Journal of Theoretical and Applied Information Technology
31st May 2021. Vol.99. No 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# A DESIGN OF A NEW HASH FUNCTION BASED ON CELLULAR AUTOMATA

[1]YOUSSEF SBAYTRI, [1] SAIIDA LAZAAR
[1]Mathematics, Computer Sciences and Applications Team (ERMIA) ENSA of Tangier, AbdelMaleek Essaadi University, Morocco
E-mail: [1]yusef.sbitri@gmail.com, [1]slazaar@uae.ac.ma

**ABSTRACT**

Cryptographic hash functions play an important role in information security. They are used in several cryptographic applications to verify the integrity and authenticity of data. The hash functions are also the basis of blockchain technology. Many hash function constructions are inspired by boolean functions. The proposed hash function algorithm in this paper is based on Elementary Cellular Automata (ECA) and 2-dimensional Cellular Automata (2DCA), which are another type of boolean functions that have excellent cryptographic properties. This algorithm has a sponge construction as such as SHA-3. The strict avalanche criterion (SAC) and NIST statistical tests suite (STS) were used to measure the security of this algorithm. The obtained results demonstrate that the proposed algorithm exhibit high sensitivity to input changes.

**Keywords:** *Cryptographic hash function, Blockchain, Sponge construction, Cellular Automata, Elementary Cellular Automata, Boolean function, NIST statistical tests suite, Avalanche Effect.*

## 1. INTRODUCTION

Cryptographic hash functions play an important role in modern cryptography since they are used for several applications such as integrity verification, message authentication, digital signatures, password storage and key derivation, and so on.

The fundamental idea of hash functions is that a hash-value (digest or digital fingerprint) of a short and fixed-length serves as a compact representative image (fingerprint) of any given message, the input message can be any type of data including character strings, binary files and TCP packets, images, etc. The hash function algorithms can be classified into keyed-hash functions and unkeyed-hash functions, the keyed-hash function need a secret key of fixed-size and a variable-length message to compute the message digest, these special digests are called Message Authentication Codes (MAC). However, the unkeyed hash function accepts a variable-length message as a single input and produce a fixed hash digest. In general, the term hash functions refer to unkeyed hash functions and the keyed hash functions are referred to as MAC.

Many hash structures have been proposed, but Merkle-Damgard construction and Sponge construction are the main used design in all NIST standardized hash functions. Both constructions based on a central component called a compression function, which consists of a series of mathematical operations (modular addition, bitwise operations, and boolean functions), Some compression functions utilize a block cipher as compression functions [1],[2], while other designs exploit the robust one-way characteristic of some mathematical problems, such as factorization problem [3], discrete logarithm problem [4], Another recent class of hash functions design is based on chaotic maps which possess high parameter sensitivity, random-like behavior and one-way computations [5],[6],[7],[8],[9],[10],[11],[12], these functions use a range of simple chaotic maps, high-dimensional chaotic maps, and spatio-temporal chaotic systems.

The first well-known hash function was the nominal MD2 (Message Digest 2), which was designed by Ron Rivest in 1989 [13]. It is the basis of all other hash functions of the MD family (MD4 [14], MD5 [15]), which are derived from Merkle-Damgard construction (Figure 1) [16], [17].

Afterwards, several hash functions with more security properties and with Merkle-Damgard

construction have been proposed such as SHA-1, MD5 and SHA-2. These later revealed weaknesses, which led them to cryptanalysis [18], [19], [20].

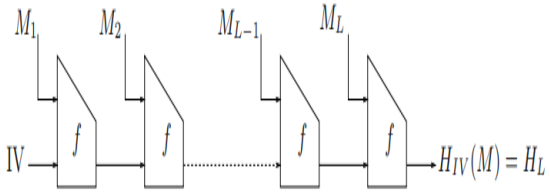


*Figure 1: Merkle-Damgard (MD) construction.*

In 2007, The National Institute of Standards and Technology (NIST) announced the competition of SHA-3. This competition ended in 2012, The Keccak design proposed by Bertoni et al was selected as a novel standard of SHA-3 [21], This algorithm was considered extremely secure due to its resistance against different cryptanalysis attacks. The Keccak design will be more detailed in section 4. In 2008, S. Nakamoto revealed the concept of blockchain [22]. Actually, this technology has many revolutionary applications [23]. The hash functions are very important to ensure the availability and security of the blockchain.

