

# An optimized 128-bit cellular automata-based hash function for authentication of data at rest and in transit

Surendra Kumar Nanda<sup>1</sup>, Suneeta Mohanty<sup>2</sup>, Prasant Kumar Pattnaik<sup>2</sup>

<sup>1</sup>Department of Computer Science, C. V. Raman Global University, Bhubaneswar, India

<sup>2</sup>School of Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University, Bhubaneswar, India

## Article Info

### Article history:

Received Apr 21, 2022

Revised Sep 22, 2022

Accepted Oct 18, 2022

### Keywords:

Cellular automata

Cryptography

Data encryption

High-performance computing

Message authentication

Reversible cellular automata

## ABSTRACT

The cryptographic hash functions are the most fundamental cryptographic concept. These functions are used as basic building blocks for digital signatures and message authentication. Boolean functions are the core of hash functions. These functions are expected to provide pseudo-randomness as well as input sensitivity. Cellular automata are a form of Boolean function that exhibits strong cryptography properties as well as chaotic behavior. This paper proposes a hash function, designed on the principle of cellular automata. The proposed algorithm is secure and meets the requirements for a successful hashing scheme. The hash function has strong statistical and cryptographic characteristics, according to the findings of the avalanche test and the National Institute of Standards and Technology (NIST) Statistical Test Suite. The modularity of different operations of this algorithm makes it suitable for a high-capacity processing environment to produce efficient performance.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Suneeta Mohanty

School of Computer Engineering, Kalinga Institute of Industrial Technology Deemed to be University

Bhubaneswar, Odisha 751024, India

Email: suneetamohanty@gmail.com

## 1. INTRODUCTION

Authentication of data is one of the very basic needs in cryptographic applications. The hash functions are used for many years to achieve that. Merkle [1] is credited with the original formulation of one-way hash functions as well as a method for generating them from random block ciphers. All techniques rely on the development of a fixed and finite-size “building block” function, which may accept any size input, and then creating the true one-way hash function from this “building block”.

Naor and Yung [2] confirmed security results in the context of polynomial-time reducibility. So far, different hash functions based on the techniques by Damgard [3] and Matyas *et al.* [4] have been devised. Hammad *et al.* [5] provided a detailed survey on lightweight cryptographic hash functions. Most of the hash functions focused on the security aspect but the efficiency and ease to compute are also desirable properties. To achieve this efficiency, we may design a hash function that will utilize the existing processing capability optimally. The cellular automata can be used for this purpose because of their inherent parallel computation capabilities.

Wolfram [6] was the first to introduce cellular automata (CA). He creates a cellular automation-based pseudo-random number generator. Szaban and Seredynski [7], [8] proposed a block encryption system using cellular automata and the design of an S-box using cellular automata and improving S-box performance. Hanin *et al.* [9] presented a lightweight cellular automata-based hash algorithm (L-CAHASH). In comparison to well-known lightweight hash algorithms, this technique provides good randomness quality and quick software hashing. The findings reveal that it fits the security requirements of radio-frequency

identification (RFID) tags. Mihaljević *et al.* [10] presented a one-way hash function that is highly adapted to small and rapid hardware implementation, which is appealing to developing smart cards for security. The cellular automata-based parameterized hash algorithms (CASH) were suggested by Kuila *et al.* [11].

Rajeshwaran and Kumar [12] present hashing technique using 1D cellular automata to generate a strong hash function. Sadak *et al.* [13] offer a novel hashing approach that incorporates both linear and non-linear cellular automata principles, as well as a salt addition mechanism to defend the hashing algorithm from cryptanalytic assaults. Jamil *et al.* [14] present a unique omega-flip permutation approach to boost diffusion in small iterations. The results show that various cellular automata rules combined with a unique omega-flip permutation technique provide a desirable SAC in a minimal number of iterations and pass all the statistical tests. John *et al.* [15] show how to create a basic hash function using cellular automata and sponge functions. Manuceau [16] created cryptographic algorithms using a 2D cellular automaton based on Conway's Game of Life. Machicao and Bruno [17] propose a chaotic CA-based encryption method using different cellular automata rules and producing a good degree of performance.

Das and Ray [18] presented a programmatically configurable parallel block encryption algorithm. Roy and Nandi [19] presented a symmetric key cryptography algorithm using cellular automata. Rani and Sasamal [20] presented showcase applications of quantum-dot cellular automata, subsequently this is extended by Qadri *et al.* [21]. Su *et al.* [22] and Naskar *et al.* [23] presented an image encryption technique using the tent map technique. Kumar and Raghava [24] presented substitution-box-based image encryption.

