Multimedia Tools and Applications (2019) 78:17973–17994
https://doi.org/10.1007/s11042-018-7122-y

Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security

Yantao Li[1] · **Guangfu Ge**[2]

Received: 1 May 2018 / Revised: 20 December 2018 / Accepted: 26 December 2018 /
Published online: 16 January 2019

Abstract
Cryptographic hash functions can map data of arbitrary size to data of fixed size (hash values), which can be used in a wide range of multimedia applications for communication security, such as integrity protection, message authentication and digital signature. In this paper, we present a cryptographic and parallel chaotic hash function based on the cross coupled map lattices for multimedia communication security. More specifically, we first utilize the piecewise linear chaotic map with secret keys to generate initial parameter sequence for the cross coupled map lattices and an initial hash value. Then, we extend the original message into a message matrix to enhance the correlation of message characters. Next, we process each of the message blocks in the matrix in parallel as the space domain input of the cross coupled map lattices and the initial parameters as the time domain input to generate intermediate hash values. After all message blocks are processed in parallel, the final h-bit hash value is obtained by logical operations with the initial and intermediate hash values. Finally, we evaluate the performance of the proposed hash function in terms of uniform distribution of hash values, sensitivity of the hash value to subtle changes of the original message, secret keys, and images, confusion and diffusion properties, collision tests, efficiency of computation speed. The cryptanalytic results demonstrate that the proposed hash algorithm has statistical properties with $\bar{B} = 64.0022$ and $P = 50.0017\%$, collision resistance with $d = 85.3944$, average computation speed of 132.0 Mbps, and better statistical performance compared with existing chaotic hash functions, which are suitable for multimedia communication security.

Keywords Multimedia communication security · Chaos · Cryptographic hash function · Cross coupled map lattices

✉ Yantao Li
yantaoli@foxmail.com; liyantao@live.com; yantaoli@cqu.edu.cn

[1] College of Computer Science, Chongqing University, Chongqing 400044, China

[2] College of Computer and Information Sciences, Southwest University, Chongqing 400715, China

1 Introduction

With the rapid development of computer and Internet technology, multimedia communication plays a significant role in multiple areas in our social society. The security of multimedia data is becoming more and more important in wired or wireless communications, such as file downloading and online payments [6, 8], image authentication [47] and watermarking [41]. Due to the high redundancy and large amount of data, multimedia data have special requirements for the security protection, such as real time, self-certification and availability. Among the various techniques proposed to address these challenges, hash algorithm has been proven to be an effective and efficient solution, which is able to protect information integrity [50], authenticate messages/images [26, 54], and generate digital signatures/watermarking [46, 69] for multimedia communication security [9, 10, 52] and mobile communication security [11–17, 71].

A hash function is a special kind of one-way function, which can be classified into two categories: (i) unkeyed hash function, whose specification dictates a single input parameter, a message; and (ii) cryptographic hash function, whose specification dictates two distinct inputs, a message and a secret key [27, 55, 61]. Unkeyed hash function $h()$ has four properties: compression, irreversibility, second-preimage resistance and collision resistance. More specifically, (a) compression: a function $h()$ maps an input M with arbitrary bit length, to an output $h(M)$ of fixed bit length l; (b) irreversibility: given a function $h()$ and an input M, it is easy to compute $h(M)$. However, it is computationally infeasible to find any input which hashes to a specific output, i.e., to find any pre-image M such that $h(M) = y$, given any y for which a corresponding input is not known; (c) second-preimage resistance: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given M, to find a second-preimage $M' \neq M$ such that $h(M) = h(M')$; (d) collision resistance: it is hard to find any two distinct inputs M and M', which hash to the same output, i.e., such that $h(M) = h(M')$. On the other hand, a cryptographic hash function h_k is a class of hash functions $\{h_k : k \in V_n\}$ indexed by a key k such that $\{h_k : M \rightarrow V_m\}$ generates a message digest with length l, where V_n denotes the n-dimensional vector space over $GF(2)$. h_k is a secure keyed one-way hash function, if it satisfies the following properties: (a) The function h_k is keyed one-way. That is: (i) given k and M, it is easy to compute $h_k(M)$; (ii) without knowledge of k, it is hard to find M when $h_k(M)$ is given; (iii) without knowledge of k, it is hard to find $h_k(M)$ when M is given; (b) the function h_k should uniformly distribute the message digest in the message digest space. This thwarts statistical attacks; (c) the function h_k is keyed collision free. That is, without the knowledge of k, it is difficult to find two distinct messages M and M' that collide under h_k; (d) the function h_k should produce a message digest with at least 128 bits to thwart birthday attacks; (e) the function h_k should have enough key space to thwart exhaustive key search [25, 31].

Traditional hash functions such as MD5 and SHA-1 are mainly based on logical operations, modular arithmetic operations or digital algebraic operations, which reveal security weakness, since attacks on these algorithms have been discovered [34, 40, 49, 56, 57]. In particular, X. Y. Wang has found an effective method to reduce the complexity of collisions of SHA-1, issued as a Federal Information Processing Standard by NIST [57]. Due to the high computational complexity of logical or xor operations on traditional hash functions, chaos has been exploited to design chaotic hash algorithms for its interesting characteristics, such as sensitivity to tiny changes in initial conditions and parameters, random-like behavior, ergodicity, unstable periodic orbits with long periods and desired diffusion and confusion. Compared with traditional hash algorithms, chaos-based hash algorithms have unstable and aperiodic orbits, and are more unstable dynamical systems with high sensitivity

to initial conditions and more suitable for large-scale data encryption [38]. K. W. Wong is the first to propose the chaotic hash function, which is built on the number of iterations of one-dimensional logistic map needed to reach the region corresponding to the character, along with a lookup table updated dynamically [63]. Then, chaotic hash functions are attracting more and more researchers to research ranging from the use of simple maps such as tent map [4, 32, 37, 70] and logistic map [1, 20, 58] to the use of more complicated maps of the sine map [21], standard map [33], piecewise linear or nonlinear chaotic maps [2, 30, 36, 53, 64, 65, 74], and high-dimensional chaotic maps [3, 22, 26, 35, 43]. For instance, M. Amin designed a chaos-based hash function based on a tent map for cryptographic applications [4]. Y. Wang provided a one-way hash function based on iterating a logistic map [58]. M. Ahmad provided a simple secure hash function scheme using multiple chaotic maps including logistic map, tent map, skew-tent map, cubic map and baker map [1]. A. Akhavan designed a hash function based on piecewise nonlinear chaotic map [2]. P. Zhang presented a parallel and collision resistance hash function with variable initial values [74]. Z. Lin developed a methodology to construct keyed hash functions based on chaotic iterations to avoid dynamic degradation caused by finite precision [36]. M. Todorova proposed a hash function based on irregularly decimated chaotic map [53]. H. S. Kwok presented a chaos-based cryptographic hash function for message authentication based on a high-dimensional cat map [26]. Z. Lin again designed an approach for constructing one-way hash function based on a message block controlled 8D hperchaotic map [35].

Although the above hash algorithms have their advantages, most of them ignore the parallel characteristic of message processing, which effectively improves the computation speed. In this paper, we present a cryptographic and parallel hash algorithm based on the cross coupled map lattices for multimedia communication security. More specifically, we first iterate the piecewise linear chaotic map with secret keys l times to generate a parameter sequence A for the cross coupled map lattices and generate an initial h-bit hash value H_0. Then, we extend an arbitrary length of message M into a message matrix M'' to enhance the correlation of message characters. Next, we iterate the cross coupled map lattices $CCML$ $2h$ times to generate a h-bit intermediate hash value H_i. There are two inputs for the CCML: the space domain input, which is the message blocks M_i'' in the matrix M'' and the time domain input, which is the parameters C from sequence A. After all message blocks are processed in parallel, the final h-bit hash value is obtained by logical operations with H_0 and $H_i (i = 1, 2, \ldots)$. Finally, we evaluate the performance of the proposed hash function in terms of uniform distribution of hash values, sensitivity of the hash value to subtle changes of the original message and secret keys, confusion and diffusion properties, collision tests, efficiency of computation speed, and the cryptographic results demonstrate that the hash algorithm has good statistical properties, strong collision resistance, and better statistical performance compared with existing chaotic hash functions. We differ in that we are among the first to exploit a two-dimensional cross coupled map lattices with space domain and time domain inputs to design a cryptographic hash function and that the proposed hash function can be performed in parallel.

