Neural Comput & Applic (2017) 28:1405–1415
DOI 10.1007/s00521-015-2158-7

ORIGINAL ARTICLE

A fast and efficient hash function based on generalized chaotic mapping with variable parameters

Yantao Li[1] · **Xiang Li**[1] · **Xiangwei Liu**[2]

Received: 23 July 2015 / Accepted: 21 December 2015 / Published online: 5 January 2016

Abstract We present a fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters in this paper. We first define a generalized chaotic mapping by utilizing piecewise linear chaotic map and trigonometric functions. Then, we convert the arbitrary length of message into the corresponding ASCII values and perform 6-unit iterations with variable parameters and message values based on the generalized chaotic mapping. The final hash value is obtained by cascading extracted bits from iteration state values. We excessively evaluate the proposed algorithm in terms of distribution of hash value, sensitivity of hash value to the message and secret keys, statistical analysis of diffusion and confusion, analysis of birthday attacks and collision resistance, analysis of secret keys, analysis of speed, and comparison with other algorithms, and the results illustrate that the suggested algorithm is fast, efficient, and enough simple and has good confusion and diffusion capabilities, strong collision resistance, and a high level of security.

Keywords Chaos · Hash function · Generalized chaotic mapping · Piecewise linear chaotic map · Variable parameter

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

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

[2] College of Mathematics and Statistics, Chongqing University of Technology, Chongqing 400054, China

1 Introduction

With the development of electronic commerce, one-way hash function has been widely developed in public-key cryptography, digital signatures [1], integrity verification [2], message authentication and dynamic password authentication [3], etc. A hash function is a one-way function that can be used to map digital data of arbitrary size to digital data with fixed size. The digital data returned by a hash function are referred to as hash value or message digest. Hash functions have the properties of sensitivity to initial conditions, diffusion and confusion, collision resistance. There are some traditional one-way hash algorithms, such as MD2, MD4, MD5, and SHA [4, 5], but most of them are based on the hypothesis of complexity that requires lots of complicated logic computing (such as XOR) or packet encryption method-based multiple iterations. Nevertheless, there are inherent defects in logic operation although the computation is simple for the further method, and the computation of the latter method is large and it is hard to find fast and reliable encryption methods simultaneously. Chaos has some inherent merits of one way, sensitivity to tiny modifications in initial conditions and parameters, mixing property and ergodicity, which can be used for designing chaotic hash functions. In the last decades, many researchers propose different hash algorithms based on chaotic maps [6–19], encouraged by the specific properties of chaotic dynamical systems such as high sensitivity to the initial conditions, ergodicity, high complexity induced by their simple analytic expressions. However, these chaotic maps in cryptanalytic studies reveal security weakness [20–27].

In this paper, we present a fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters. A piecewise linear chaotic map with variable parameter p is chosen, and a generalized chaotic mapping is defined by utilizing the piecewise linear chaotic map and trigonometric functions [such as $\sin(x)$ and $\cos(x)$]. Then, we convert the arbitrary length of message into the corresponding ASCII values and perform 6-unit iterations with variable parameters and message values based on the generalized chaotic mapping, where the variable parameter p is updated by the generalized chaotic mapping. The final hash value is obtained by cascading extracted bits from iteration state values. We excessively evaluate the proposed algorithm in terms of distribution of hash value, sensitivity of hash value to the message and secret keys, statistical analysis of diffusion and confusion, analysis of birthday attacks and collision resistance, analysis of secret keys, analysis of speed, and comparison with other algorithms, and the results illustrate that the suggested algorithm is fast, efficient, and enough simple and has good confusion and diffusion capabilities, strong collision resistance, and a high level of security.

The main contributions of this work can be summarized as follows:

- We present a fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters.
- We define a generalized chaotic mapping by utilizing piecewise linear chaotic map and trigonometric functions.
- We excessively evaluate the proposed algorithm, and the results illustrate that it has good capabilities of confusion and diffusion, strong collision resistance.

The remainder of this paper is organized as follows: Sect. 2 introduces the preliminaries of piecewise linear chaotic map and defines a generalized chaotic mapping used in the hash algorithm. In Sect. 3, we design the chaotic hash algorithm based on a generalized chaotic mapping with variable parameters in detail. We excessively evaluate the performance of the proposed hash algorithm in Sect. 4 and present conclusions in Sect. 5.