Any hash function ***H*** should have the following crucial properties:

- ✓ Pre-image resistance. Hash functions should be one-way functions, which means it's infeasible to find a pre-image of a digest (reverse a message digest ***y*** to recover the original message ***m*** where ***H(m) = y*** ).
- ✓ Second-pre-image resistance. This property means that, for a given message ***m*** and its digest ***y***, then it is computationally impossible to find a second message ***m'*** (***m = m'***) with the same message digest, i.e., ***H(m) = H(m')***.
- ✓ Collision resistance. It means that it is computationally impossible to find ***m*** and ***m'*** with ***m= m'*** where ***H (m) = H (m')***. A one-way function is both pre-image and second pre-image resistant. the collision resistance implies the second-pre-image resistance

The remainder of the paper is organized as follows: Section 2 gives a brief presentation of Cellular automata. Section 3 discusses the use of CA as a model for hash functions. In section 4 the Keccak hash functions structure is presented, Section 5 explains the detailed design of the proposed hash function algorithm. While section 6 discusses the security of our algorithm. Finally, the conclusion is presented in section 7.

## 2. CELLULAR AUTOMATA

Cellular automata (CA) were primarily proposed by Ulam and Von Neumann in 1940 to come up with a formal framework for studying the behaviour of complex systems [24]. They become more exploited in several fields due to their parallel structure, their simplicity and the wide potential that they offer for designing complex systems [25].

A cellular automaton is a discrete dynamical system where space, time, and the state are discrete. It can be represented by a finite or infinite grid of cells. Each cell in time ***t*** have a state, at time ***t +1*** the new state of a cell depends only on its old state, the states of its nearby neighbours at time t and according to a local rule**.** All cells on the grid are updated synchronously.

Mathematically, a cellular automaton ***A*** can be represented as a quadruple ***A = (S, N, d, f)*** where ***S*** is a finite set of possible states, ***N*** is the cellular neighbourhood, ***d*** $\epsilon$ $Z^+$ is the dimension of ***A***, ***f*** is the local cellular interaction rule (or transition function).

Elementary Cellular Automata (ECA) are the simplest one-dimensional CA with only two states {0;1}, two neighbours (left, right):
***d = 1, S = {0;1}; N = (-1;0;1)*** and ***f: $S^3 \rightarrow S$.***

In ECA, the local cellular interaction rule depends only on the nearest neighbor's states. As a result, the evolution of an ECA can be described by the next generation, based on the state of the cell on the left side, the own state of the current cell and the state of the cell to its right. Hence, there are 8 ($2^3$) possible binary states for the 3 cells neighboring to a given cell, Hence, there are 256 ($2^8$) possible ECA. Figure 2 shows the next generation of a central cell according to the ECA rule 30.

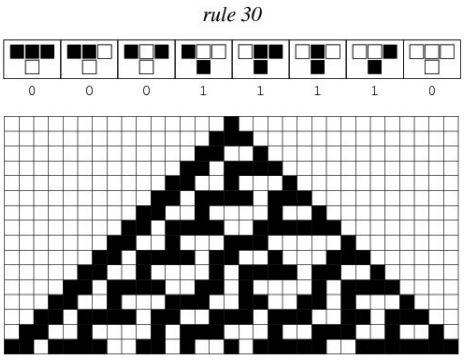


*Figure 2. Next State Configuration for ECA Rules 30*

In [25], [26] Wolfram has defined minimal Boolean formulas for ECA rules, which use the minimum possible number of logical operators $\neg, \wedge, \vee$ and $\oplus$ denoted NOT, AND, OR and XOR respectively. For example, the boolean formula for the rule-30 is $p \oplus (q \vee r)$ where p, q and r are the left neighbors, the right neighbor and the current cell sequentially.

The two-dimensional CA (2CA) have also the same properties as ECA, Von Neumann neighbourhoods (a) and Moore neighborhoods (b) (Figure 3) are the well-known neighborhoods of 2CA.

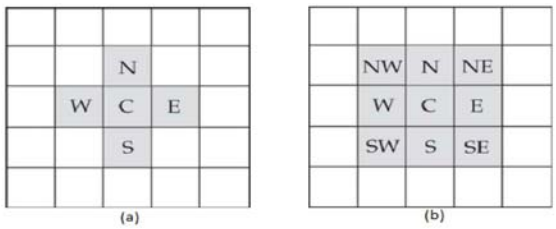


*Figure 3: Example of CA Neighborhoods.*

## 3. CA AND HASH FUNCTIONS: RELATED WORKS

Cellular automata have several properties that favour their ability to produce quite complex systems [27], [28] and [29]. Hence, these characteristics can be used as a basis for cryptographic algorithms design, such as cipher algorithms, PRNGs and hash functions [30], [31], and [32].