The main contributions of this work can be summarized as:

- We present a cryptographic and parallel hash algorithm based on the cross coupled map lattices for multimedia communication security. Different from the chaotic maps-based parallel keyed hash scheme [64], we are among the first to exploit the cross coupled map lattices to construct the hash algorithm.
- We utilize message blocks as the space domain input and parameter sequence from the piecewise linear chaotic map as the time domain input for the cross coupled map lattices

to generate intermediate hash values. We differ from the 2D coupled map lattices-based hash scheme [59] in that the cross coupled map lattices have space and time domain inputs, which significantly compress the message information into intermediate hash values.

- We evaluate the performance of the proposed hash algorithm, and the cryptanalytic results demonstrate that the hash algorithm has good statistical properties, strong collision resistance, and better statistical performance compared with existing chaotic hash functions. Comparing to [23], our algorithm shows better performance in the average computation speed, statistical analysis, and collision resistance.

The remainder of this paper is organized as follows: Section 2 briefly describes the piecewise linear chaotic map and the crossing coupled map lattices. In Section 3, we present the design of the cryptographic and parallel hash algorithm based on the cross coupled map lattices in detail. We evaluate the performance of the proposed hash algorithm in Section 4 and conclude the work in Section 5.

2 Preliminaries

In this section, we briefly describe the one-dimensional piecewise linear chaotic map and the two-dimensional crossing coupled map lattices used in the proposed hash algorithm, respectively.

2.1 Piecewise linear chaotic map (*PWLCM*)

We select the one-dimensional piecewise linear chaotic map in the proposed hash algorithm, which is expressed as in (1):

$$x_{i+1} = \mathrm{PWLCM}(\alpha, x_i) = \begin{cases} x_i/\alpha & 0 \le x_i < \alpha; \\ (x_i - \alpha)/(0.5 - \alpha) & \alpha \le x_i < 0.5; \\ (1 - x_i - \alpha)/(0.5 - \alpha) & 0.5 \le x_i < 1 - \alpha; \\ (1 - x_i)/\alpha & 1 - \alpha \le x_i \le 1. \end{cases} \tag{1}$$

where x_i represents the iteration trajectory value, and α denotes the control parameter. When α is assigned values in (0, 0.5), x_i evolves into a chaotic state in range of (0, 1), and there is no dynamical degradation of $PWLCM$ because only rounded values of x_i (0 or 1) are needed in the hash algorithm [24]. The $PWLCM$ has properties of uniform distribution, good ergodicity, confusion and diffusion, therefore, it can provide chaotic random sequences. An explicit analysis of the bifurcation diagram of the $PWLCM$ shows that with the specified initial value x_0 and parameter α, its iterative values are fixed, which are listed in Table 1. The map is running in a chaotic state within the range (0, 1), except for the specified values in Table 1. Therefore, we use the map as a key generator to produce the

Table 1 The fixed iterative values of $PWLCM$ with the specified initial value x_0 and parameter α

x_0	α	$x_i (i = 1, 2, \ldots)$
< 0.25	$2 \times x_0$	0.5,1,0,0, …
< 0.5	x_0	0, 0, 0, 0, …
0.5	(0, 1)	1, 0, 0, 0, …
(0, 1)	0.25	0.5, 1, 0, 0, …

four initial buffers for our hash algorithm. Moreover, according to Ref. [5], $\{x_i\}$ is ergodic, uniformly distributed in the interval (0, 1) and the autocorrelation function is δ-like.

2.2 Cross coupled map lattices (*CCML*)

We select the two-dimensional cross coupled map lattice of size L with nearest neighbor coupling in the proposed hash algorithm, which is a mixed model of coupled map lattices with a diffusion process and local reaction process. $CCML$ is defined by (2):

$$x_{n+1}(i) = \mathrm{CCML}(\varepsilon, q_i) = \begin{cases} \frac{f(x_n(i))}{1+\varepsilon} + \frac{\varepsilon}{2(1+\varepsilon)}(x_n(i-1)) + x_n(i+1) & i \bmod 2 = 0; \\ \frac{f(x_n(i))}{1+\varepsilon} + \frac{\varepsilon}{2(1+\varepsilon)}(x_{n+1}(i-1)) + x_{n+1}(i+1) & i \bmod 2 = 1. \end{cases} \tag{2}$$

where n denotes the number of discrete-time steps in $[1, N]$, referred to as the time dimension, and i represents the number of discrete-space steps in $[1, L]$, referred to as the space dimension. $x_n(i)$ denotes the state of site i (or ith lattice) at time n, and ε indicates a coupling coefficient in range of (0, 1). Periodic boundary conditions meet $x_n(i) = x_n(i + L)$. Lattice mapping function $f()$ is an asymmetric tent map that creates the local dynamics, which is defined by (3):

$$x_{n+1} = \begin{cases} \frac{x_n}{q} & 0 \le x_n < q; \\ \frac{1-x_n}{1-q} & q \le x_n \le 1. \end{cases} \tag{3}$$

where q ranges in (0, 1). Since the asymmetric tent map $f()$ is a chaotic map, $CCML$ model is chaotic. $CCML$ uses the way of diffusion in both time dimension (x-axis in Fig. 2) and space dimension (y-axis). In even lattice point of each iteration (black dots), the current value depends on even value in previous iteration step and both-side odd values in previous iteration step. In odd lattice point of each iteration (white circle), the current value depends on odd value in previous iteration step and both-side even values in current iteration step. Therefore, even lattice point reacts first and continues completing diffusion process before odd lattice point does, and then odd lattice point performs reaction and diffusion processes. The state of odd lattice points differs half step unit time than that of even lattice points. We cascade the two chaotic maps ((2) and (3)) to counteract the degradation of various extents [62].

We believe $CCML$ is superior to other maps for our specified hash algorithm, because: 1) we exclusively design the two-dimensional $CCML$ to enhance the security of the hash function construction, since it considers the message blocks as the space domain input and the parameter sequence from $PWLCM$ as the time domain input; and 2) we are among the first to exploit $CCML$ to design a cryptographic and parallel hash algorithm.

3 Cryptographic and parallel hash function based on *CCML*

In this section, we describe the proposed cross coupled map lattices based parallel chaotic hash function in detail. We first detail the proposed hash function in four steps: parameter initialization, message extension, message processing, and hash value generation, as shown in the structure of the hash function in Fig. 1. Then we present the hash function in Algorithm 1, and illustrate the flowchart of the proposed hash function in Fig. 3. As illustrated

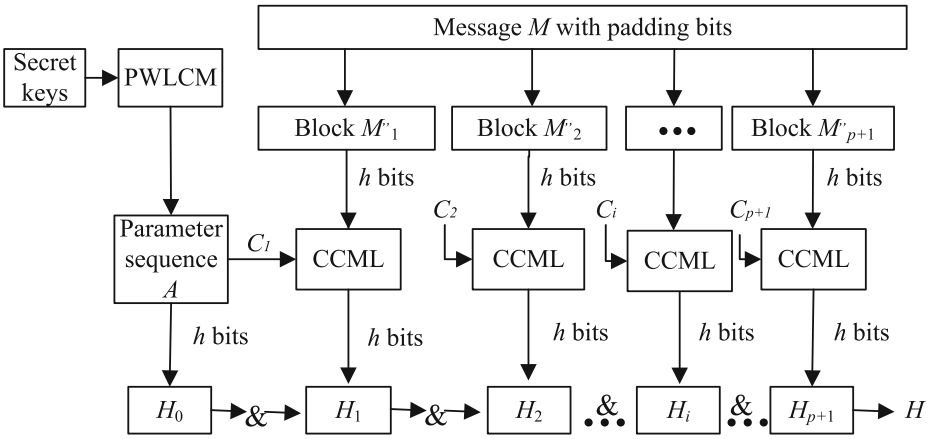


Fig. 1 The structure of the chaotic hash algorithm.

in Algorithm 1 and Fig. 3, the inputs are an arbitrary length of message M and secret keys $\{x_0, p_0, \varepsilon, q_0\}$, and the output is h-bit hash value H, where $h = 128$.

Step 1: Parameter Initialization With initial secret keys $\{x_0, p_0\}$, we iterate $PWLCM$ l times and intend to generate a parameter sequence A for $CCML$, where $l = (n'+h)\times 2$ and $n' = \lceil \frac{n}{h-1} + 1 \rceil \times h$. Let n denote the number of characters in the arbitrary length of message M. Since parameter p is variable, for each iteration i $(i = 1, 2, \ldots, l)$, $p_i = \frac{1}{2}(p_{i-1} + x_{i-1})$ and $x_i = \frac{1}{2}(x_i + m_{i-2h} \times \frac{1}{2h})$ if $i \geq 2h$. After l iterations, we obtain a chaotic sequence $S = \{s_i = x_i | i = 1, 2, \ldots, l\}$ and then the parameter sequence $A = \{a_j = x_i | i = 2h+1, 2h+2, \ldots, l;\ j = 1, 2, \ldots, 2n'\}$ is generated. With $A = \{a_i | i = 1, 2, \ldots, 2n'\}$, we calculate $T = \{t_i = round(a_i) | i = 1, 2, \ldots, 2n'\}$ and then conduct XOR operation on each h-bit binaries in T, and finally cascade them to generate the h-bit initial hash value H_0.

