IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 7, 2021, accepted September 15, 2021, date of publication September 17, 2021,
date of current version September 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3113855

Keyed Parallel Hash Algorithm Based on Multiple Chaotic Maps (KPHA-MCM)

ROAYAT ISMAIL ABDELFATAH, ESRAA ABDELKHALEK BAKA, AND MOHAMED E. NASR, (Life Senior Member, IEEE)
Department of Electronics and Electrical Communications Engineering, Faculty of Engineering, Tanta University, Tanta 31527, Egypt
Corresponding author: Roayat Ismail Abdelfatah (royat_esmaeel@f-eng.tanta.edu.eg)

ABSTRACT Hash functions are considered as the core of cryptography, which are the basic technique used for data security. Cryptographic hash functions also can be used to achieve the integrity of large data such as the data stored in a hard disk and set of financial data. So, in the era of large data and increasing capacity of data in applications, fast hash schemes with parallel operations are extremely desirable which effectively increases the computational speed. Functions of random behavior like chaotic maps are used in hash functions to generate the fixed length message digest from the original message. This paper proposes a parallel hash algorithm based on multiple chaotic functions by mixing logistic map, tent map, and sine function. In the proposed scheme the structure of coupling lattice is changed and using diamond lattice as new structure. This algorithm is flexible to generate 128, 256 or longer hash value. The simulation analysis such as hash distribution, key sensitivity, confusion and diffusion statistical properties, and collision resistance are executed. The results demonstrated that the proposed hash is an efficient, simple and fast algorithm comparing with some recent hash algorithms based on chaotic maps.

INDEX TERMS Hash function, multiple chaotic function, security, cryptography, tent map.

I. INTRODUCTION

A hash function must be a one-way function that means the hash can convert any length of message to fixed length. Hash function classified into unkeyed and keyed hash functions. Unkeyed hash need no keys and this type includes traditional hash function such as MD5, SHA-1, SHA-2, and SHA-3. Keyed hashing needs two inputs; message and secret keys [1], [2]. The hash function must have the following qualities in order to be an efficient cryptographic algorithm: 1) Pre-image resistance (one-way feature): this indicates that it is difficult to reverse the hash function computationally. To put it another way, for any hash function H that outputs hash value D, it must be a difficult procedure to discover an input value X that hashes to D. This feature makes the algorithm more robust against the attacker that has only the hash value and trying to find the input. 2)Second pre-image resistance (weak collision resistance): This indicates that given input and its associated hash, getting -another input with the same hash would be impossible. In other words, for a given input H(X) that is the hash value of an input X, it should be difficult to identify any other input value Y such that H(Y) = H(X). This feature keeps the algorithm safe against the attacker who possesses the input and its hash digest and aims to find another input that can be replaced instead of the original input. 3) Collision resistance (strong collision resistance): this property indicates the difficulty of finding two different inputs that hashes to the same digest. In other meaning, this property indicates the difficulty of finding two different inputs that hashes to the same digest. In other meaning, it would be difficult to find two inputs X and Y that obtain H(X) = H(Y). This collision-free feature guarantees that these collisions are difficult to detect, and makes it highly difficult for the attacker to get two distinct input values with the same hash digest. Furthermore, if a hash function assumes the collision-resistant characteristic, it also assumes the second pre-image resistant property. In cryptography, a keyed hash function h_k must possess some properties; a) for any given h_k and input x, it is easy to generate the output y. b) when y is given without knowing the key k, it is hard to find x. c) when x is given without knowing k, it is hard to find y [3]. The keyed hash function can be indicated as Message Authentication Code (MAC) and is aimed to use for authentication and integrity [4], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Yanjiao Chen.

Cryptographic Hash Functions have several applications in term of network security. It's used to achieve some of security objectives such as message authentication, entity authentication, message integrity, and also used to execute digital signatures (non-repudiation). **Message Authenticity and Integrity:** Verifying the authenticity and integrity of data is the major necessity in network and computer systems. These properties are the most important application of the hash functions. These properties can achieve by using Message Authentication Code (MAC), and Hashed Message Authentication Code (HMAC). **Digital Signature:** With the rapid revolution of digital data communication and data storage, digital data can be copied, stored, modified and transported easily than analog data. These desired features are very valuable although having several security issues; so, digital dealing is considered unreliable when authentication, privacy, and data integrity are important concern unless some security processing are applied to it. There are several field such as contracts, transactions, and comparable others where users are concerned about stealing data and unauthorized access. Traditional signatures can't solve this issue as they are contained in the document as an integral part of it. Digital Signature is able to solve these issues as when a document is signed digitally, the signature is send as a separated document. The receiver gets the message and signature, and apply a technique to verify the authentication of the of message and signature. Digital Signature guarantee the data privacy and prevent unauthorized people to access. Currently, digital signature is used in several applications that include: *Government data, Financial transactions, Banking services, Business to business (B2B), and Healthcare*. **Password Verification:** Commonly, password verification is depending on cryptographic hash functions. Storing the passwords of users as clear text can lead to a massive breach in security issues if the file of passwords is exposed. In order to reduce this risk, the hash digest of passwords only stored instead of all the password. When authenticate a user, the user presents his password and then this password is converted to its corresponding hash digest. The digest is compared with the stored one. The original passwords can't be regenerated from the stored hash digest. **Wireless Sensor Network:** In the recent, Wireless Sensor Networks (WSNs) is considered a fast growing and exciting field that has considerable attention by researchers. This has been prompted by recent incredible technology developments in the production of low-cost sensor equipment with wireless network interface. The development of large-scale sensor networks with hundreds to thousands of sensor nodes brings up a slew of technological obstacles as well as limitless application opportunities. Sensor networks have several applications in various domains including: *Military Applications, Monitoring the Environmental Conditions, Environmental Applications, Health Care Applications, Home Intelligence, Agriculture, Industrial Process Control, and Structural Monitoring.* **Blockchain Technology:** Cryptography is the backbone of blockchain technology; without it, the technology would be unable to link each block of data together in order to form a secure, immutable chain. In the Blockchain, this is two types of cryptographic schemes; asymmetric-key algorithms, and hash functions are used to achieve the functionality of a single view of blockchain to every participant. Hash functions play an important role in connecting blocks and ensuring the integrity of the data stored within each block. Any change in the block data might cause inconsistency and disrupt the blockchain, rendering it invalid. This criterion is met by a characteristic of hash functions known as the avalanche effect. According to this, when even a minor adjustment is performed to the hash function's input, a completely unrelated output is returned compared to the original output. The Blockchain has several applications that include: *Bitcoin, Voting, Intellectual Property Rights, Banking and Financial Data, Storing Federal Government Data, and Healthcare*.

Hash function attacks have never ended. For example, the conventional hash function such as MD-5 has an attack investigated in 2008 that broke the security of the construction [6]. SHA-0 and SHA-1 have a theoretical weakness that was founded in 2005 [7], [8]. Another collision in SHA-1 is founded by Google in 2017 [9]. For SHA-2, there are limitations in the structure [4] such as multi-collision attacks [10], herding attacks [11], and second pre-images attacks [12]. In 2014, The Keccak algorithm has been chosen by NIST as the standard of SHA-3 as Secure Hash Standard [13]. SHA-3 which is based on sponge structure demonstrated increased resistance to attacks related to hashing. But the possibility of detecting attacks on SHA-3 is investigated in [14], [15]. As a result, conventional hash algorithms are considered to have some inherent weaknesses. This promotes researchers to implement alternatives with simple computations for effective hash functions.

