Journal of Ambient Intelligence and Humanized Computing (2020) 11:2137–2161
https://doi.org/10.1007/s12652-019-01244-y

ORIGINAL RESEARCH

Design and security analysis of two robust keyed hash functions based on chaotic neural networks

Nabil Abdoun[1] · Safwan El Assad[1] · Olivier Deforges[2] · Rima Assaf[3] · Mohamad Khalil[3]

Received: 10 August 2018 / Accepted: 6 February 2019 / Published online: 22 February 2019

Abstract

In this paper, we designed, implemented, and analyzed the performance, in terms of security and speed, of two proposed keyed Chaotic Neural Network (CNN) hash functions based on Merkle–Dåmgard (MD) construction with three output schemes: CNN-Matyas–Meyer–Oseas, Modified CNN-Matyas–Meyer–Oseas, and CNN-Miyaguchi–Preneel. The first hash function's structure is composed of two-layer chaotic neural network while the structure of the second hash function is formed of one-layer chaotic neural network followed by non-linear layer functions. The obtained results of several statistical tests and cryptanalytic analysis highlight the robustness of the proposed keyed CNN hash functions, which is fundamentally due to the strong non-linearity of both the chaotic systems and the neural networks. The comparison of the performance analysis with some chaos-based hash functions of the literature and with standard hash functions make the proposed hash functions suitable for data integrity, message authentication, and digital signature applications.

Keywords Keyed hash functions · Chaotic neural networks · Chaotic activation function · Merkle–Dåmgard · Statistical tests · Brute force attacks · Cryptanalytical attacks · Speed analysis

1 Introduction

During the last decade, information security has become a hot issue. Developers are usually concerned about five main services regarding information exchange over non-secure channels (e.g., internet): confidentiality, authenticity, integrity, non-repudiate, and availability. Hash functions are one of the most useful primitives in cryptography that play an important role in data security. They can achieve data integrity, message authentication (Islam 2014), and digital signature (Chain and Kuo 2013). Hash function is a one-way function that maps an arbitrary finite large message data into a fixed-length hash value. It should achieve some security properties, such as message sensitivity, key sensitivity, confusion–diffusion, preimage, second preimage, and collision resistance. Also, it should be immune against brute force and cryptanalytical attacks. Nowadays, the most popular standard secure hash functions are unkeyed Secure Hash Algorithms SHA-2 (Secure Hash Standard and FIPS Publication 2002) and SHA-3 (SHA-3), commonly used by many SSL certificate authorities, whereas keyed hash functions include: Very fast Message Authentication Code—VMAC, Keyed-Hash MAC—HMAC, Galios/Counter Mode—GCM, Cipher-based MAC—CMAC, Destination MAC—DMAC, Cipher Block Chaining Message Authentication Code—CBC-MAC and BLAKE 2.

Alternatively, a new direction in the construction of chaos-based hash functions appeared in 2002. Due to the strong non-linearity of chaotic systems and neural network structures, some designers usually combine these two systems to build robust hash functions. Indeed, a chaotic system is characterized by important security features, such

✉ Nabil Abdoun
nabil.abdoun@etu.univ-nantes.fr

Safwan El Assad
safwan.elassad@univ-nantes.fr

Olivier Deforges
olivier.deforges@insa-rennes.fr

Rima Assaf
rima.assaf@ul.edu.lb

Mohamad Khalil
mohamad.khalil@ul.edu.lb

1 Institut d'Électronique et de Télécommunications de Rennes, UMR CNRS 6164/Site Polytech Nantes, Nantes, France

2 Institut d'Électronique et de Télécommunications de Rennes/Site INSA Rennes, Rennes, France

3 Lebanese University, Beirut, Lebanon

as sensitivity to initial conditions, random-like behavior, and unstable periodic orbits. Also, a neural network is characterized by its confusion–diffusion and compression properties that are required to design secure hash functions.

However, many researchers developed hashing schemes based on simple chaotic maps, such as logistic map, high-dimensional discrete map, piecewise linear chaotic map, tent map, and Lorenz map or on 2D coupled map lattices (Akhavan et al. 2009; Amin et al. 2009; Arumugam et al. 2007; Kwok and Tang 2005; Li et al. 2012b; Liu et al. 2012; Maqableh et al. 2008; Wang et al. 2007, 2008, 2011; Wong 2003; Xiao et al. 2005; Yi 2005; Zhang et al. 2009). In 2007, Zhang et al. (2007) proposed a novel chaotic keyed hash algorithm using a feed forward-feedback non-linear filter. Other researchers proposed combined hashing and encryption schemes based on chaotic neural network (Deng et al. 2009, 2010; He et al. 2013; Li et al. 2011a, b, 2013; Lian et al. 2006a, b; Liu and Xiu 2008; Xiao and Liao 2004; Xiao et al. 2009a, b; Yang et al. 2009).

Since 2010, there has been a real turning point in building new secure hash algorithms based on chaotic maps and neural network. Huang (2011) proposed an enhancement of Xiao's parallel keyed hash function based on chaotic neural network (Xiao et al. 2009b). Indeed, in Xiao's scheme, the secret keys are not nonce numbers, which might produce a potential security flaw. Jiteurtragool et al. (2013), proposed a topologically simple keyed hash function based on circular chaotic sinusoidal map network that uses more complex map, i.e., the Sine map. In 2014, Teh et al. (2015) introduced a parallel chaotic hash function based on the shuffle-exchange network that runs in parallel to improve hashing speed. In 2015, Abdoun et al. (2015; 2016) proposed a new efficient structure that consists of two parts: an efficient chaotic generator and a three or two-layer neural network. Chenaghlu et al. (2016) published a new keyed parallel hashing scheme based on a new hyper sensitive chaotic system with compression ability. High-dimensional chaotic maps have also been used in hash functions for higher complexity and better mixing (Akhavan et al. 2013; Guesmi et al. 2016; Nouri et al. 2012). Xiao et al. (2008) designed a parallel keyed chaos-based hash function, where a mechanism of both changeable-parameter and self-synchronization is used to establish a close relation of the keystream with the algorithm key, the content, and the order of each message block.

This paper proposes two robust keyed CNN hash functions based on Merkle–Dåmgard construction, that having better hash throughput as compared to the other chaos-based hash functions in literature. Indeed, the structures of the proposed CNN hash functions are based on neural network layer(s) and non-linear layer functions. Each neuron uses a chaotic activation function based on an efficient chaotic generator using Discrete Skew Tent map (DSTmap) and a Discrete Piecewise Linear Chaotic map (DPWLCmap) (El Assad 2012; El Assad and Noura 2014).

The rest of this paper is organized as follows: Sect. 2 presents the properties, and the general model of Merkle–Dåmgard construction formed by preprocessing and compression phases. Section 3 introduces in detail the structures of the two proposed keyed CNN hash functions based on MD with their components i.e., chaotic generator, output schemes, neural network, and non-linear functions. Section 4 presents the obtained results, in terms of security and computational performance, of the proposed hash functions and compares their performance with other hash functions found in literature. Section 5 concludes our contribution and outlines the direction of future work.

2 Preliminaries

2.1 Properties of cryptographic hash functions

A cryptographic hash function H aims to guarantee a number of properties, which makes it very useful for information security (Kim et al. 2017; Lee et al. 2014; Liu et al. 2015; Menezes et al. 1996; Stallings 2014). H must verify at least the following two implementation properties:

1. Compression: H maps an input message M of arbitrary finite bit-length to a hash value h of fixed bit-length u bits.
2. Ease of computation: given H and an input message M, $H(M)$ is easy to compute.

Nevertheless, two important requirements are needed to realize the cryptographic hash functions: the *hardness* to find collisions and the appearance of *randomness*. Also, H has the following three security properties (Bellare et al. 1996):

1. Preimage resistance (one-way): for all the pre-specified hash values h, it is computationally infeasible to find any message input that is hashed to the chosen hash value.
2. Second preimage resistance (weak collision resistance): it is computationally infeasible to find any second input that has the same hash value as a specified input message M.
3. Collision resistance (strong collision resistance): it is computationally infeasible to find any two distinct message inputs (M, M') hashed to the same hash value, such that $H(M) = H(M')$. It should be noted that, the users are free to choose both input messages.

2.2 Structures of hash functions

In cryptography, many structures are used to construct different hash functions (Denton and Adhami), such as Merkle–Dåmgard (Damgård 1989; Merkle et al. 1979), Wide Pipe (Lucks 2004), Fast Wide Pipe (Nandi and Paul 2010), HAIFA (Dunkelman and Biham 2006), and Sponge construction (Bertoni et al. 2007). The Merkle–Dåmgard construction was used in the design of many popular hash algorithms, such as MD5 (Rivest 1992), SHA-1 (Pub 1995), and SHA-2 (Secure Hash Standard and FIPS Publication 2002). The Sponge construction was used in the design of SHA-3 (SHA-3). This paper proposes novel hash functions based on Chaotic System and Neural Network. The proposal uses the structure of Merkle–Dåmgard with a proposed compression function based on Chaotic Neural Network (CNN). To understand the proposed hash functions, it is necessary to introduce the Merkle–Dåmgard construction (Fig. 1) and the model of Strengthened Merkle–Dåmgard (Fig. 2).

*Merkle–D*åmgard construction: preprocessing and compression: Fig. 1 shows the structure of Merkle–Dåmgard construction where the compression function is defined by $C : \{0,1\}^l \times \{0,1\}^{|M_i|} \to \{0,1\}^l$. C takes as inputs a chaining or state variable $h_i (i = 0, ..., q-1)$ of size l bits and a message block $M_i (i = 1, ..., q)$ of size $|M_i|$ bits, to produce the updated chaining variable $h_i (i = 1, ..., q)$ of size l bits. Thus, to allow the usage of input messages of arbitrary length, the Merkle–Dåmgard structure needs a padding, which transforms the input message into a padded message M of length multiple of $|M_i|$ bits. Indeed, a simple padding is insufficient because, in this case, the generated hash value is vulnerable to different attacks due to collision between the latest blocks. We will consider the Strengthened Merkle–Dåmgard padding with length strengthening (Figs. 2 and 3). It uses a padding function named "is-pad", which appends the binary value of the message length L at the end of the message to generate the padded message. Additionally, the Strengthened Merkle–Dåmgard construction employs a predefined initialization vector IV used as the first state value of the structure. The Strengthened Merkle–Dåmgard hash function $SMD_C(M)$ is defined as follow:

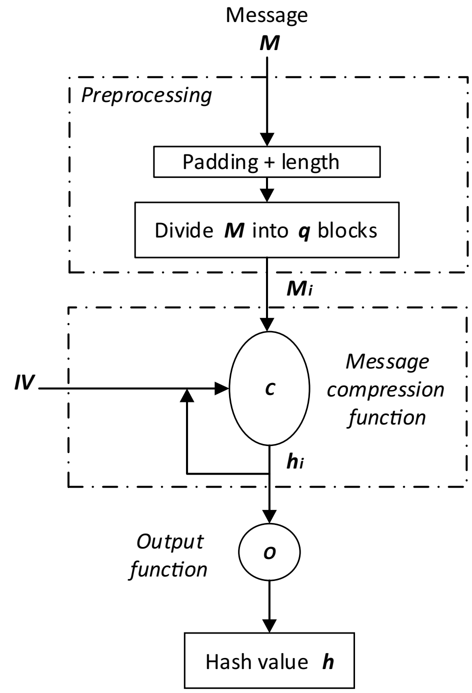


Fig. 2 Model of Strengthened Merkle–Dåmgard construction

$M_1 \parallel M_2 \parallel ... \parallel M_q \leftarrow \}\}is - pad(M)''$

$h_0 \leftarrow IV$

$for\ i = 1, ..., q\ do\ h_i \leftarrow C(h_{i-1}, M_i)$

$h \leftarrow O(h_q)$

$return\ h.$

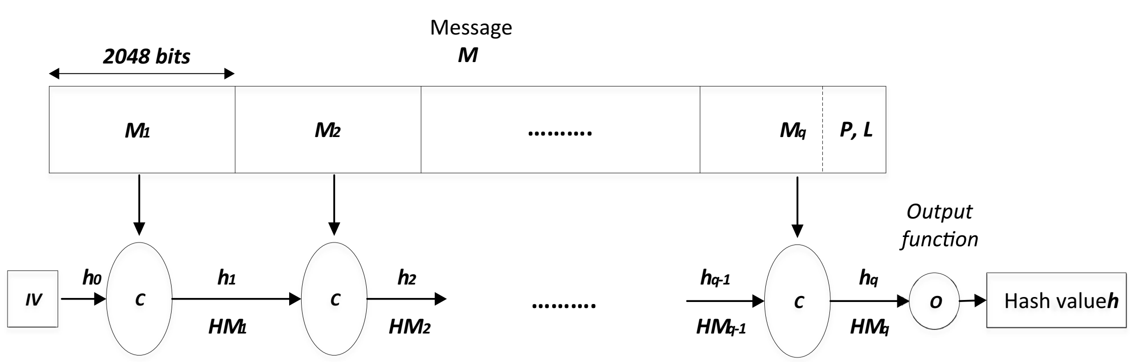


Fig. 1 Strengthened Merkle–Då mgard construction

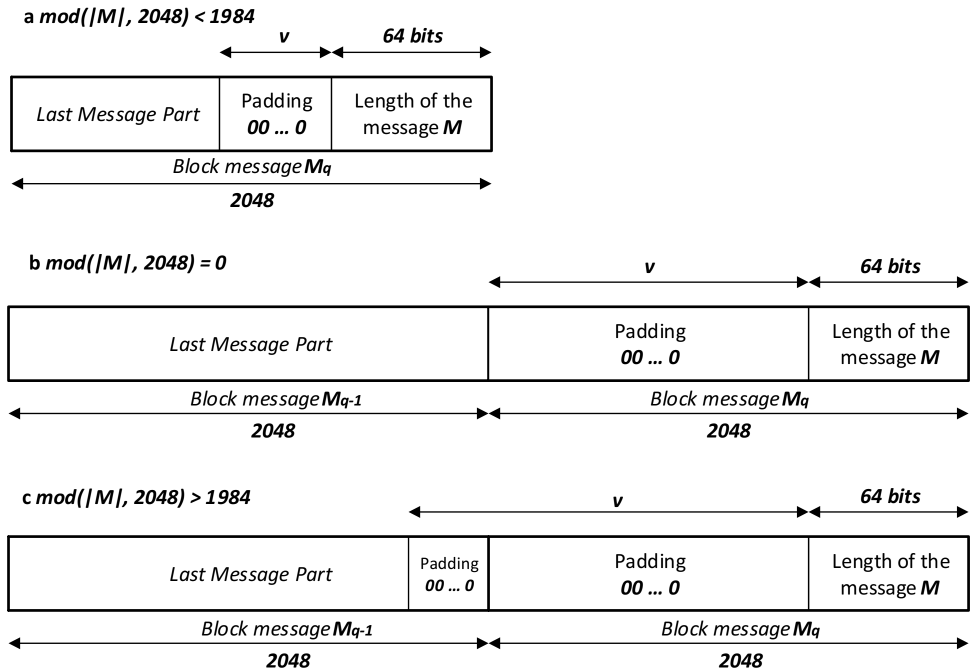


Fig. 3 The padding of input message in the proposed hash functions

M is padded with the bit pattern 00...0 of length v bits, as shown in Eq. (1). The remaining 64 bits is used by "is-pad" function to denote L.

$$v = |M_i| - mod[(L + 64), |M_i|] \tag{1}$$

It should be noted that, if L exceeds 2^{64}, then $L \; mod \; 2^{64}$ is taken as the message length instead of L (Menezes et al. 1996).