Step 2: Message Extension Given the arbitrary length of message M, we extend it into a matrix M'' to enhance the correlation of message characters. In order to make the original message M a multiple length of $((h-1)\times 8)$ bits, we first append $\{1010\ldots10\}_2$ bits to message M, and then preserve 64 bits rightmost denoting the length of original message M. Then we convert each 8-bit character of the padded message into the corresponding ASCII code value (a decimal integer) and store them into an array m $(m = \{m_1, m_2, \ldots\})$. Finally, we obtain a $p \times (h-1)$ message matrix M' by:

$$M' = \begin{bmatrix} m'_{1,1} & m'_{1,2} & \cdots & m'_{1,h-1} \\ m'_{2,1} & m'_{2,2} & \cdots & m'_{2,h-1} \\ \vdots & \vdots & \ddots & \vdots \\ m'_{p,1} & m'_{p,2} & \cdots & m'_{p,h-1} \end{bmatrix} = \{m'_{i,j} = m_{(h-1)(i-1)+j} | i = 1, 2, \ldots, p;\ j = 1, 2, \cdots, h-1\}.$$

Then, we generate a transform sequence $B = \{b_i = \lfloor a_i \times 2h \rfloor | i = 1, 2, ..., 2n'\}$, where $a_i \in A$. Based on M' and B, we obtain elements of M'': $m''_{i,j} = m'_{i,j} \oplus b_k + b_{n'+k}$, $r_i = \sum_{j=1}^{h-1} m''_{i,j}$, $c_j = \oplus_{i=1}^{p} m''_{i,j}$, where "$\oplus$" represents bitwise exclusive OR operation, "+" denotes addition modulo 2^8, and $k = (h-1)\times(j-1)+j$. Then, we extend the original message M into a $(p+1)\times h$ matrix M'' as shown:

$$M'' = \begin{bmatrix} m''_{1,1} & m''_{1,2} & \cdots & m''_{1,h-1} & r_1 \\ m''_{2,1} & m''_{2,2} & \cdots & m''_{2,h-1} & r_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m''_{p,1} & m''_{p,2} & \cdots & m''_{p,h-1} & r_p \\ c_1 & c_2 & \cdots & c_{h-1} & c_h \end{bmatrix}$$

Step 3: Message Processing Each message block M''_i $(i = 1, 2, ..., p+1)$ in matrix M'' will be processed by $CCML$ in parallel and the corresponding intermediate hash value H_i will be generated. We select M''_i as an instance to demonstrate the process, which is illustrated in Fig. 2. In time domain, based on the parameter sequence A, we obtain sequence $C_i = \{c_{i,j} = a_{(i-1)\times 2h+j} | j = 1, 2, \ldots, 2h\}$, which are used as the lattice boundary values for step "i" (green dots in Fig. 2) in $CCML$. In space domain, the elements $M''_i = \{m''_{i,1}, m''_{i,2}, ..., m''_{i,h}\}$ in matrix M'' are used as initial values for step "n" (red dots) in $CCML$, respectively. We iterate $CCML$ $2h$ times with secret keys $\{\varepsilon, q_0\}$ and variable parameter $q_{i+1} = q_i + b_{i,j} \times 10^{-3}$ to generate h-bit binaries (yellow dots), that is, one bit (a yellow dot) generation is associated to one element $m''_{i,j}$ (a red dot) in current message block M''_i by $CCML$ iterations. For h elements in M''_i, h-bit binaries are generated, which are cascaded sequentially as intermediate hash value H_i (Fig. 3).

Step 4: Hash value generation After all message blocks in M'' are processed in parallel, we obtain the final hash value $H = H_0 \& H_1 \& \ldots \& H_p \& H_{p+1}$, where "&" is defined as (4):

$$\& = \begin{cases} (H \oplus H_i) << 1 \ \ i \bmod 2 == 0; \\ (H \oplus H_i) << 2 \ \ i \bmod 2 == 1. \end{cases} \tag{4}$$

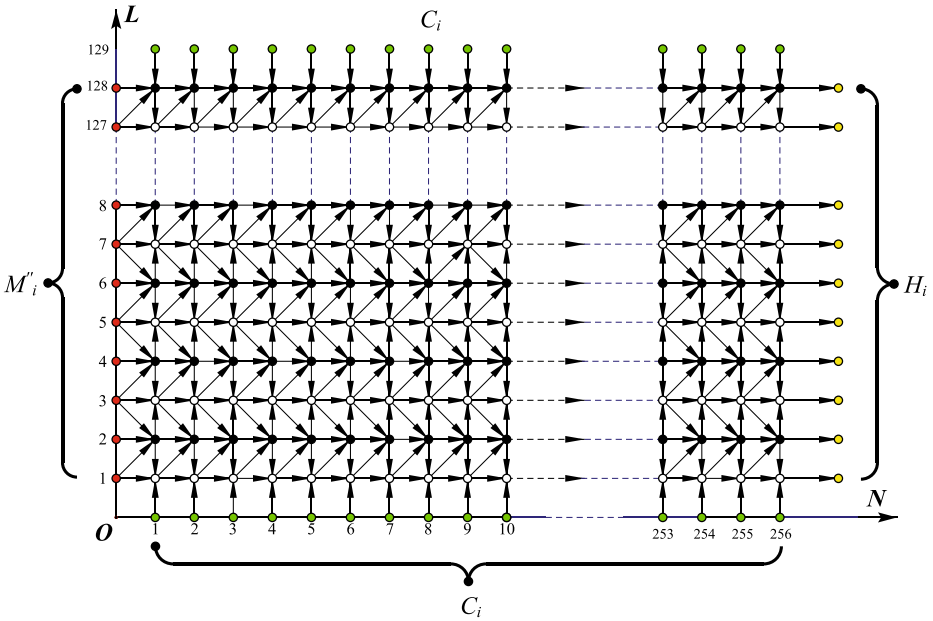


Fig. 2 The detailed process of message block M''_i

Algorithm 1 Cryptographic and parallel hash function based on $CCML$.

Input: an arbitrary-length message: M and secret keys: $\{x_0, p_0, \varepsilon, q_0\}$
Output: h-bit hash value: H

1 Iterate $PWLCM$ l ($l = (n' + h) \times 2$ and $n' = \lceil \frac{n}{h-1} + 1 \rceil \times h$) times with secret keys $\{x_0, p_0\}$ ($p_i = \frac{1}{2}(p_{i-1} + x_{i-1})$ and $x_i = \frac{1}{2}(x_i + m_{i-2h} \times \frac{1}{2h})$ if $i \geq 2h$, for $i = 1, 2, \ldots, l$);
2 Generate a chaotic sequence $S = \{s_i = x_i | i = 1, 2, \ldots, l\}$ and a parameter sequence $A = \{a_j = x_i | i = 2h + 1, 2h + 2, \ldots, l;\ j = 1, 2, \ldots, 2n'\}$ for $CCML$;
3 Calculate $T = \{t_i = round(a_i) | i = 1, 2, \ldots, 2n'\}$ ($A = \{a_i | i = 1, 2, \ldots, 2n'\}$);
4 Conduct XOR operation on each h-bit binaries in T;
5 Cascade h-bit binaries to generate H_0; //$Parameter\ initialization$
6 Extend an arbitrary-length message M to a message with a multiple length of $(h-1) \times 8$ bits;
7 Convert each 8-bit character of extended message M into the corresponding ASCII code value to obtain a $p \times (h-1)$ message matrix $M' = \begin{bmatrix} m'_{1,1} & m'_{1,2} & \cdots & m'_{1,h-1} \\ m'_{2,1} & m'_{2,2} & \cdots & m'_{2,h-1} \\ \vdots & \vdots & \ddots & \vdots \\ m'_{p,1} & m'_{p,2} & \cdots & m'_{p,h-1} \end{bmatrix}$ $= \{m'_{i,j} = m_{(h-1)(i-1)+j} | i = 1, 2, \ldots, p;\ j = 1, 2, \cdots, h-1\}$;
8 M' is extended to a $(p+1) \times h$ matrix $M'' = \begin{bmatrix} m''_{1,1} & m''_{1,2} & \cdots & m''_{1,h-1} & r_1 \\ m''_{2,1} & m''_{2,2} & \cdots & m''_{2,h-1} & r_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m''_{p,1} & m''_{p,2} & \cdots & m''_{p,h-1} & r_p \\ c_1 & c_2 & \cdots & c_{h-1} & c_h \end{bmatrix}$ ($m''_{i,j} = m'_{i,j} \oplus b_k + b_{n'+k}$, $r_i = \sum_{j=1}^{h-1} m''_{i,j}$, $c_j = \oplus_{i=1}^{p} m''_{i,j}$ and $B = \{b_i = \lfloor a_i \times 2h \rfloor | i = 1, 2, \ldots, 2n'\}$; //$Message\ extension$
9 **for** $\forall\ M''_i$ ($i = 1, 2, \ldots, p+1$) **do**
10 Iterate $CCML$ $2h$ times with secret keys $\{\varepsilon, q_0\}$ ($q_{i+1} = q_i + b_{i,j} \times 10^{-3}$), and inputs of $C_i = \{c_{i,j} = a_{(i-1)\times 2h+j} | j = 1, 2, \ldots, 2h\}$ and $M''_i = \{m''_{i,1}, m''_{i,2}, \ldots, m''_{i,h}\}$ to generate h-bit hash value of H_i of M''_i; //$Message\ processing$
11 **end**
12 Return $H = H_0 \& H_1 \& \ldots \& H_p \& H_{p+1}$; //$Hash\ value\ generation$