2 Preliminaries

To learn about the proposed hash algorithm, we should have some related preliminaries. In this section, we first introduce piecewise linear chaotic map, which is utilized for constructing a generalized chaotic mapping with trigonometric functions, and then define the generalized chaotic mapping in detail, which is used to iterate the message to generate hash values in the algorithm.

2.1 Piecewise linear chaotic map (PWLCM)

The chaotic tent map chosen in the hash algorithm is one-dimensional and piecewise linear chaotic map expressed as follows:

$$X(t+1) = F_p(X(t)) = \begin{cases} X(t)/p & 0 \le X(t) < p, \\ (X(t)-p)/(0.5-p), & p \le X(t) < 0.5, \\ (1-p-X(t))/(0.5-p), & 0.5 \le X(t) < 1-p, \\ (1-X(t))/p, & 1-p \le X(t) \le 1, \end{cases}$$

where $X(t)$ represents the iteration trajectory value and p denotes the control parameter. When p is assigned values in (0, 0.5), $X(t)$ evolves into a chaotic state in range of (0, 1). That is, $X(t)$ sequence values vary in (0, 1) and distribute independently. 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 p, 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

Table 1 Fixed iterative values of PWLCM with the specified initial value $X(0)$ and parameter p

$X(0)$	p	$X(t)$ (t = 1, 2,…)
<0.25	2 × $X(0)$	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,…

map as a key generator to produce the four initial buffers for our hash algorithm. Moreover, based on Ref. [28], we believe$\{X(t)\}$is ergodic and uniformly distributed in the interval (0, 1) and the autocorrelation function is δ-like [52].

2.2 Generalized chaotic mapping (GCM)

To realize multiple chaotic mappings under a unified structure, a key factor is that it should ensure each input of these chaotic maps does not exceed its value range [29]. Therefore, chaotic maps with same value range will be given preferential consideration to construct the generalized chaotic mapping. Then, we define a generalized chaotic mapping model as follows:

$$\begin{aligned} f(t+1) = G(c(t+1), X(t), f(t)) = a_1 \times \frac{c(t+1)}{256} + a_2 \\ \times |\sin(c(t+1))| + a_3 \times |\sin(c(t+1))|^2 \\ + a_4 \times |\sin(c(t+1))|^3 + a_5 \times |\cos(c(t+1))| \\ + a_6 \times (1 - X(t)^2) + a_7 \times (1 - f(t)^3) \end{aligned}$$

where a_i (i = 1, 2, 3,…, 7) only have two values: 0 or 1, and each variable element has the same value range of (0, 1). $c(t + 1)$ is the $(t + 1)$th element of an array composed of the ASCII code values of message characters. $X(t)$ is the tth iteration value of PWLCM. When a_i (i = 1, 2, 3,…, 7) take different values, GCM illustrates different chaotic maps, and this kind of structure can easily control the range of the results. As we can see, the GCM is composed of outputs of PWLCM and trigonometric functions.

3 Description of the proposed algorithm

In this section, we design the fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters. The input is an arbitrary length of message M, and the output is l-bit hash value, where $l = 128$. The reason we choose the 128-bit length of hash value is that it is sufficient enough to ensure the security. We describe the hash algorithm in three steps of message preprocessing, hash algorithm description, and hash value generation, in the following:

Step 1 (Message preprocessing) We first convert the original message M into the corresponding ASCII code values and then store these values into an array c, where we denote the length of the array by n.