In general, we have three cases of padding:

$$case\ a\ :\ mod(|M|, |M_i|) < |M_i| - 64.$$

$$case\ b\ :\ mod(|M|, |M_i|) = 0.$$

$$case\ c\ :\ mod(|M|, |M_i|) > |M_i| - 64.$$

Now, let's take a look at the three cases of padding where $|M_i| = 2048$ bits (Fig. 3), which is as follows:

$$case\ a\ :\ if\ L = 6066\ bits\ :$$

$$v = 2048 - mod[(6066 + 64), 2048] = 14\ bits.$$

$$case\ b\ :\ if\ L = 6144\ bits\ :$$

$$v = 2048 - mod[(6144 + 64), 2048] = 1984\ bits.$$

$$case\ c\ :\ if\ L = 6086\ bits\ :$$

$$v = 2048 - mod[(6086 + 64), 2048] = 2042\ bits.$$

Then, the padded message is processed as a sequence of message blocks $M_1 \parallel M_2 \parallel ... \parallel M_q$.

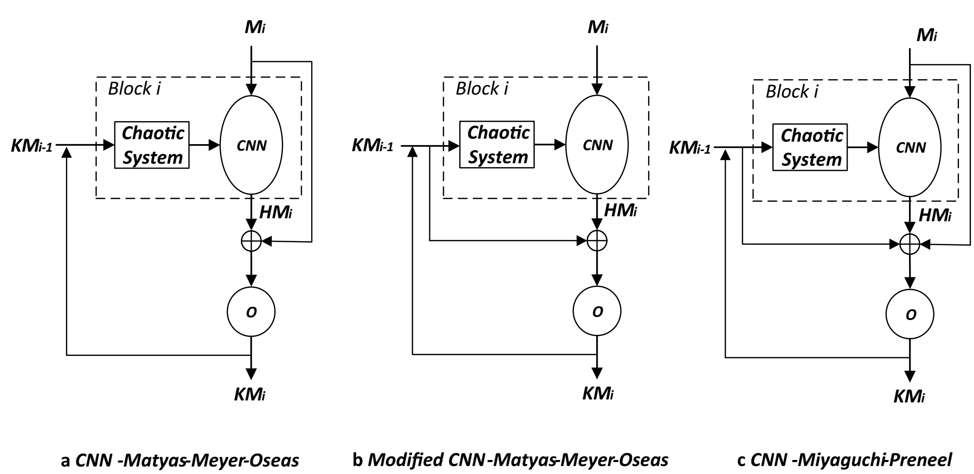


Fig. 4 The proposed Merkle–Dåmgard compression functions based on CNN with output schemes

3 Chaotic neural network structure of the proposed keyed hash functions

This paper proposes two keyed hash functions based on Chaotic Neural Network (CNN), and for each one, three output schemes are suggested as presented in Fig. 4. The first CNN hash function uses two-layer neural network structure (named Structure 1), whereas the second hash function uses one-layer neural network followed by a combination of Non-Linear (NL) functions (named Structure 2). The next sub-section describes the three suggested output schemes based on Matyas–Meyer–Oseas (Bartkewitz 2009; Brachtl et al. 1990; Matyas 1985) and Miyaguchi–Preneel models (Miyaguchi et al. 1989, 1990; Prencel et al. 1989; Preneel et al. 1993).

3.1 Suggested output schemes

*Matyas–Meyer–Oseas (MMO)*output scheme: In this output scheme, the message block M_i is xored with the chaining variable HM_i, which is the output of the CNN that takes as inputs M_i and the output of the Chaotic System (Fig. 4a). The state value KM_{i-1} is the key of the Chaotic System. Due to the possible different bit-length, an output function O precedes the generation of the final output KM_i, which represents the key of the next block, which is as follows:

$$KM_i = O(HM_i \oplus M_i) \tag{2}$$

where i: the block index; $1 \le i \le q$.

$for\ i = 1\ :\ KM_0 = K$: the secret key.

$for\ i = q\ :\ KM_q = h$: the final hash value.

Modified Matyas–Meyer–Oseas (MMMO) output scheme: This output scheme is similar to MMO output scheme except for the xor operation. Indeed in this case, HM_i is xored with KM_{i-1} (Fig. 4b), where the final output KM_i is defined by:

$$KM_i = O(HM_i \oplus KM_{i-1}) \tag{3}$$

where i: the block index; $1 \le i \le q$.

$for\ i = 1\ :\ KM_0 = K$: the secret key.

$for\ i = q\ :\ KM_q = h$: the final hash value.

Miyaguchi–Preneel (MP) output scheme: This output scheme can be considered as an extension of the MMO output scheme, where KM_{i-1} is also added to the xor operation between M_i and HM_i (Fig. 4c). The final output KM_i is defined by:

$$KM_i = O(HM_i \oplus M_i \oplus KM_{i-1}) \tag{4}$$

where i: the block index; $1 \le i \le q$.

$for\ i = 1\ :\ KM_0 = K$: the secret key.

$for\ i = q\ :\ KM_q = h$: the final hash value.

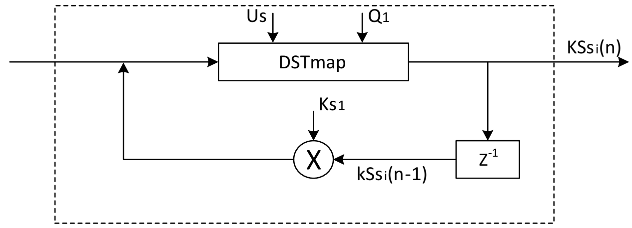


Fig. 5 The structure of the chaotic system

3.2 Chaotic system

The proposed Chaotic System is used to generate the parameters concerning the CNN compression function introduced in Fig. 4. It comprises the DSTmap with one recursive cell (delay equal to 1) (Fig. 5). Its outputs are defined as follows:

$$KSs(n) = DSTmap(KSs(n-1), Q1)$$
$$= \begin{cases} 2^N \times \frac{KSs(n-1)}{Q1} & if\ 0 < KSs(n-1) < Q1 \\ 2^N - 1 & if\ KSs(n-1) = Q1 \\ 2^N \times \frac{2^N - KSs(n-1)}{2^N - Q1} & if\ Q1 < KSs(n-1) < 2^N \end{cases} \tag{5}$$

where *Q1*, the control parameter, and $KSs(n)$ range from 1 to $2^N - 1$. N is the finite precision and is equal to 32 bits. The secret key K, used for the first block M_1, is composed of the necessary parameters and initial conditions of the simplified version of the Chaotic Generator patent (El Assad and Noura 2014) and it is given by the following equation:

$$K = \{KSs(0), Ks, KSs(-1), U_s, Q1\} \tag{6}$$

where $KSs(0)$ and $KSs(-1)$ are the initial values, U_s is an additional initial value used only to generate the first sample, Ks is the coefficient, and $Q1$ is the control parameter of the Chaotic System. The components of K are samples of 32 bits length and its size is given as follows:

$$\begin{aligned} |K| &= |KSs(0)| + |Ks| + |KSs(-1)| + |U_s| + |Q1| \\ &= 160\ bits \end{aligned} \tag{7}$$

3.3 Keyed hash functions based on two-layer CNN structure (Structure 1)

The architecture of the proposed keyed hash function is composed of the defined chaotic system (DSTmap with delay 1) and two-layer CNN (Fig. 6) (Abdoun et al. 2016). Each layer is composed of eight neurons, where each one uses a chaotic activation function (Fig. 7). The chaotic activation function consists of two xored chaotic maps: a Discrete Skew Tent map (DSTmap) and a Discrete Piecewise Linear Chaotic map (DPWLCmap) (Desnos et al. 2014; El Assad 2012;

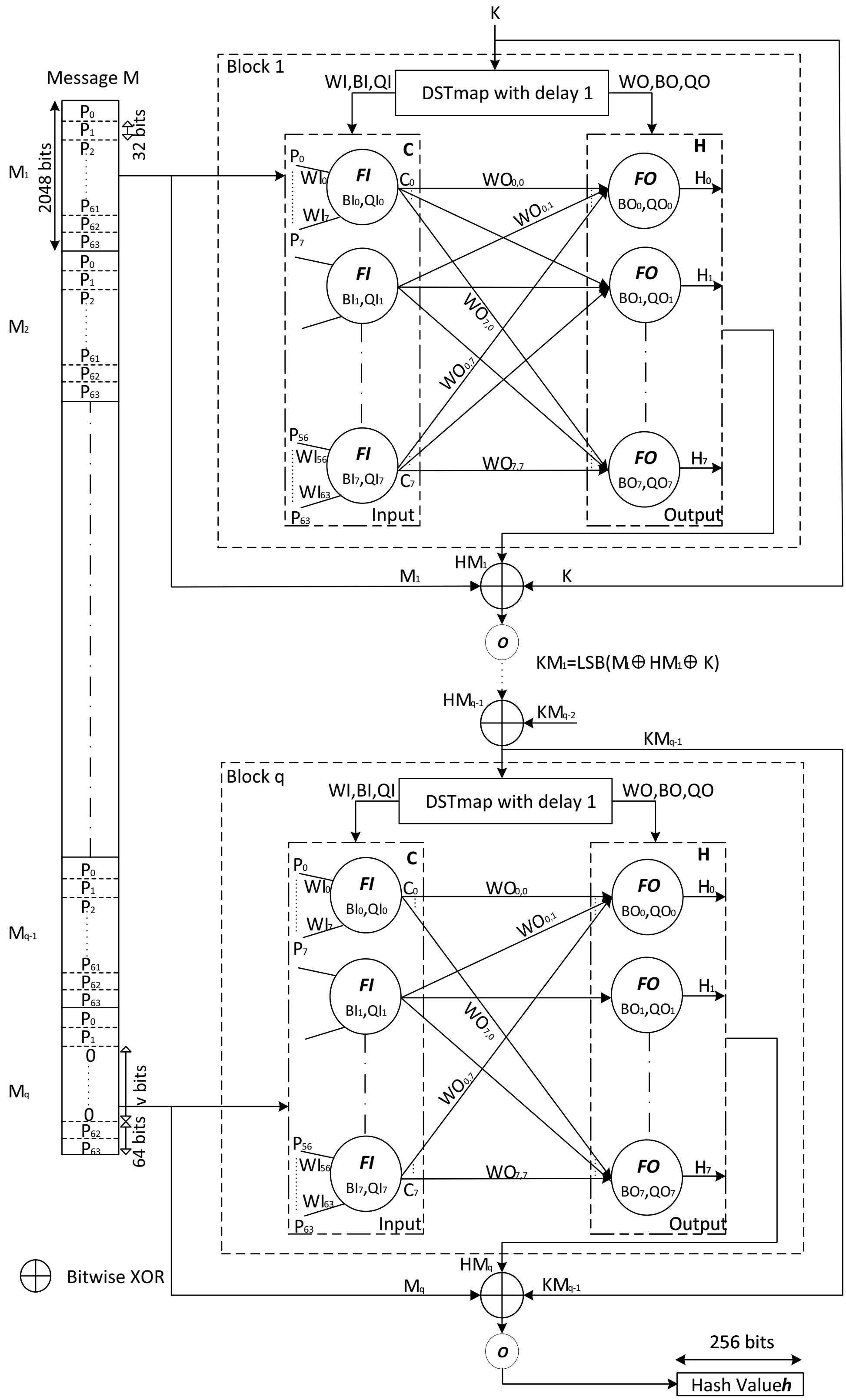


Fig. 6 The proposed keyed hash function based on two-layer CNN with MP output scheme

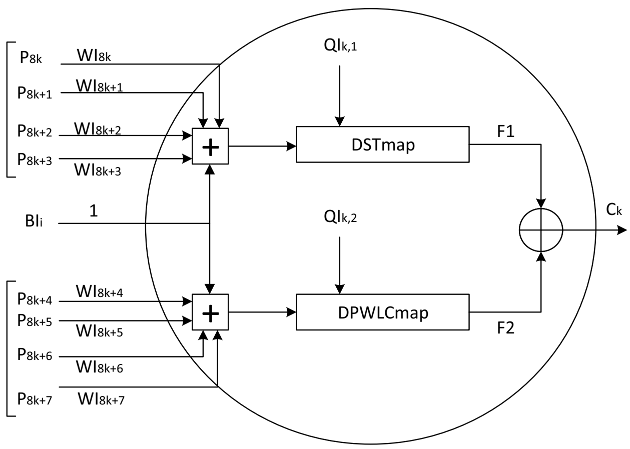


Fig. 7 A detailed structure of the k^{eme} neuron in input layer of the two proposed hash functions

El Assad and Noura 2014). Each map is iterated T times (by experiment, we choose the transient phase tr = 30 for Structure 1 and tr = 20 for Structure 2), before generating the first useful sample for maintaining the randomness of the output. The outputs of the DPWLCmap are defined as follows:

$$KSp(n) = DPWLCmap(KSp(n-1), Q2)$$
$$= \begin{cases} 2^N \times \frac{KSp(n-1)}{Q2} & if\ 0 < KSp(n-1) \le Q2 \\ 2^N \times \frac{KSp(n-1)-Q2}{2^{N-1}-Q2} & if\ Q2 < KSp(n-1) \le 2^{N-1} \\ 2^N \times \frac{2^N - KSp(n-1)-Q2}{2^{N-1}-Q2} & if\ 2^{N-1} < KSp(n-1) \le 2^N - Q2 \\ 2^N \times \frac{2^N - KSp(n-1)}{Q2} & if\ 2^N - Q2 < KSp(n-1) \le 2^N - 1 \\ 2^N - 1 - Q2 & otherwise \end{cases} \tag{8}$$

where $Q2$ is the control parameter of DPWLCmap and ranges from 1 to 2^{N-1} (N = 32 bits).