4 Performance evaluation

In this section, we implement the cross coupled map lattices ($CCML$) based cryptographic and parallel hash function for performance evaluation by utilizing secret keys $x_0 = 0.676767$, $p_0 = 0.232323$, $\varepsilon = 0.333333$ and $q_0 = 0.375281$. We evaluate the parallel hash algorithm in terms of uniform distribution of hash values, sensitivity of the hash value to subtle changes of the original message, secret keys and images, confusion and diffusion properties, collision tests, efficiency of computation speed, and comparison with other algorithms. An arbitrary of length of message M for evaluating the performance of the proposed hash algorithm is randomly chosen as:

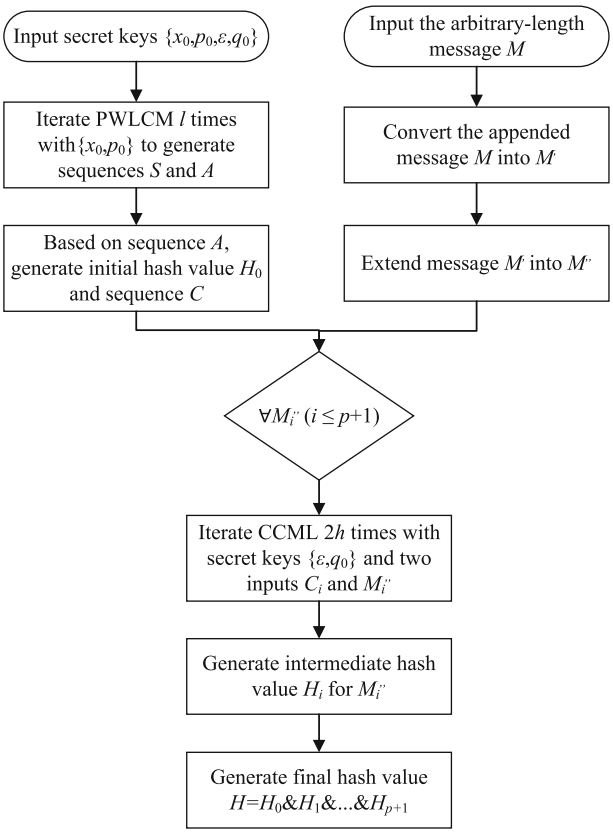


Fig. 3 The flowchart of the chaotic hash algorithm

Southwest University (SWU) is a key comprehensive university, under the direct administration of the Ministry of Education. It was newly established in July 2005 through the incorporation of former Southwest China Normal University and Southwest Agricultural University upon the approval of the Ministry of Education. SWU is situated nearby the beautiful Jialing River, and is located at the foot of Jinyun Mountain, a state level scenic spot, in Beibei District, Chongqing Municipality.

4.1 Uniform distribution of hash values

Uniform distribution of hash values indicates that hash values are uniformly randomly distributed into "buckets", which is directly related to the security of hash functions. The uniform distribution of a hash value is one of the significant security features of hash functions. We evaluate the uniform distribution of hash values by implementing the proposed hash algorithm with a randomly chosen message, and then plot the distribution of the message and the corresponding hash value. As demonstrated in Fig. 4a, the original message spreads in a range of [32, 126], which fits the range of ASCII code values of printable characters (such as message) in ASCII code chart. As illustrated in Fig. 4b, the hexadecimal hash value spreads around randomly and uniformly, which hides the statistical information

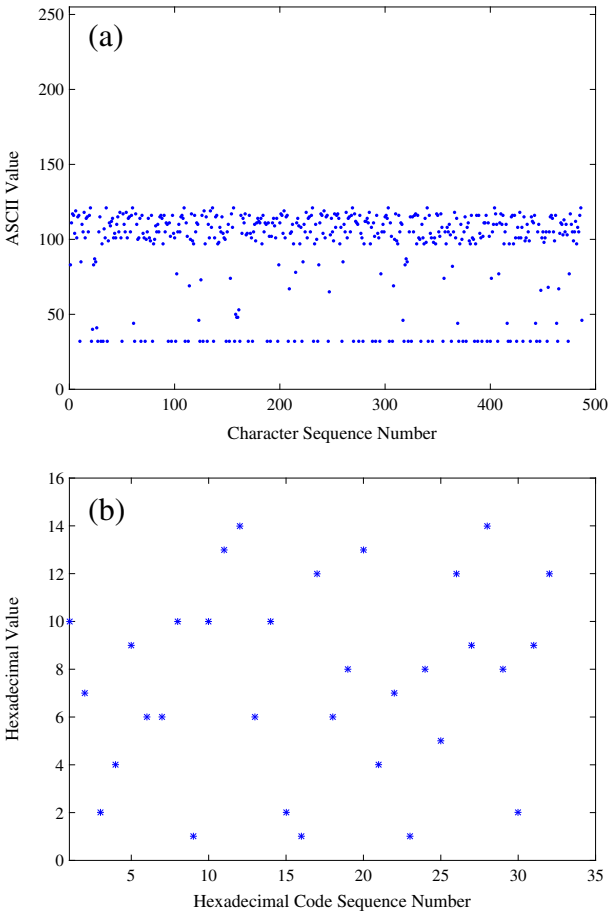


Fig. 4 Spread of message and hash value: **a** distribution of the message in ASCII code; **b** distribution of the hash value in hexadecimal format (A724966A1ADE6A21C68D47185C9E829C)

of the message. In contrast, we evaluate the proposed algorithm on an extreme case - "blank space" message with the same length, and then plot the distribution of the particular message and the hash value as well. As shown in Fig. 5, even under such an extreme condition, the distribution of hash value is still uniform. These distributions are well uniform enough to hide information and act as a strong security measure. Therefore, the proposed hash algorithm has a good characteristic of uniform distribution on hash values.

4.2 Hash sensitivity

The irreversibility property indicates that it is computationally infeasible to find any input message for a given hash value, which entails that a hash algorithm should have excellent message and key sensitivity. That is, a good hash algorithm should be sensitive to tiny modifications in messages, secret keys, as well as images. According to Hamming distance, any slight modifications on messages, secret keys or images will lead to a 50% difference in the hash value. We evaluate the hash sensitivity of the proposed hash algorithm to the

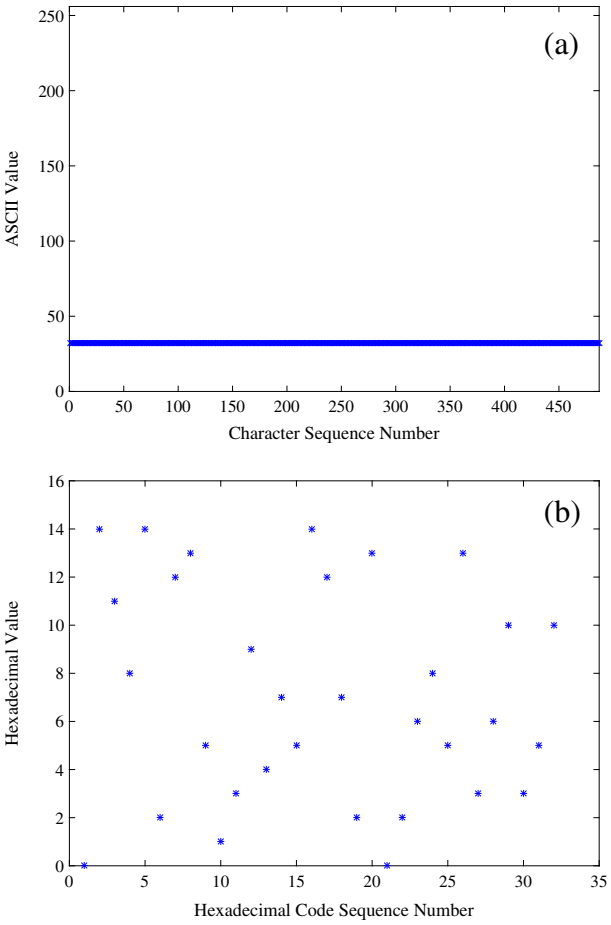


Fig. 5 Spread of all "blank space"-message and hash value: **a** distribution of all "blank space"-message; **b** distribution of the hash value in hexadecimal format (0EB8E2CD5139475EC72D02685D36A35A)

original message and secret keys under ten different conditions (Conditions 1 to 10) and to a grey-scale Lena image with 256×256 image size in Fig. 6 under three different conditions (Conditions 11 to 13):