Chaos-based cryptosystems structures provide high statistical performance and efficient attack resistance. Chaotic maps have inherent characteristics such as random behavior, one-way feature, high sensitivity to parameters, and requested confusion and diffusion [16]. These characteristics boost generating effective and secure hashes. In recent years, researchers suggested several hash algorithms based on chaotic maps using various methods. Teh *et al.* [17] invented keyed hash scheme based on logistic map and representation of fixed point, and in [18] proposed an unkeyed hash function depending on sponge construction with arithmetic of fixed-point that solved the issue of excessive computational complexity of the floating-point representation. Liu *et al.* [2] designed keyed hash function with perturbation of time-varying parameters and system of hyperchaotic. Li and Ge [3] investigated a parallel hash scheme based on lattices of cross-coupled map. Ahmad *et al.* [4] used multiple chaotic maps to propose a simple and secure hash scheme.

[19] presented keyed hash algorithm proposed by Liu *et al.* that based on one dimensional enhanced quadratic chaotic map (EQM) with using varying parameters. In [20], Liu *et al.* invented hash function algorithm by constructing a 3D exponent chaotic map (3D-ECM) and using parallel impulse

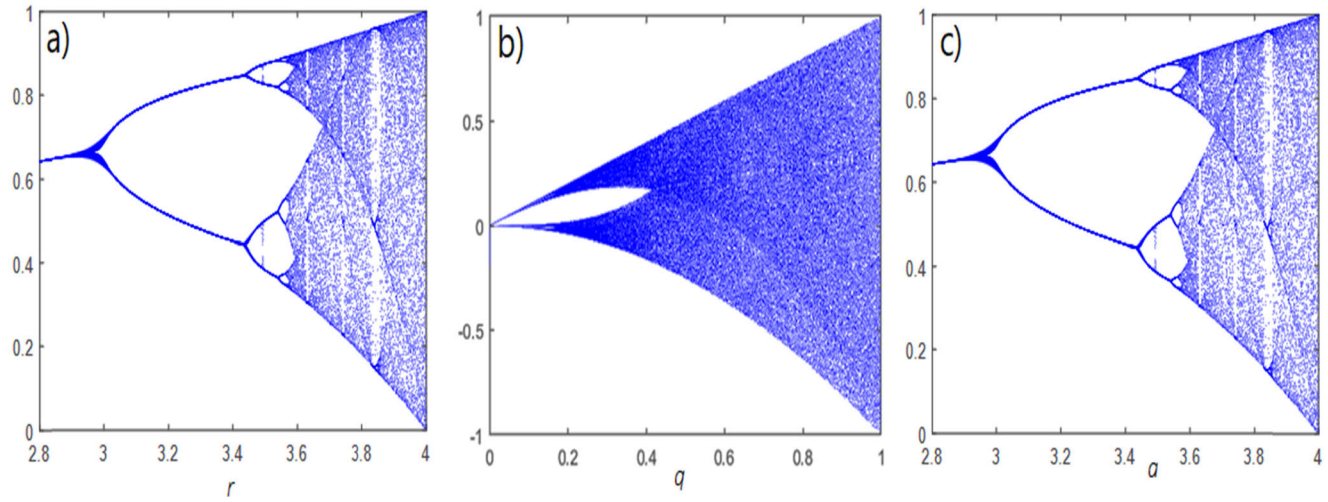

FIGURE 1. Bifurcation diagrams of a) logistic map b) tent map c) sine map.

perturbation. Alawida *et al.* [21] designed a chaos-based hash function that based on a structure known as the deterministic chaotic finite state automata (DCFSA), and another hash function that depend on a chaotic sponge structure and DNA sequence [22].

The hash schemes based on chaotic systems are mostly based on one or two chaotic maps. In this paper, a unique medley of multiple chaotic maps is examined to build an efficient, secure, and simple hash function. The major contribution in this paper contains the following:

(a) A secure hash algorithm is exhibited using combined chaotic maps of the tent, sine, and logistic map.

(b) Enlarge the secret keys size as a result of using multiple chaotic functions.

(c) Change the structure of coupling lattice and using diamond lattice as a new structure with keeping the ease of computations.

(d) Using parallel iterative structure to increase the speed advantage when handling large files.

After the introduction, the paper will be arranged as follow; Sect II includes an introductory of the proposed scheme and definition of used chaotic maps, Sect. III illustrates the proposed hash function, Sect IV investigated the theoretical analysis and computer simulation results, Sect V showed comparative analysis with some recent chaos-based hashes, Sect.VI presented the conclusion.

II. PRELIMINARIES

Chaos is a random long-period behavior in a dynamical system which has good response to initial conditions variation. In this section, chaotic maps used in this suggested scheme are briefly illustrated.

1) Logistic map: It is a simple and widely 1D chaotic map used in cryptography and characterized by low computations. It is defined in mathematical as

$$x_{i+1} = rx_i(1 - x_i), \tag{1}$$

where $x_i \in [0, 1]$ is the chaotic system state and $r \in [0, 4]$ is the control parameter. The bifurcation map behavior with parameter r is displayed in Figure 1a. The behavior of logistic map is depending on its control parameter that can be analyzed by using Lyapunov exponent. When λ is positive value, this means the chaotic maps has chaotic behavior, and for negative λ or zero, the Lyapunov exponent estimates non-chaotic behavior. The Lyapunov exponent of the logistic map as a function of its control parameter r displayed that the logistic map has aperiodic behavior when $r \in [3.57, 4]$.

2) Tent map: It is 1D chaotic map, and it is a piecewise function that defined as

$$x_{i+1} = f(x) = \begin{cases} qx_i, & x_i < 0.5 \\ q(1 - x_i), & x_i \geq 0.5, \end{cases} \tag{2}$$

where $x_i \in [0, 1]$ is the chaotic system state and $q \in [0, 2]$ is the control parameter. The bifurcation behavior of tent map with its parameter r is plotted in Figure 1b. The Lyapunov exponent of the tent map that depends on control parameter q shown that the tent map has chaotic behavior when $r \in [1, 2]$.

3) Sine map: It is a simple 1D chaotic map and its output falls between [0,1]. Mathematically, sine map is defined as

$$x_{i+1} = \sin(\pi x_i). \tag{3}$$

The bifurcation map behavior is displayed in Figure 1c.

III. DESCRIPTION OF PROPOSED ALGORITHM

In this section keyed parallel hash scheme based on multiple chaotic maps is described. The algorithm description will be composed of message pre-processing, initiate control parameters and variables, message handling, and produce the hash value. The whole structure of the hash algorithm is shown in Figure 2. Where M_i indicates to i^{th} message block, A refers to the first logistic map used to generate the initial values of tent map in the next stage, F_1 is the tent map, F_2 is the sine map, and F_3 is the logistic map.

