stages is then assigned with an omega-flip instruction. Each omega-flip instruction uses the output from the previous omega-flip instruction and input for the next omega-flip instruction is generated. The first omega-flip instruction takes the original input and the final result is generated by the last omega-flip instruction.We slightly modify the original omega-flip network as follows:

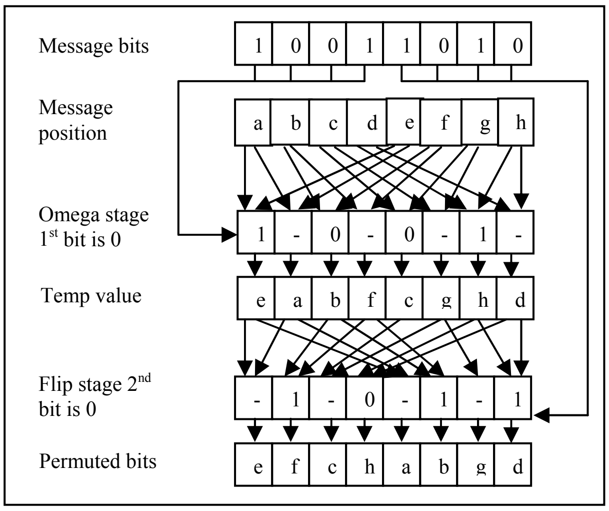
i. Steps (i), (ii) and (iii) as in the original omega-flip network are retained.

ii. For omega stage, the first four left-most bits are inserted alternatively with most significant bit in the left-most register.

iii. For flip stage, the first four bits from right are inserted alternatively with least significant bit in the left-most register.

The example of implementation of the original and the modified omega-flip network are described in Fig. 1 and Fig. 2 respectively.





Control bits in the original omega-flip permutation network is the message bits in the modified omega-flip permutation. In the omega stage, bit '1' maps the first left arrow pointed to it and bit '0' maps the first right arrow pointed to it. In a case for both situations where the position whose arrow has been mapped to the next position, it will take the second arrow pointed to it. This mapping is applied to both the original and modified omega-flip permutation network. The only difference is on the way the message bits are arranged in the array. In the modified omega-flip permutation, message bit '1' is assigned in the left most array in the omega stage and in the second most left array in the flip stage. We say that the permutation is deterministic and collision resistance, since the final bits are determined by a fixed position. We believe that the change has no big difference in term of performance as the original omega-flip network.

4.2. Rule 134 and 30 As Boolean Functions

All In CA, the 256-rules are also referred as Wolfram rules. He undertook a detailed study of one dimensional cellular automata and their relationship to dynamical systems [28]. Rule 30 is one of the rules that fall under this class. Based on a result from [6] we conjecture that Rule 30 alone is not sufficient to strengthen the primitives for cryptographic hash functions. Therefore we select randomly a few non-linear CA rules and integrate it with Rule 30. The selected rules that we test are Rule 126, Rule134 and Rule 166. The Boolean form representation of these rules is described below:

Rule 30: $p\otimes(q\vee r)$, rule 126: $(p\otimes q)\vee(p\otimes r)$, rule 134: $(p\wedge(q\vee r))\otimes q\otimes r$, rule 166: $(p\wedge q)\otimes q\otimes r$, where p is the left neighborhood cell, q is the center cell and r is the right neighborhood cell.

There are two properties that we test over the integration of rule 30 and the selected rules, namely Strictly Avalanche Criterion (SAC) and randomness. SAC [8] is a property that any one-bit change in the input would result in the change of at least half of its corresponding output bits. This effect is also referred as the diffusion effect. For randomness, we test the hybrid rules using a statistical battery package distributed by the National Institute of Standards and Technology (NIST) [5]. The passing of all the statistical tests is one of the indications that the primitives would survive against statistical attacks. The result is discussed in the following section.

5. Security Analysis

We have performed a preliminary security analysis of our proposal both in terms of avalanche effect and randomness. Our experiments take 100 samples of 512-bit of message blocks as input and outputs 256 bits for each of the sample. We iterate the building blocks (CA operations and omega-flip permutations) for 128, 256, 512 and 1024 rounds. The experiment is done only on this small component consists of CA operation and permutation. The following table shows the result of SAC and statistical tests this particular component.

5.1. Analysis on Strict Avalanche Criterion (SAC)

The SAC was originally proposed by Webster and Tavares [2] in 1986. It states that for whenever one input bit is changed, every output bit is changed with probability 0.5. In our experiments, we want to achieve a probability of 0.5 on average, for all the sample inputs. Note that we did not test the samples for 128 and 256 rounds since we notice that the CA rules do not achieve the desired probability even for higher rounds. Since the result is quite massive, we only show the average probabilities obtained from the test. For the purpose of justifying the omega-flip permutation as one of the building blocks of our cryptographic hash function primitives, we divide the test into two parts. The first part is the SAC test for hybrid CA rules without the omega-flip permutation. The second part is the SAC test for hybrid CA rules with omega-flip permutation. The results of both parts are shown in Table 1 and Table 2, respectively.