Step 2 (Hash algorithm description) We assign values to secret keys ($f(0)$, $X(0)$) as $f(0) = 0$, $X(0) = 0.2323$, where $f(0)$ is the input of GCM and the output is used to dynamically change the control parameter p of PWLCM, and $X(0)$ is the initial value of PWLCM. Then, we perform 6-unit iterations, 1st-nth, $(n + 1)$th–$2n$th, $(2n + 1)$th–$3n$th, $(3n + 1)$th–$4n$th, $(4n + 1)$th–$5n$th, $(5n + 1)$th–$6n$th, on message array c. For each iteration, we first assign values to parameters a_1, a_2, a_3,…, a_7 in GCM: For $i = 1$ to n, we calculate $s(i) = c(i) \mod 8$ and then assign values as $a_1,\ldots,a_{s(i)} = 1$ and $a_{s(i+1)},\ldots,a_7 = 0$. There are two exceptions: One is that when $s(i) = 0$, then we set $s(i) = 7$; the other is that, for some special conditions, for example, when all messages are all the same (such as all blank-space message), let coefficients of variables $\frac{c(i+1)}{256}$ and $(1 - X(i))^2$ be 1 and others be 0. Then, we iterate GCM n times to generate $f(n)$, then recalculate $f(n) = |f(n) - \lfloor f(n) \rfloor - 0.5|$ such that $f(n)$ is in the range of (0,0.5), and compute $X(n) = F_{f(n)}(X(n - 1))$, which are then used as the inputs for the next iteration. The detailed 6-unit iteration processes are described in Algorithm 1:

```
for $i = 1$ to $n$
    $s(i) = c(i) \bmod 8$;
    $a_1, \ldots, a_{s(i)} = 1$;
    $a_{s(i+1)}, \ldots, a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_1 \times \frac{c(i)}{256} + a_2 \times |\sin(c(i))| + a_3 \times |\sin(c(i))|^2 + a_4 \times |\sin(c(i))|^3 + a_5 \times |\cos(c(i))| + a_6 \times (1 - X(i-1)^2) + a_7 \times (1 - f(i-1)^3)$;
end for
$f_1(n) = |f(n) - \lfloor f(n) \rfloor - 0.5|$;
$X(n) = F_{f_1(n)}(X(n-1))$;
for $i = n+1$ to $2n$
    $s(i-n) = c(i-n) \bmod 8$;
    $a_1, \ldots, a_{s(i-n)} = 1$;
    $a_{s(i-n+1)}, \ldots, a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_2 \times \frac{c(n)}{256} + a_3 \times |\sin(c(n))| + a_4 \times |\sin(c(n))|^2 + a_5 \times |\sin(c(n))|^3 + a_6 \times |\cos(c(n))| + a_1 \times (1 - X(n-1)^2) + a_7 \times (1 - f(n-1)^3)$;
end for
$f_2(2n) = |f(2n) - \lfloor f(2n) \rfloor - 0.5|$;
$X(2n) = F_{f_2(2n)}(X(2n-1))$;
for $i = 2n+1$ to $3n$
    $s(i-2n) = c(i-2n) \bmod 8$;
    $a_1, \ldots, a_{s(i-2n)} = 1$;
    $a_{s(i-2n+1)}, \ldots, a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_3 \times \frac{c(n)}{256} + a_4 \times |\sin(c(n))| + a_5 \times |\sin(c(n))|^2 + a_6 \times |\sin(c(n))|^3 + a_1 \times |\cos(c(n))| + a_2 \times (1 - X(n-1)^2) + a_7 \times (1 - f(n-1)^3)$;
end for
$f_3(3n) = |f(3n) - \lfloor f(3n) \rfloor - 0.5|$;
$X(3n) = F_{f_3(3n)}(X(3n-1))$;
for $i = 3n+1$ to $4n$
    $s(i-3n) = c(i-3n) \bmod 8$;
    $a_1, \ldots, a_{s(i-3n)} = 1$;
    $a_{s(i-3n+1)}, \ldots, a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_4 \times \frac{c(n)}{256} + a_5 \times |\sin(c(n))| + a_6 \times |\sin(c(n))|^2 + a_1 \times |\sin(c(n))|^3 + a_2 \times |\cos(c(n))| + a_3 \times (1 - X(n-1)^2) + a_7 \times (1 - f(n-1)^3)$;
end for
$f_4(4n) = |f(4n) - \lfloor f(4n) \rfloor - 0.5|$;
$X(4n) = F_{f_4(4n)}(X(4n-1))$;
for $i = 4n+1$ to $5n$
    $s(i-4n) = c(i-4n) \bmod 8$;
    $a_1, \ldots, a_{s(i-4n)} = 1$;
    $a_{s(i-4n+1)}, \ldots, a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_5 \times \frac{c(n)}{256} + a_6 \times |\sin(c(n))| + a_1 \times |\sin(c(n))|^2 + a_2 \times |\sin(c(n))|^3 + a_3 \times |\cos(c(n))| + a_4 \times (1 - X(n-1)^2) + a_7 \times (1 - f(n-1)^3)$;
end for
```

```
$f_5(5n) = \left| f(5n) - \lfloor f(5n) \rfloor - 0.5 \right|$;
$X(5n) = F_{f_5(5n)}(X(5n-1))$;
for $i = 5n+1$ to $6n$
    $s(i\text{-}5n) = c(i\text{-}5n) \bmod 8$;
    $a_1,\ldots,a_{s(i\text{-}5n)} = 1$;
    $a_{s(i\text{-}5n+1)},\ldots,a_7 = 0$;
    $f(i) = G(c(i), X(i-1), f(i-1)) = a_6 \times \frac{c(n)}{256} + a_1 \times |\sin(c(n))| + a_2 \times |\sin(c(n))|^2$
        $+ a_3 \times |\sin(c(n))|^3 + a_4 \times |\cos(c(n))| + a_5 \times (1 - X(n-1)^2) + a_7 \times (1 - f(n-1)^3)$;
end for
$f_6(6n) = \left| f(6n) - \lfloor f(6n) \rfloor - 0.5 \right|$;
$X(6n) = F_{f_6(6n)}(X(6n-1))$;
```