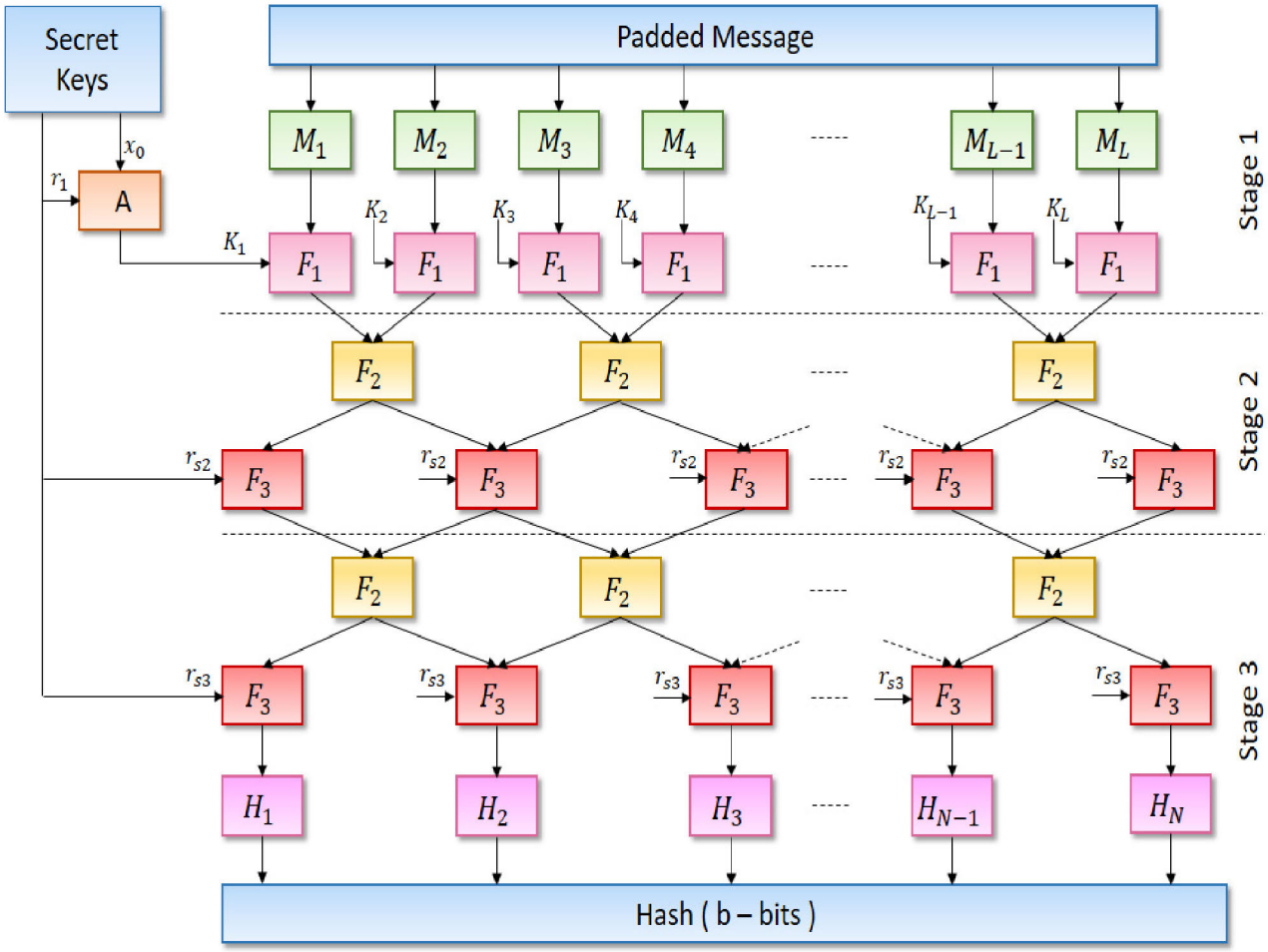

FIGURE 2. The structure of the keyed parallel hash algorithm.

A. MESSAGE PRE-PROCESSING

First, the message M with a random length is padded to multiples of 256-bits by adding a bit of $\{100\ldots.00\}_2$ at the end of message and the rightmost 64-bits are reserved to indicate the length of message M. Then, the binary message is divided into L blocks each of 256-bits and the blocks are converted to ASCII codes according to the chart of ASCII code.

B. MESSAGE PRE-PROCESSING

First, the message M with a random length is padded to multiples of 256-bits by adding a bit of $\{100\ldots.00\}_2$ at the end of the message, and the rightmost 64-bits are reserved to indicate the length of message M. Then, the binary message is divided into L blocks each of 256-bits and the blocks are converted to ASCII codes according to the chart of ASCII code.

C. PARAMETER CONFIGURATION

Initiate the first logistic map with $x_0 = 0.98765$ and $r_1 = 3.9393$, then iterate the first logistic map several times equal to the number of message blocks L to obtain the array of initial values of tent maps.

i) The control parameter of the tent map is set to $q = 1.5151$.

ii) The control parameters of second and third logistic maps are set to $r_{s2} = 3.83$, and $r_{s3} = 3.73$ respectively.

iii) A proportion coefficient p is set to $p = 0.6543$.

D. MESSAGE HANDLING

After parameters initialization, the proposed hash function consists of three stages which are executed as following (the blocks are implemented in parallel so, M_i is used to demonstrate the algorithm);

Stage (1):

i) M_i consists of 32 ASCII codes that lie between [0,255].

Modify the initial state x_0 of tent map F_1 by using the proportion coefficient p and iterating the Eq. (4) several times equal to 32.

$$x_{i+1} = \left| (p * x_i) + (\frac{\text{ASCII}}{255} - p) \right|. \tag{4}$$

ii) The new x_0 is used in Eq. (2) and iterating it 32 times.

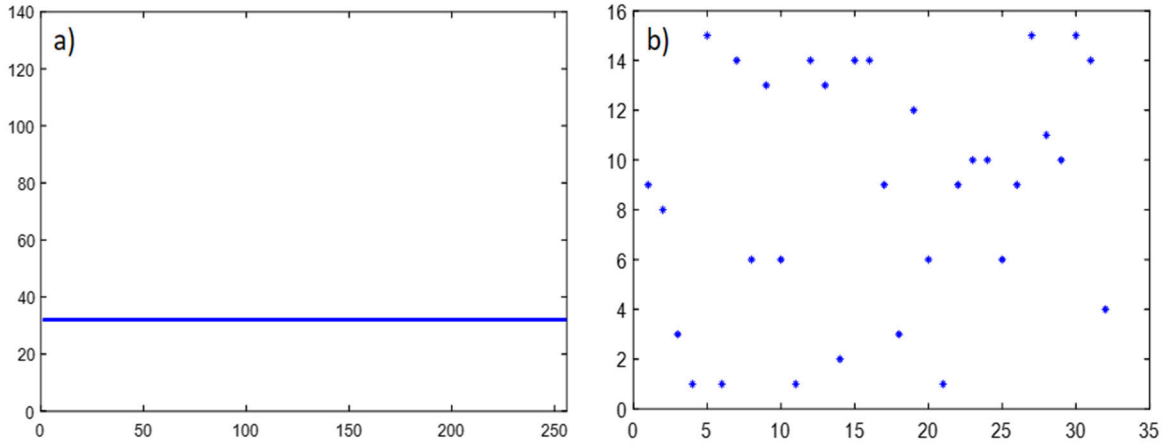

FIGURE 3. Sequence number of a) ASCII code distribution of all blank spaces message and b) Hexadecimal code distribution.

iii) The output of tent map x_{31} is the input of the second stage.