The idea of using CA for the hash design was first time proposed by Damgard in [33], where he exploited wolfram's CA-based PRNG design [34] to create a new compression function. However, these schemes were broken by Daemen [35], and he successively promoted two CA-based hash functions CellHash and SubHash [36], [37]. The two schemas were broken later by Chang [39]. In [38] J.C Jeon presented a scheme for hash function algorithm based on linear and nonlinear CA rules. Kuila et al. presented a hash function based on CA and inspired by the sponge construction [39]. This algorithm has the same security properties as well-known hash functions like SHA-3 [40] and SPONGENT [41]. Recently K. Rajeshwaran et al. proposed a CA-based Hashing algorithm (CABHA) by using CA rules 30 and 134 [42]. In [43] authors presented a scheme for a lightweight hash function inspired by sponge construction and based on linear and non-linear Cellular Automata.

## 4. KECCAK HASH FUNCTIONS

### 4.1 Sponge Construction

The Keccak algorithm is a hash function family based on sponge construction [44], [45]. The sponge function has its own state, which is a binary array of b-bits. Figure 4 exhibits the whole data structure used in Keccak sponge construction.

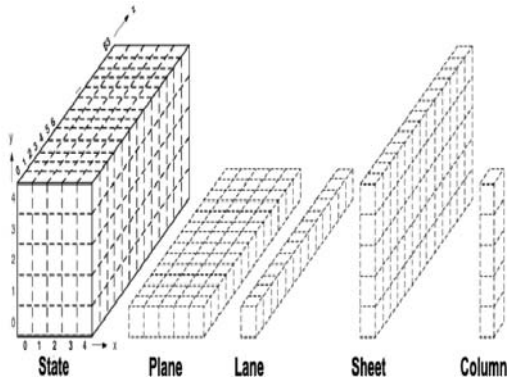


*Figure 4: State data structures used in Keccak.*

The state bits are divided into two parts: the outer part consisting of the first r bits and the inner part of c bits, where c is called the capacity and r is the bit rate, and with the condition $b = r+c$. The sponge construction, as shown in Figure 5, has two main phases absorbing and squeezing:

- ✓ **Absorbing phase**: In this phase, the message is divided into r-bits blocks, then a bitwise XOR is applied between the first block and the outer part of the initial 0-

state, the whole resulting state is processed by a permutation function ***f***. This process continues interlarding the bitwise-XOR operations with the implementation of the permutation function ***f*** for a fixed number of rounds. When all the input blocks are consumed, the sponge construction moves to the squeezing phase.

- ✓ **Squeezing phase**: This phase aims to generate a message digest of the desired length. Therefore, every generated r-bit block from the absorption phase is extracted and it undergoes a squeezed transformation to forms the final hash digest. If the length of the squeezed out bits is more than the required hash digest length, it must be truncated to match the needed length.

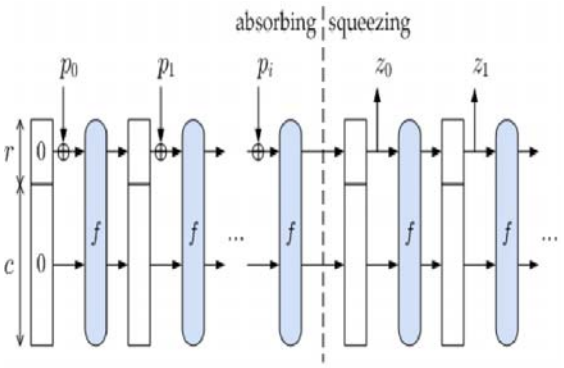


*Figure 5: The sponge construction.*

### 4.2 Keccak permutation functions

The permutation function is the main operation used in Keccak algorithm. It is represented by ***Keccak-f[b]***, where ***b ϵ {25, 50 100, 200, 400, 800, 1600}*** is the width of the permutation. These permutations are successive constructions including a series of a similar number of rounds ***nr***, as demonstrated in **Algorithm 1**. The number of the round depends on the permutation width ***b*** and is defined as ***nr = 12 + 2l***, where ***2l = b/25***.

**Algorithm 1** Keccak-$\int$ permutation

```
function KECCAK(state)
    for i ← 0 to n_r − 1 do
        state ← Round[b](state, RC[i])
    end for
    return state
end function
```

Each round in **Algorithm 1** consists of five steps denoted **Theta(θ), Rho(ρ), Pi(π), Chi(χ)** and **Iota(ι)** as indicated in the equations 1 to 6. Each operating on the state is organized as an array of 5×5x64 (64 is the bit-size of each lane in the case of b=1600bits).