Condition 1: The original randomly chosen message;
Condition 2: Change the first character "*S*" in the original message into "*T*";
Condition 3: Change the word "*direct*" in the original message into "*directly*";
Condition 4: Swap "*Southwest China Normal University*" and "*Southwest Agricultural University*" in the original message;
Condition 5: Change the full stop "." at the end of the original message into comma ",";
Condition 6: Add a blank space to the end of the original message;
Condition 7: Change the initial value $x_0 = 0.676767$ to $x_0 = 0.676767000001$;
Condition 8: Change the control parameter $p_0 = 0.232323$ to $p_0 = 0.232323000001$;
Condition 9: Change the coupling coefficient $\varepsilon = 0.333333$ to $\varepsilon = 0.333333000001$;
Condition 10: Change the parameter $q_0 = 0.375281$ to $q_0 = 0.375281000001$;

Fig. 6 The standard grey-scale Lena image with 256 × 256 image size

Condition 11: The original grey-scale Lena image with 256 × 256 image size (Fig. 6);
Condition 12: Add 1 to the gray value of the pixel located at the upper left corner;
Condition 13: Subtract 1 to the gray value of the pixel located at the upper right corner.

We illustrate the corresponding hash values in binary format associated to the thirteen conditions in Fig. 7, and tabulate the hash values in hexadecimal format as well as Hamming distances from Condition 1 for Conditions 2 to 10 and from Condition 11 for Conditions 12 and 13 in Table 2. As depicted in Fig. 7, the hash sensitivity property of the proposed algorithm to text is good, since any subtle change of the message (Conditions 2 to 6) causes large difference in hash values, and any tiny modification of the secret keys (Conditions 7 to 10) leads to huge difference. It also shows a good sensitivity property to images that 1 bit of the gray value change causes much difference in hash values (Conditions 12 and 13). As illustrated in Table 2, Hamming distances from Condition 1 have an average value of 64.6, which is significantly close to the ideal value of 64 (half of hash value size), and from Condition 11 have an average value of 66. These prove that the proposed hash algorithm

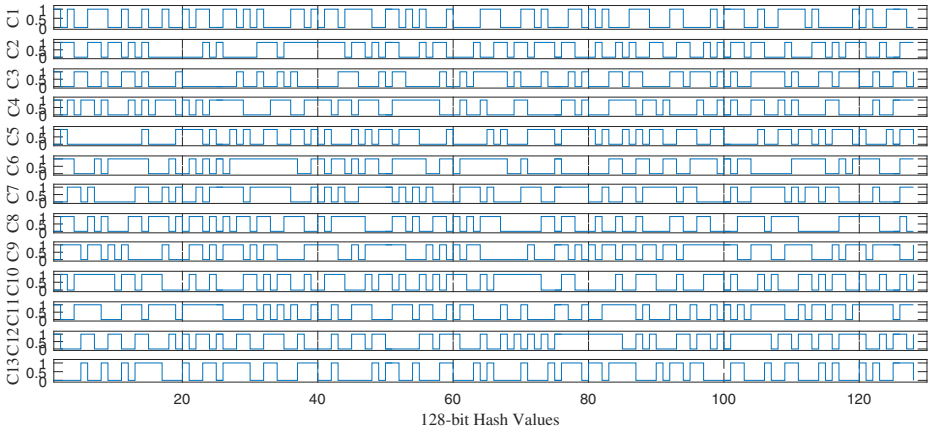


Fig. 7 Hash values in binary format under thirteen different conditions

Table 2 The corresponding hash values in hexadecimal format to the thirteen conditions

Message	Hash value in hexadecimal format	Hamming distance
Condition 1	A724966A1ADE6A21C68D47185C9E829C	0
Condition 2	66D40283BFED62C26DD894CCAD88D727	65
Condition 3	993420125038B04BE88A361BA9F46E4D	64
Condition 4	AD15D5F0E7B73F890C1A3DA12994304F	65
Condition 5	40043D299A8ADC20A3DF4ACE609B5D16	56
Condition 6	E2FC4ABFF36B3E6FE4203333D607CBC7	64
Condition 7	340CA5F7E2A7D50E873E4DF609F5B047	70
Condition 8	651BAD5B397C34D5C0D8921987BC0FC4	60
Condition 9	76A08D7587CBF15602356DB7F9CF1779	66
Condition 10	5FB7090D64E2D2B97F181388E8A35925	71
Condition 11	9E3BDF8D524E3B2C24327D1255F2DBF7	0
Condition 12	8CCC5699C84D83712ABFF28DAFDB6F2D	65
Condition 13	092F13798E416462845DBC5B0C4C514E	67

satisfies the irreversibility property of a cryptographic hash function. Therefore, our hash algorithm shows high hash sensitivity to messages, secret keys, and images.

4.3 Confusion and diffusion

In cryptography, confusion and diffusion are two properties of the operation of a secure cipher, which are identified by Claude Shannon [48]. Confusion refers to making the relationship between the key and the ciphertext as complex and as involved as possible, and diffusion refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. In our evaluation, confusion refers to as the relationship between a message and its corresponding hash value must be complex and unpredictable, while diffusion refers to as the hash value is highly dependent on the message.

We conduct the diffusion and confusion experiment for the proposed hash algorithm: a message is randomly selected and the hash value for the message is generated; then one bit of the message is modified randomly, and a new hash value is generated. The two hash values are compared with each other, and the number of different bits at the same position in the two hash values is counted. We introduce six statistical metrics for evaluation of confusion and diffusion: minimum changed bit number $B_{\min} = \min\{B_1, B_2, ..., B_N\}$, maximum changed bit number $B_{\max} = \max\{B_1, B_2, ..., B_N\}$, mean changed bit number $\bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$, mean changed probability $P = \frac{\bar{B}}{h} \times 100\%$, standard variance of the changed bit number $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \bar{B})^2}$, and standard variance $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{h} - P)^2} \times 100\%$, where B_i denotes the changed bit number, N indicates the test time of the experiment, and h represents the length of hash value.

The experiment is performed N times on the proposed hash algorithm, where $N = 256$, 512, 1024, 2048, and 10000, respectively. The corresponding results of $B_{\min}$, $B_{\max}$, $\bar{B}$, P, ΔB, and ΔP are tabulated in Table 3. The corresponding distribution of changed bit number $\bar{B}$, is illustrated in Fig. 8.

Table 3 Statistics of number of changed bits

N	$N = 256$	$N = 512$	$N = 1024$	$N = 2048$	$N = 10000$	$Mean$
$B_{\max}$	78	78	79	82	85	80.4000
$B_{\min}$	50	47	47	45	42	46.2000
$\bar{B}$	63.9844	64.0176	64.0518	63.9438	64.0133	64.0022
$P(\%)$	49.9878	50.0137	50.0404	49.9561	50.0104	50.0017
ΔB	5.500067	5.554012	5.279875	5.607089	5.600470	5.5083
$\Delta P(\%)$	4.2969	4.3391	4.1249	4.3805	4.3754	4.3034

As illustrated in Table 3, the proposed hash algorithm has a mean changed bit number $\bar{B} = 64.0022$ and mean changed probability $P = 50.0017\%$ that are extremely close to the ideal values of 64 bits and 50%, respectively. The values of ΔB and ΔP are very small, which shows a strong capability for confusion and diffusion. As depicted in Fig. 8, the plot