Stage (2):

i) This stage is consist of two functions F_2 and F_3 which are sine and logistic maps respectively.

ii) For F_2, the output of tent map for two blocks M_i and M_{i+1} are used to calculate the initial value of sine map listed in Eq. (3) as defined in Eq. (5)

$$x_{0(\sin)}(j) = x_{31(M_i)} + x_{31(M_{i+1})}. \tag{5}$$

The Eq. (3) is repeated only once and obtain $x_{1(\sin)}(j)$

iii) The output of sine map is the initial state of F_3 as following Eq.

$$x_{0(\log)} = x_{1(\sin)}(j) + x_{1(\sin)}(j+1), \tag{6}$$

if $x_{0(\log)} \geq 1$ it divided by 2, then using in Eq.(1) and iterating it 8 times with $r = r_{s2}$. The output of this logistic map is the input of the third stage.

Stage (3):

For 128-bit hash; the Eq.(1) is iterating 8 times with $r = r_{s3}$, the eight outputs of logistic map iterations are stored and converted into integer numbers by multiplied by 10^8 and then each integer number is converted to its corresponding binary representation then truncates the output to 16-bits from the right side, so there are 8 outputs each of 16-bits length that joined together to obtain the k^{th} 128-bit hash value.

For 256-bit hash; the Eq.(1) is iterating 16 times with $r = r_{s3}$, the sixteen outputs of logistic map iterations are stored and converted into integer numbers by multiplied by 10^{16} and then each integer number is converted to its corresponding binary representation then truncates the output to 16-bits from the right side, so there are 16 outputs each of 16-bits length that joined together to obtain the k^{th} 256-bit hash value.

E. HASH VALUE GENERATION

For k^{th} block of F_3 in stage (3), the obtained hash value is H_k, and the final hash value is computed as following

$$H = H_1 \oplus H_2 \oplus \ldots \oplus H_k \oplus \ldots \oplus H_z. \tag{7}$$

IV. THEORETICAL ANALYSIS AND SIMULATION RESULTS

The estimation security strength of the proposed hash is evaluated in this section. The performance analysis is assessed in terms of hash distribution, sensitivity analysis, Confusion and diffusion properties, collision analysis and flexibility. The parameters and initial values used in simulation are assigned as $x_0 = 0.98765$, $p = 0.6543$, $r = 3.9393$, $q = 1.5151$, $r_{s2} = 3.83$, $r_{s3} = 3.73$. The algorithm is executed in Wolfram Mathematica 12.1.

A. HASH DISTRIBUTION ANALYSIS

The hash values distribution is one of the important security features and it is must be uniform. To evaluate the hash distributive, a message of blank spaces is stored in ASCII code representation and plotted in Figure 3-a), then the message corresponding hash is generated and described in Figure 3-b). The process is repeated with another random message and illustrated in Figure 4. As shown, the message spread lies between [32,121]. From Figure 3 and Figure 4, it is obvious that the original messages spread uniformly in a small section of the plot. However, the hexadecimal hashes spread uniformly and randomly in the plot, that made the message statistical information is hided.

B. KEY SPACE ANALYSIS

To avoid brute-force attacks, the key space of every cryptosystem is very important which must to be very large to resist these attacks. For this algorithm, the keys are initial value x_0, control parameters r_1, r_{s2}, r_{s3}, q, and proportion coefficient p. The computational precision of each key is 10^{-15} [23], [24], so the size of the secret key is $10^{15} * 10^{15} * 10^{15} * 10^{15} * 10^{15} = 10^{90}$ which means that the key

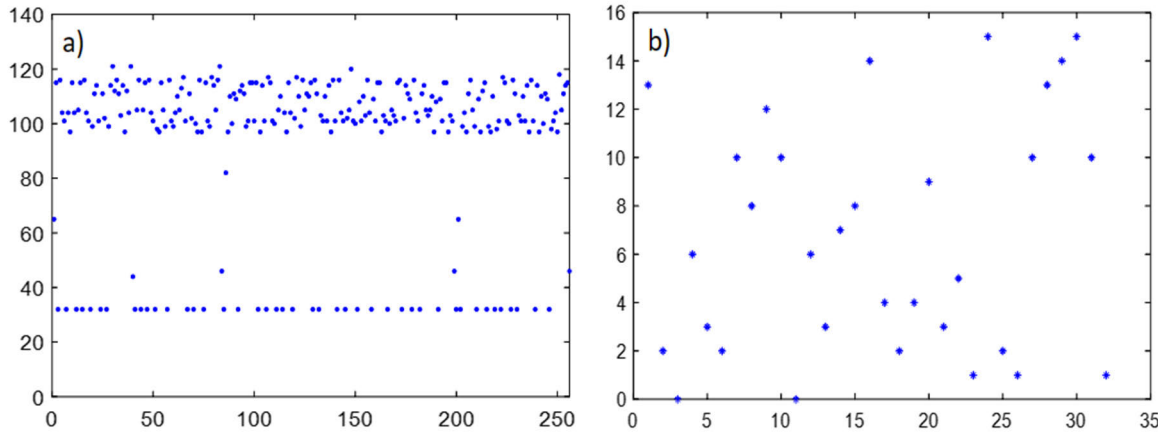

FIGURE 4. Sequence number of a) ASCII code distribution of random message and b) Hexadecimal code distribution.

TABLE 1. Key space comparison.

	Proposed	Ref[2]	Ref[24]	Ref[25]
Key space size	2^{298}	2^{249}	2^{106}	2^{279}

space is $2^{298} \gg 2^{100}$. Comparing with some other algorithms in Table 1., the proposed scheme has a larger key space that is big enough to stand against the brute-force attack.

C. SENSITIVITY ANALYSIS

The non-reverse property denotes that, finding the input message from its corresponding hash value is infeasible to compute. So, the hash function ought to have good sensitivity to small changes in the message and secret keys. The sensitivity simulation of the suggested scheme is evaluated under the following ten different conditions.

Condition 1: The original message "As the hash is the core of cryptography, it is the basic technique of the data security. Random processes are used in hash functions to generate the fixed length message digest from the original message. A chaotic map can be used to generate random behaviors.".
Condition 2: Modify the first letter to lowercase.
Condition 3: Replace the character 'A' with 'B'.
Condition 4: Change the comma to a dotted comma.
Condition 5: Modify $x_0 = 0.98765$ to $x_0 = 0.9876500001$.
Condition 6: Modify $p = 0.6543$ to $p = 0.654300001$.
Condition 7: Modify $r = 3.9393$ to $r = 3.939300001$.
Condition 8: Modify $q = 1.5151$ to $q = 1.515100001$.
Condition 9: Modify $r_{s2} = 3.83$ to $r_{s2} = 3.8300001$.
Condition 10: Modify $r_{s3} = 3.73$ to $r_{s3} = 3.7300001$.