Step 3 (Hash value generation) After 6-unit iterations, we obtain *X(n)*, *X*(2*n*), *X*(3*n*), *X*(4*n*), *X*(5*n*), *X*(6*n*), then convert them to their corresponding binary formats, extract 20, 20, 20, 20, 24, 24 bits after their decimal point, respectively, and finally combine them to generate a 128-bit final hash value.

4 Performance analysis

We evaluate the proposed hash function based on a generalized chaotic mapping with variable parameters in terms of distribution of hash value, sensitivity of hash value to the message and secret keys, statistical analysis of diffusion and confusion, analysis of birthday attacks and collision resistance, analysis of secret keys, analysis of speed, and comparison with other algorithms. The arbitrary length of message for evaluating the performance of the proposed hash algorithm is randomly chosen as:

> 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 Distribution of hash value

One of the most important properties of a hash function is the uniform distribution of hash value, which is directly related to the security of the hash function. We conduct the hash simulation experiment on the randomly chosen message and then use two two-dimensional graphs to display the distribution of the message and the final hash value. First of all, we plot the randomly chosen message in Fig. 1a, and we can see that the decimal ASCII code values of the message distribute in a small range of [32, 127], while in Fig. 1b, the distribution of the corresponding hash value in hexadecimal format is very irregular. To make a comparison, we choose an extreme message, "all 0"-message, and we can see that the distribution of the message in Fig. 2a and the distribution of the corresponding hash value in hexadecimal format in Fig. 2b also spread

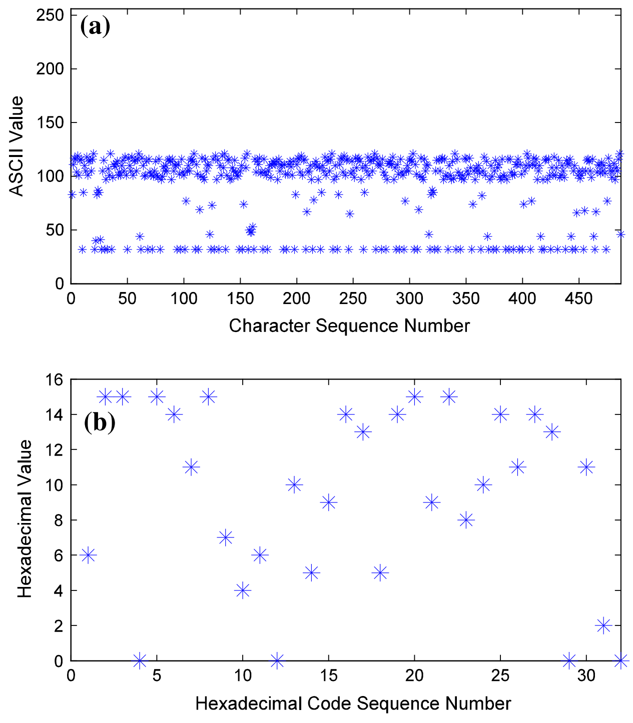


Fig. 1 Spread of message and hash value: **a** distribution of the message in ASCII code and **b** distribution of the hash value in hexadecimal format (6FF0FEBF7460A59ED5EF9F8AEBED0B20)