Wang *et al.* [25] designed a new circuit based on a quantum-dot CA-based programmable logic array. The comparison with other existing algorithms suggests better efficiency of the quantum dot-based CA algorithm. Reyes *et al.* [26] modelled a complex adaptive network using CA and Li *et al.* [27] designed an integrated circuit using quantum dot which consumes less power and has better operational speed. Bykov [28] uses a modified Wolfram's rule 184 to simulate a heterogeneous traffic flow for a smart city. A new technique is proposed by Angulo *et al.* [29] to reduce noise in a digital image. The experimental result shows this technique is also effective for reducing salt and pepper noises. Alexan *et al.* [30] proposed an image encryption technique using CA and Lorenz systems. The result shows similar performance to other researchers' techniques, but it requires less processing time. Mohanty *et al.* [31] represent a cloud-based implementation of privacy preservation.

We propose a 128-bit cellular automata-based hash function (CABHF-128), an optimized 128 bits cryptographic hash function based on cellular automata principles in this paper. This hash function uses linear and non-linear rules for message digest generation. CABHF-128 demonstrates good pseudorandom behavior, according to the National Institute of Standards and Technology (NIST) Statistical Test Suite. CABHF-128 displays pseudorandom behavior, statistical independence between the input and output, and algorithm sensitivity to changes in the input, according to these tests.

The work is organized in 5 sections. Section 2 provides the theoretical basis for the method of cellular automation and cryptographic hash function. We have presented our proposed work in section 3, which is to develop a new cellular automaton-based cryptographic hash function. The discussion of the findings obtained is in section 4. Lastly, the work's future scope is discussed in section 5.

## 2. THEORETICAL BACKGROUND

Wolfram describes the first examples of cellular automata. They were very simple to implement and had a basic structure. Cellular automata were the subject of many scientific papers during the last decades. The study of cellular automata was carried out by various researchers from different disciplines. In this work, we use a cellular automata-based hash function with a block size of 128 bits. We have used 7 cellular automata-based rules which are one-directional and simple to implement. These rules satisfied the basic principle of hash functions. It is explained in detail in subsection 2.1.

### 2.1. Cryptographic hash function

In cryptography, cryptographic hash functions are commonly employed. A hash function accepts any number of bits as input and returns an n-bit integer as an output. The one-way hash function should maintain collision resistance property for its generated output. Furthermore, the hash function must produce a truly unique, unrelatable result when given two messages with slight differences. If a hash function with the following three features is termed a strong hash function.

A hash function should compress data while also being simple to implement. A hash function received any length input and converts it into a fixed-length output. Furthermore, it must fulfill the following fundamental security standards to be labelled as "cryptographic".

It is difficult to find the message  $M1'$  such that  $h(M1')=k$ , given the hash value  $k=h(M1)$ . Given  $M1$ , it is hard to find  $M1'$  such that  $h(M1')=h(M1)$ . It is hard to find  $M1$  and  $M1'$  such that  $h(M1)=h(M1')$ .

**2.2. Cellular automaton**

A discrete parallel computation model built of a finite array of  $n$  cells is known as a one-dimensional cellular automaton. In a discrete-time  $t$ , each cell communicated with its neighbors. Each center cell  $x$  updates its state  $s_1 \in \{0, 1\}$  by applying a local rule and a radius ( $r$ ). For radius  $r=1$  the neighborhood consists of a total of  $2*r+1$  numbers of cells, including the central cell  $x$ . If the  $radius=1$  then it is called one-dimensional cellular automaton, the neighborhood consists of a central cell and one cell each toward the left and right. Figure 1 depicts the transition of states in finite-size cellular automata with periodic boundaries. The center cell's transition from stage 0 to stage 1 is determined by the stage 0 neighborhood and a transition function known as the "cellular automata rule". The pseudo-random number is generated by Wolfram using cellular automata rule 30 with  $radius (r)=1$ . The working procedure of cellular automaton with periodic boundary using rule 90 and  $radius=1$  is shown in Figure 1. Figure 2 represents the time-space diagram for rule 90.