**Theta($\theta$) Step**: $(0 \le x,y \le 4)$

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4] \quad (1)$$

$$D[x] = C[x-1] \oplus ROT(C[x+1],1) \quad (2)$$

$$A[x,y] = A[x,y] \oplus D[x] \quad (3)$$

**Rho($\rho$) and Pi($\pi$) Step**: $(0 \le x,y \le 4)$

$$B[y,2x+3y] = ROT(A[x,y],r[x,y]) \quad (4)$$

**Chi($\chi$) Step**: $(0 \le x,y \le 4)$

$$A[x,y] = B[x,y] \oplus ((NOT\, B[x+1,y]) AND\, B[x+2,y]) \quad (5)$$

**Iota($\iota$)**:

$$A[0,0] = A[0,0] \oplus RC \quad (6)$$

In the equation. (1), ***A*** represents the state array of 1600-bits and *A[x, y]* is a specific 64-bit lane in that state. ***B[x, y], C[x]*** and ***D[x]*** are transitional parameters. ***XOR AND*** and ***NOT*** represents bitwise logical operations. ***r*** and ***RC*** are specific constants. A more detailed description can be found in [44], [45] and [46].

## 5. PROPOSED HASH FUNCTION ALGORITHM

The proposed Hash function algorithm is inspired by Keccak design; it consists of two phases, Absorbing and Squeezing. In the absorbing phase, the input message of any size ***n*** is converted into a state of 1600-bits, while in the squeezing phase the resulting state transformed into a message digest of fixed size (192-bits or 256-bits). The two phases exploit a CA-based permutation function denoted ***CA_f***. Table 1 shows the required parameters of this algorithm.

*Table 1: Parameters of proposed CA-based hash Algorithm.*

| Sponge State (b) | Rate (r) | Capacity (c) | Hash length | Number of Rounds |
|---|---|---|---|---|
| 1600-bits | 832-bits | 768-bits | 192-bits | 4 |
| 1600-bits | 576-bits | 1024-bits | 256-bits | 4 |

### 5.1 Permutation function *CA_f*

This function consists of successive rounds (***nr = 4***). Each round calls the function ***Round_CA***, which uses as entry a sponge state of size 1600-bits, the state is converted to a 3D array of dimension 5x5x64-bit words. The main process of the function ***Round_CA*** starts by browsing the whole array. Next, in each iteration, a special mechanism (in a specific iteration, two neighbours from eight is selected, which gives $A_8^2 = 56$ possibilities) based on the Moore Neighbours (Figure 3 (b)) was defined to determine the left and the right neighbors for each lane. After that, a series of boolean functions (the boolean formula of ECA, - as shown in Table 2- ***30, 146, 22, 105, 105, 146, 22, 146, 105, 126, 22, 146*** respectively) was applied between the value of the current lane - denoted center- and their temporary neighbor's lanes (left; right).

The last operation on each iteration is the application of ECA rule 30; **Algorithm 2** presents the structure of the ***Round_CA*** function.

**Algorithm 2** the *Round_CA*

```
1: function Round_CA(state)
2:     for x ← 0 to 5 do
3:         for y ← 0 to 5 do
4:             center ← state[x][y]
5:             left, right ← Get_Neighbours(state, x, y)
6:             center ← Boolean_Formula(left, center, right)
7:             state ← ECA30(center)
8:         end for
9:     end for
10:    return state
11: end function
```

Table 2 describes the boolean formulas of the ECA rules used in our algorithm. The symbols $\neg$, $\wedge$, $\vee$ and $\oplus$ represent NOT, AND, OR and XOR boolean operators respectively.

*Table 2: The Boolean Formula of ECA rules*

| ECA rule | The Boolean formula of the ECA rule |
|---|---|
| 30 | left $\oplus$ (center $\vee$ right) |
| 146 | left$\oplus$(center$\vee$right)$\wedge$center$\oplus$right |
| 22 | left $\oplus$ left $\wedge$ center $\wedge$ right $\oplus$ center $\oplus$ right |
| 105 | left $\oplus$ center $\oplus \neg$right |
| 126 | left $\oplus$ center $\vee$ left $\oplus$ right |

## 6. SECURITY ANALYSIS

### 6.1 The Avalanche effect and Strict Avalanche Criterion

While defining a new cryptographic algorithm it is necessary to measure whether the system reaches a certain optimum level of security or not, the avalanche effect and the strict avalanche criterion are good cryptographic properties widely used to prove the security of any cryptographic algorithm [47], [48]. These criteria are interesting because, statistical-based cryptanalysis such as linear and differential cryptanalysis are related to them [49], [50], [51].

The avalanche effect measures the percentage of changed bits in the output message (digest in the case of a hash function) when changing one bit in the input message. If this percentage is equal to 0%, this means that the hash output does not change. When the input varies by a single bit. However, if it's 100%, it means that the bit is bound to reverse on change of input by a single bit.