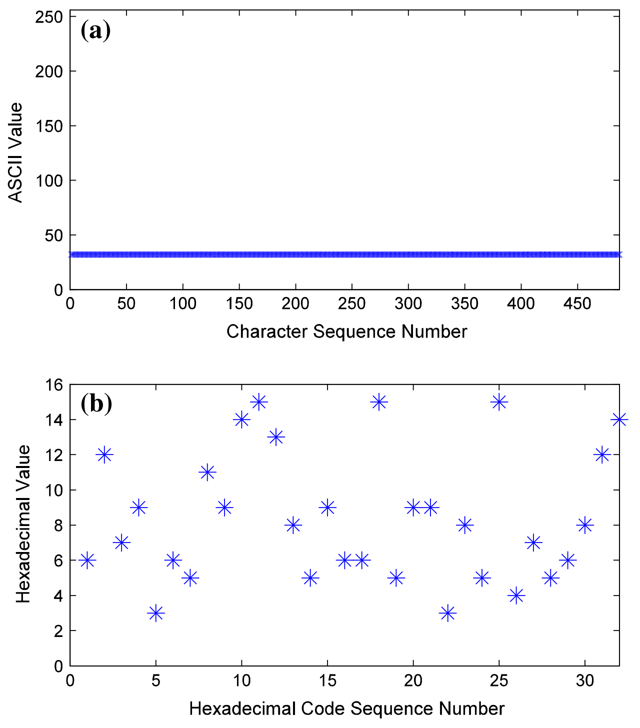


Fig. 2 Spread of "all 0"-message and hash value: **a** distribution of all "all 0"-message and **b** distribution of the hash value in hexadecimal format (6C79365B9EFD85966F599385F47568CE)

irregularly. The simulation results indicate that no information (including the statistic information) of the message can be left after the diffusion and confusion.

4.2 Sensitivity of hash value to the message and secret keys

According to the characteristics of a hash function, it is hard to find the original message if the hash value is known, which indicates that a hash function must be sensitive to tiny modifications in original message or secret keys. In particular, any slight modifications on message or secret keys will lead to a 50 % difference in hash value, according to a Hamming distance of approximately $l/2$ (l denotes the length of hash value) between the two hash values. In order to show the sensitivity of hash value to the message and secret keys, we conduct hash algorithm simulation experiment under the following seven conditions:

Condition 1: The original message chosen as the one in Sect. 4.1;
Condition 2: Change the first character "*S*" to "*s*";
Condition 3: Change the word "*key*" to "*non-key*";
Condition 4: Swap the word "*Normal*" with "*Agricultural*";
Condition 5: Change the full stop "." at the end of the message into a comma ",";
Condition 6: Add a blank space to the end of the message;
Condition 7: Change the secret key $X(0) = 0.2323$ to $X(0) = 0.2323000000000001$.

The corresponding hash values in hexadecimal format are obtained, followed by the corresponding number of different bits compared with the hash value of Condition 1:

Condition 1: 6FF0FEBF7460A59ED5EF9F8AEBED0B20.
Condition 2: 54E86ABB3E709568B05A090BBA548938 (50).
Condition 3: 554B06A95906DE6E192456A46CD8FC36 (71).
Condition 4: 9AB0E492C1C6A3CF89BE84116F092CB1 (57).
Condition 5: AEF66656E072DFD495B5A579522BB5C7 (62).
Condition 6: FAA7F8AE9E6FAAD150DAC41810D9D88F (67).
Condition 7: 85D8AD49506CD53A3A6AC15A15A49010 (62).

The graphical display of binary sequences is plotted in Fig. 3.