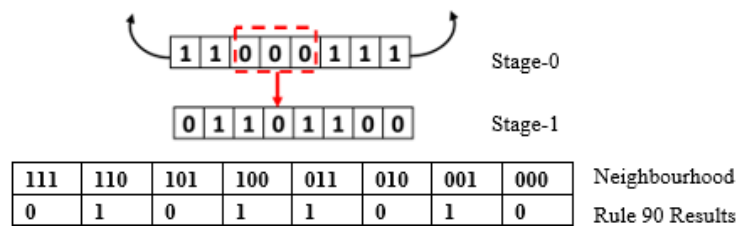


Figure 1. Cellular automaton rule 90 with Radius=1

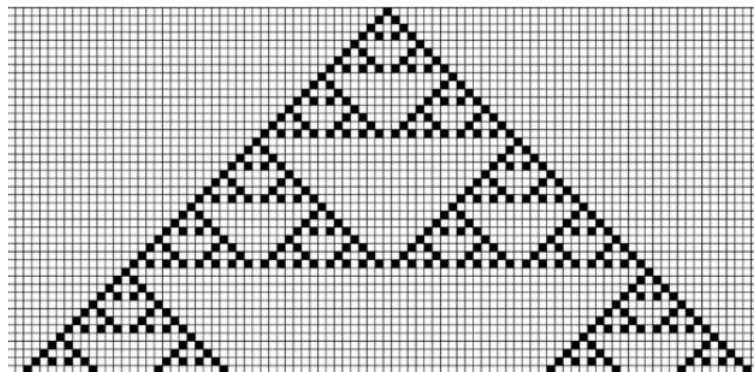


Figure 2. Time-space diagram representation of rule 90

**3. METHOD**

In this section, the working process of the proposed algorithm is described. This algorithm produces a message digest of 128 bits and is known as CABHF-128. The CABHF-128 broadly consists of two phases. Pre-processing of the message occurs in the first phase. The pre-processed message is then transformed into a message digest through the hash generation phase. The hash generation phase uses a hash function based on cellular automata rules to produce a message digest. A detailed description of both phases of CABHF-128 is presented in the following subsections.

**3.1. Pre-processing phase**

The pre-processing phase is designed to divide the message into multiple batches so that each batch can be processed parallelly without depending on the other batch. It may happen that the last batch will be incomplete, in that case, we will do padding of additional bits to make it complete. After splitting and padding, we will perform XOR with seed to add confusion to the message and prepare for the hash generation phase.

**3.1.1. Splitting and padding**

In the first phase of pre-processing an arbitrary length of a message is split into  $N$  batches. The batches are consisting of 128 blocks of 128 bits each. If the last batch is incomplete, then additional bits are padded to it to make it complete and multiple of 128.

The CABHF-128 employs a padding technique in which a single “1” bit is padded at the end of the final data bit and a sequence of “0” bits. We perform this because each block should have an equal number of bits. If the data blocks are incomplete, then only we will add padding. These padding bits will be dropped during decryption.

### 3.1.2. Initial transformation

In the second phase of pre-processing, each batch of the message is applied to an initial transformation function which performs an XOR operation on a block of 128 bits of message with a seed value. The seed for the first block is a random number of 128 bits. The subsequent blocks use the hash value generated by the previous block. The initial transformation function can be represented as:

$$h_0 = f(\text{block}_0, \text{initial seed})$$

$$h_1 = f(\text{block}_1, h_0)$$

$$h_n = f(\text{block}_n, h_{n-1})$$

### 3.2. Hash generation phase

In the hash function generation phase, a cellular automata-based hash function is applied to the output generated by the initial transformation. Here, we used a cellular automata rule based on the block number. The rule selection criteria are block number modulo 7. The output of this phase is the message digest.

The cellular automata-based hash function uses one-dimensional CA rules with radius 1 with periodic boundary. The selected 8 CA rules are arranged in the following order with a *hash\_ruleset*.

$$\text{hash\_ruleset} = \{30, 90, 150, 45, 135, 120, 165, 195\}.$$

The message digest will be obtained as:

$$\text{Message Digest}_i = f(\text{CA}_{\text{select}}, \text{seed}, h_i)$$

$\text{CA}_{\text{select}}$  is the selected rule for that block from *hash\_ruleset*. The concatenation of all individual message digests will generate the final message digest.

### 3.3. CABHF-128 algorithm

The algorithm is broadly divided into 2 phases. The pre-processing phase helps in the splitting of a message into equal size blocks. The block generation steps are followed by the initial transformation of data. The hash generation phase converts the initial transformed data to a message digest. All the message digests are combined to get the final message digest.