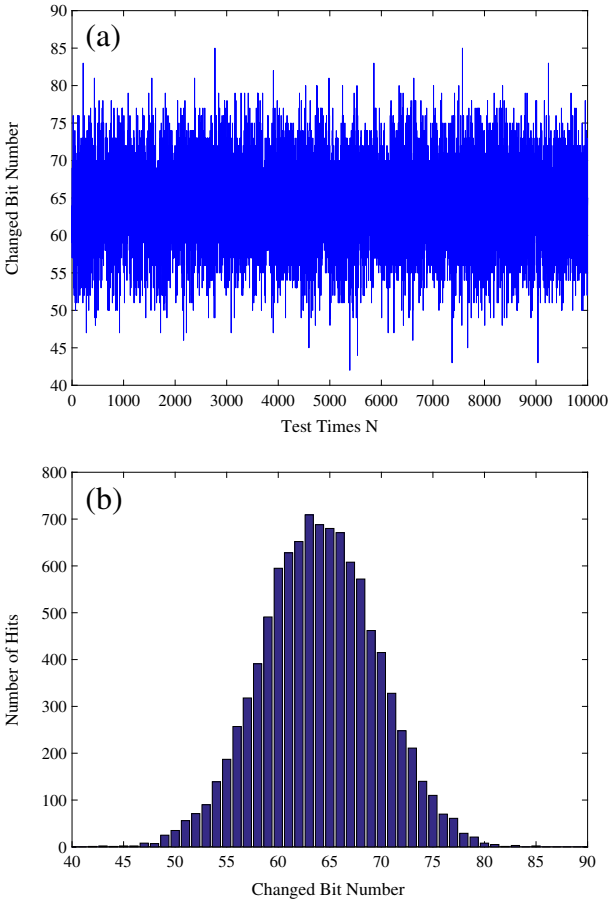


Fig. 8 Spread of changed bit number: **a** Plot of B_i, **b** Histogram of B_i

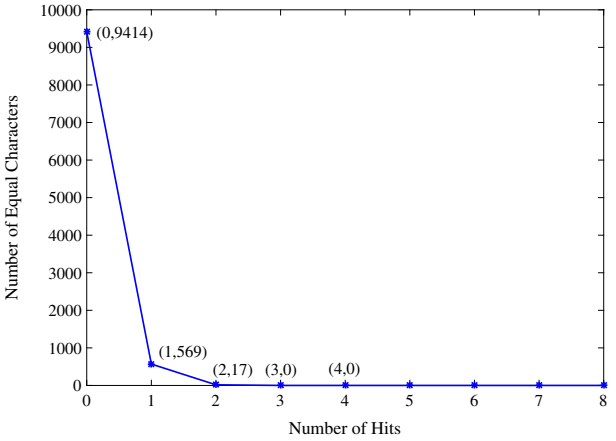


Fig. 9 Distribution of the number of the same ASCII characters at the same location in hash value

of B_i shows that its value is evenly distributed (Fig. 8a), and the histogram of B_i has a normal distribution centering on the ideal value of 64 (Fig. 8b). The statistical confusion and diffusion results ensure that the proposed algorithm exhibits the competency to mitigate any kind of linear or differential attacks related to hash values. Therefore, the proposed hash function has a near-ideal confusion and diffusion strength.

4.4 Collision resistance

Collision refers to as two distinct messages produce the same hash value, while collision attack indicates that it tries to find two arbitrary messages that collide. Collision resistance is an important property of a secure hash function, which refers to as it is hard to find two different message with the same hash value.

In the proposed hash algorithm, the state of the chaotic $CCML$ is related to message blocks (space domain input) and the sequence of $PWLCM$ (time domain input). The sequence of $PWLCM$ is affected by the control parameter and initial conditions, which will be assigned values if the algorithm is designed. Therefore, the state of $CCML$ is directly related to each message bit in message blocks. These ensure that each bit of the final hash value is related to all the bits of the message. That is, even 1-bit change in the message would lead to a completely different in hash values.

We conduct collision resistance experiment for the proposed hash algorithm: a hash value for a randomly chosen message is generated and stored in ASCII code format; then a new hash value for the message with a bit randomly modified is generated and stored in ASCII code format as well. The two hash values are compared with each other, and the number of the same ASCII character at the same location (the number of hits) is counted. The collision resistance experiment is conducted $N = 2048$ times, and the distribution of the number of hits is plotted in Fig. 9. As described in Fig. 9 and Table 4, 2 tests hit twice, 117 tests hit

Table 4 Number of hits

Number of equal characters	0	1	2	3	4	5	6	7	8
The proposed algorithm	1930	117	2	0	0	0	0	0	0

once, while in 1930 tests, no hit occurs. The maximum number of equal characters is only 2, therefore, the collision on our hash algorithm is very low.

Moreover, we calculate the absolute difference of the two hash values by using equation of $d = \sum_{i=1}^{N} |t(e_i) - t(e_i')|$, where e_i denote the ith ASCII character of the original hash value while e_i' represents the ith ASCII character of the new hash value, respectively, and the function $t()$ converts the the entries into the equivalent decimal values. We perform the collision test $N = 10000$ times, and the corresponding maximum, minimum, mean and mean/character values of the absolute difference d for two hash values are 2316, 553, 1366.3105, and 85.3944, respectively. For our algorithm, the mean/character of absolute difference d of two hash values is 85.3944, which is very close to the theoretical mean/character value 85.3333 computed in [45, 67]. Therefore, our mean/character value is a near theoretical value, and the analysis on collision shows that our hash algorithm has strong collision resistance.

4.5 Efficiency

In order to analyze the efficiency of computation speed, we implement the proposed hash algorithm in C99 on a PC with 2.50 GHz Intel Pentium IV Dual-core, 2G Memory and Ubuntu 10.10 operation system and the test message consists of 10000 ASCII codes. The proposed parallel hash function is implemented in a distributed memory architecture, where the message is split and saved to the local memory for each message block. In theory, parallel computation optimizes the use of all processors in a multicore computer. In implementation, it subjects to the number of cores on a computer. Therefore, the degree of parallelism for the proposed algorithm is 2 message blocks, each with a size of 128 bits. The overhead of message separation is quite low, which can be ignored. Furthermore, we implement the widely used MD5 [45] and SHA-1 [42] algorithms in C99 with optimized codes on the same conditions to our algorithm as well. Finally, we present the average computation speed comparison based on the same platform as ours in Table 5. As illustrated in Table 5, the average computation speed of our algorithm (132.0 Mbps) is higher than Li's algorithm (131.1 Mbps) [31], SHA-1 (114.5 Mbps) [45], Guo's algorithm (131.3 Mbps) [18], while it is very close to Li's algorithm [33] (132.1 Mbps) and MD5 (132.1 Mbps) [42].

Table 5 Average computation speed comparison

Algorithms	Platform (on PC)	Average computation speed
Li's [31]	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	131.1 Mbps
Li's [33]	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	132.1 Mbps
Rivest's [45]	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	132.1 Mbps
NIST [42]	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	114.5 Mbps
Guo's [18]	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	131.3 Mbps
This scheme	C99, Pentium IV 2.50 GHz Dual-core CPU, 2G RAM under Ubuntu 10.10	132.0 Mbps

Table 6 Comparison on statistical performance with $N = 2048$ random tests and 128-bit hash value

Algorithm	Statistical performance of the algorithms			
	$\bar{B}$	$P(\%)$	ΔB	$\Delta P(\%)$
Li's [31]	63.873	49.901	5.581	4.360
Ahmad's [1]	64.129	50.101	5.605	4.378
Li's [33]	63.99	49.99	5.66	4.40
Li's [30]	63.57	49.66	7.43	5.80
Xiao's [64]	63.92	49.94	5.62	4.39
Xiao's [65]	64.09	50.07	5.48	4.28
Zhang's [74]	128.014	50.006	5.450	2.123
Lin's with MT [36]	63.78	49.83	5.56	4.35
Todorova's [53]	63.99	49.99	5.51	4.33
Kanso's [22]	63.94	49.95	5.69	4.44
Lin's [35]	63.98	49.99	5.71	4.46
MD5[45]	64.03	50.02	5.66	4.42
Wang's [59]	63.98	49.98	5.53	4.33
Zhang's [72]	64.43	49.96	5.57	4.51
Deng's [7]	63.84	49.88	5.88	4.59
Guo's [19]	63.40	49.53	7.13	6.35
Kanso's [23]	64.01	50.01	5.61	4.38
Li's [28]	64.17	50.13	5.75	4.49
Ren's [44]	63.92	49.94	5.78	4.52
Li's [29]	63.81	49.85	5.76	4.50
Teh's [51]	64.01	50.01	5.66	4.26
Wang's [60]	64.15	50.11	5.77	4.51
Xiao's [66]	64.01	50.01	5.72	4.47
Xiao's [68]	64.18	50.15	5.67	4.41
Zhang's [73]	63.91	49.92	5.58	4.36
This scheme	63.94	49.96	5.61	4.38