It should be noted that in the proposed structures, the padded message M is divided into q blocks, where M_i $(1 \le i \le q)$ is the i^{eme} input block of the message M, KM_i $(0 \le i \le q-1)$ is the i^{eme} key, and HM_i $(1 \le i \le q)$ is the i^{eme} hash value of block M_i $(1 \le i \le q)$. For the first block M_1, $K = KM_0$ is the secret key (El Assad 2012). For the final block M_q, h is the final hash value of the entire message M (Fig. 6).

For each block M_i at the input layer, each neuron has 8 input-data: $P_j(j = 0, ..., 7)$ for neuron 0, $P_j(j = 8, ..., 15)$ for neuron 1 and so on until reaching $P_j(j = 56, ..., 63)$ for neuron 7. Each $P_j(j = 0, ..., 63)$ is weighted by $WI_j(j = 0, ..., 63)$, where both are the samples (integer values) of 32 bits length. The Chaotic System generates the necessary samples (Key Stream (KS)) to supply the CNN of each block i, which is as follows:

$$KS = \{WI, BI, QI, WO, BO, QO\} \tag{9}$$

and its size is written as:

$$|KS| = |WI| + |BI| + |QI| + |WO| + |BO| + |QO| \\ = 176\ samples \tag{10}$$

where $|WI| = 64$ samples, $|BI| = 8$ samples, $|QI| = 16$ samples, $|WO| = 64$ samples, $|BO| = 8$ samples, and $|QO| = 16$ samples, each of the 32 bits length.

The chaotic activation function of each neuron k $(k = 0, ..., 7)$ for the input layer is now explained as an example, (the activation function for the output layer has similar description). As we can see in Fig. 7, the first four inputs $P_j(j = 8k, ..., 8k+3)$ are weighted by the $WI_j(j = 8k, ..., 8k+3)$ and then added together with the bias BI_k (weighted by 1) to form the input of DSTmap. The second four inputs $P_j(j = 8k+4, ..., 8k+7)$ are weighted by $WI_j(j = 8k+4, ..., 8k+7)$ and then added together with the same bias BI_k to form the input of DPWLCmap. $QI_{k,1}$ and $QI_{k,2}$ are the control parameters of DSTmap and DPWLCmap, respectively. The biases BI_k are necessary in case the input message is null.

The outputs of the chaotic activation function are denoted C_k for the input layer, which is given by Eq. (11), and H_k for the output layer, which is given by Eq. (12).

$$C_k = mod\{[F1 + F2], 2^N\}\ where$$
$$\begin{cases} F1 = DSTmap\{mod\left(\left[\sum_{j=8k}^{8k+3}(WI_j \times P_j)\right] + BI_k,\ 2^N\right), \\ \qquad QI_{k,1}\} \\ F2 = DPWLCmap\{mod\left(\left[\sum_{j=8k+4}^{8k+7}(WI_j \times P_j\right)\right] + BI_k, \\ \qquad 2^N), QI_{k,2}\} \end{cases} \tag{11}$$

$$H_k = mod\{[G1 + G2, 2^N]\}\ where$$
$$\begin{cases} G1 = DSTmap\{mod\left(\left[\sum_{j=0}^{3}(WO_{k,j} \times C_j)\right] + BO_k, 2^N\right), \\ \qquad QO_{k,1}\} \\ G2 = DPWLCmap\{mod(\left[\sum_{j=4}^{7}(WO_{k,j} \times C_j)\right] + BO_k, \\ \qquad 2^N), QO_{k,2}\} \end{cases} \tag{12}$$

where $k = 0, 1, ..., 7$.

The outputs C_k of the input layer, weighted by $WO_{k,k}(k = 0, ..., 7)$, and the output biases $BO_k(k = 0, ..., 7)$, weighted by 1, are the inputs of the activation function of the output layer. Both $WO_{k,k}$ and BO_k are samples of 32 bits length. For each neuron, DSTmap and DPWLCmap are iterated once. The output $HM_i (i = 1, ..., q)$ of each block is the concatenation vector of $H_k (k = 0, ..., 7)$ (Fig. 6). Then, the final hash value of length 256 bits is given by the following equation:

$$\begin{aligned} h &= O[KM_{q-1} \oplus HM_q \oplus M_q] \\ &= O[(KM_{q-2} \oplus HM_{q-1} \oplus M_{q-1}) \oplus HM_q \oplus M_q] \\ &= \cdots = O[(K \oplus HM_1 \oplus M_1) \oplus HM_2 \\ &\oplus M_2 \oplus \cdots \oplus HM_q \oplus M_q] \end{aligned} \tag{13}$$

where O is the Least Significant Bit (LSB) output function.

3.4 Keyed hash functions based on one-layer CNN with non-linear output layer (Structure 2)

Thus, to efficiently increase the hash throughput while keeping the necessary security requirements, we replace the output layer neural network of Fig. 6 by a combination of non-linear functions used in the standard SHA-2. However in our implementation, the round constant $K_i (i = 0, ..., 63)$ and the message schedule array $W_i (i = 0, ..., 63)$ are not useful (Fig. 8). As we can see in the Fig. 8, the non-linear functions take eight 32-bit inputs $D_k (k = 0, ..., 7)$ and generates eight 32-bit outputs $H_k (k = 0, ..., 7)$. The four boxes (Ch, Ma, $\Sigma 0$, and $\Sigma 1$) combine the input data in non-linear ways to generate H_0 and H_4, while the other outputs $H_k (k = 1, 2, 3, 5, 6, 7)$ are connected directly to D_k, which is as follows: $H_k = D_{k-1}$ $(k = 1, 2, 3, 5, 6, 7)$. These non-linear functions are defined as follow (Secure Hash Standard and FIPS Publication 2002):

$$\begin{cases} Ch(D_4, D_5, D_6) = (D_4 \wedge D_5) \oplus (\neg D_4 \wedge D_6) \\ Ma(D_0, D_1, D_2) = (D_0 \wedge D_1) \oplus (D_0 \wedge D_2) \oplus (D_1 \wedge D_2) \\ \Sigma 0(D_0) = ROTR^2(D_0) \oplus ROTR^{13}(D_0) \oplus ROTR^{22}(D_0) \\ \Sigma 1(D_4) = ROTR^6(D_4) \oplus ROTR^{11}(D_4) \oplus ROTR^{25}(D_4) \\ ROTR^n(x) = (x \gg n) \vee (x \ll (32 - n)) \end{cases} \tag{14}$$

where $\wedge$: *AND logic*, $\neg$: *NOT logic*, $\oplus$: *XOR logic*, $\vee$: *OR logic*, $\gg$: *BinaryShift Right operation, and* $\ll$: *Binary Shift Left operation.*

The structure of the proposed CNN is given in Fig. 9. To supply the CNN, the Chaotic System generates the necessary samples (Key Stream (KS)) of each block i, which are as follows:

$$KS = \{WI, BI, QI, WO\} \tag{15}$$

and its size is given as follows:

$$\begin{aligned} |KS| &= |WI| + |BI| + |QI| + |WO| \\ &= 96 \textit{ samples} \end{aligned} \tag{16}$$

where $|WI| = 64$ samples, $|BI| = 8$ samples, $|QI| = 16$ samples, and $|WO| = 8$ samples, each of 32 bits length. The outputs $C_k (k = 0, ..., 7)$ of the chaotic activation function given by Eq. (11) are weighted by $WO_{k,k} (k = 0, ..., 7)$ to form the inputs of the NL layer. The outputs $H_k (k = 0, ..., 7)$ are given by Eq. (17).

$$\begin{cases} H_0 = Ch(D_4, D_5, D_6) \oplus D_7 \oplus \Sigma 1(D_4) \oplus Ma(D_0, D_1, D_2) \\ \quad \oplus \Sigma 0(D_0) \\ H_1 = D_0, \ H_2 = D_1, \ H_3 = D_2 \\ H_4 = Ch(D_4, D_5, D_6) \oplus D_7 \oplus \Sigma 1(D_4) \oplus D_3 \\ H_5 = D_4, \ H_6 = D_5, \ H_7 = D_6 \end{cases} \tag{17}$$

We iterate the non-linear functions until the necessary security requirements are met. From experimental results (given in performance analysis paragraph), the number of rounds r equals to 8, which is sufficient. The final hash value h of length 256 bits is given in Eq. (13).

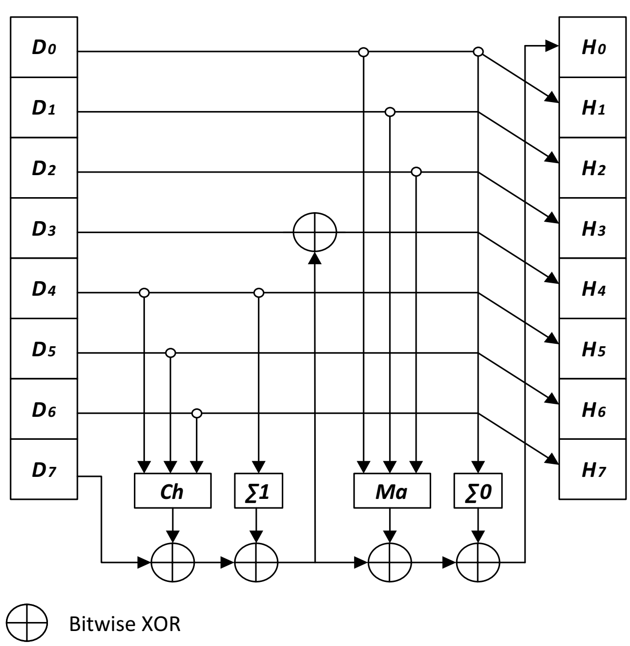
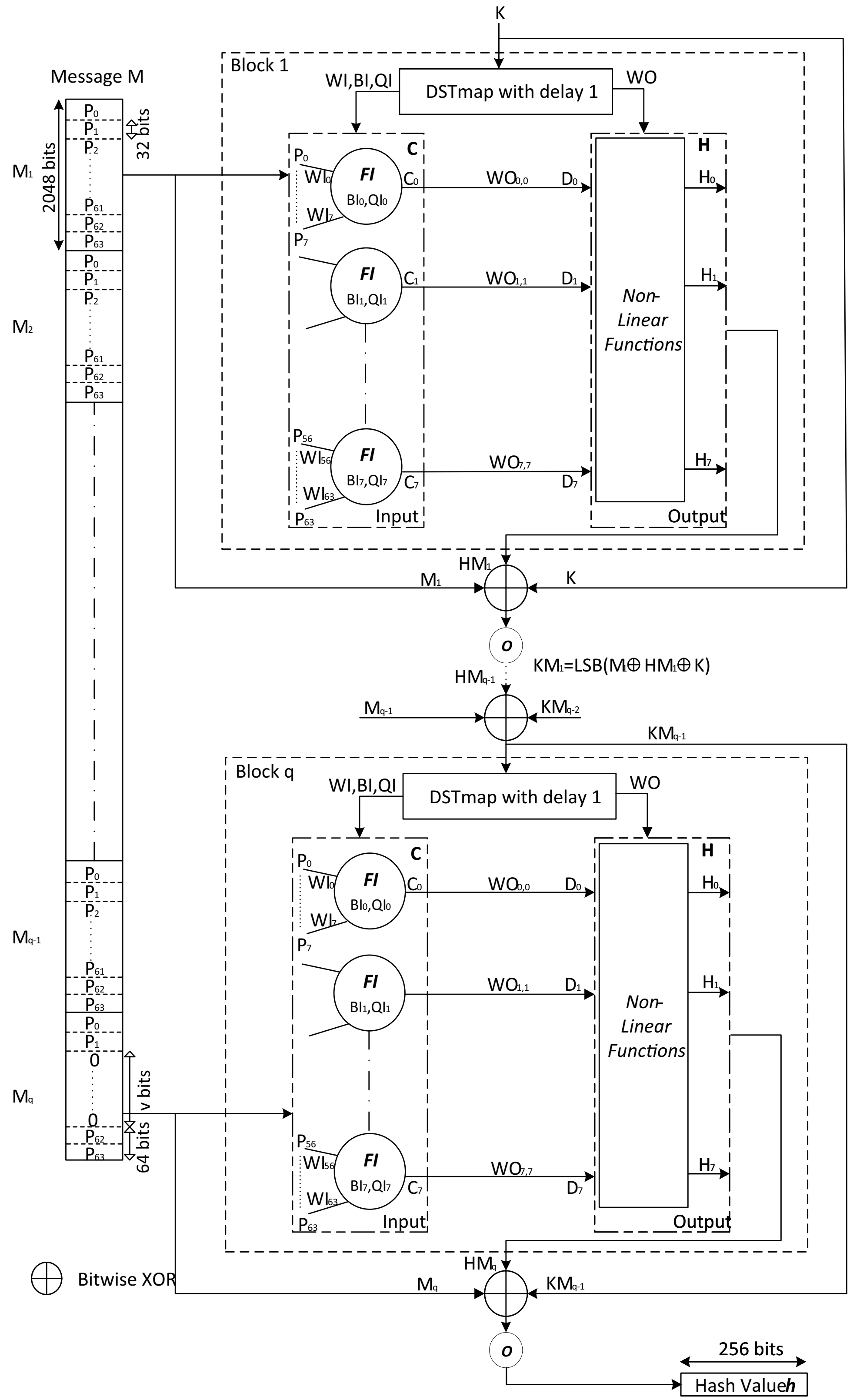


Fig. 8 Non-linear functions

4 Performance analysis

To evaluate the performance, in terms of security and speed computation, of the two proposed structures for each suggested output schemes, we perform the following experiments and analysis. Then, we compare their performance with most chaos-based hash functions in the literature and SHA-2. First, the one-way property (preimage resistance) is showed and then

Fig. 9 The proposed keyed hash function based on one-layer NL CNN with MP output scheme



the statistical tests, the brute force, and cryptanalytical attacks of the proposed hash functions are analyzed.