The strict avalanche criterion (SAC) is a generalization of the avalanche effect [52], [53]. A cryptographic algorithm is said to satisfy the SAC, when a single bit change between two input messages $\boldsymbol{m}$ and $\boldsymbol{m_0}$ produces a percentage of changed bits around 50% in the output messages $\boldsymbol{d}$ and $\boldsymbol{d_0}$. The SAC test aims to measure if any flipped bit of the input message bit affects the digest bits as if it is a random mapping. If the SAC is not satisfied then the probability of a successful attack on the hash algorithm increases considerably.

To perform these tests, 1000 messages of size 64-bits and 8192-bits respectively were generated,

for each series of messages, if we considered a generated message $m_0$ of size n-bits, so a copy of $m_0$ is kept unchanged, $n-1$ one-bit flipped messages $m_1$, $m_2$,..., $m_n$ are deducted from $m_0$, next we calculate the hash-digests of $h_0$, $h_1$, $h_2$,..., $h_n$ of messages $m_0$, $m_1$, $m_2$,..., $m_n$ respectively, the last step is the determination of avalanche effect value between $h_0$ and their correspondent hash-digests $h_1$, $h_2$,..., $h_n$. These steps were performed to the 1000-messages of size 64-bits and 8192-bits. The Figure 6 and 7 illustrate the results of the SAC tests.

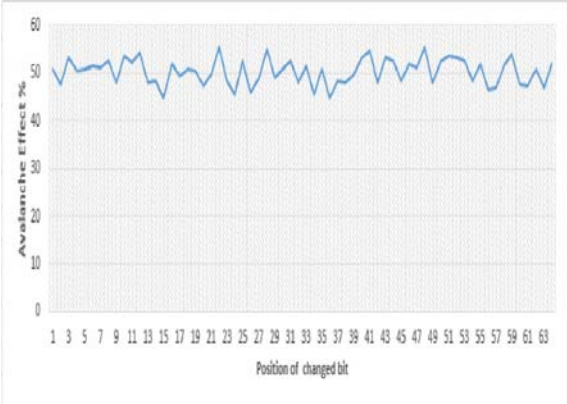


*Figure 6: Average avalanche effect % versus one-bit-flipped alterations to an input message of 64-bits.*

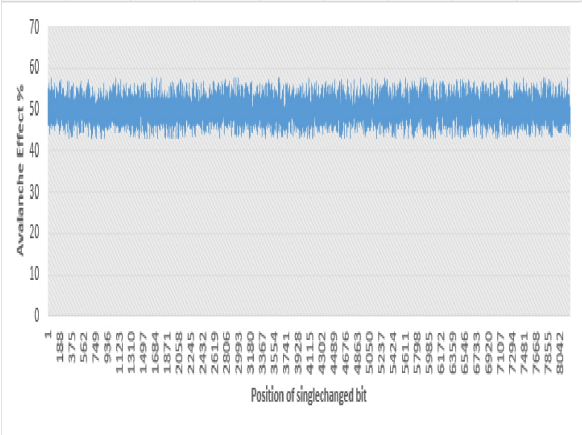


*Figure 7: Average avalanche effect % versus one-bit-flipped alterations to an input message of 64-bits.*

The obtained results in Figure 6 and 7 demonstrate that for both series of the message of size 64-bits and 8192-bits the avalanche effect values due to only one-bit flip are concentrated around 50%, which means the proposed hash function algorithm provides a good avalanche effect criterion, which is one of the most important features of secure cryptographic hash functions.

### 6.2 NIST Statistical Test

The statistical randomness tests are very useful to study the security of cryptographic algorithms. One of the well-known tools used for this purpose is the NIST statistical test suite (STS), the STS was developed by NIST and was used to select the AES and SHA-3 algorithms. It consists of 15 main statistical tests; a more detailed description of these tests can be found in [54]. Each test generates a "p-value" to verify the randomness of tested bit sequences.

A significance level denoted $\alpha$ ($\alpha$ = 0.001) is used to validate the randomness of the generated bit-sequence, if the obtained ***p-value*** greater than $\alpha$, the bit-sequence is considered as random with a trust-coefficient of 99.9%. Otherwise, if the ***p-value*** $< \alpha$, the sequence is considered non-random.

The main goal of this test is to ensure that the proposed algorithm has a property that the redundancies at the input don't leak any information in the output. To reach this purpose we construct three data sets of different structures as defined below.