Table 7 Comparison on statistical performance with $N = 10000$ random tests and 128-bit hash value

Algorithm	Statistical performance of the algorithms			
	$\bar{B}$	$P(\%)$	ΔB	$\Delta P(\%)$
Ahmad's [1]	63.781	49.828	5.937	4.638
Li's [33]	64.00	50.00	5.72	4.47
Todorova's [53]	64.00	50.01	5.6	4.37
Wang's [59]	63.90	49.91	5.58	4.36
Kanso's [23]	63.94	49.95	5.64	4.41
Lin's [35]	63.95	49.96	5.62	4.39
Li's [28]	64.04	50.03	5.79	4.53
Ren's [44]	64.00	50.00	5.62	4.39
Teh's [51]	63.85	49.88	5.67	4.43
Wang's [60]	63.99	49.99	5.61	4.39
Zhang's [73]	63.96	49.97	5.52	4.32
This scheme	64.01	50.01	5.60	4.37

Table 8 Comparison on collision resistance with $N = 2048$ random tests and 128-bit hash value

Algorithm	Number of hits				Absolute difference			
	0	1	2	3	Min.	Max.	Mean	Mean/Char.
Li's [33]	1928	116	4	0	695	2017	1359	84.94
Xiao's [64]	1924	120	4	0	658	2156	1431.1	89.46
Xiao's [65]	1915	132	1	0	812	2034	1349.1	84.32
Lin's with MT [36]	1931	132	3	0	N/A	N/A	N/A	83.79
Lin's [35]	1912	132	4	0	496	2908	N/A	88.76
Deng's [7]	1940	104	4	0	583	2206	1399.8	87.49
Kanso's [23]	1954	92	2	0	731	2230	1368	85.50
Li's [28]	1945	103	0	0	685	1972	1255	78.44
Li's [29]	1928	118	2	0	687	2220	1432.1	89.51
Xiao's [66]	1926	120	2	0	605	1952	1227.8	76.74
Xiao's [68]	1932	114	2	0	573	2224	1401.1	87.56
Luo's [39]	1928	117	3	0	796	2418	1598.6	99.91
This scheme	1930	117	2	0	688	2019	1350	84.38

4.6 Comparison with other hash algorithms

We perform a comparison between the proposed hash function and some significant chaos-based hash functions as well as MD5, which is based on statistical performance and collision resistance. Tables 6 and 7 describe the comparison of statistical performance between the proposed algorithm and selected existing algorithms. Note that the results reported in Table 6 are based on $N = 2048$ random tests and 128-bit hash value, while the results of Table 7 focus on $N = 10000$ random tests and 128-bit hash value. Based on the results, our algorithm shows better statistical performance.

In addition, Table 8 presents the comparison of the number of ASCII characters with the same value at the same location and absolute difference in 128-bit hash values based on $N = 2048$ random tests between our algorithm and selected existing algorithms. Based on the results, the proposed algorithm shows better collision resistance.

5 Conclusions

In this paper, we design, implement and evaluate a cryptographic and parallel chaotic hash function based on the two-dimensional cross coupled map lattices for multimedia communication security. This work includes three main contributions: 1) presents a cryptographic and parallel hash algorithm based on the cross coupled map lattices; 2) utilizes message blocks as the space domain input and parameter sequence from the piecewise linear chaotic map as the time domain input for the $CCML$ to generate intermediate hash values; 3) evaluates the performance of the proposed hash algorithm, and the cryptanalytic results demonstrate that the hash algorithm has good statistical properties, strong collision resistance, and better statistical performance compared with existing chaotic hash functions. Comparing with other related works, it is the first time to exploit the two-dimensional cross coupled map lattices with space domain and time domain inputs to design a cryptographic hash function

and the proposed hash function can be performed in parallel. We believe the proposed hash function is suitable for multimedia communication security.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China (Grant nos. 61672119, 61528206 and 61402380), the Natural Science Foundation of CQ CSTC (Grant nos. cstc2015jcyjA40044, and cstc2014jcyjA40030), the Fundamental Research Funds for the Central Universities (Grant no. XDJK2015B030), U.S. National Science Foundation (Grant nos. CNS-1253506 (CAREER) and CNS-1618300), and the Opening Project of State Key Laboratory for Novel Software Technology (Grant No. KFKT2016B13).

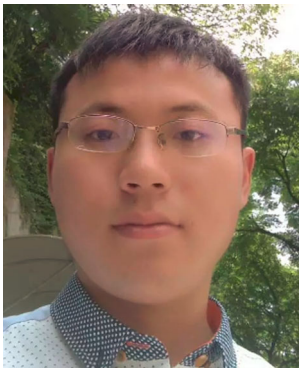
References

1. Ahmad M, Khurana S, Singh S, AlSharari HD (2017) A simple secure hash function scheme using multiple chaotic maps. 3D research 8(2), article 13
2. Akhavan A, Samsudin A, Akhshani A (2009) Hash function based on piecewise nonlinear chaotic map. Chaos, Soliton and Fractals 42:1046–1053
3. Akhavan A, Samsudin A, Akhshani A (2013) A novel parallel hash function based on 3D chaotic map. EURASIP Journal on Advances in Signal Processing 1:1–12
4. Amin M, Faragallah OS, El-Latif AAA (2009) Chaos based hash function (CBHF) for cryptographic applications. Chaos, Soliton and Fractals 42(2):767–772
5. Bakhtiari S, Safavi-Naini R, Pieprzyk J (1996) Keyed hash function. Proceedings of the Cryptography: Policy and Algorithms, Lecture Notes in Computer Science 1029:201–214
6. 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
7. 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
8. Deng S, Zhan Y, Xiao D, Li Y (2011) Analysis and improvement of a hash-based image encryption algorithm. Commun Nonlinear Sci Numer Simul 16(8):3269–3278
9. Elhoseny M, El-Minir HK, Riad AM, Yuan X (2016) A secure data routing schema for WSN using elliptic curve cryptography and homomorphic encryption. Journal of King Saud University - Computer and Information Sciences 28(3):262–275
10. Elhoseny M, Yuan X, El-Minir HK, Riad AM (2016) An energy efficient encryption method for secure dynamic WSN. Security and Communication Networks 9(13):2024–2031
11. Elhoseny M, Farouk A, Zhou N, Wang M-M, Abdalla S, Batle J (2017) Dynamic multi-hop clustering in a wireless sensor network: performance improvement. Wirel Pers Commun 95(4):3733–3753
12. Elhoseny M, Shehab A, Yuan X (2017) Optimizing robot path in dynamic environments using genetic algorithm and Bezier curve. J Intell Fuzzy Syst 33(4):2305–2316
13. Elhoseny M, Tharwat A, Farouk A, Hassanien AE (2017) K-coverage model based on genetic algorithm to extend WSN lifetime. IEEE Sensors Letters 1(4):1–4
14. Elhoseny M, Tharwat A, Hassanien AE (2018) Bezier curve based path planning in a dynamic field using modified genetic algorithm. Journal of Computational Science 25:339–350
15. Elhoseny M, Tharwat A, Yuan X, Hassanien A (2018) Optimizing K-coverage of mobile WSNs. Expert Syst Appl 92:142–153
16. Elsayed W, Elhoseny M, Sabbeh S, Riad A (2018) Self-maintenance model for wireless sensor networks. Comput Electr Eng 70:799–812
17. FarouK A, Batle J, Elhoseny M, Naseri M, Lone M, Fedorov A, Alkhambashi A, Ahmedand SH, Abdel-Aty M (2018) Robust general N user authentication scheme in a centralized quantum communication network via generalized GHZ states. Front Phys 13(2):130306
18. Guo XF, Zhang JS (2006) Keyed one-way hash function construction based on the chaotic dynamic S-Box. Acta Phys Sin 55:4442–4449
19. Guo W, Wang X, He D, Cao Y (2009) Cryptanalysis on a parallel keyed hash function based on chaotic maps. Phys Lett A 373(36):3201–3206