The 128-bit hash digests in hexadecimal format are given as following:

$\boldsymbol{H_1}$: DF6F04F7CAF72B7D8EB4C6EE0E5BE65
$\boldsymbol{H_2}$: 8D9F4D80422DA43EA43AAC4098A03D42
$\boldsymbol{H_3}$: C9640B1614BDB7272DC8F71BD1CB6106
$\boldsymbol{H_4}$: 7AC09136B6309EB78E2DEA8C7B8FDB44
$\boldsymbol{H_5}$: 97AB3B3CABB387BEF25FAB8929D8A332
$\boldsymbol{H_6}$: 709EA4A91C2EED65AFF534BC705A53EE
$\boldsymbol{H_7}$: 2735B4EA8D86352D21D11290E5D7BF8A
$\boldsymbol{H_8}$: C574A3DF786363FC804306FFB3D7F216
$\boldsymbol{H_9}$: D47F51D72A67B82DAA84E1FDB74505E
$\boldsymbol{H_{10}}$: DF5F05A7CB272A2D8044DDEE06CA39C

The 256-bit hash digests in hexadecimal format are given as following:

$\boldsymbol{H_1}$: D06A9C22502F04E9919DD3DD9217AB86FB98D21B2768BABB86230B557DCB0768
$\boldsymbol{H_2}$: D8EDB93F06464936AF3837E845278BE482CB9715AA11CE28F957D6AD094B2E68
$\boldsymbol{H_3}$: 5DA7C5FA72A44D5B734697B38F08F5331E3BCDAEB6A1872C063C589413ED1970
$\boldsymbol{H_4}$: 69F78F6C7E21170EC24E1AFE7B7834EF90E6DB2F010F308CE7DB6C32EBD27417
$\boldsymbol{H_5}$: 323202D4DA766E4DE0E4A6AC6EECA74106956BA15A8FE050A8D1B66EAF2D6429
$\boldsymbol{H_6}$: 4258912F8F6222307F79163B08C7ECF3B02B5C63C9C1B734AE233CBA167253F6
$\boldsymbol{H_7}$: 956105CE9571E08F137381FC8E73656DB8C915977D468E26CBB929917FA4BEA2
$\boldsymbol{H_8}$: AE03F1E953D831BD1C3D02331C1D8D59ACC140F0B799C0AB75CD31E89D77DD9C
$\boldsymbol{H_9}$: A6C17E0E687EFC1970BDACB71935D24E082C21AA9F03F8B2B95ED21A36A8C63D
$\boldsymbol{H_{10}}$: 672527D3FE19889B90A33F792F413E5899B9862B9CDC15AEA6E24F6E5A912372

The sequence of the 128-bit binary hash values are plotted in Figure 5. As shown in hash values and in the Figure 5, the proposed scheme has high sensitivity to any variation in message and secret keys.

D. CONFUSION AND DIFFUSION ANALYSIS

As stated by Claude Shannon, confusion and diffusion properties considered as two significant features to secure

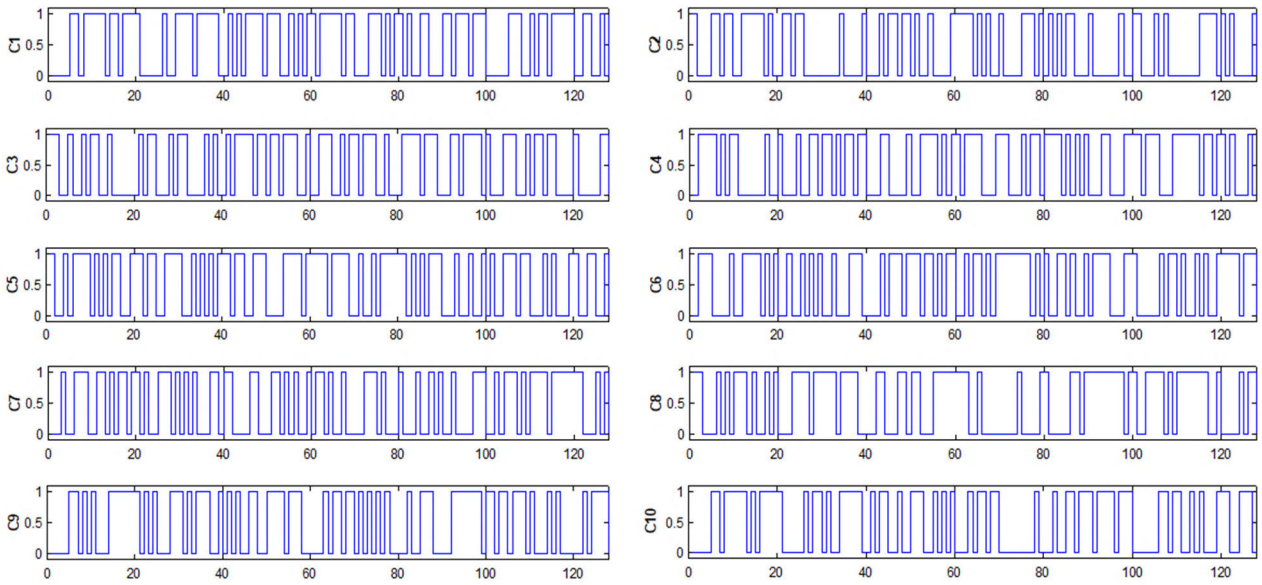

FIGURE 5. Hash values of the ten conditions in binary representation.

hash algorithm [26]. The confusion purpose is to complex the relationship that connect the message and its corresponding hash digest, and the diffusion purpose is to make the hash value highly dependent on the input message. For an efficient hash scheme, the ideal ratio for probability of hash change as a result of any bit change in message or in initial conditions is 50%. In order to evaluate these features for the suggested scheme, a random message is chosen and generate its corresponding hash value, then a random bit of the message is modified, the hash value of new message is produced. The two generated hash values are compared bit by bit and count the changed bits B_i. For N times of the test and hash length of $len = h$ bits, The capabilities of confusion and diffusion are evaluated through the following six metrics:

Minimum number of changed bits

$$B_{\min} = \min(B_i), i = 1, 2, \ldots, N. \tag{8}$$

Maximum number of changed bits

$$B_{\max} = \max(B_i), i = 1, 2, \ldots, N. \tag{9}$$

Mean number of changed bits

$$\bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i. \tag{10}$$

Mean probability of changed bits

$$P = \frac{\bar{B}}{h} \times 100\%. \tag{11}$$

Standard deviation of bit change

$$\Delta\bar{B} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(B_i - \bar{B}\right)^2}. \tag{12}$$

Standard deviation of probability

$$\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{B_i}{n} - P\right)^2} \times 100\%. \tag{13}$$

The test is repeated N times on the intended scheme with hash length of 128-bit and 256-bit where $N = 256, 512, 1024$, 2048, and 10000, and the result of the previous metrics are listed in Table 2, and Table 3.