- ✓ Low-density bit-sequence (**LD seq**): The low-density bit-sequence is a binary sequence of random length (between 64-bits and 1024-bits), it is formed by one bit equal '1' and the rest equal '0'.
- ✓ High-density bit-sequence (**HD seq**): This binary sequence has the same characteristic as the low-density sequence, except for the distribution of '1' and '0', which is the opposite of the low-density sequence.
- ✓ Random bit-sequence (**Rand seq**): The random bit-sequence is a binary sequence generated randomly; it contains a random distribution between '0' and '1'.

For the three types of data, 400000 bit-sequences was prepared, which makes up a total of

1200000 bits tested in this experiment. Table 3 shows the obtained results by the NIST STS tool. We remark that the generated p-value by the 15 tests are greater than the supposed value of α = 0.001, which means that our algorithm succeeds the NIST statistical randomness tests.

*Table3: NIST Test Results of our hash function algorithm.*

| Test Name | P-Value | | | Inter-pretation |
|---|---|---|---|---|
| | LD seq | Rand seq | HD seq | |
| Frequency Test | 0.23 | 0.99 | 0.65 | Pass |
| Frequency Test within a Block | 0.64 | 0.18 | 0.69 | Pass |
| Run Test | 0.18 | 0.51 | 0.98 | Pass |
| Longest Run of Ones in a Block | 0.69 | 0.05 | 0.97 | Pass |
| Binary Matrix Rank Test | 0.71 | 0.86 | 0.94 | Pass |
| Discrete Fourier Transform Test | 0.55 | 0.89 | 0.21 | Pass |
| Non-Overlapping Template Matching Test | 0.93 | 0.67 | 0.74 | Pass |
| Overlapping Template Matching Test | 0.89 | 0.08 | 0.85 | Pass |
| Maurer's Universal Statistical test | 0.83 | 0.89 | 0.76 | Pass |
| Linear Complexity Test | 0.07 | 0.18 | 0.65 | Pass |
| Serial test | 0.32 | 0.29 | 0.36 | Pass |
| Approximate Entropy Test | 0.35 | 0.89 | 0.28 | Pass |
| Cumulative Sums Test | 0.41 | 0.77 | 0.15 | Pass |
| Random Excursions Test | 0.43 | 0.77 | 0.38 | Pass |
| Random Excursions Variant Test | 0.52 | 0.71 | 0.64 | Pass |

### 6.3 Resistance against Brute Force Attacks

The security of the proposed hash function algorithm is more reliant on the internal permutation function ***CA_f*** presented in sub-section 5.1. This function is composed of a series of non-linear boolean functions and bit-operations, which produces a strong dependence between the input message bits because, in each round, a lane has a dependency on other 56 neighbor lanes. The application of ECA-rule 30 creates a strong reliance between every bit and their left and right neighbors, thence, the resistance to pre-image, $2^{nd}$ pre-image and collision attacks can be more robust.

The proposed length for the hash digest is also important because for a hash value of size n (192-bits; 256-bits), $2^n$ ($2^{192}$; $2^{256}$ ) operations are needed to find a pre-image or a $2^{nd}$ pre-image and $2^{n/2}$ ($2^{96}$ ; $2^{128}$) operations are required to find a collision.

## 7. CONCLUSION

In this paper, we proposed a new cryptographic hash function based on irreversible cellular automata of one and two dimension. The structure of the proposed scheme is inspired by Keccak sponge construction. Our proposed algorithm succeeded the SAC and NIST statistical tests; it also demonstrated a high resistance against pre-image and $2^{nd}$ preimage collision attacks. Our future work will be focused on strengthening our designed hash function by conducting advanced cryptanalysis tests.

## REFERENCES

[1] Preneel B, Govaerts R, and Vandewalle J. (1993). Hash Functions Based on Block Ciphers: A Synthetic Approach. In Crypto '93, volume 773 of LNCS, pages 368–378. Springer-Verlag.

[2] Black J, Rogaway P, and Shrimpton T. (2002). Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Crypto'02, volume 2442 of LNCS, pages 320–335. Springer-Verlag.

[3] Lenstra, A.K.; Stam, M.; Page, D.; Discrete logarithms variants of VSH, Proceedings Vietcrypt 2006, LNCS 4341, 229–242, Springer-Verlag 2006.

[4] Buchmann J and Paulus S. (1997). A One Way Function Based on Ideal Arithmetic in Number Fields. In Crypto '97, volume 1294 of LNCS, pages 385–394. Springer-Verlag.