4.1 One-way property

In the two proposed structures, we will show that it is extremely difficult to compute the message M and the secret key K when only the hash value h is known. For the first structure, the hash H is written in a general form, which is as follows (Eqs. 11, 12):

$$\begin{aligned} H &= G[(WO \times C + BO), QO] \\ &= G[(WO \times F((WI \times P + BI), QI), QO)] \end{aligned} \tag{18}$$

For the second structure, the hash H can be written as follows:

$$\begin{aligned} H &= NL^r(WO \times C) \\ &= NL^r[WO \times F((WI \times P + BI), QI)] \end{aligned} \tag{19}$$

A brute force attack, as defined in Sect. 4.3.1, tries for a given secret key K to find a message M, of which its hash is equal to a given hash value. The attacker needs to try, on average, 2^{u-1} values of M, to find the desired hash value h. As u is the length of the hash value equal to 256 bits in the two proposed structures, then according to today's computing ability, this attack is infeasible (Lian et al. 2006a, b; Xiao et al. 2005; Yi 2005).

4.2 Statistical tests

This paragraph lists down the analysis of the following tests: Collision resistance, Distribution of hash value, Sensitivity of hash value h to the message M, Sensitivity of hash value h to the secret key K, and Diffusion effect.

4.2.1 Analysis of collision resistance

This test is usually conducted to evaluate the quantitative analysis of collision resistance (Xiao et al. 2005; Wong 2003). First, the hash value h of a random message is generated and stored in the ASCII format. Next, a bit in the message is randomly selected, toggled, and then a new hash value h' is generated and stored in the ASCII format. The two hash values are represented by: $h = \{c_1, c_2, ..., c_s\}$ and $h' = \{c'_1, c'_2, ..., c'_s\}$, where c_i and c'_i are the ith ASCII character of the two hash values h and h', respectively. The size s of the hash value in the ASCII code is equal to $s = \frac{u}{k=8} = 32$ characters. The two hash values are compared with each other and the number of characters with the same value at the same location, namely the number of hits ω, is counted according to the following:

$$\begin{aligned} &\omega = \sum_{i=1}^{s=32} f(T(c_i), T(c'_i)) \\ &where\, f(x,y) = \left\{ \begin{array}{ll} 1 & if \;\; x = y \\ 0 & if \;\; x \neq y \end{array} \right\} \end{aligned} \tag{20}$$

Table 1 Number of hits ω according to the number of rounds r of Structure 2 for 2048 tests

Number of rounds		Number of hits ω										
r		0	1	2	3	4	16	17	24	25	26	28
Output schemes												
MMO	1	1778	240	24	1	0	0	0	5	0	0	0
	2	1784	248	11	0	0	5	0	0	0	0	0
	4	1790	243	14	1	0	0	0	0	0	0	0
	8	1825	207	15	1	0	0	0	0	0	0	0
	16	1811	222	14	1	0	0	0	0	0	0	0
	24	1817	225	6	0	0	0	0	0	0	0	0
MMMO	1	1757	232	11	0	0	0	0	45	1	2	0
	2	1725	259	15	1	0	45	3	0	0	0	0
	4	1828	206	14	0	0	0	0	0	0	0	0
	8	1800	237	10	1	0	0	0	0	0	0	0
	16	1801	233	14	0	0	0	0	0	0	0	0
	24	1810	230	7	1	0	0	0	0	0	0	0
MP	1	1744	238	17	1	0	0	0	46	0	1	1
	2	1773	215	11	1	0	45	3	0	0	0	0
	4	1783	251	13	1	0	0	0	0	0	0	0
	8	1817	215	16	0	0	0	0	0	0	0	0
	16	1813	218	16	1	0	0	0	0	0	0	0
	24	1815	226	7	0	0	0	0	0	0	0	0

Table 2 Number of hits ω regarding the proposed structures with the three output schemes for 2048 tests

	Output schemes	Number of hits ω			
		0	1	2	3
Structure 1	MMO	1833	200	15	0
	MMMO	1799	237	12	0
	MP	1803	232	13	0
Structure 2	MMO	1825	207	15	1
r = 8	MMMO	1800	237	10	1
	MP	1817	215	16	0
Structure 2	MMO	1817	225	6	0
r = 24	MMMO	1810	230	7	1
	MP	1815	226	7	0

Table 3 Number of hits ω of the proposed structures with MP output scheme for J = 512, 1024, and 2048 tests

	Number of tests	Number of hits ω			
		0	1	2	3
Structure 1	512	444	64	4	0
	1024	905	111	8	0
	2048	1803	232	13	0
Structure 2	512	446	62	4	0
r = 8	1024	899	117	8	0
	2048	1817	215	16	0
Structure 2	512	452	58	2	0
r = 24	1024	905	116	3	0
	2048	1815	226	7	0

where the function $T(.)$ converts the entries to their equivalent decimal values.

For J independent experiments and under the assumption of uniform and random distribution of hash value, the theoretical number of tests denoted by $W_J(\omega)$ with a number of hits $\omega = 0, 1, 2, ..., s$, is given by (Zhang et al. 2007):

$$W_J(\omega) = J \times Prob\{\omega\} = J \frac{s!}{\omega!(s-\omega)!} \left(\frac{1}{2^k}\right)^{\omega} \left(1 - \frac{1}{2^k}\right)^{s-\omega} \tag{21}$$

Thus, to find the optimal number of round r for Structure 2, we calculate, using the Eq. (20), the number of hits ω according to r (r = 1, 2, 4, 8, 16, 24) in the worst case, where the number of tests J = 2048 tests.

As we can see from the results obtained in Table 1, with MMO output scheme, as an example, for r = 8 rounds, there are zero hits for 1825 tests, one hit for 207 tests, two hits for 15 tests, and three hits for 1 test. For r = 24 rounds, there are zero hits for 1817 tests, one hit for 225 tests, and two hits for six tests. Similar results are obtained for other output schemes as well. The number of rounds r equals 8, whereas 24 seems to be adequate for the three output schemes. We choose $r = 24$, for more robustness and the number $r = 8$ is a compromise between robustness and hash throughput.

Table 2 represents the number of obtained hits ω, for the proposed structures for the three output schemes, with J = 2048 tests and for r = 8, 24 rounds for Structure 2. We remark that, for r = 8 rounds, the obtained results with Structure 2 are similar to the results obtained with Structure 1, irrespective of the considered output scheme. For $r = 24$ rounds, the obtained results with Structure 2, as are slightly bit better than that of Structure 1.

Table 4 Theoretical values of the number of hits ω according to the number of tests J

		ω				
		0	1	2	3	32
J	512	451.72	56.68	3.44	0.13	4.42×10^{-75}
	1024	903.45	113.37	6.89	0.27	8.84×10^{-75}
	2048	1806.91	226.74	13.78	0.54	1.76×10^{-74}

Table 5 Mean, mean/charater, minimum, and maximum of the absolute difference d for the proposed structures with the three output schemes and J = 2048 tests

	Output schemes	Mean	Mean/character	Minimum	Maximum
Structure 1	MMO	2721.43	85.04	1736	3723
	MMMO	2764.05	86.37	1829	3757
	MP	2633.17	82.28	1471	3779
Structure 2	MMO	2616.94	81.77	1559	3574
r = 8	MMMO	2854.76	89.21	1845	4195
	MP	2861.93	89.43	1707	3951
Structure 2	MMO	2746.07	85.81	1696	3807
r = 24	MMMO	2856.03	89.25	1545	3981
	MP	2615.44	81.73	1540	3671

Table 6 Mean, mean/charater, minimum, and maximum of the absolute difference d for the proposed structures with MP output scheme and J = 512, 1024, and 2048 tests

	Number of tests	Mean	Mean/character	Minimum	Maximum
Structure 1	512	2637.00	82.40	1471	3779
	1024	2637.99	82.43	1471	3779
	2048	2633.17	82.28	1471	3779
Structure 2	512	2872.23	89.75	1828	3872
r = 8	1024	2868.04	89.62	1707	3951
	2048	2861.93	89.43	1707	3951
Structure 2	512	2603.32	81.35	1764	3671
r = 24	1024	2620.85	81.90	1626	3671
	2048	2615.44	81.73	1540	3671

Thus, to evaluate the influence of the test number J (J = 512, 1024, and 2048 tests) on the number of hits, we calculate ω for the proposed structures with MP output scheme, and for $r = 8$, 24 rounds for the second structure. The obtained results presented in Table 3 for Structures 1 and 2 with $r = 8$ rounds are similar, while with $r = 24$ rounds of Structure 2, the number of hits is smaller than that of the other cases. We remark that the number of hits increases with the number of tests J. These results are in sync with the theoretical values of $W_J(\omega)$ calculated from Eq. (21) and are represented in Table 4.

The collision resistance is also quantified by the absolute difference d of two hash values given by Eq. (22). We evaluated and presented the mean, mean/character, minimum, and maximum of d for the two proposed hash functions in Tables 5 and 6.

$$d = \sum_{i=1}^{s=32} |T(c_i) - T(c_i')| \tag{22}$$

From the results given in Table 5 for J = 2048 tests, we observe that the mean/character value with the MMO output scheme for Structure 1 (mean/character = 85.04) and Structure 2—r = 24 rounds (mean/character = 85.81) are close to the expected value 85.33 given in Eq. (23). The results presented in Table 6 with J (J = 512, 1024, and 2048 tests) show that, when J is increasing, the mean/character converge to the expected value E. For two hash, i.e., $h = \{c_1, c_2, ..., c_s\}$ and $h' = \{c_1', c_2', ..., c_s'\}$, with independent and uniformly distributed ASCII character having equal probabilities, the expected value of the mean/character is calculated by (Preneel 1993):

$$E[T(c_i) - T(c_i')] = \frac{1}{3} \times L = 85.33 \tag{23}$$

where $T(c_i)$ and $T(c_i') \in \{0, 1, 2, ..., 255\}$ and $L = 256$ (L is the number of levels).

4.2.2 Distribution of hash value

A hash function H should produce uniform distribution of hash value h. To verify this property, we perform the following test: for a given message M, *With the wide application of Internet and computer technique, information security becomes more and more important. As we know, hash function is one of the cores of cryptography and plays an important role in information security. Hash function takes a message as input and produces an output referred to as a hash value. A hash value serves as a compact representative image (sometimes called digital fingerprint) of input string and can be used for data integrity in conjunction with digital signature schemes.*, we calculate its hash value h, for the proposed Structure 1 with MP output scheme, before drawing two-dimensional graphs. The first graph shows the ASCII values of the message according to their index positions (Fig. 10a). The second graph exhibits the hexadecimal values of the hash value h according to their index positions (Fig. 10b). As we can see, the distribution of original message is mostly localized around a small area, while the distribution of hexadecimal values spreads around the entire area. This property of hash value h must be true under the worst case of null input message (Figs. 10c, 10). Similar results are obtained for the two proposed hash functions with their different output schemes.

4.2.3 Sensitivity of hash value h to the message M

An efficient hash function H should be extremely sensitive to any input message M, which means that any slight change in the input message should produce a completely different hash value h_i. To verify this property, we calculate, for a given secret key K, the hash value h_i in hexadecimal format, the number of bits changed $B_i(h, h_i)$ (bits), and the sensitivity of the hash value h to the original message M measured by Hamming Distance $HD_i(h, h_i)(\%)$:

$$B_i(h, h_i) = \sum_{k=1}^{|h|} [h(k) \oplus h_i(k)] \; bits \tag{24}$$

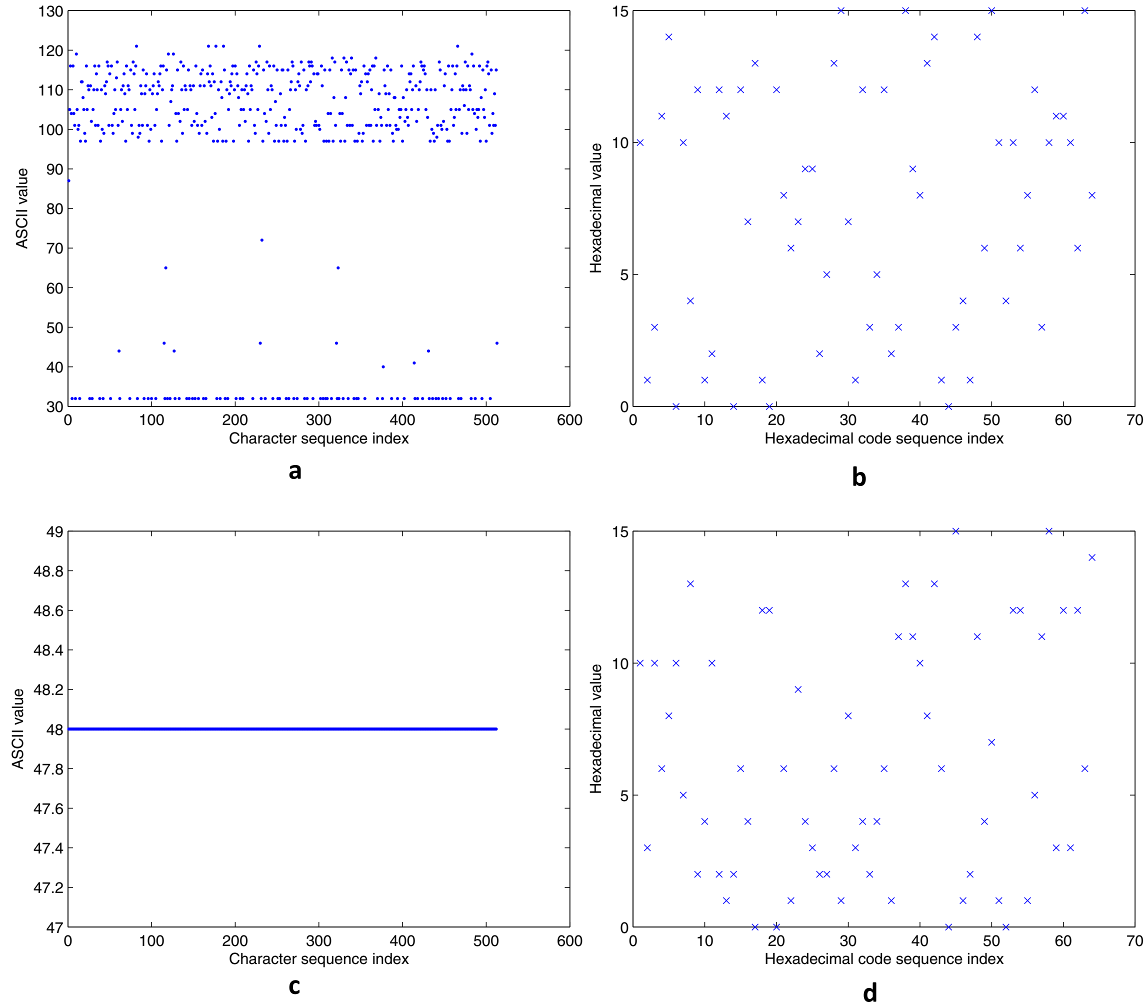


Fig. 10 Distribution of hash value for Structure 1 with MP output scheme

$$HD_i(h, h_i)_{\%} = \frac{B_i(h, h_i)}{|h|} \times 100\% \quad (25)$$

The message variants are obtained under the following conditions:

Condition 1: The original message M is the one given in Sect. 4.2.2.