Table 1: The SAC result for hybrid CA rules without omega-flip permutation

Rules	Probability for 512 rounds	Probability for 1024 rounds
Rule 126&30	0.00277	0.00278
Rule 134&30	0.415	0.418
Rule 166&30	0.008804	0.0126

Table 2: The probability result for hybrid CA rules with omega-flip permutation

Rules	Probability for 128 rounds	Probability for 256 rounds
Rule 126&30	0.28	0.29
Rule 134&30	0.48	0.49
Rule 166&30	0.38	0.39

Table 1 shows the result of the SAC tests on hybrid CA rules without the omega-flip permutation that run for 512 and 1024 rounds. The number of rounds for this experiment is higher than that of the second experiment which results are depicted in Table 2. This is because we conjecture that the rules need to be iterated more if they want to generate the desired probability, which is 0.5. Nevertheless two hybrid CA rules namely rules 126 and 30, and rules 166 and 30 do not produce the desired probability. Only hybrid rules 134 and 30 show a probability close to the desired one. In Table 2, the hybrid CA rules 134 and 30 together with omega-flip permutation shows a good probability, which is approaching 0.5. We believe that with a proper number of rounds, it can achieve a full avalanche. Time (in second) in both tables show the time taken to complete this component which consist of hybrid CA rules operation with and without the omega-flip permutation. Note that the time taken is not considered as really fast and we conjecture that this is because of the non-optimized coding in our implementation of omega-flip permutation.

5.2. Analysis on Statistical Randomness Tests

Statistical tests are a set of functions that take arbitrary length input and produce a real number between 0 and 1, or also called as a *p*-value. This is done by evaluating certain randomness properties of the given input bits. There are many statistical randomness test suites available but we opted to use the battery of statistical tests distributed by NIST[5] in 2001. This test suite consists of 16 different tests to observe the randomness properties of the sequence bits. However, the Fast Fourier Transform test was discarded due to a problem observed by NIST in 2009 and only 7 proper tests for 256-bit sequences are taken namely Frequency test, Frequency Test within a Block, Runs test, test for the Longest Run of Ones in a Block, Serial test, Approximate Entropy test and Cumulative Sums test. These tests are briefly described in [1]. For the statistical randomness test, the p-value is supposed to be greater than 0.001 for the sequence bits to be considered as random. For example in the Frequency test, the p-value depends on the weight of the sequence bits.

We found out that the hybrid CA rules 126 and 30, and rules 166 and 30 (both with omega-flip permutation) do not appear to pass the frequency test for a few samples. Since they did not pass this test, we did not perform the remaining tests. However, hybrid rules 134 and 30 with omega-flip permutation passes all the seven statistical randomness tests. It is believed that any cryptographic algorithm that passes a frequency test could also pass the other tests. Due to massive results from the statistical tests, we only show the average result of p-values of the successful ones. The following table shows the p-values for all the seven tests and the average *p*¬-value for the hybrid CA rules 134 and 30 with omega-flip permutation.

Table 3: The statistical randomness result for hybrid CA rule 134 and rule 30 with omega-flip permutation

Test	*p*-value
Frequency	0.571
Approximate Entropy	0.355
Block frequency	0.855
Serial	0.498
Cumulative	0.857
Runs	0.205
Longest Runs	0.833
Average *p*-value	**0.596**

6. Conclusion

In this paper, we have studied and investigated a few selected CA rules that exhibit interesting behavior desirable in a cryptographic algorithm namely CA rules 30, 126, 134 and 166. Though rule 30 was shown not to be random enough but we manage to show that the combination of CA rule 30 and other non-linear CA rules with some permutations by omega-flip network can help randomize the bits and achieve a full avalanche. The hybrid CA rules 134 and 30 shows an ideal avalanche effect which is very closely to 0.5. Even if CA rules 134 and 30 run without our customized omega-flip permutation, they can still show almost good avalanche effect. We believe that with proper choosing of other permutation methods, it can help to minimize the implementation speed. We will further investigate CA rules 134 and 30 because they have initially showed a desired characteristic. We will also work further to investigate the potential of CA rules in performing permutation on their own to better perform both in hardware and software.

7. References

[1] A. Doganaksov, B. Ege, O. Kocak, F. Sulak. Statistical analysis of reduced round compression functions of SHA-3 second round candidates, *Cryptology ePrint Archive*, 2010.

[2] A. F. Webster and S. E. Tavares. On the design of s-boxes, Lecture Notes in Computer Science; 218 on *Advances in cryptology-CRYPTO 85*, New York, USA, pp. 523-534, Springer Verlag New York, Inc., 1986.

[3] A. Joux. Multicollisions in iterated hash functions, Advances in Cryptology, *proceedings of CRYPTO 2004*, Lecture Notes in Computer Science 3152, pp. 306C, Springer-Verlag, 2004.