20. Hong D, Kim D-C, Kwon D, Kim J (2016) Improved preimage attacks on hash modes of 8-round AES-256. Multimed Tools Appl 75(22):14525–14539
21. Jiteurtragool N, Ketthong P, Wannaboon C, San-Um W (2013) A topologically simple keyed hash function based on circular chaotic sinusoidal map network. In: International conference on advanced communication technology, pp 1089–1094
22. Kanso A, Ghebleh M (2013) A fast and efficient chaos-based keyed hash function. Commun Nonlinear Sci Numer Simul 18:109–123
23. Kanso A, Ghebleh M (2015) A structure-based chaotic hashing scheme. Nonlinear Dyn 81(1):27–40
24. 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
25. Kim H, Kim D-W, Yi O, Kim J (2018) Cryptanalysis of hash functions based on blockciphers suitable for IoT service platform security. Mutimedia Tools and Applications. https://doi.org/10.1007/s11042-018-5630-4
26. Kwok HS, Tang WKS (2005) A chaos-based cryptographic hash function for message authentication. Int J Bifurcation Chaos 15(12):4043–4050
27. Li Y (2016) Collision analysis and improvement of a hash function based on chaotic tent map. Optik 127(10):4484–4489
28. Li Y, Li X (2016) Chaotic hash function based on circular shifts with variable parameters. Chaos, Soliton and Fractals 91:639–648
29. Li Y, Deng S, Xiao D (2011) A novel Hash algorithm construction based on chaotic neural network. Neural Comput & Applic 20(1):133–141
30. Li Y, Xiao D, Deng S, Han Q, Zhou G (2011) Parallel hash function construction based on chaotic maps with changeable parameters. Neural Comput & Applic 20(8):1305–1312
31. Li Y, Xiao D, Deng S (2012) Keyed hash function based on a dynamic lookup table of functions. Inform Sci 214:56–75
32. Li Y, Xiao D, Deng S (2012) Secure hash function based on chaotic tent map with changeable parameter. High Technol Lett 18(1):7–12
33. Li Y, Ge G, Xia D (2016) Chaotic hash function based on the dynamic S-Box with variable parameters. Nonlinear Dyn 84(4):2387–2402
34. Liang J, Lai X (2005) Improved collision attack on hash function MD5, Technical report
35. Lin Z, Yu S, Lu J (2017) A novel approach for constructing one-way hash function based on a message block controlled 8D hyperchaotic map. Int J Bifurcation Chaos 27(7):1750106
36. Lin Z, Guyeux C, Yu S, Wang Q, Cai S (2017) On the use of chaotic iterations to design keyed hash function. Clust Comput. https://doi.org/10.1007/s10586-017-1062-6
37. Liu J, Wang X, Yang K, Zhao C (2012) A fast new cryptographic hash function based on integer tent mapping system. J Comput 7(7):1671–1680
38. Liu H, Kadir A, Sun X, Li Y (2018) Chaos based adaptive double-image encryption scheme using hash function and S-boxes. Multimed Tools Appl 77:1391–1407
39. Luo Y, Du M (2012) One-way hash function construction based on the spatiotemporal chaotic system. Chinese Physics B 21(6):060503
40. Mendel F, Nad T, Schlaffer M (2013) Improving local collisions: new attacks on reduced SHA-256. Advances in Cryptology-EUROCRYPT, lecture notes in computer science 7881:262–278
41. Mihcak K, Venkatesan R, Liu T (2005) Watermarking via optimization algorithms for quantizing randomized semi-global image statistics. Multimedia Systems 11(2):185—200
42. NIST (2001) Secure hash standard. http://csrc.nist.gov/CryptoToolkit/tkhash.html
43. Nouri M, Khezeli A, Ramezani A, Ebrahimi A (2012) A dynamic chaotic hash function based upon circle chord methods. In: 6th international symposium on telecommunications, pp 1044–1049
44. Ren H, Wang Y, Xie Q, Yang H (2009) A novel method for one-way hash function construction based on spatiotemporal chaos. Chaos, Soliton and Fractals 42(4):2014–022
45. Rivest R (1992) The MD5 message-digest algorithm. IETF network working group
46. Rompel J (1990) One-way functions are necessary and sufficient for secure signatures. Proceedings of the 22th annual ACM symposium on theory of computing: 387–394
47. Schneider M, Chang SF (1996) A robust content based digital signature for image authentication. In: Proceedings IEEE conf image processing, vol 3, pp 227–230
48. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28(4):656–715
49. Stevens M (2013) New collision attacks on SHA-1 based on optimal joint local-collision analysis. Advances in Cryptology-EUROCRYPT 2013, Lecture Notes in Computer Science 7881:245–261
50. Tang KW, Tang WK, Man KF (2007) A chaos-based pseudo-random number generator and its application in voice communications. Int J Bifurcation Chaos 17(3):923–933

51. Teh JS, Samsudin A, Akhavan A (2015) Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dyn 81(3):1067–1079
52. Tharwat A, Elhoseny M, Hassanien A, Gabel T, Kumar A (2018) Intelligent Beziér curve-based path planning model using chaotic particle swarm optimization algorithm. Clust Comput. https://doi.org/10.1007/s10586-018-2360-3
53. Todorova M, Stoyanov B, Szczypiorski K, Kordov K (2018) SHAH: hash function based on irregularly decimated chaotic map. arXiv:1808.01956
54. Tsudik G (1992) Message authentication with one-way hash functions. ACM SIGCOMM Computer Communication Review 22:29–38
55. Wang S, Shan P (2011) Security analysis of a one-way hash function based on spatiotemporal chaos. Chin Phys B 20(9):090504–090507
56. Wang X, Feng D, Lai X, Yu H (2004) Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint archive, report 2004/199
57. Wang X, Yin Y, Yu H (2005) Finding collisions in the full SHA-1. Advances in Cryptology-CRYPTO 2005, Lecture Notes in Computer Science 3621:17–6
58. Wang Y, Yang D, Du M, Yang H (2007) One-way hash function construction based on iterating a chaotic map. In: 2007 international conference on computational intelligence and security workshops, pp 791–794
59. Wang Y, Liao X, Xiao D, Wong K (2008) One-way hash function construction based on 2D coupled map lattices. Inform Sci 178(5):1391–1406
60. Wang Y, Wong KW, Xiao D (2011) Parallel hash function construction based on coupled map lattices. Commun Nonlinear Sci Numer Simul 16:2810–2821
61. Wang S, Li D, Zhou H (2012) Collision analysis of a chaos-based hash function with both modification detection and localization capability. Commun Nonlinear Sci Numer Simul 17(2):780–784
62. Wang Q, Yu S, Li C, Lu J, Fang X, Guyeux C, Bahi JM (2016) Theoretical design and FPGA-based implementation of higher-dimensional digital chaotic systems. IEEE Trans Circuits Syst Regul Pap 63(3):401–412
63. Wong KW (2003) A combined chaotic cryptographic and hashing scheme. Phys Lett A 307:292–298
64. Xiao D, Liao X, Deng S (2008) Parallel keyed hash function construction based on chaotic maps. Phys Lett A 372:4682–4688
65. Xiao D, Liao X, Wang Y (2009) Improving the security of a parallel keyed hash function based on chaotic maps. Phys Lett A 373:4346–4353
66. Xiao D, Liao X, Wang Y (2009) Parallel keyed hash function construction based on chaotic neural network. Neurocomputing 72:2288–2296
67. Xiao D, Peng W, Liao X, Xiang T (2010) Collision analysis of one kind of chaos-based hash function. Phys Lett A 374(10):1228–1231
68. Xiao D, Shih FY, Liao XF (2010) A chaos-based hash function with both modification detection and localization capabilities. Commun Nonlinear Sci Numer Simul 15:2254–2261
69. Xie EY, Li C, Yu S, Lu J (2017) On the cryptanalysis of Fridrich's chaotic image encryption scheme. Signal Process 132:150–154
70. Yi X (2005) Hash function based on chaotic tent maps. IEEE Trans Circuits Syst Express Briefs 52:354–357
71. Yuan X, Elhoseny M, El-Minir HK, Riad AM (2017) A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. J Netw Syst Manag 25(1):21–46
72. Zhang H, Wang X, Li Z, Liu D (2005) One way hash function construction based on spatiotemporal chaos. Acta Phys Sin 54:4006–4011
73. Zhang J, Wang X, Zhang W (2010) Chaotic keyed hash function based on feedforward-eedback nonlinear digital filter. Phys Lett A 362:439–448
74. Zhang P, Zhang X, Yu J (2017) A parallel hash function with variable initial values. Wirel Pers Commun 96(2):2289–2303

Yantao Li is a Professor in the College of Computer Science at the Chongqing University, China and has been a Postdoctoral Research Associate in the Department of Computer Science at the College of William and Mary, USA, since July of 2016. He received the PhD degree from the College of Computer Science at Chongqing University, in December 2012. From September 2010 to September 2012, he was a visiting scholar supported by China Scholarship Council working with Prof. Gang Zhou at the Department of Computer Science in the College of William and Mary, USA. He was an Associate Professor in the College of Computer and Information Sciences at the Southwest University, China, from March 2013 to December 2018. His research area includes wireless communication and networking, sensor networks and ubiquitous computing, and information security.



Guangfu Ge is working toward the Master's degree in the College of Computer and Information Sciences at Southwest University, China. He received his B.S. degree from Southwest University in 2015. His research area includes mobile computing, wireless networks and information security.