Condition 2: We change the first character W in the original message to X.

Condition 3: We change the word *With* in the original message to *Without*.

Condition 4: We change the *dot* at the end of the original message to *comma*.

Condition 5: We add a *blank space* at the end of the original message.

Condition 6: We exchange the first message block M_1, *with the wide application of internet and computer technique, information security becomes more and more important. As we know, hash function is one of the cores of cryptography and plays an important role in information security. Hash function takes a mes*, with the second message block M_2, *sage as input and produces an output referred to as a hash value. A hash value serves as a compact representative image (sometimes called digital fingerprint) of input string and can be used for data integrity in conjunction with digital signature schemes.*

In Tables 7, 8, and 9, we present the obtained results of h_i, B_i, and $HD_i(\%)$ under each condition for the two proposed hash functions with their output schemes, i.e., MMO, MMMO, and MP.

In Table 10, we reassessed the obtained results and even for a single test, the results were inside the normal range. Therefore, the proposed hash functions have high message sensitivity. These results were in sync with precision in the diffusion test, which was realized over a large number of tests.

Table 7 Sensitivity of hash value to the message for the proposed structures with MMO output scheme

	Message variants	Hexadecimal hash values	B_i	HD_i%
Structure 1	1	bedf7967520105d114e2cdf3399f52394a53e276bb104307345bacf93e317ef6	-	-
	2	48def8102016f2e3a5f8e7d8dc782b5b4e3e930cc207f925176ab87f380ad03d	125	48.82
	3	d486760a20882b71746704d35ffdcd0f07c5ffe23cad86bd8117737205dd163c	127	49.60
	4	b8bdc0f41686695f582a4d2e5b37f9b98813ab9c1cc42ba64024ee1769b422e7	113	44.14
	5	e82980358f548044d0328f613a640fe23d1cb8465325dc223a7881ae65ef360d	136	53.12
	6	76e33a9b2f6599542c557bdac7bee94f25dddbc615b222653201fd484ae8ce1c	130	50.78
	Average	–	126.2	49.29
Structure 2	1	1d6238873699dd1c252e02c88e1d2a380d9b5ea8e6c09c788fa4d3955b959975	-	-
r = 8	2	6ea75e2045e994639a7d547ece06a6a399397a6cc501f52ffe4d4727030bedb2	128	50
	3	72dcdf42b4c6d47352b75a7f2a7bbb3d9144c519e99e10cdd1a04237433730bf	131	51.17
	4	308f5d0ce0c0a7b140cc7c179ae4697fba8ea270433c50b015095877c2047267	141	55.07
	5	ebeac17eb7b2d842ef21971f6c9da59771f7a0e0612ecd96e37a97691eb0c1cd	135	52.73
	6	1e56b053ebe94fc4eb36f8ed74981da9c01a861cdbe93b3c176ecfab8102a336	122	47.65
	Average	–	131.4	51.32
Structure 2	1	af5e7ca7c83a72c77f0e9b7d47df11b0f66cadc862d6f522d592dc5ad9bae938	-	-
r = 24	2	46f051a065a716de24405e782adaccb29b3a85b0b75b34a9ba0757644bcdcc33	127	49.60
	3	4f9c3863d40a2a1094d8d7483acc0724cbd9f2b68648db7fe8c0609327c8f318	130	50.78
	4	2b4ff84285427b479d6948d20dd00eb389956dd325894d6036e510b99b20055d	126	49.21
	5	15e6695fca52780d8694f83b0bba7b5fb43bc29329e78018287bd87776cdf459	132	51.56
	6	d5a2f663581034f865ba7a2bc93d29232b0f57f99f8d33a8ef50e1070c84ae88	133	51.95
	Average	–	129.6	50.62

Table 8 Sensitivity of hash value to the message for the proposed structures with MMMO output scheme

	Message variants	Hexadecimal hash values	B_i	HD_i%
Structure 1	1	719adf0e0cdf5b149edc54efdbc09bb6df5a0ce3d3ac9bccc39ac5a64ea65531	-	-
	2	a9472c054759a85c0c172e27bc1b957f09488c40329424c48aac1d1141dd8297	132	51.56
	3	1bee2969559824929f8d53fda2c541288a4a04491a0a11670b3b907fa0d5dd91	119	46.48
	4	27c29f1e040d922b31559e0e3f4e36edc9bdad55cf058d7f0eaa7a9f9eda6d98	124	48.43
	5	65489772dff489621f3188237c1ff84c8bf686d7a4f5c6ff1e114b740c72c922	133	51.95
	6	64755b1267f7243f2dbf243d698db2dd40ff63df7375f645886d064b2d05fdb2	135	52.73
	Average	-	128.6	50.23
Structure 2	1	a594a994aa162adca654e889dea0e6344190aa02328302465570df8f0084f5e6	-	-
r = 8	2	9d698ca7855b104a526a075a36cbf158da31c872257db0d8d589502f60a8115f	135	52.73
	3	fe77f2939687110cc6f383ed0ac2990e89b513ed1425c2a2ded04ce8ab26331e	129	50.39
	4	d906ae7eaf90974ce664e8adb535e71b798873bfdc77827e3715715bb6b5cbb5	113	44.14
	5	f1ea83b16b7fecd5d523573d35f52a424e35a8dc38af6e013f9d2020f0825c35	136	53.12
	6	29e7a1e00480ff09b86d357982d28ab641758c071cee1a2095452cb583740194	121	47.26
	Average	–	126.8	49.53
Structure 2	1	6abbd825d6b17184a5fc558670f9f78d91b3812c899c8a062ef855507b4a81e5	-	-
r = 24	2	c7c8654da6fd4fb838f8f9bea4baa223b8298a1c1e0cda2181a23e612cbb8446	122	47.65
	3	0fe4ee2f96a9092f539a4fd229466b381a794db148da178e635022d9a690eabf	130	50.78
	4	9b01f686addb2e2f6dbd7046b985b4ae1b5b39a7da3aec544ecb6c8efd310a00	128	50.00
	5	9901ff0d69138df2f70a5930ede63447875c859830bc87e4164a83b083a6a193	131	51.17
	6	c5035924044140a2009837907fba710d05efbcbe12ff9c1d14d9090961bd054e	113	44.14
	Average	–	124.8	48.75

Table 9 Sensitivity of hash value to the message for the proposed structures with MP output scheme

	Message variants	Hexadecimal hash values	B_i	HD_i%
Structure 1	1	a005e50f9673ecee6e80c07c550e53f8a950cb4a91176a2a340b5822ec2f28c4	-	-
	2	d4ecfadcc796f46d63762eb8f0c7af6233ded0d61ea901541db1f8890f999755	141	55.07
	3	3f8b28e72a453ad31e798a60ec46b64ab4eb3e95674b28d535a5d2feb8a7cdd8	139	54.29
	4	b40f8be0ee3c28fc7c76578d6e8b49f56ea25aa0c2944475691746a7c2f23387	129	50.39
	5	c0f0b6c0fee17303c94ab30ad6d7b1ecd50d9606e4fab176e726b20a3c229b5d	139	54.29
	6	551eb7f04ec0ae2f0ceec2bb451a2b67682305697a0ffef418e221bdaad4a09c	129	50.39
	Average	–	135.40	52.89
Structure 8	1	31882869cce69d7734f0078d29f297841b99d3f9786a1cf522688de9561826ee	-	-
r = 8	2	d8da2ae1aacca231e26931237f8ba1388aef0faf2372dde8876d329564bb4f39	129	50.39
	3	0b43925c8865869e7dde5c67cfd976f839bd8f5c8fda2814c2c61ce4c926b380	130	50.78
	4	d9e813e6f36a7a960664ab422b1eb1892be71f43a28229399bdcf51a5ab0df8d	131	51.17
	5	d0f1dcbf0670f8a3ef2771d0f0d8404c6068ab43b303d1aa9e335d9a757ddb6b	149	58.20
	6	1441805beb1753d9c81bd16d9059f3f2e57752732c1f2e539ec606555f2d9042	137	53.51
	Average	–	135.2	52.81
Structure 2	1	a86e4c2ff1450a08a173b2d9ef27d941fcb9a06f76ad1e70108192ce3cd02a16	-	-
r = 24	2	22e2025f1d0bdb5b20098e8f2d81a63b27e722c9e2eb521e87e00943f7af1dbe	132	51.56
	3	366d73069aa3e7238773a6ba39bbfc29203f28ffd05f8fec06060ececc54fc2e	113	44.14
	4	cd1fcb9c2c9a1caab20b4c8bf1ff18493533b42004d9f7741f957ab1850831db	128	50.00
	5	a0ef7aa8c7200a711f30101de786e2450f7a7f1e884a44831aba30c77f46b478	122	47.65
	6	bbf12b6acb919c42edb035fe0945b414bf0809b666bbb536976139bee4ea9bdd	124	48.43
	Average	–	123.8	48.35

Table 10 A comparison of average B_i and $HD_i(\%)$ for message sensitivity

	Output scheme	B_i	HD_i%
Structure 1	MMO	126.2	49.29
	MMMO	128.6	50.23
	MP	135.40	52.89
Structure 2	MMO	131.4	51.32
r = 8	MMMO	126.8	49.53
	MP	135.2	52.81
Structure 2	MMO	129.6	50.62
r = 24	MMMO	124.8	48.75
	MP	123.8	48.35

4.2.4 Sensitivity of hash value h to the secret key K

Thus, to evaluate the sensitivity of hash value h to the secret key K, hash simulation experiments were conducted under five different conditions (the original input message M is fixed), which are as follows:

Condition 1: The original secret key K is used.

In each of these conditions, we flip the LSB in the aforementioned initial conditions and parameters.

Condition 2: We change the initial condition $KSs(0)$ in the secret key.

Condition 3: We change the parameter Ks in the secret key.

Condition 4: We change the initial condition $KSs(-1)$ in the secret key.

Condition 5: We change the control parameter $Q1$ in the secret key.

In Tables 11, 12, and 13, we present the obtained results of h_i, B_i, and $HD_i(\%)$ under each condition for the two proposed structures with their output schemes, i.e., MMO, MMMO, and MP.

In Table 14, we reassessed the obtained results and even for a single test, the results are inside the normal range. Therefore, the proposed hash functions have high key sensitivity.

4.2.5 Statistical analysis of diffusion effect

Since confusion and diffusion were first proposed by (Shannon 1949), they have been extensively used to evaluate the security of cryptographic primitives. In the context of hash functions, confusion is defined as the complexity of the relation between the secret key K and the hash value h for a given message M, whereas diffusion is defined as the complexity of the relationship between the message M and the hash value h for a given key K. The confusion

Table 11 Sensitivity of hash value to the secret key for the proposed structures with MMO output scheme

	Message variants	Hexadecimal hash values	B_i	HD_i%
Structure 1	1	bedf7967520105d114e2cdf3399f52394a53e276bb104307345bacf93e317ef6	–	–
	2	60f63ae88faea074964bc5e71022d77003f61ed4dddd8b027c7826e8f31725ff	116	45.31
	3	3e7a24001b11a0a5376d55d073e5910e1bb3b98e4736793ca8bcdf4b5da27b41	127	49.60
	4	fd8fe49f2c5013871f1e291d6c74ceefeb9c4eead9a236d6b923bb04da3c7f4b	135	52.73
	5	054c289004f47fde2fd041e5e830cd4a74d9b586ba2b79835fb5ee13c7289717	139	54.29
	Average	–	129.25	50.48
Structure 2	1	1d6238873699dd1c252e02c88e1d2a380d9b5ea8e6c09c788fa4d3955b959975	–	–
r = 8	2	aab2bfb971b64b4349a5045d277421df6ee299dc209b0bf0ce9bfccff8bbbe8b	138	53.90
	3	c5667f505bcb289ec52be2fce9a168b72ad0de3fae396b7654f34cf419309b0f	123	48.04
	4	54b21e25c1ee818897c54e84eca15d2ddbd7b505ef81ba2c099a5c852db33b51	121	47.26
	5	f6e6702867e3c3ee86a4d86a6153b1266f58847a704665417fbc66fc39d8179f	132	51.56
	Average	–	128.5	50.19
Structure 2	1	af5e7ca7c83a72c77f0e9b7d47df11b0f66cadc862d6f522d592dc5ad9bae938	–	–
r = 24	2	f922e9e31c36e932ffb098930fa2726b29a1ce91c5c62b1f16981609b9b2453b	125	48.82
	3	3566ab26fff9c3a232368b624267c3397ab1099ba744ff5f6ec97a7cbc483fa5	126	49.21
	4	3b6a773dfe06e246ab3f53c3c9a0af08123346bb8a0e58a17caf6046992e08a7	130	50.78
	5	40ed183aa3cfb41d9d6f7e304d9ab05a0007044b0db84f039f4315c046051641	146	57.03
	Average	–	131.75	51.46

Table 12 Sensitivity of hash value to the secret key for the proposed structures with MMMO output scheme

	Message variants	Hexadecimal hash values	B_i	HD_i%
Structure 1	1	719adf0e0cdf5b149edc54efdbc09bb6df5a0ce3d3ac9bccc39ac5a64ea65531	–	–
	2	f2d4772a5a605c729e8ad2c3db016a20135f617b98c4366bb9b44cea418afe92	114	44.53
	3	23c5a8b268979416f80a32c7aa272c23cd293e20fe3547f8a621815276b3ebab	130	50.78
	4	75c848fa05415217403dbc2235da6d8fa7fa18b7526b376e4fbb89497303c340	120	46.87
	5	22c9b90204e4522181389ccff6ab7d24547415b87c8cbd3425c83929c3221024	118	46.09
	Average	–	120.50	47.07
Structure 2	1	a594a994aa162adca654e889dea0e6344190aa02328302465570df8f0084f5e6	–	–
r = 8	2	96ebc3ab71912e96b77b6c0db2ad2b0b300484abec4c326bbf10e7b5263ba545	127	49.60
	3	67d10bee9dedd7e06d58ee10aca74ca3336000f1984a54591d4f9e33face2a1a	138	53.90
	4	d2db99f2d01e0b5933c37fd86f8983577893b03f490abe2683e2e11870d1df69	123	48.04
	5	6d5b61d74e75cd983b4f0bf3913211dd991aa35f378842bb187d734f708a49db	126	49.21
	Average	–	128.5	50.19
Structure 2	1	6abbd825d6b17184a5fc558670f9f78d91b3812c899c8a062ef855507b4a81e5	–	–
r = 24	2	8741188aadde9edba0310e69541c85936202a4c7ef4de93e9906bdd970931948	149	58.20
	3	e10308d6126ebaef0ed5982b03e0c27a521060a570aa0a2cf692e63d2d149336	137	53.51
	4	e8818d36b227e849ed6e3a121745f8d8803bf9425384745fba6a2b1b7adbe32c	119	46.48
	5	26354f0bc5a4e6385ac23c715acccf65c2d2b28785e504a4a2966f21189b8fde	132	51.56
	Average	–	134.25	52.44

effect is naturally obtained in hash functions and it is very strong in chaos-based hash functions, due to the inherent properties of chaos. In cryptographic hash functions, strong diffusion is required. The ideal diffusion effect is obtained when any single bit change in the message causes a change with a 50% probability for each bit of a hash value (binary format). This is often referred to the *avalanche effect* in literature (Feistel 1973).