From Tables 2 and 3, it is apparent that the proposed scheme achieves a mean number of changed bit $\bar{B}$ which very close to the ideal value of $(len/2)$, and mean probability of changed bit $P(\%)$ is highly near to ideal rate of 50%. For all tests the hash function also has so small values of $\Delta\bar{B}$ and ΔP, which means that the proposed hash algorithm has very stable confusion and diffusion capabilities. Figure 5 illustrate the number of changed bit distribution of $len = 128$ bit hash as a) distribution of changed bit $\bar{B}$ for $N = 10000$ and b) shows the histogram of $\bar{B}$ which centered near to the ideal value of 64.

E. COLLISION ANALYSIS

An experiment is performed to describe the collision resistance property as it is one of the most critical feature of efficient cryptosystem. This experiment is done as follow:

i) A random message with arbitrary size is chosen as the main message and generate the corresponding hash value then stored in ASCII format.

ii) Change one random bit of the original message and calculate the new hash then store it in ASCII format.

iii) The two hash value are compared with each other and the ASCII characters that located in the same places and have the same values (number of hits) are counted.

TABLE 2. Changed bit statistical results of 128-bit hash value.

	N=256	N=512	N=1024	N=2048	N=10000	Mean
$B_{\min}$	50	50	50	47	47	48.8
$B_{\max}$	81	78	83	83	83	81.6
$\bar{B}$	64.332	64.217	64.198	64.202	64.012	64.192
$P(\%)$	50.259	50.169	50.155	50.158	50.009	50.15
$\Delta\bar{B}$	5.307	5.365	5.537	5.555	5.609	5.475
$\Delta P(\%)$	4.146	4.192	4.326	4.339	4.382	4.277

TABLE 3. Changed bit statistical results of 256-bit hash value.

	N=256	N=512	N=1024	N=2048	N=10000	Mean
$B_{\min}$	117	105	104	100	99	105
$B_{\max}$	147	148	149	150	150	148.8
$\bar{B}$	128.62	128.438	128.622	128.388	127.789	128.372
$P(\%)$	50.242	50.171	50.243	50.151	49.918	50.145
$\Delta\bar{B}$	7.485	7.569	7.576	7.664	7.926	7.644
$\Delta P(\%)$	2.924	2.957	2.959	2.994	3.096	2.986

iv) Calculate the absolute difference as: $d = \sum_{i=1}^{N} \left| t\left(e_i\right) - t(e_i') \right|$, Where e_i is the i^{th} ASCII character of the first hash value while e_i' is the i^{th} ASCII character of the second hash value, and the function $t(\ldots)$ indicate to the decimal equivalent of ASCII value.

To evaluate the theoretical values to the number of hits, let ω is the number of hits where $\omega = 0, 1, 2, \ldots, s$, and $W_N(\omega)$ is the number of hits that happened in N tests that defined as

$$F = \begin{cases} W_N\left(\omega\right) = N \times \dfrac{s!}{\omega!(s-\omega)!} \times \left(\dfrac{1}{2^8}\right)^{\omega} \times \left(1 - \frac{1}{2^8}\right)^{s-\omega} \\ \sum\limits_{\omega=0}^{s} W_N\left(\omega\right) = W_N\left(0\right) + W_N\left(1\right) + \ldots + W_N\left(s\right) = N \end{cases} \tag{14}$$

where $s = \frac{L}{8}$, L is the hash value length while 8 is the number of bits required to ASCII representation [22]. The test is performed for $N = 2048$ and 10000 times and for hash length of $len = 128$ and 256 bits, the number of hits are compared with ideal values and listed in Table 4.

TABLE 4. Number of hits for $N = 2048$ and 10000 of a) 128-bit hash and b) 256-bit hash.

a) 128-bit hash:

N	Hits No.	0	1	2	3	4
2048	Ideal	1924	121	4	0	0
	Proposed	1931	113	3	0	0
10,000	Ideal	9393	589	17	0	0
	Proposed	9393	584	22	0	0

b) 256-bit hash:

N	Hits No.	0	1	2	3	4
2048	Ideal	1807	227	14	0	0
	Proposed	1809	224	14	0	0
10,000	Ideal	8823	1107	67	0	0
	Proposed	8744	1188	65	2	0

F. ABSOLUTE DIFFERENCE

It is important to compute the theoretical value of the mean absolute difference to compare with the value of the proposed scheme. The discrete uniform distribution H has a range of 0 to 255 and the ideal value of this uniform distribution is equal to half of the maximum distribution value that occurs when all possible characters are equal to 255. Using the assumption that the two different hash digests are ideally uniform, the absolute difference sum of these two hashes is equal to $^2/_3$ of the uniform distribution mean value [21]. For 128-bit hash length, the maximum value is $\frac{255\times16}{2} = 2,040$, so the theoretical mean absolute difference is equal to $^2/_3 \times 2,040 = 1,360$. Whereas, for 256-bit hash, the uniform distribution mean value is $\frac{255\times32}{2} = 4,080$, and the theoretical mean absolute difference is equal to $^2/_3 \times 4,080 = 2,720$. For the proposed scheme, when $len = 128$ and 256, the mean absolute difference are 1367.46 and 4079 respectively which are extremely near to the theoretical value.

For $N = 10000$, the minimum, maximum, mean absolute difference, and mean/character values are reported in Table 5. As it apparent, the mean/character value of $len = 128$ and