Algorithm 1. Pre-processing phase

- Step 1 : Split the message of size M into N batches of 2 KB (16385 bits) each.
- Step 2 : Split each batch into 128 blocks that contain 128 bits.
- Step 3 : If required add padding to the last batch. The padding will start from a ‘1’ bit followed by multiple ‘0’ bits.
- Step 4 : Apply the initial transformation function to each block of 128 bits of data from a batch as follows:
  - If Block number=0
    - $h_0 = \text{block}_0 \text{ XOR initial seed}$
  - else
    - $h_i = \text{block}_i \text{ XOR } h_{i-1}$
- Step 6 : Concatenate  $h_0$  to  $h_{127}$  to generate the initial transformed value for one batch
- Step 7 : Repeat Step 1 to Step 6 for all batches
- Hash generation phase:
- Step 8 : Repeat the following activities for  $i$  from 0 to 127 (for each Batch)
  - Find  $\text{CA}_{\text{select}}$  from *hash\_ruleset*. Where  $\text{select} = i \text{ MOD } 7$ . Calculate  $\text{Message Digest}_i = f(\text{CA}_{\text{select}}, \text{seed}, h_i)$
- Step 9 : Concatenate all message digests to form message digest for the batch.
- Step 10 : Repeat Step 8 and Step 9 for each batch and concatenate all outputs to find the final message digest.

### 3.4. Block diagram of CABHF-128 algorithm

The block diagram of CABHF-128 algorithm is represented in Figure 3. The block diagram summarizes the different operations of the algorithm. All operations of the algorithm are broken into 3 categories. The first one is splitting and padding to convert data into equal size blocks. The second phase is to apply the initial seed and transformation function. The last phase is to produce a message digest.

## 4. RESULTS AND DISCUSSION

We have executed the proposed algorithm and HCAHF proposed by Sadak *et al.* [13] in an high-performance computing (HPC) cluster having 96 cores and 64 GB RAM. The results of the implementation of both algorithms are compared in subsection 4.3. The result shows proposed algorithm produces better execution performance. The results of avalanche tests and the NIST test on the proposed cellular automata-based hash function method are presented in this section. Passing these tests implies that the algorithm has a high level of security, but it does not mean that it is immune to different attacks. Statistical independence between input and output is the most critical aspect of any hash function.