To evaluate the performance of the two proposed structures with different output schemes, i.e., MMO, MMMO, and MP, we performed the following diffusion test:

The previous defined message M is chosen and a hash value h is generated. Next, a bit in the message is randomly

Table 13 Sensitivity of hash value to the secret key for the proposed structures with MP output scheme

	Message variants	Hexadecimal hash values	B_i	$HD_i\%$
Structure 1	1	a005e50f9673ecee6e80c07c550e53f8a950cb4a91176a2a340b5822ec2f28c4	–	–
	2	27de6d91694c777474b94f2a4ec3ed8c5b5b0da8c38fed5b4c75e2e2bf97972f	143	55.85
	3	3fa8a997b46131a1429d0006b6c03f181898632313a64f3da8143d1cadd66925	122	47.65
	4	f670f60cfc1daecb0c81988735b736c8c18851cebe5b94a6f1234f49bd4d5209	117	45.70
	5	7c68bc63287bfe02badbceb99cdde6a0ef5e9e7429d1dc3d2a9bf90b34a6402c	123	48.04
	Average	-	126.25	49.31
Structure 2	1	31882869cce69d7734f0078d29f297841b99d3f9786a1cf522688de9561826ee	–	–
r = 8	2	0b840b10ffda4c9feb4dabf4ab2f642ffe55f730386b8d295534368af526fa33	136	53.12
	3	2f65ed46a3cb9b0ebb1cf7cd52558de58e2ebc7474b01f169a6b30067e20e5a5	134	52.34
	4	cf524afe65de3a8123e43e61540a28180f0be21669a3ca4b4d62fdca34f538b5	139	54.29
	5	27d7a12c3a95c9f52148b43d60c7dbd3acd0b774c885d712bf2bb7673b77443e	131	51.17
	Average	–	135	52.73
Structure 2	1	a86e4c2ff1450a08a173b2d9ef27d941fcb9a06f76ad1e70108192ce3cd02a16	–	–
r = 24	2	37235dea611e13421ca8545078d0ec3a88654cfbc4e24bd64dd110ce2ed4ea3e	121	47.26
	3	7f60df23e3570ba37890a0b199e891835757fabc67b96e2cbbd02d0f64629cb7	120	46.87
	4	d3bd1e2064cecd5851624b61019a097a00eca137bd1cff0d50b1af161185581e	127	49.60
	5	149bb7e22e3a018254a5cfb711e192471971857c96663e6ec189762548f09ca3	139	54.29
	Average	–	126.75	49.51

Table 14 A comparison of average B_i and $HD_i(\%)$ for key sensitivity

	Output scheme	B_i	$HD_i\%$
Structure 1	MMO	129.25	50.48
	MMMO	120.50	47.07
	MP	126.25	49.31
Structure 2	MMO	128.5	50.19
r = 8	MMMO	128.5	50.19
	MP	135	52.73
Structure 2	MMO	131.75	51.46
r = 24	MMMO	134.25	52.44
	MP	126.75	49.51

selected and toggled and a new hash value is generated. Then, the number of bits changed B_i between the two hash values is calculated. This test is performed at J-time, where J = 512, 1024, and 2048 tests. The six statistical values concerning this test are calculated as follows:

1. Minimum number of bits changed:
   $B_{min} = min(\{B_i\}_{i=1,\ldots,J})\ bits$
2. Maximum number of bits changed:
   $B_{max} = max(\{B_i\}_{i=1,\ldots,J})\ bits$
3. Mean number of bits changed:
   $\bar{B} = \frac{1}{J}\sum_{i=1}^{J} B_i\ bits$
4. Mean changed probability (mean of $HD_i(\%)$):
   $P = (\frac{\bar{B}}{256}) \times 100\%$
5. Standard variance of the changed bit number:
   $\Delta B = \sqrt{\frac{1}{J-1}\sum_{i=1}^{J}(B_i - \bar{B})^2}$
6. Standard variance of the changed probability:

Table 15 Diffusion statistical-results for the two proposed structures

		Output schemes		
		MMO	MMMO	MP
Structure 1	B_{min}	98	98	100
	B_{max}	158	158	154
	$\bar{B}$	127.98	127.90	127.95
	P	49.99	49.96	49.98
	ΔB	8.01	8.12	8.03
	ΔP	3.13	3.17	3.13
Structure 2	B_{min}	99	98	103
r = 8	B_{max}	157	154	157
	$\bar{B}$	128.31	128.18	127.97
	P	50.12	50.07	49.99
	ΔB	8.03	8.17	8.01
	ΔP	3.13	3.19	3.13
Structure 2	B_{min}	101	103	100
r = 24	B_{max}	155	156	157
	$\bar{B}$	127.81	127.70	127.88
	P	49.92	49.88	49.95
	ΔB	8.23	8.06	7.94
	ΔP	3.21	3.15	3.10

Table 16 Diffusion statistical-results for the two proposed structures with MP output scheme

		Number of tests		
		512	1024	2048
Structure 1	B_{min}	100	100	100
	B_{max}	149	152	154
	$\bar{B}$	128.11	128.22	127.95
	P	50.04	50.08	49.98
	ΔB	8.11	8.17	8.03
	ΔP	3.16	3.19	3.13
Structure 2	B_{min}	104	104	103
r = 8	B_{max}	150	151	157
	$\bar{B}$	127.98	127.88	127.97
	P	49.99	49.95	49.99
	ΔB	7.92	7.98	8.01
	ΔP	3.09	3.12	3.13
Structure 2	B_{min}	100	100	100
r = 24	B_{max}	153	153	157
	$\bar{B}$	127.85	127.96	127.88
	P	49.95	49.98	49.95
	ΔB	8.22	8.10	7.94
	ΔP	3.21	3.16	3.10

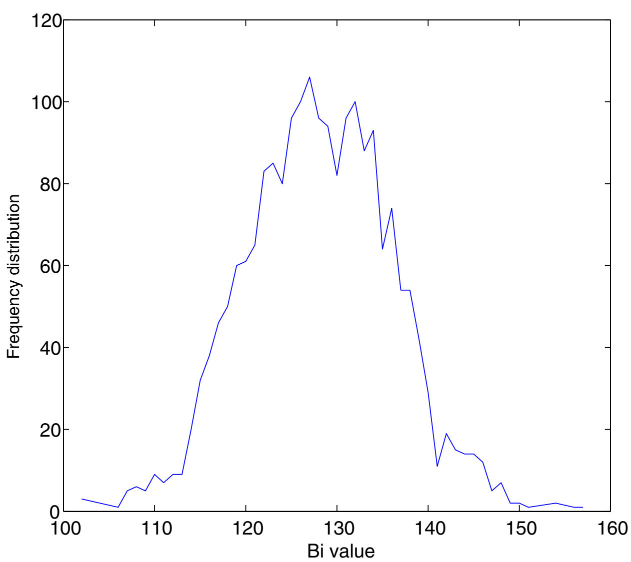


Fig. 11 Histogram of B_i

$$\Delta P = \sqrt{\frac{1}{J-1}\sum_{i=1}^{J}\left(\frac{B_i}{256} - P\right)^2} \times 100\%.$$

The obtained statistical results of diffusion presented in Table 15 with 2048 tests demonstrates that the diffusion effect is close to the expected one. Indeed, irrespective of the used structure and the output schemes, both $\bar{B}$ and P are very close to the ideal values (128 bits and 50%, respectively), while ΔB and ΔP are very low, which indicates that the diffusion is extremely stable. These results, presented in Table 16, are also confirmed through the tests with J = 512 and 1024, for Structures 1 and 2 with MP output scheme.

In addition, we draw the histogram B_i (Fig. 11) of Structure 1 with MP output scheme to show that the values of B_i are centered on the ideal value 128 bits. Similar results are obtained for the other proposed hash functions as well.

4.3 Cryptanalysis

The attackers make use of some general attack methods that are available to them, which can be applied to any Unkeyed or Keyed hash functions. These attacks depend only on the hash value length u for the unkeyed hash function and on the hash value length u and the secret key length $|K|$ for the keyed hash function. If the cryptanalyst can find a method to retrieve K, the system is entirely compromised (during the key life time) (Lucks 2004; Mironov 2005).

4.3.1 Brute force attacks

A brute-force attack on a keyed hash function is more difficult than a brute-force attack on an unkeyed hash function. There are two possible types of attacks, which are as follows:

1. Attacks on the hash value h, namely *Preimage attack, second preimage attack*, and *collision resistance attack*.
2. Attack on the secret key K, namely *Exhaustive key search attack*.

For the first type of attacks, for a given secret key K, the fastest way to compute a first or second preimages and collision resistance is through a brute force attack that consists of randomly selecting values of M and try each value until a collision occurs. For exhaustive key search attack, the attacker requires known {message, hash} pairs.

Preimage and second preimage attacks(Aoki and Sasaki 2008): In a preimage attack, given only the hash value h, the attacker tries to find the original message M in a way such that $H(M) = h$ without attempting to recover the secret key K.

In a second preimage attack, the adversary has more information. Specifically, he knows the hash value h for a given message M and he tries to find another message M' that produces the same hash value h.

For the first and second preimage attacks, the adversary would have to try, on average, 2^{u-1} values of M to find one that generates the given hash value h. Our proposed structures produce hash values of length 256 bits, so that the minimum amount of work required by an attacker to violate the preimage or second preimage resistance property should be 2^{256-1} operations, which is considered very high. Thus, the proposed hash functions are robust against first and second preimage attacks.

Collision resistance attack (*Birthday attack*) (Flajolet et al. 1992): In the collision resistance attack, the attacker tries to find two messages (M, M') that collide with the same hash value h. The minimum amount of work required by an attacker to violate the collision resistance property is approximately $2^{u/2}$ operations. This required effort is proven by a mathematical result referred to as the birthday paradox.

Also, for collision resistance attack, the length of hash value h determines the security and the proposed hash functions are secure against these kinds of attacks because an attacker needs, on average, 2^{128} tries.

Exhaustive key search attack(Safavi-Naini and Pieprzyk 1995; Preneel 1993): In keyed CNN hash functions, if the attacker has access to a pair (message, digest), then normally the key can be found by exhaustive searching and, on average, the attacker needs $2^{|K|-1}$ tries, where $|K|$ is the length of the secret key K. Thus, the level of effort for brute force attack on keyed hash functions can be expressed, on average, as $min(2^{|K|-1}, 2^{u/2})$. As $|K| = 160$ bits, consequently, the proposed hash functions are immune against these kinds of attacks.

4.3.2 Cryptanalytical attacks

Cryptanalytic attacks seek to exploit some properties of the keyed hash function to perform some attacks other than brute force attacks. An ideal keyed hash function should require a cryptanalytic effort greater than or equal to the brute force effort. Far less research has been conducted on developing such attacks. A useful survey of some methods for specific keyed hash functions is developed in (Preneel and van Oorschot 1996). In the following paragraphs, we apply the two most common cryptanalytic attacks of the literature on the proposed hash functions:

1. Length extension attack (Padding attack)
2. Meet-in-the-middle preimage attack

Length extension attack(Has; MD5): In cryptography and computer security, a length extension attack is a type of attack where an attacker can use $H(M)$ and the length of M to calculate $H(M \| EM)$ for an attacker-controlled extended message EM. In our proposed hash functions, the secret key K is not pre-pended to the message M but used as an input for the Chaotic System to produce the necessary supplies to CNN. Then, such an attack can not be conducted.

Meet-in-the-middle preimage attack (MITM)(Aoki and Sasaki 2009; Wei et al. 2011): the meet-in-the-middle preimage attack is a generic cryptanalytic approach that is originally applied to the cryptographic systems based on block ciphers (Chosen plain-text attack). In 2008, Aoki and Sasaki (2009) noticed that the MITM attack could be applied to hash functions, to find preimage, second preimage, or collision for intermediate hash chaining values instead of the hash value h. This attack has successfully broken several designs: the MD hash family includes MD5 (Sasaki and Aoki 2009), round-reduced SHA-0, and SHA-1 (Aoki and Sasaki 2009), round-reduced SHA-2 (Aoki et al. 2009), some Davies–Meyer hash constructions, e.g., Tiger (Guo et al. 2010), reduced HAS-160 (Hong et al. 2009) and HAVAL (Sasaki and Aoki 2008). As our hash functions are preimage resistant, the effort to succeed the meet-in-the-middle attack with probability 0.632 is $2^{u/2} = 2^{128}$ tries.