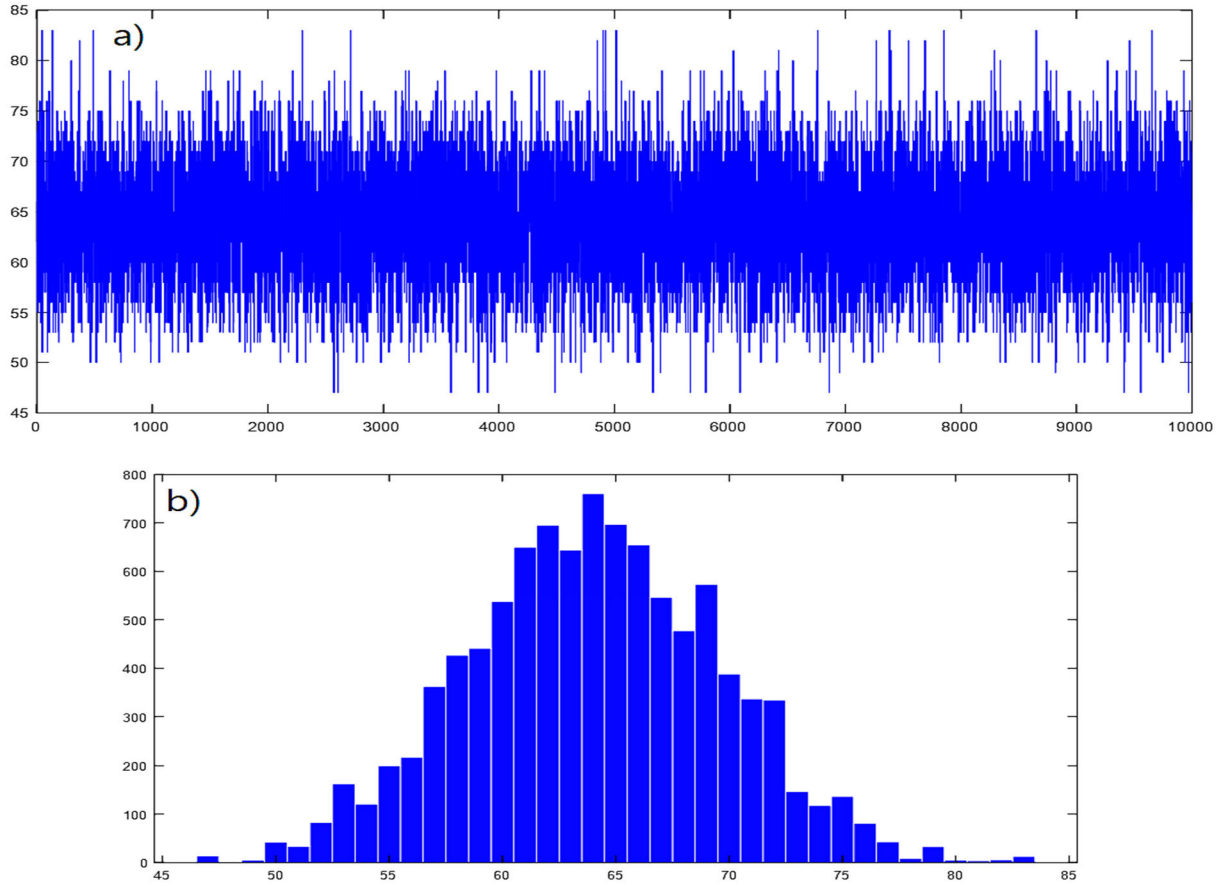

FIGURE 6. Changed bit number distribution of a) $\bar{B}$ and b) Histogram of $\bar{B}$.

TABLE 5. Absolute difference results of *len* = 128 and 256-bit hash value for $N = 10000$.

Hash length	Minimum	Maximum	Mean	Mean/Ch.
128-bit hash	554	2368	1367.46	85.466
256-bit hash	1474	4079	2733.18	85.41

256 are 85.466 and 85.41 respectively which are extremely near to the theoretical value that is $^1/_3 \times 256 = 85.333$ %.

TABLE 6. Speed analysis comparison.

Algorithm	Speed (Gbps)	Platform
Ref [2]	0.697	Intel Core-i7, 16GB RAM
Ref [27]	0.761	Intel Core-i3, 4GB RAM
Ref [28]	0.681	Quad-core machine
Ref [28]	0.314	dual-core machine
This scheme	0.883	Intel Core-i7, 8GB RAM

G. SPEED ANALYSIS

One of the main requirements of effective hash algorithm is the processing speed. To analyze the speed of proposed hash, the algorithm is implemented in Wolfram Mathematica 12.1. on a PC with 2.50 GHz Intel(R) Core^{TM}i7, 8 GB RAM and message length of 100 MB. The proposed algorithm compression speed with comparison is listed in Table 6.

V. COMPARATIVE ANALYSIS WITH OTHER HASH SCHEMES

A comparison between the suggested hash scheme and other different chaos-based hash schemes is performed in terms of statistical performance, resistance of collision, and speed analysis. Tables 7 and 8 depict the statistical performance comparison between the proposed hash algorithm and selected hash algorithms. The results in Table 7 are based on the hash length of 128-bit hash, while in Table 8 the results are based on 256-bit hash. According to the results listed in the tables, this algorithm exhibits preferable statistical performance. Furthermore, a comparison of the number of hits and absolute difference in 128 and 256- bit hash values is performed based on $N = 10000$ and the results are shown in Tables 9 and 10 respectively. The results

TABLE 7. Statistical performance comparison of 128-bit hash value.

Algorithm	$B_{\min}$	$B_{\max}$	$\bar{B}$	$P(\%)$	ΔB	$\Delta P(\%)$
This scheme	49	82	64.35	50.15	5.48	4.28
Ref [18]	44	84	64.02	50.01	5.64	4.40
Ref [3]	46	80	64.002	50.002	5.51	4.30
Ref [4]	47	82	64.13	50.10	5.61	4.38
Ref [30]	47	80	64.04	50.03	5.74	4.48
Ref [28]	45	86	63.99	49.99	5.68	4.37

TABLE 8. Statistical performance comparison of 256-bit hash value.

Algorithm	$B_{\min}$	$B_{\max}$	$\bar{B}$	$P(\%)$	ΔB	$\Delta P(\%)$
This scheme	105	149	128.372	50.145	7.644	2.986
Ref [31]	100	154	127.96	49.98	8.04	3.14
Ref [20]	100	156	128	50	8.03	3.14
Ref [2]	97	157	128.30	50.12	7.85	3.06

TABLE 9. Comparison of collision test results of 128-bit hash value and $N = 10{,}000$ a) Number of hits and b) Absolute difference.

a) Number of hits

Algorithm	0	1	2	3	4	5
Ideal	9393	589	17	0	0	0
This scheme	9393	584	22	0	0	0
Ref [18]	9411	577	12	0	0	0
Ref [4]	9387	586	27	0	0	0
Ref [30]	9495	505	0	0	0	0
Ref [28]	9969	31	0	0	0	0

a) Absolute difference

Algorithm	Minimum	Maximum	Mean	Mean/Ch.
Ideal	-	-	1360	85.333
This scheme	554	2368	1367.46	85.466
Ref [18]	584	2321	1366	85.375
Ref [4]	796	2436	1374	85.88
Ref [30]	685	1972	1253	78.31
Ref [28]	575	2390	1379	86.19

TABLE 10. Absolute difference of 256-bit hash wit $N = 10{,}000$.

Algorithm	Minimum	Maximum	Mean	Mean/Ch.
Ideal	-	-	2720	85.333
This scheme	1474	4079	2733.18	85.41
Ref [31]	1663	3930	2741	85.66
Ref [20]	1568	4054	2789	87.16
Ref [2]	1883	4051	2887	90.22

in Tables 9 and 10 show that the proposed scheme has high collision resistance and absolute difference near to the ideal mathematical value. In addition, the proposed algorithm can be used as keyless hash function when the secret keys from initial values and control parameters are be constant and known to the public, in this case the proposed scheme can be used such SHA-3 in image encryption application as in [29].

VI. CONCLUSION

In this paper, a keyed hash algorithm based on multiple chaotic maps with parallel diamond lattice structure is implemented. A chaotic logistic map, sine function, and tent map are chosen, with certain parameters, so the proposed scheme proved to have high security. Also, the algorithm is flexible to generate different lengths of hash value and in this paper hash values of lengths 128 and 256-bit are evaluated. Simulation results in terms of distribution, performance, security, and sensitivity demonstrated that the investigated hash has high capabilities of confusion and diffusion and strong collision attack furthermore high speed. These properties make the algorithm efficient for authentication, digital signature, and other communication security application.