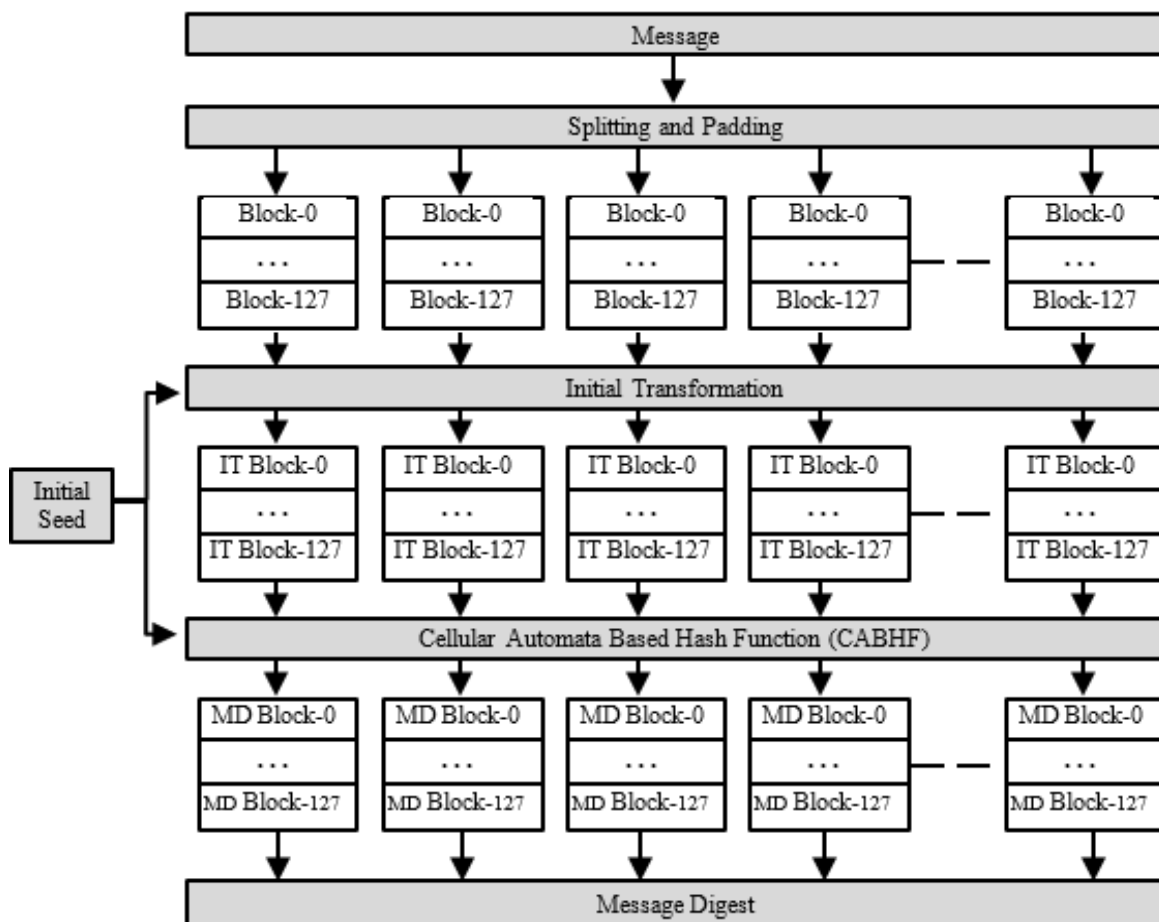


Figure 3. Block diagram of CABHF-128 algorithm

### 4.1. Avalanche test

The avalanche test states that a minor modification in input value generates a substantial difference in the derived output. An algorithm that shows a good avalanche effect normally produces a hamming distance between the message and modified message at least half of the message digest size. A hash function is highly non-linear if it shows an avalanche effect.

To test the avalanche test for CABHF-128, 50 messages of 2 KB each were used. The CABHF-128 generates 2 message digests, one for the original message *Msgi* and one for the corresponding 1-bit change replica. The hamming distance between hash values is then determined.

$$Hamming\_distance(f(Msg_i), f(Msg_{ireplica}))$$

The result obtained from the avalanche test shows the average avalanche value concentrated around 51. The detailed result is represented in Figure 4. This shows that CABHF-128 generates a good avalanche effect and is hence statistically independent concerning its input values.

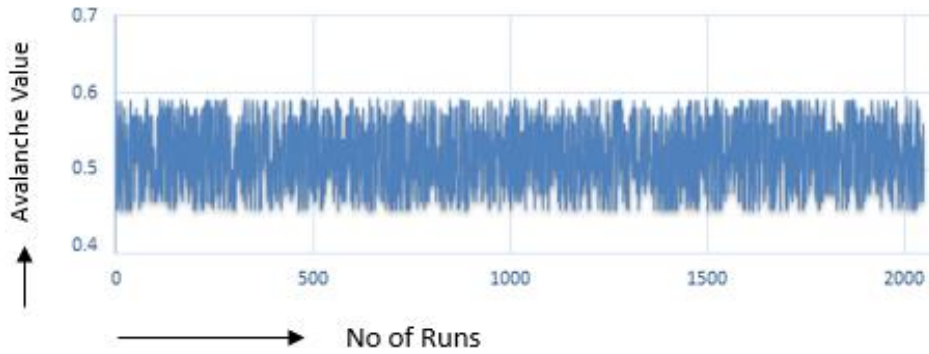


Figure 4. Result of avalanche test

**4.2. NIST statistical test**

A hash function should produce good pseudorandom behavior. To test this pseudorandom behavior of CABHF-128 results, NIST Statistical Test Suite is used. It calculates the p-values and verifies the randomness. To perform NIST statistical test suite, a message digest of size 15 MB is used. This message digest is produced by the CABHF-128 algorithm. The result obtained passed this test because all values are present within the range of 0.001 to 0.999. The detailed results are represented in Table 1.

Table 1. Results of NIST statistical test

Name of the Test	p-value	Status
Frequency	0.5412	Success
Block frequency	0.5343	Success
Cumulative frequency	0.5327	Success
Runs	0.4951	Success
Longest run	0.5100	Success
Fourier transform	0.5210	Success

**4.3. Performance evaluation**

We used 15 MB of data to produce a message digest using both algorithms and measures results for 50 concurrent runs. The results show that CABHF-128 produces better performance as compared to the HCAHF algorithm. The results of 50 runs are represented in Figure 5 and Table 2.