[5]. Kanso, A., Ghebleh, M.: A fast and efficient chaos-based keyed hash function. Commun. Nonlinear Sci. Numer. Simul. 18(1), 109–123 (2013)

[6]. Kanso, A.,Ghebleh,M.:Astructure-based chaotic hashing scheme. Nonlinear Dyn. 81(1–2), 27–40 (2015)

[7]. Li, Y., Xiao, D., Deng, S.: Keyed hash function based on a dynamic lookup table of functions. Inform. Sci. 214(23), 56–75 (2012)

[8]. Teh, J.S., Samsudin, A., Akhavan, A.: Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dyn. 81(3), 1067–1079 (2015)

[9]. Xiao, D., Liao, X., Deng, S.: Parallel keyed hash function construction based on chaotic maps. Phys. Lett.A 372(26), 4682–4688 (2008)

[10] M. Ahmad, S. Khurana, S. Singh, and H. D. AlSharari, ``A simple secure hash function scheme using multiple chaotic maps," 3D Res., vol. 8, no. 2, p. 13, Jun. 2017.

[11] M. Todorova, B. Stoyanov, K. Szczypiorski, and K. Kordov, ``SHAH: Hash function based on irregularly decimated chaotic map," 2018.

[12] J. S. Teh, M. Alawida, and J. J. Ho, ``Unkeyed hash function based on chaotic sponge construction and fixed-point arithmetic," Nonlinear Dyn., vol. 100, pp. 713_729, Feb. 2020.

[13] Kaliski, B.: RFC 1319 - The MD2 Message-Digest Algorithm. RSA Laboratories (April 1992).

[14] R.L. Rivest, The MD4 message digest algorithm, in Advances in Cryptology—CRYPTO '90, Proceedings, eds. by A. Menezes, S.A. Vanstone. Lecture Notes in Computer Science, vol. 537 (Springer, Berlin, 1991), pp. 303–311.

[15] R.L. Rivest, The MD5 Message-Digest Algorithm, April 1992. Network Working Group, Request for Comments: 1321.

[16] R.C. Merkle, One-way hash functions and DES, in Advances in Cryptology—CRYPTO '89, Proceedings, ed. by G. Brassard. Lecture Notes in Computer Science, vol. 435 (Springer, Berlin, 1990), pp. 428–446

[17] I. Damgard, A design principle for hash functions, in Advances in Cryptology—CRYPTO '89, Proceedings, ed. by G. Brassard. Lecture Notes in Computer Science, vol. 435 (Springer, Berlin, 1990), pp.416–427

[18] Szydlo, M.: SHA-1 collisions can be found in 263 operations. Crypto BytesTechnical Newsletter (2005)

[19] Xiaoyun Wang, X.L., Feng, D., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, pp. 1–4 (2004), http: // eprint.iacr.org/2004/ 199

[20] Stevens, M.: Fast collision attack on MD5. ePrint-2006-104, pp. 1–13 (2006), http: //eprint.iacr.org/2006/104.pdf

[21] National Institute of Standards and Technology (NIST). SHA-3 Winner announcement, http://www.nis.gov/itl/csd/sha-100212.cfm

[22] Nakamoto S. (2008). Bitcoin: a peer-to-peer electronic cash system. Accessed at http://bitcoin.org/bitcoin.pdf

[23] Priteshkumar Prajapati, Krutarth Dave, Dr. Parth Shah:" A Review of Recent Blockchain Applications", INTERNATIONAL JOURNAL OF SCIENTIFIC and TECHNOLOGY RESEARCH VOLUME 9, ISSUE 01, JANUARY 2020.

[24] J. VON NEUMANN. Theory of Self-Reproducing Automata. University of Illinois Press, Illinois, 1966. Edited and Completed by A. W. Burks.

[25] S. Wolfram, "A new kind of science," Champaign, IL: Wolfram Media, Inc., pp. 55, 2002.

[26] The Wolfram Atlas of Simple Programs website; http://atlas.wolfram.com/01/01/

[27] Nandi, S., B.K. Kar, and P. Pal Chaudhuri. 'Theory and Applications of Cellular Automata in Cryptography'. IEEE Transactions on Computers 43, no. 12 (December 1994): 1346–57.

[28] Wolfram, Stephen. 'Cryptography with Cellular Automata'. In Advances in Cryptology—CRYPTO '85 Proceedings, edited by Hugh C. Williams, 429–32. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1986.

[29] Jarkko Kari," Theory of cellular automata: A survey", 0304-3975$ - see front matter © 2005 Elsevier B.V. All rights reserved. doi: 10.1016/j.tcs.2004.11.021

[30] Sbaytri, Youssef, and Saiida Lazaar. 'A Lightweight Cellular Automata-Based Cryptosystem Evaluated with NIST Statistical Tests'. In Advanced Intelligent Systems for Sustainable Development (AI2SD'2019), 1105:21–29. Advances in Intelligent Systems

and Computing. Cham: Springer International Publishing, 2020.