[4] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. Handbook of applied cryptography, chapter Hash Functions and Data Integrity, pages 321-383. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

[5] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *NIST special publication 800-22 revision 1a*, 2001. The publication is available at http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf. Last access date: 20th January 2010.

[6] D. gage, E. Laub, and B. McGarry, Cellular automata: is rule 30 random?, preprint 2005.

[7] E. Andreeva, C. Bouillaguet, P. A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir, and S. Zimmer. Second preimage attacks on dithered hash functions, *Eurocrypt 2008*.

[8] H. M. Heys, and S. E. Tavares. Avalanche characteristics of substitution-permutation encryption networks. *IEEE Transaction on Computer*,vol. 44, no. 9, pp. 1131-1139, September 1995.

[9] I. Damgard. A design principle for hash functions. In *Advances in Cryptology-Crypto '89*, volume 435 of Lecture Notes in Computer Science, pages 416-427. Springer-Verlag, 1989.

[10] I. Damgard. Design principles for hash functions, in *Advances in Cryptology: Proceedings of Crypto '89*, 416-427, Springer-Verlag, 1990.

[11] J. Black, M. Cochran, and T. shrimpton. On the impossibility of highly-efficient block cipher-based hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of Lecture Notes in Computer Science, pages 526-541. Springer-Verlag, 22-26 May 2005.

[12] J. C. Jeon. Non-linear and non-group cellular automata chaining technique for cryptographic applications, *Journal of Mathematical and Computer Modelling*, vol. 51, pp. 995-999, April 2010.

[13] J. Daemen, R. Govaerts and J. Vandewalle. A framework for the design of one-way hash functions including cryptanalysis of Damgard's one-way function based on cellular automaton, *Advances in Cryptology-ASIACRYPT '91*, Lecture Notes in Computer Science, vol. 739, 1993.

[14] J. Liang, and X. Lai. Improved collision attack on hash function MD5, *Cryptology ePrint Archive 2005/425.*

[15] J. Randall and M. Szydlo. Collisions for SHA0, MD5, HAVAL, MD4, and RIPEMD, but SHA1 still secure. A technical note on the RSA website, August 2004. The note is available at http://www.rsasecurity.com/rsalabs/node.asp?id=2738. Last access date: 17th December 2007.

[16] K. Cattel and J. C. Muzio. Synthesis of one dimensional linear hybrid cellular automata, *IEEE Trans. On CAD*, 15:325-335, 1996.

[17] M. Mihaljevic, Y. Zheng, and H. Imai. A cellular automaton based fast one-way hash function suitable for hardware implementation, *1998 International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, Japan, Yokohama. Feb, 1998.

[18] M. Mihaljevic, Y. Zheng, H. Imai. A fast cryptographic hash function based on linear cellular automata over GF(q), *Mathematical Institute, Serb. Acad. Sci. & Arts*, Kneza Mihaila 35, Belgrade, Yugoslavia.

[19] O. Martin, A. M. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. Comm. Math. Phys., 93:219-258, 1984.

[20] P. D. Hortensius, R. D. McLeod, W. Pries, and H. C. Card. Cellular automata based pseudorandom number generators for built-in self test. IEEE Trans. On CAD, 8(8):842-859, August 1989.

[21] P Pal Chaudhuri, D Roy Chowdary, S. Nandi, and S. Chatterjee. Additive cellular automata – theory and applications, volume 1. *IEEE Computer Society Press*, California, USA, ISBN 0-8186-7717-1, 1997.

[22] R. B. Lee, Z. Shi, and X. Yang. Efficient permutation instructions for fast software cryptography, *IEEE Micro*, vol. 21, no. 6, pp. 56-69, Dec. 2001.

[23] R. C. Merkle, One way hash functions and DES. In *Advances in Cryptology*, proceedings of Crypto 1989, Lecture Notes in Computer Science 435, pp. 428C446, Springer-Verlag, 1990.

[24] R. Rivest. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force, April 1992. This RFC is available at http://www.faqs.org/rfcs/rfc1321.html. Last access date: 25th December 2007.

[25] S. Contini, and Y. L. Yin. Forgery and partial key recovery attacks on hmac and nmac using hash collisions. *12th International Conference on the Theory and Application of Cryptology and Information Security*, Advances in Cryptology – ASIACRYPT 2006, Shanghai, China, December 3-7, 2006, vol. 4284, pp. 37-53. Springer 2006.

[26] S. Tripathy, S., Nandi, S.: LCASE: Lightweight cellular automata-based symmetric key encryption. *International Journal of Network Security 8(2)*, 243-252 (2009).

[27] S. Wolfram. A new kind of science. Illinois: Stephen Wolfram LLC, 2002.

[28] S. Wolfram. Statistical mechanics of cellular automata. Rev. Mod. Phys., 55(3):601-644, July 1983.

[29] S. Wolfram. Universality and complexity in cellular automata. Physica D, 10:1-35, 1984.