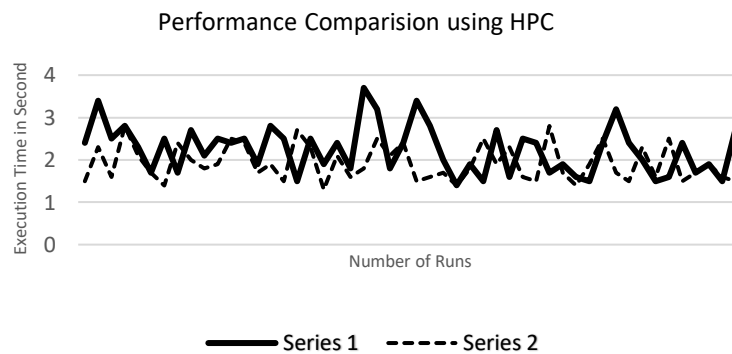


Figure 5. Execution time comparison between HCAHF and CABHF-128

Table 2. Execution time of HCAHF and CABHF-128

Runs	HCAHF	CABHF-128	Runs	HCAHF	CABHF-128
1	2.4	1.5	26	3.4	1.5
2	3.4	2.3	27	2.8	1.6
3	2.5	1.6	28	2	1.7
4	2.8	2.8	29	1.4	1.4
5	2.3	2.1	30	1.9	1.8
6	1.7	1.7	31	1.5	2.5
7	2.5	1.4	32	2.7	1.9
8	1.7	2.4	33	1.6	2.3
9	2.7	2	34	2.5	1.6
10	2.1	1.8	35	2.4	1.5
11	2.5	1.9	36	1.7	2.8
12	2.4	2.5	37	1.9	1.7
13	2.5	2.4	38	1.6	1.4
14	1.9	1.7	39	1.5	1.9
15	2.8	1.9	40	2.4	2.5
16	2.5	1.5	41	3.2	1.7
17	1.5	2.7	42	2.4	1.5
18	2.5	2.3	43	2	2.3
19	1.9	1.3	44	1.5	1.6
20	2.4	2.1	45	1.6	2.5
21	1.8	1.6	46	2.4	1.5
22	3.7	1.8	47	1.7	1.7
23	3.2	2.5	48	1.9	1.9
24	1.8	2.1	49	1.5	1.6
25	2.4	2.4	50	2.7	1.5

#### 4.4. Security analysis

It is not straightforward to prove a cryptographic algorithm's resistance to attacks targeted at it formally. To perform a brute force attack on the hash function, someone requires  $2^n$  trials to determine the original message  $Msg$  from a given hash value this means in the case of CABHF-128 the attacker required  $2^{128}$  operations to successfully decode it. Therefore, CABHF-128 provides a good level of defense against brute force attack

One can target the hashing algorithm through cryptanalytic attacks. The use of a random number of 128 bits as the initial seed during the initial transformation and hash generation stages makes it very difficult for the attacker to perform cryptanalytic attacks. Again, the use of linear and non-linear cellular automata rules during hashing operations prevents the cryptanalytic attack. The algorithm uses both confusion and diffusion properties of basic cryptography through XOR operation and hash generation making it difficult to break this hash function.

## 5. CONCLUSION AND FUTURE WORK

This paper proposes cellular automata based on a new cryptographic hash function. The CABHF-128 algorithm is designed to produce a message digest of 128 bits at a time. This algorithm consists of a pre-processing stage and a hash generation phase. The input data is then padded using the message digest padding technique and then splits into N batches that contain 128 blocks of 128 bits each. The initial transformation function is applied to different batches of messages to complete the pre-processing stage. An initial seed, which is a random number, is used during the initial transformation function as well as in the hash generation stage. The hash generation function selects a CA rule from the *hash\_ruleset* and applies it with the block of data to generate a message digest. The CABHF-128 uses the XOR operator and cellular automata to provide confusion and diffusion qualities. The avalanche test indicates an average of 51% of the avalanche effect and is hence statistically independent concerning its input values. The NIST statistical test suite confirms that CABHF-128 shows good pseudorandom behavior. The performance comparison of the CABHF-128 with HCAHF in an HPC environment shows this algorithm produces better performance than HCAHF. These experiments reveal that CABHF-128 exhibits pseudorandom behavior, statistical independence between the input and output, and algorithm sensitivity to changes in the input. In the future, the CABHF-128 can be implemented to authenticate data at rest. This cryptographic hash function can also be used extensively in blockchain technology for authentication. As cellular automata are inherently parallel computing systems, this algorithm can be optimized to provide optimal performance for high-performance computing systems.