[31] Sbaytri, Youssef, Saiida Lazaar, Hafssa Benaboud, and Said Bouchkaren. 'A New Secure Cellular Automata Cryptosystem for Embedded Devices'. In Mobile, Secure, and Programmable Networking, 11557:259–67. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019.

[32] Manzoni L, Mariot L (2018) Cellular Automata Pseudo-Random Number Generators and Their Resistance to Asynchrony. In: Hoshi M, Seki S (eds) Developments in Language Theory. Springer International Publishing, Cham, pp 428–437.

[33] I. B. Damgarrd, "A Design Principle for Hash Functions", proceeding of Crypto'89, LNCS 435, (1989), pp. 416-442.

[34] S. Wolfram, "Random Sequence Generation by Cellular Automata", Advanced in Applied Mathematics, Academic Press, Orlando, 1986, Vol. 7, pp. 123-169.

[35] 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 a Cellular Automaton", proceeding of Asiacrypto'91, LNCS 739, (1993), pp. 82-96.

[36] J. DAEMEN, R. GOVAERTS, J. VANDEWALLE, A hardware design model for cryptographic algorithms,Computer Security – ESORICS 92. Proc. Second European Symposium on Research in Computer Security, LNCS 648, (1992) pp. 419–434. Springer-Verlag.

[37] D. Chang, Preimage attacks on CellHash, SubHash and strengthened versions of CellHash and SubHash,IACR Cryptology ePrint Archive 2006: 412, 2006.

[38] J.C. Jeon, —One-way hash function based on cellular automata,IT Convergence and Security 2012 Lecture Notes in Electrical Engineering, pp. 21–28, Nov. 2012

[39] S. Kuila, D. Saha, M. Pal, and D. R. Chowdhury, CASH: Cellular automata based parameterized hash, Security, Privacy, and Applied Cryptography Engineering Lecture Notes in Computer Science, pp. 59–75, 2014.

[40] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, The making of KECCAK, Cryptologia, vol. 38, no. 1, pp. 26–60, Feb. 2014.

[41] A. Bogdanov, M. Kneˇzevi´c, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, SPONGENT: A lightweight hash function, Cryptographic Hardware and Embedded Systems – CHES 2011 Lecture Notes in Computer Science, pp. 312–325, 2011

[42] Rajeshwaran, Kartik, and Kakelli Anil Kumar. 'Cellular Automata Based Hashing Algorithm (CABHA) for Strong Cryptographic Hash Function'. In 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 1–6. Coimbatore, India: IEEE, 2019.

[43] Zhang, Xing, Qinbao Xu, Xiaowei Li, and Changda Wang. 'A Lightweight Hash Function Based on Cellular Automata for Mobile Network'. In 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), 247–52. Shenzhen, China: IEEE, 2019. https://doi.org/10.1109/MSN48538.2019.00055.

[44] G. Bertoni, J. Daemen, M. Peeters, and G. Assche, "The Keccak reference," Submission to NIST (Round 3), January, 2011.

[45] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak sponge function family main document," Submission to NIST (Round 2), 2009.

[46] Pub NF, FIPS PUB (2015) 202. SHA-3 standard: permutation based hash and extendable-output functions Federal Information Processing Standards Publication.

[47] 33. H. Feistel, "Cryptography and computer privacy," Sci. Am. 228, 15-23 (1973).

[48] A. Kumar, N.amita Tiwari, "Effective Implementation and Avalanche Effect of AES." 2013 International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, 2012, pp 31-35

[49]. C. Blondeau and K. Nyberg, "New links between differential and linear cryptanalysis," Lect. Notes Comput. Sci. 7881, 388-404 (2013). 36. V. Goyal, A. O'N

[50]. S. Gupta and S. Yadav, "Performance analysis of cryptographic hash functions," Int. J. Sci. Res. 4, 2319-7064 (2015).

[51]. I. Moon, F. Yi, Y. H. Lee, and B. Javidi, "Avalanche and bit independence characteristics of double random phase encoding in the fourier and fresnel domains," J. Opt. Soc. Am. A 31, 1104-1111 (2014).

[52] J. C. H. CASTROA, J.M. SIERRAB, A. SEZNECA, A. IZQUIERDOA, A.

RIBAGORDAA, The strict avalanche criterion randomness test. Mathematics and Computers in Simulation, 68, (2005) pp. 1–7.

[53] R. FORRE, The strict avalanche criterion: spectral properties of booleans functions and an extended definition. Advances in cryptology, in: Crypto'88, Lecture Notes in Computer Science, 403, (S. GOLDWASSER, Ed.), (1990) pp. 450–468. Springer-Verlag.

[54] L. E. Bassham et al.," A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications - NIST" Special Publication (NIST SP) - 800-22 Rev 1a, Sep. 2010.