REFERENCES

[1] Y. Li, "Collision analysis and improvement of a hash function based on chaotic tent map," *Optik*, vol. 127, no. 10, pp. 4484–4489, May 2016.
[2] H. Liu, A. Kadir, and J. Liu, "Keyed hash function using hyper chaotic system with time-varying parameters perturbation," *IEEE Access*, vol. 7, pp. 37211–37219, 2019.
[3] Y. Li and G. Ge, "Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security," *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 17973–17994, 2019.
[4] 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, 2017.
[5] M. A. AlAhmad and I. F. Alshaikhli, "Broad view of cryptographic hash functions," *Int. J. Comput. Sci. Issues*, vol. 10, no. 4, p. 239, 2013.
[6] A. Sotirov, M. Stevens, J. Appelbaum, A. K. Lenstra, D. Molnar, D. A. Osvik, and B. de Weger, "MD5 considered harmful today, creating a rogue CA certificate," in *Proc. 25th Annu. Chaos Commun. Congr.*, 2008, pp. 1–24.
[7] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Proc. Annu. Int. Cryptol. Conf.*, 2005, pp. 17–36.

[8] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collisions of SHA-0 and reduced SHA-1," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 36–57.
[9] T. Fox-Brewster, *Google Just 'Shattered' an Old Crypto Algorithm-Here's Why That's Big for Web Security*. Waltham, MA, USA: Forbes, 2017.
[10] A. A. L. Selvakumar and C. S. Ganadhas, "Survey of hash function: Resistance to finding attacks," *Int. J. Inf. Technol. Knowl. Manage.*, vol. 2, no. 1, pp. 15–20, Jan./Jun. 2009.
[11] J. Kelsey and T. Kohno, "Herding hash functions and the nostradamus attack," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2006, pp. 183–200.
[12] J. Kelsey and B. Schneier, "Second preimages on n-bit hash functions for much less than 2^n work," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 474–490.
[13] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," Federal Inf. Process. Standards Publication, Tech. Rep. FIPS 202, Aug. 2015.
[14] I. Dinur, O. Dunkelman, and A. Shamir, "New attacks on Keccak-224 and Keccak-256," in *Proc. Int. Workshop Fast Softw. Encryption*, 2012, pp. 442–461.
[15] I. Dinur, O. Dunkelman, and A. Shamir, "Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials," in *Proc. Int. Workshop Fast Softw. Encryption*, 2013, pp. 219–240.
[16] J. M. Amigó, L. Kocarev, and J. Szczepanski, "Theory and practice of chaotic cryptography," *Phys. Lett. A*, vol. 366, no. 3, pp. 211–216, Jun. 2007.
[17] J. S. Teh, K. Tan, and M. Alawida, "A chaos-based keyed hash function based on fixed point representation," *Cluster Comput.*, vol. 22, no. 2, pp. 649–660, 2019.
[18] 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, no. 1, pp. 713–729, Mar. 2020.
[19] H. Liu, A. Kadir, C. Ma, and C. Xu, "Constructing keyed hash algorithm using enhanced chaotic map with varying parameter," *Math. Problems Eng.*, vol. 2020, pp. 1–10, Jul. 2020.
[20] H. Liu, X. Wang, and A. Kadir, "Constructing chaos-based hash function via parallel impulse perturbation," *Soft Comput.*, vol. 25, pp. 11077–11086, 2021.
[21] M. Alawida, J. S. Teh, D. P. Oyinloye, M. Ahmad, and R. S. Alkhawaldeh, "A new hash function based on chaotic maps and deterministic finite state automata," *IEEE Access*, vol. 8, pp. 113163–113174, 2020.
[22] M. Alawida, A. Samsudin, N. Alajarmeh, J. S. Teh, M. Ahmad, and W. H. Alshoura, "A novel hash function based on a chaotic sponge and DNA sequence," *IEEE Access*, vol. 9, pp. 17882–17897, 2021.
[23] Y. Q. Zhang and X. Y. Wang, "A symmetric image encryption algorithm based on mixed linear-nonlinear coupled map lattice," *Inf. Sci.*, vol. 273, pp. 329–351, Jul. 2014.
[24] C. Li, G. Luo, K. Qin, and C. Li, "An image encryption scheme based on chaotic tent map," *Nonlinear Dyn.*, vol. 87, no. 1, pp. 127–133, 2017.
[25] Y. Zhou, L. Bao, and C. P. Chen, "A new 1D chaotic system for image encryption," *Signal Process.*, vol. 97, no. 11, pp. 172–182, 2014.
[26] D. Selent, "Advanced encryption standard," *Rivier Acad. J.*, vol. 6, no. 2, pp. 1–14, 2010.
[27] M. A. Chenaghlu, S. Jamali, and N. N. Khasmakhi, "A novel keyed parallel hashing scheme based on a new chaotic system," *Chaos, Solitons Fractals*, vol. 87, pp. 216–225, Jun. 2016.
[28] J. S. Teh, A. Samsudin, and A. Akhavan, "Parallel chaotic hash function based on the shuffle-exchange network," *Nonlinear Dyn.*, vol. 81, no. 3, pp. 1067–1079, Aug. 2015.
[29] G. Ye, K. Jiao, and X. Huang, "Quantum logistic image encryption algorithm based on SHA-3 and RSA," *Nonlinear Dyn.*, vol. 104, no. 3, pp. 2807–2827, May 2021.
[30] Y. Li and X. Li, "Chaotic hash function based on circular shifts with variable parameters," *Chaos, Solitons Fractals*, vol. 91, pp. 639–648, Oct. 2016.
[31] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assch. (2011). *The Keccak Sponge Function Family: Specifications Summary*. [Online]. Available: http://keccak.noekeon.org/specs_summary.html

ROAYAT ISMAIL ABDELFATAH received the B.Sc. degree from the Electronics and Electrical Communications Engineering Department, Tanta University, Egypt, in 2000, and the M.Sc. and Ph.D. degrees from Tanta University, in 2005 and 2011, respectively. Her M.Sc. is dedicated in encryption techniques and its applications. Her Ph.D. is devoted to introduce new encryption, digital signature, signcryption, and hash functions algorithms for securing digital data over communication networks. She has promoted as an Assistant Professor with the Electronics and Electrical Communications Engineering Department, Faculty of Engineering, Tanta University, in 2020.

ESRAA ABDELKHALEK BAKA received the B.Sc. degree in electronics and electrical communications engineering from Tanta University, Egypt, in 2014, where she is currently pursuing the M.Sc. degree in information security and encryption techniques.

MOHAMED E. NASR (Life Senior Member, IEEE) received the B.Sc. degree (Hons.) in electronics, in 1975, the M.Sc. degree in electronics and communication, in 1979, and the Ph.D. degree in digital communication systems, in 1985. From 2001 to 2007, he worked as the Head of the Department. From 2006 to 2009, he was the President of Administration Board for Information and Communication Technology Project (ICTP) at Tanta University, Egypt, and leading the Technical Team to manage and modify Tanta University network. From September 2009 to January 2010, he was the Chief Information Officer (CIO) and a Co-ordinator at Tanta University. From February 2010 to August 2014, he was the Dean of Ras EL-Bar Higher Institute for Computers, Damietta. From September 2014 to August 2016, he was the Dean of Alexandria Higher Institute of Engineering and Technology, Alexandria, Egypt. He is currently a Professor of digital communications and computer networks with the Department of Electronics and Communications, Faculty of Engineering, Tanta University. He has published more than 170 journal articles and conference scientific papers. His current research interests include speech and audio coding, packet voice transmission over data networks, wireless communication systems, and networking security. He was the Vice Chairman of the Twenty Fifth National Radio Science Conference (NRSC, 2008), Tanta, Egypt.