## REFERENCES




- [1] R. C. Merkle, "One-way hash functions and DES," in *Advances in Cryptology-CRYPTO' 89 Proceedings*, 1990, pp. 428–446, doi: 10.1007/0-387-34805-0\_40.
- [2] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing-STOC '89*, 1989, pp. 33–43, doi: 10.1145/73007.73011.
- [3] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology-CRYPTO' 89 Proceedings*, vol. 435, New York, NY: Springer New York, 1990, pp. 416–427.
- [4] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Technical Disclosure Bulletin*, vol. 27, no. 10, pp. 5658–5659, 1985.
- [5] B. T. Hammad, N. Jamil, M. E. Rusli, and M. R. Z'aba, "A survey of lightweight cryptographic hash function," *International Journal of Scientific and Engineering Research*, vol. 8, no. 7, pp. 806–814, 2017.
- [6] S. Wolfram, "Cryptology with cellular automata," in *Advances in Cryptology-CRYPTO '85 Proceedings*, vol. 218, Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 429–432.
- [7] M. Szaban and F. Seredynski, "Cellular automata-based S-Boxes vs. DES S-Boxes," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5698, Springer Berlin Heidelberg, 2009, pp. 269–283.
- [8] M. Szaban and F. Seredynski, "Improving quality of DES S-boxes by cellular automata-based S-boxes," *The Journal of Supercomputing*, vol. 57, no. 2, pp. 216–226, Aug. 2011, doi: 10.1007/s11227-010-0398-y.
- [9] C. Hanin, B. Echandouri, F. Omary, and S. El Bernoussi, "L-CAHASH: a novel lightweight hash function based on cellular automata RFID," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10542, Springer International Publishing, 2017, pp. 287–298.
- [10] M. Mihaljević, Y. Zheng, and H. Imai, "A cellular automaton based fast one-way hash function suitable for hardware implementation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1431, Springer Berlin Heidelberg, 1998, pp. 217–233.
- [11] S. Kuila, D. Saha, M. Pal, and D. R. Chowdhury, "CASH: cellular automata based parameterized hash," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8804, Springer International Publishing, 2014, pp. 59–75.
- [12] K. Rajeshwaran and K. Anil Kumar, "Cellular automata based hashing algorithm (CABHA) for strong cryptographic hash function," in *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Feb. 2019, pp. 1–6, doi: 10.1109/ICECCT.2019.8869146.
- [13] A. Sadak, F. Ezzahra, B. Echandouri, C. Hanin, and F. Omary, "HCAHF: a new family of CA-based hash functions," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, pp. 502–510, 2019, doi: 10.14569/IJACSA.2019.0101267.
- [14] N. Jamil, R. Mahmood, M. R. Z'aba, N. I. Udzir, Zuriati, and A. Zukarnaen, "A new cryptographic hash function based on cellular automata rules 30, 134 and omega-flip network," in *International Conference on Information and Computer Networks (ICIN 2012)*, 2012.
- [15] A. John, A. Reji, A. P. Manoj, A. Premachandran, B. Zachariah, and J. Jose, "A novel hash function based on hybrid cellular automata and sponge functions," in *Advances in Intelligent Systems and Computing*, Springer Nature Singapore, 2022, pp. 221–233.
- [16] V. Manuceau, "About a fast cryptographic hash function using cellular automata ruled by far-off neighbours," *International Journal of Engineering Trends and Technology*, vol. 69, no. 2, pp. 39–41, Feb. 2021, doi: 10.14445/22315381/IJETT-V69I2P206.
- [17] J. Machicao and O. M. Bruno, "A cryptographic hash function based on chaotic network automata," *Journal of Physics: Conference Series*, vol. 936, no. 1, Dec. 2017, doi: 10.1088/1742-6596/936/1/012058.
- [18] D. Das and A. Ray, "A parallel encryption algorithm for block ciphers based on reversible programmable cellular automata," Jun. 2010, *arXiv:1006.2822*.
- [19] S. Roy, S. Nandi, J. Dansana, and P. K. Pattnaik, "Application of cellular automata in symmetric key cryptography," in *International Conference on Communication and Signal Processing*, Apr. 2014, pp. 572–576, doi: 10.1109/ICCSP.2014.6949906.
- [20] S. Rani and T. N. Sasamal, "A new clocking scheme for quantum-dot cellular automata based designs with single or regular cells," *Energy Procedia*, vol. 117, pp. 466–473, Jun. 2017, doi: 10.1016/j.egypro.2017.05.172.
- [21] S. U. R. Qadri, Z. A. Bangi, M. T. Banday, G. M. Bhat, and M. R. Beigh, "A novel comparator-a cryptographic design in quantum dot cellular automata," *International Journal of Digital Signals and Smart Systems*, vol. 4, 2020, doi: 10.1504/IJDSS.2020.106078.
- [22] Y. Su, Y. Wo, and G. Han, "Reversible cellular automata image encryption for similarity search," *Signal Processing: Image Communication*, vol. 72, pp. 134–147, Mar. 2019, doi: 10.1016/j.image.2018.12.008.
- [23] P. K. Naskar, S. Bhattacharyya, D. Nandy, and A. Chaudhuri, "A robust image encryption scheme using chaotic tent map and cellular automata," *Nonlinear Dynamics*, vol. 100, no. 3, pp. 2877–2898, May 2020, doi: 10.1007/s11071-020-05625-3.
- [24] A. Kumar and N. S. Raghava, "An efficient image encryption scheme using elementary cellular automata with novel permutation box," *Multimedia Tools and Applications*, vol. 80, no. 14, pp. 21727–21750, Mar. 2021, doi: 10.1007/s11042-021-10750-1.
- [25] R.-J. Wang, Y.-P. Xu, C. She, and M. Nasution, "A new design for programmable logic array based on QCA-based nanotechnology," *Optik*, vol. 253, Mar. 2022, doi: 10.1016/j.ijleo.2022.168581.
- [26] L. Reyes and D. Laroze, "Cellular automata for excitable media on a complex network: the effect of network disorder in the collective dynamics," *Physica A: Statistical Mechanics and its Applications*, vol. 588, Feb. 2022, doi: 10.1016/j.physa.2021.126552.
- [27] Y. Li, F. Peng, G. Li, and G. Xie, "A crucial step of quantum-dot cellular automatic placement and routing," in *2022 International Conference on Power Energy Systems and Applications (ICoPESA)*, Feb. 2022, pp. 96–100, doi: 10.1109/ICoPESA54515.2022.9754396.
- [28] N. V. Bykov, "Cellular automata simulation of heterogeneous freeway traffic flow in a smart city," in *4<sup>th</sup> International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, Mar. 2022, pp. 1–5, doi: 10.1109/REEPE53907.2022.9731490.
- [29] K. V. Angulo, D. G. Gil, and H. E. Espitia, "Modeling and numerical validation for an algorithm based on cellular automata to reduce noise in digital images," *Computers*, vol. 11, no. 3, Mar. 2022, doi: 10.3390/computers11030046.
- [30] W. Alexan, M. ElBeltagy, and A. Aboshousha, "Lightweight image encryption: cellular automata and the Lorenz system," in *International Conference on Microelectronics (ICM)*, Dec. 2021, pp. 34–39, doi: 10.1109/ICM52667.2021.9664961.