Table 17 Hashing time, hashing throughput, and the number of cycles per Byte for Structures 1 and 2 with MMO output scheme and 2048 random tests

Message length	Structure 1			Structure 2—r = 8			Structure 2—r = 24		
	HT	HTH	NCpB	HT	HTH	NCpB	HT	HTH	NCpB
513	8.60	57.37	43.70	4.47	112.02	22.71	6.73	73.21	34.20
1024	15.24	64.98	38.75	8.18	124.18	20.79	8.02	124.17	20.30
2048	27.02	72.66	34.33	13.82	143.44	17.56	15.11	132.90	19.20
4096	51.13	76.50	32.46	25.73	153.06	16.34	26.99	146.33	17.13
10^4	122.15	78.18	31.76	60.16	159.42	15.64	62.30	153.79	16.20
10^5	1211.30	79.14	31.49	590.16	162.70	15.34	626.89	154.21	16.29
10^6	11972.02	79.73	31.12	5910.81	162.14	15.36	6185.43	155.61	16.08

Table 18 Hashing time, hashing throughput, and the number of cycles per Byte for Structures 1 and 2 with MMMO output scheme and 2048 random tests

Message length	Structure 1			Structure 2—r = 8			Structure 2—r = 24		
	HT	HTH	NCpB	HT	HTH	NCpB	HT	HTH	NCpB
513	8.53	57.72	43.34	5.16	99.80	26.21	6.89	71.12	35.02
1024	15.11	65.65	38.42	7.78	127.88	19.77	8.03	124.46	20.40
2048	27.21	72.30	34.56	13.47	145.78	17.11	14.32	137.94	18.19
4096	51.71	75.81	32.83	25.40	154.57	16.13	26.67	147.56	16.93
10^4	122.50	78.05	31.85	59.71	160.27	15.52	63.25	152.32	16.44
10^5	1216.68	78.70	31.63	603.15	159.79	15.68	632.82	153.17	16.45
10^6	11935.23	79.97	31.03	6015.73	160.38	15.64	6272.66	153.96	16.30

Table 19 Hashing time, hashing throughput, and the number of cycles per Byte for Structures 1 and 2 with MP output scheme and 2048 random tests

Message length	Structure 1			Structure 2—r = 8			Structure 2—r = 24		
	HT	HTH	NCpB	HT	HTH	NCpB	HT	HTH	NCpB
513	8.67	57.19	44.04	4.45	111.99	22.61	6.76	73.19	34.36
1024	14.77	66.84	37.55	7.72	128.94	19.62	7.94	124.42	20.19
2048	27.05	72.73	34.35	13.81	143.17	17.55	16.03	127.37	20.36
4096	51.52	76.12	32.71	27.42	145.93	17.41	28.16	141.84	17.88
10^4	122.12	78.32	31.75	59.73	160.25	15.53	63.87	151.23	16.60
10^5	1232.16	78.32	32.03	585.29	163.83	15.21	631.08	153.34	16.40
10^6	11866.13	80.42	30.85	5864.95	163.29	15.24	6250.05	154.55	16.25

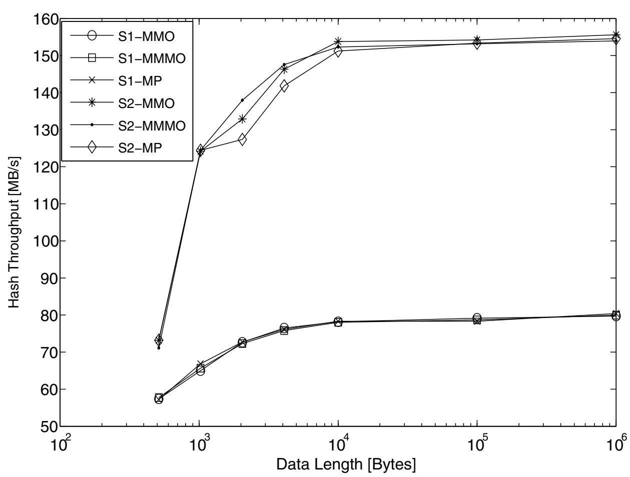


Fig. 12 Comparison of HTH for Structure 1 and Structure 2—r = 24 rounds with MMO, MMMO, and MP output schemes

4.4 Speed analysis

We evaluated the computing performance of the two proposed hash functions with their output schemes for different message lengths. For this purpose, we calculated the average hashing time HT (micro second), the average hashing throughput HTH (MBytes/s) and the needed number of cycles to hash one Byte NCpB (cycles/Byte).

$$HTH\ (MBytes/s) = \frac{Message\ size(MBytes)}{Average\ hashing\ time(s)} \tag{26}$$

$$NCpB\ (cycles/Byte) = \frac{CPUspeed(Hz)}{HTH(Byte/s)} \tag{27}$$

To test the speed of our proposed hash functions, we implement these hash functions in C code using a computer with a 2.6 GHZ Intel core i5-4300M CPU with 4 GB of RAM running Ubuntu Linux 14.04.1 (32-bit) operating system. In Tables 17, 18, and 19, the average HT, the average HTH, and the average NCpB for the two structures with their output schemes are presented. It was observed that, irrespective of the output schemes, the computing performance of Structure 2 is approximately twice better than the computing performance of Structure 1, even for $r = 24$ rounds. To focus more on these results, the HTH for the two structures with their output schemes were drawn in Fig. 12.

The variation of computing performance according to the size of the message is due to the transition phase of both chaotic system and chaotic activation function of a neuron.

Table 20 Comparison in terms of collision resistance of the proposed structures with MP output scheme with some chaos-based hash functions

Hash function	Number of hits ω				Absolute difference $\boldsymbol{d}$			
	0	1	2	3	Mean	Mean/character	Minimum	Maximum
Xiao et al. (2005)	–	–	–	–	1506	94.12	696	2221
Xiao et al. (2009b)	1926	120	2	0	1227.8	76.73	605	1952
Deng et al. (2010)	1940	104	4	0	1399.8	87.49	583	2206
Yang et al. (2009)	–	–	–	–	–	93.25	–	–
Xiao et al. (2009a)	1915	132	1	0	1349.1	84.31	812	2034
Li et al. (2012b)	1901	146	1	0	1388.9	86.81	669	2228
Wang et al. (2007)	1917	126	5	0	1323	82.70	663	2098
Huang (2011)	1932	111	5	0	1251.2	78.2	650	1882
Li et al. (2011a)	1928	118	2	0	1432.1	89.51	687	2220
Li et al. (2011b)	1899	124	25	0	1367.6	85.47	514	2221
Li et al. (2013)	1920	124	4	0	1319.5	82.46	603	2149
He et al. (2013)	1926	118	4	0	1504	94	683	2312
Xiao et al. (2008)	1924	120	4	0	1431.3	89.45	658	2156
Yu-Ling et al. (2012)	1928	117	3	0	1598.6	99.91	796	2418
Xiao et al. (2010)	1932	114	2	0	1401.1	87.56	573	2224
Li et al. (2012c)	1920	122	6	0	–	–	–	–
Li et al. (2012a)	1905	135	8	0	1335	83.41	577	2089
Structure 1	1931	114	3	0	1291.64	80.72	480	2038
Structure 2—r = 8	1929	114	5	0	1426.23	89.13	730	2213
Structure 2—r = 24	1942	106	0	0	1338.85	83.67	629	2071

Indeed, the cost of the transition phase is approximately equal $2\times \text{tr} \times 4 = 240$ Bytes for Structure 1 (tr = 30) and 160 Bytes for Structure 2 (tr = 20) in our implementation.

4.5 Performance comparison with other chaos-based hash functions of literature and standards hash functions

We compared the performance of the proposed hash functions with some hash functions of literature in terms of statistical analysis and NCpB. Table 20 presents the comparison with chaos-based hash function in terms of collision resistance for MP output scheme with 2048 tests. As we can see, except Li et al. (2011b) our obtained results are more close to the expected values. Table 21, additionally, presents the comparison of statistical results of diffusion. We observed that the obtained results for all cited references are closed to the expected values. It should be noted that besides the two references (Chenaghlu et al. 2016; Jiteurtragool et al. 2013), all the other references in Tables 20 and 21present structures that work with hash value $h = 128$ bits. For comparison purposes, we took the 128 LSB hash values.

Tables 22 and 23present the comparison of the proposed chaos-based hash functions with standard hash function in terms of collision resistance and diffusion. Aside the values of Structure 2—$r = 8$ rounds, the obtained results are similar to those obtained by standard hash functions.

The speed performance, in terms of the number of cycles to hash one Byte (NCpB), of the proposed keyed chaos-based hash functions is compared to that of some chaos-based hash functions of literature and with the main standards of the unkeyed and keyed hash functions, which are presented in Tables 24 and 25, respectively. We observed that the NCpB of the Structure 2 is approximately twice as fast as the best NCpB obtained by (Teh et al. 2015), but it is a little bit slower than the SHA-2's NCpB and approximately four times slower than the main keyed hash functions.

5 Conclusion

We designed, realized and analyzed the security and computation performance of the two keyed chaotic neural network hash functions, based on Merkle–Dåmgard

Table 21 Comparison of the statistical results of diffusion for the proposed structures with MP output scheme with some chaos-based hash functions

Hash function	B_{min}	B_{max}	$\bar{B}$	$P(\%)$	ΔB	ΔP %
Xiao et al. (2005)	–	–	63.85	49.88	5.78	4.52
Lian et al. (2006b)	–	–	63.85	49.88	5.79	4.52
Zhang et al. (2007)	46	80	63.91	49.92	5.58	4.36
Wang et al. (2008)	–	–	63.98	49.98	5.53	4.33
Xiao et al. (2009b)	–	–	64.01	50.01	5.72	4.47
Deng et al. (2009)	–	–	63.91	49.92	5.58	4.36
Deng et al. (2010)	–	–	63.84	49.88	5.88	4.59
Yang et al. (2009)	–	–	64.14	50.11	5.55	4.33
Xiao et al. (2009a)	–	–	64.09	50.07	5.48	4.28
Amin et al. (2009)	–	–	63.84	49.88	5.58	4.37
Li et al. (2012b)	45	81	63.88	49.90	5.37	4.20
Wang et al. (2007)	–	–	63.90	49.93	5.64	4.41
Akhavan et al. (2009)	42	83	63.91	49.92	5.69	4.45
Huang (2011)	–	–	63.88	49.91	5.75	4.50
Li et al. (2011a)	–	–	63.80	49.84	5.75	4.49
Wang et al. (2011)	44	82	64.15	50.11	5.76	4.50
Li et al. (2011b)	–	–	63.56	49.66	7.42	5.80
Li et al. (2013)	–	–	63.97	49.98	5.84	4.56
He et al. (2013)	45	83	64.03	50.02	5.60	4.40
Jiteurtragool et al. (2013)	43	81	62.84	49.09	5.63	4.40
Teh et al. (2015)	–	–	64.01	50.01	5.61	4.38
Chenaghlu et al. (2016)	–	–	64.12	50.09	5.63	4.41
Akhavan et al. (2013)	43	82	63.89	49.91	5.77	4.50
Nouri et al. (2012)	–	–	64.08	50.06	5.72	4.72
Xiao et al. (2008)	47	83	63.92	49.94	5.62	4.39
Yu-Ling et al. (2012)	–	–	64.17	50.14	5.74	4.49
Xiao et al. (2010)	–	–	64.18	50.14	5.59	4.36
Li et al. (2012c)	–	–	64.07	50.06	5.74	4.48
Li et al. (2012a)	–	–	63.89	49.91	5.64	4.41
Ren et al. (2009)	–	–	63.92	49.94	5.78	4.52
Guo et al. (2006)	–	–	63.40	49.53	7.13	6.35
Yu et al. (2011)	45.6	81.8	63.98	49.98	5.73	4.47
Zhang et al. (2005)	–	–	64.43	49.46	5.57	4.51
Jiteurtragool et al. (2013)	101	153	126.75	49.51	7.98	3.12
Chenaghlu et al. (2016)	101	168	128.08	50.03	8.12	3.21
Structure 1	45	86	64.05	50.03	5.65	4.41
Structure 2—r = 8	42	84	63.88	49.91	5.66	4.42
Structure 2—r = 24	43	85	63.90	49.92	5.60	4.37

Table 22 Comparison in terms of collision resistance of the proposed structures with MP output scheme with some chaos-based hash functions

Hash function	Number of hits ω				Absolute difference d			
	0	1	2	3	Mean	Mean/character	Minimum	Maximum
SHA2-256 (Secure Hash Standard and FIPS Publication 2002)	1817	220	11	0	2707.10	84.59	1789	3819
Structure 1	1803	232	13	0	2633.17	82.28	1471	3779
Structure 2—r = 8	1817	215	16	0	2861.93	89.43	1707	3951
Structure 2—r = 24	1815	226	7	0	2615.44	81.73	1540	3671

Table 23 Comparison of the statistical results of diffusion for the two proposed structures with MP output scheme and SHA2-256

Hash function	B_{min}	B_{max}	$\bar{B}$	$P(\%)$	ΔB	ΔP %
SHA2-256 (Secure Hash Standard and FIPS Publication 2002)	104	154	128.01	50.00	7.94	3.10
Structure 1	100	154	127.95	49.98	8.03	3.13
Structure 2—r = 8	103	157	127.97	49.99	8.01	3.13
Structure 2—r = 24	100	157	127.88	49.95	7.94	3.10

Table 24 Comparison of NCpB of the proposed structures with three output schemes with some chaos-based hash functions

Hash function	Structure 1			Structure 2—r = 8			Structure 2—r = 24			Wang (2008)	Akhavan (2009)	Teh (2015)
	MMO	MMMO	MP	MMO	MMMO	MP	MMO	MMMO	MP			
NCpB	31.12	31.03	30.85	15.36	15.64	15.24	16.08	16.30	16.25	122.4	105.5	28.45

Table 25 Comparison of NCpB of the proposed hash functions with the unkeyed and keyed standards

Hash function	Structure 1			Structure 2—r = 8			Structure 2—r = 24			SHA2-256
	MMO	MMMO	MP	MMO	MMMO	MP	MMO	MMMO	MP	
NCpB	31.12	31.03	30.85	15.36	15.64	15.24	16.08	16.30	16.25	11.87