- [31] S. Mohanty, P. K. Pattnaik, and G. B. Mund, "Privacy preserving auction based virtual machine instances allocation scheme for cloud computing environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 5, pp. 2645–2650, Oct. 2017, doi: 10.11591/ijece.v7i5.pp2645-2650.

## BIOGRAPHIES OF AUTHORS






**Surendra Kumar Nanda**    received the M.Tech. Degree in Computer Science Engineering from Biju Pattnaik University of Technology, Odisha, 2011. Currently, he is pursuing his Ph.D. from Kalinga Institute of Industrial Technology (KIIT) Deemed to be University, Odisha, India. His research interests include cryptography, network security, cellular automata, and data privacy. He can be contacted at [situnanda@gmail.com](mailto:situnanda@gmail.com).



**Suneeta Mohanty**    has 14 years of teaching experience in computer science and engineering. She earned her M.Tech. degree in computer science and engineering from College of Engineering and Technology which is a constituent college of BPUT, Odisha. She earned her Ph.D. from KIIT Deemed to be University, Odisha. She has been a life member of ISCA, ISTE, and IET. Her research contribution includes 02 co-edited books published by Springer Nature, more than 45 research publications in reputed conferences, book chapters, and journals indexed in Scopus/SCI/Web of Science. She can be contacted at [suneetamohanty@gmail.com](mailto:suneetamohanty@gmail.com).



**Prasant Kumar Pattnaik**    (Senior Member, IEEE) received his Ph.D. degree in computer science. He is currently working as a professor with the School of Computer Engineering, KIIT University, Bhubaneswar, India. He has more than a decade of teaching and research experience. He has published a number of research papers in peer-reviewed international journals and conferences. His research interests include mobile computing, cloud computing, brain-computer interface, and privacy preservation. He is a fellow of IETE. He can be contacted at [patnaikprasant@gmail.com](mailto:patnaikprasant@gmail.com).