Hash function	VMAC	HMAC	GCM	CMAC	DMAC	CBC-MAC	BLAKE 2
NCpB	0.42	14.42	0.42	4.41	4.40	2.88	2.58

construction with three output schemes MMO, MMMO, and MP. The obtained results quantified the robustness of the proposed hash functions for using them in data integrity, message authentication, and digital signature applications. The very good performance is due to the strong one-way property of the combined chaotic system with neural network structure. Indeed, the neuron's activation functions are based on a secure and efficient chaotic generator. Compared to some chaos-based hash functions of literature, the proposed CNN hash functions are more robust and show good results in terms of computation performance. Our future work will focus on the design of keyed CNN hash functions based on Sponge construction, adopted in the standard SHA-3, and its Duplex construction for authenticated encryption application. Also, the hardware implementation of our proposed hash functions will be studied.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Abdoun N, El Assad S, Taha MA, Assaf R, Deforges O, Khalil M (2015) Hash function based on efficient chaotic neural network. In: International conference on internet technology and secured transactions, pp. 32–37

Abdoun N, El Assad S, Taha MA, Assaf R, Déforges O, Khalil M (2016) Secure hash algorithm based on efficient chaotic neural network. In: The 11th international conference on communications, page comm 2016

Akhavan A, Samsudin A, Akhshani A (2009) Hash function based on piecewise nonlinear chaotic map. Chaos Solitons Fractals 42(2):1046–1053

Akhavan A, Samsudin A, Akhshani A (2013) A novel parallel hash function based on 3D chaotic map. EURASIP J Adv Signal Process 2013(1):126

Amin M, Faragallah OS, El-Latif AAA (2009) Chaos-based hash function (CBHF) for cryptographic applications. Chaos Solitons Fractals 42(2):767–772

Aoki K, Guo J, Matusiewicz K, Sasaki Y, Wang L (2009) Preimages for step-reduced SHA-2. In: ASIACRYPT, vol 5912, pp 578–597. Springer, New York

Aoki K, Sasaki Y (2008) Preimage attacks on one-block MD4, 63-step MD5 and more. In: International workshop on selected areas in cryptography, pp 103–119. Springer, New York

Aoki K, Sasaki Y (2009) Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Advances in cryptology-CRYPTO 2009, pp 70–89. Springer, New York

Arumugam G, Praba VL, Radhakrishnan S (2007) Study of chaos functions for their suitability in generating message authentication codes. Appl Soft Comput 7(3):1064–1071

Bartkewitz T (2009) Building hash functions from block ciphers, their security and implementation properties. Ruhr-University, Bochum

Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Annual international cryptology conference, pp 1–15. Springer, New York

Bertoni G, Daemen J, Peeters M, Van Assche G (2007) Sponge functions. In: ECRYPT hash workshop, vol 2007

Brachtl BO, Coppersmith D, Hyden MM, Matyas Jr, SM, Meyer CH, Oseas J, Pilpel S, Schilling M (1990) Data authentication using modification detection codes based on a public one way encryption function. US Patent 4908861

Chain K, Kuo W-C (2013) A new digital signature scheme based on chaotic maps. Nonlinear Dyn 74(4):1003–1012

Chenaghlu MA, Jamali S, Khasmakhi NN (2016) A novel keyed parallel hashing scheme based on a new chaotic system. Chaos Solitons Fractals 87:216–225

Damgård IB (1989) A design principle for hash functions. In: Conference on the theory and application of cryptology, pp 416–427. Springer, New York

Deng S, Li Y, Xiao D (2010) Analysis and improvement of a chaos-based hash function construction. Commun Nonlinear Sci Numer Simul 15(5):1338–1347

Deng S, Xiao D, Li Y, Peng W (2009) A novel combined cryptographic and hash algorithm based on chaotic control character. Commun Nonlinear Sci Numer Simul 14(11):3889–3900

Denton B, Adhami R (2011) Modern hash function construction

Desnos K, El Assad S, Arlicot A, Pelcat M, Menard D (2014) Efficient multicore implementation of an advanced generator of discrete chaotic sequences. In: 2014 9th international conference for internet technology and secured transactions (ICITST), pp 31–36. IEEE, New York

Dunkelman O, Biham E (2006) A framework for iterative hash functions: Haifa. In: 2nd NIST cryptographich hash workshop, vol 22

El Assad S (2012) Chaos based information hiding and security. In: 2012 international conference for internet technology and secured transactions, pp 67–72. IEEE, New York

El Assad S, Noura H (2014) Generator of chaotic sequences and corresponding generating system. US Patent 8781116

Feistel H (1973) Cryptography and computer privacy. Sci Am 228:15–23

Flajolet P, Gardy D, Thimonier L (1992) Birthday paradox, coupon collectors, caching algorithms and self-organizing search. Discrete Appl Math 39(3):207–229

Guesmi R, Farah M, Kachouri A, Samet M (2016) A novel chaos-based image encryption using dna sequence operation and secure hash algorithm SHA-2. Nonlinear Dyn 83(3):1123–1136

Guo J, Ling S, Rechberger C, Wang H (2010) Advanced meet-in-the-middle preimage attacks: first results on full tiger, and improved results on MD4 and SHA-2. In: ASIACRYPT, vol 6477, pp 56–75. Springer, New York

Guo X-F, Zhang J-S (2006) Keyed one-way hash function construction based on the chaotic dynamic s-box

Hash length extension attacks—Java code geeks—2017. https://www.javacodegeeks.com/2012/07/hash-length-extension-attacks.html. Accessed 07 Nov 2017

He B, Lei P, Pu Q, Liu Z (2013) A method for designing hash function based on chaotic neural network. In: International workshop on cloud computing and information security (CCIS)

Hong D, Koo B, Sasaki Y (2009) Improved preimage attack for 68-step has-160. In: ICISC, vol 5984, pp 332–348. Springer, New York

Huang Z (2011) A more secure parallel keyed hash function based on chaotic neural network. Commun Nonlinear Sci Numer Simul 16(8):3245–3256

Islam SH (2014) Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps. Nonlinear Dyn 78(3):2261–2276

Jiteurtragool N, Ketthong P, Wannaboon C, San-Um W (2013) A topologically simple keyed hash function based on circular chaotic sinusoidal map network. In: 2013 15th international conference on advanced communication technology (ICACT) , pp 1089–1094. IEEE, New York

Kim B-K, Oh S-J, Jang S-B, Ko Y-W (2017) File similarity evaluation scheme for multimedia data using partial hash information. Multimed Tools Appl 76(19):19649–19663

Kwok HS, Tang WK (2005) A chaos-based cryptographic hash function for message authentication. Int J Bifurc Chaos 15(12):4043–4050

Lee S-H, Hwang W-J, Kwon K-R (2014) Polyline curvatures based robust vector data hashing. Multimed Tools Appl 73(3):1913–1942

Li Y, Deng S, Xiao D (2011a) A novel hash algorithm construction based on chaotic neural network. Neural Comput Appl 20(1):133–141

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

Li Y, Xiao D, Deng S (2012b) Secure hash function based on chaotic tent map with changeable parameter. High Technol Lett 18(1):7–12

Li Y, Xiao D, Deng S, Han Q, Zhou G (2011b) Parallel hash function construction based on chaotic maps with changeable parameters. Neural Comput Appl 20(8):1305–1312

Li Y, Xiao D, Deng S, Zhou G (2013) Improvement and performance analysis of a novel hash function based on chaotic neural network. Neural Comput Appl 22(2):391–402

Li Y, Xiao D, Li H, Deng S (2012c) Parallel chaotic hash function construction based on cellular neural network. Neural Comput Appl 21(7):1563–1573

Lian S, Liu Z, Ren Z, Wang H (2006a) Hash function based on chaotic neural networks. In: Proceedings of the 2006 IEEE international symposium on circuits and systems, 2006. ISCAS 2006, p 4. IEEE, New York

Lian S, Sun J, Wang Z (2006b) Secure hash function based on neural network. Neurocomputing 69(16):2346–2350

Liu X, Xiu C (2008) Hysteresis modeling based on the hysteretic chaotic neural network. Neural Comput Appl 17(5–6):579–583

Liu J, Wang X, Yang K, Zhao C (2012) A fast new cryptographic hash function based on integer tent mapping system. JCP 7(7):1671–1680

Liu C, Ling H, Zou F, Wang Y, Feng H, Yan L (2015) Local and global structure preserving hashing for fast digital fingerprint tracing. Multimed Tools Appl 74(18):8003–8023

Lucks S (2004) Design principles for iterated hash functions. IACR Cryptol ePrint Arch 2004:253

Maqableh M, Samsudin AB, Alia MA (2008) New hash function based on chaos theory (CHA-1). Int J Comput Sci Netw Secur 8(2):20–27

Matyas SM (1985) Generating strong one-way functions with cryptographic algorithm. IBM Tech Discl Bull 27:5658–5959

MD5 length extension attack revisited—v's inner peace. https://web.archive.org/web/20141029080820/http://vudang.com/2012/03/md5-length-extension-attack/. Accessed 07 Nov 2017
Menezes AJ, Van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, New York
Merkle RC, Charles R et al (1979) Secrecy, authentication, and public key systems
Mironov I et al (2005) Hash functions: theory, attacks, and applications. Microsoft Research, Silicon Valley Campus
Miyaguchi S, Iwata M, Ohta K (1989) New 128-bit hash function. In: Proceedings of the 4th international joint workshop on computer communications, Tokyo, pp 279–288
Miyaguchi S, Ohta K, Iwata M (1990) Confirmation that some hash functions are not collision free. In: Workshop on the theory and application of cryptographic techniques, pp 326–343. Springer, New York
Nandi M, Paul S (2010) Speeding up the wide-pipe: secure and fast hashing. In: Indocrypt, vol 6498, pp 144–162. Springer, New York
Nouri M, Khezeli A, Ramezani A, Ebrahimi A (2012) A dynamic chaotic hash function based upon circle chord methods. In: 2012 Sixth international symposium on telecommunications (IST), pp 1044–1049. IEEE, New York
Prencel B, Bosselaers A, Govaerts R, Vandewalle J (1989) Collision-free hashfunctions based on blockcipher algorithms. In: Proceedings of the international Carnahan conference on security technology, pp 203–210. IEEE, New York
Preneel B (1993) Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit te Leuven
Preneel B, Govaerts R, Vandewalle J (1993) Hash functions based on block ciphers: a synthetic approach. In: Crypto, vol 93, pp 368–378. Springer, New York
Preneel B, van Oorschot P (1996) On the security of two MAC algorithms. In: Advances in cryptology EUROCRYPT96, pp 19–32. Springer, New York
Pub F (1995) Secure hash standard. Public Law 100:235
Ren H, Wang Y, Xie Q, Yang H (2009) A novel method for one-way hash function construction based on spatiotemporal chaos. Chaos Solitons Fractals 42(4):2014–2022
Rivest R (1992) The MD5 message-digest algorithm
Safavi-Naini R, Pieprzyk J et al (1995) Cryptographic hash functions: a survey. Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Australia
Sasaki Y, Aoki K (2008) Preimage attacks on 3, 4, and 5-pass Haval. In: International Conference on the theory and application of cryptology and information security, pp 253–271. Springer, New York
Sasaki Y, Aoki K (2009) Finding preimages in full MD5 faster than exhaustive search. In: Eurocrypt, vol 5479, pp 134–152. Springer, New York
Secure Hash Standard and FIPS Publication (2002) US Government FIPS Publication 180-2; 1:72
SHA-3 (2015) Standard: permutation-based hash and extendable-output functions. FIPS PUB 202:2015
Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28(4):656–715
Stallings W (2014) Cryptography and network security: principles and practice, international edition. Pearson Higher Education, London
Teh JS, Samsudin A, Akhavan A (2015) Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dyn 81(3):1067–1079
Wang Y, Du M, Yang D, Yang H (2007) One-way hash function construction based on iterating a chaotic map. In: International conference on computational intelligence and security workshops, 2007. CISW 2007, pp 791–794. IEEE, New York
Wang Y, Liao X, Xiao D, Wong K-W (2008) One-way hash function construction based on 2D coupled map lattices. Inf Sci 178(5):1391–1406
Wang Y, Wong K-W, Xiao D (2011) Parallel hash function construction based on coupled map lattices. Commun Nonlinear Sci Numer Simul 16(7):2810–2821
Wei L, Rechberger C, Guo J, Wu H, Wang H, Ling S (2011) Improved meet-in-the-middle cryptanalysis of ktantan (poster). In: Australasian conference on information security and privacy, pp 433–438. Springer, New York
Wong K-W (2003) A combined chaotic cryptographic and hashing scheme. Phys Lett A 307(5):292–298
Xiao D, Liao X (2004) A combined hash and encryption scheme by chaotic neural network. Adv Neural Netw ISNN 2004:13–28
Xiao D, Liao X, Deng S (2005) One-way hash function construction based on the chaotic map with changeable-parameter. Chaos Solitons Fractals 24(1):65–71
Xiao D, Liao X, Deng S (2008) Parallel keyed hash function construction based on chaotic maps. Phys Lett A 372(26):4682–4688
Xiao D, Liao X, Wang Y (2009a) Improving the security of a parallel keyed hash function based on chaotic maps. Phys Lett A 373(47):4346–4353
Xiao D, Liao X, Wang Y (2009b) Parallel keyed hash function construction based on chaotic neural network. Neurocomputing 72(10):2288–2296
Xiao D, Shih FY, Liao X (2010) A chaos-based hash function with both modification detection and localization capabilities. Commun Nonlinear Sci Numer Simul 15(9):2254–2261
Yang H, Wong K-W, Liao X, Wang Y, Yang D (2009) One-way hash function construction based on chaotic map network. Chaos Solitons Fractals 41(5):2566–2574
Yi X (2005) Hash function based on chaotic tent maps. IEEE Trans Circuits Syst II Express Briefs 52(6):354–357
Yu H, Lu Y-F, Yang X, Zhu Z-l (2011) One-way hash function construction based on chaotic coupled map network. In: 2011 fourth international workshop on chaos-fractals theories and applications (IWCFTA), pp 193–197. IEEE, New York
Yu-Ling L, Ming-Hui D (2012) One-way hash function construction based on the spatiotemporal chaotic system. Chin Phys B 21(6):060503
Zhang H, Wang X-F, Li Z-H, Liu D-H (2005) One way hash function construction based on spatiotemporal chaos
Zhang J, Wang X, Zhang W (2007) Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. Phys Lett A 362(5):439–448
Zhang Q-H, Zhang H, Li Z-H (2009) One-way hash function construction based on conservative chaotic systems. In: Fifth international conference on information assurance and security, 2009. IAS'09, vol 2, pp 402–405. IEEE, New York