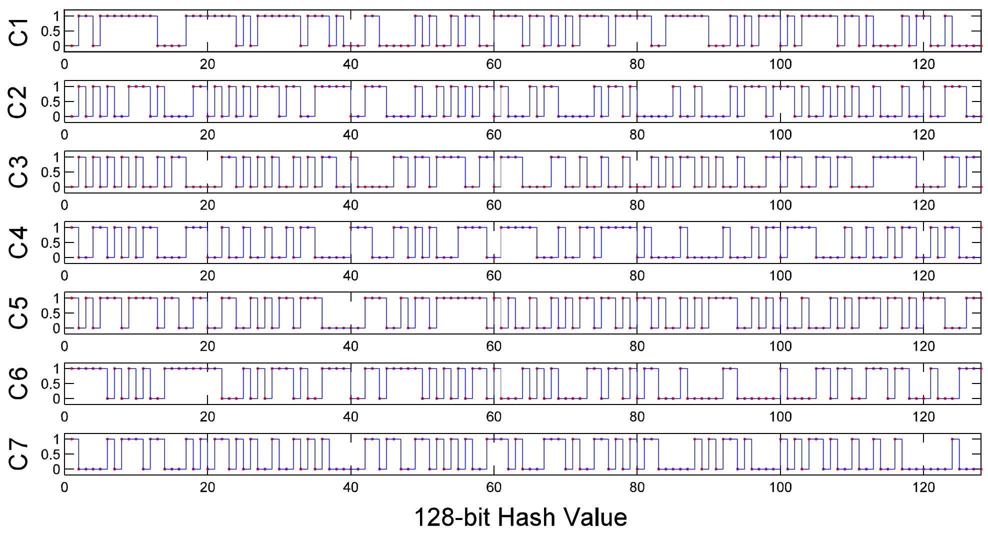
As shown in the seven different hash values in hexadecimal format and Fig. 3, we conclude that the simulation results indicate the sensitivity property of the proposed hash algorithm is so perfect that any tiny difference in the message or secret keys will cause huge changes in the final hash value. Note that the reason we increase the value of $X(0)$ with 10^{-17} in Condition 7 is the fact that 10^{-17} is the least value for the sensitive test of the hash algorithm, which has been verified in Sect. 4.5.

4.3 Statistical analysis of diffusion and confusion

Diffusion and confusion are two essential elements to hide message redundancy in the encryption algorithms that are introduced by Shannon [30]. The diffusion refers to as spreading out of the influence of a single plaintext bit over many cipher text bits so as to hide the statistical structure of the plaintext, while the confusion refers to as the use of transformations that complicate dependence of the statistics of cipher text on the statistics of plaintext.

We conduct the following experiment to capture qualitative characteristics of the diffusion and confusion: We first conduct the hash algorithm simulation on a randomly chosen message. Then, we randomly modify a bit in the

Fig. 3 Hash values under seven different conditions



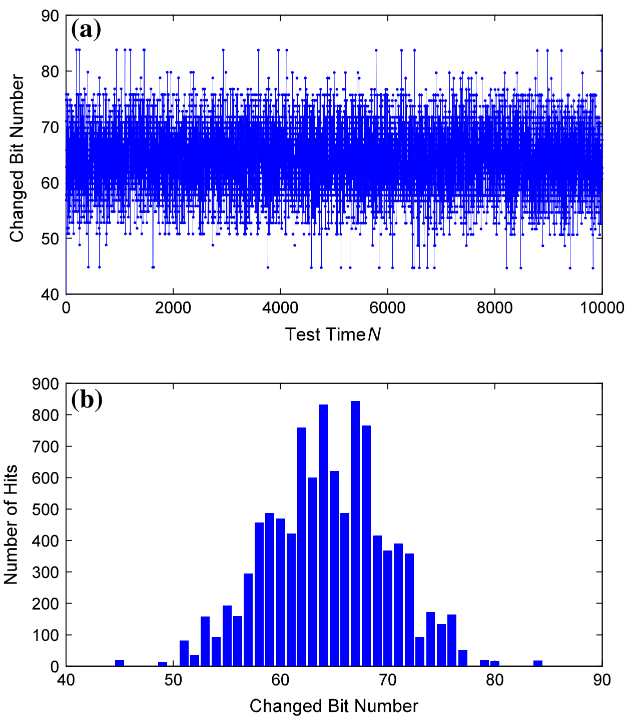


Fig. 4 Distribution of changed bit number: **a** plot of B_i and **b** histogram of B_i

message and then conduct the simulation. Finally, we count the difference between the two hash values in binary format. We denote the number of changed bits as B_i. We perform $N = 10{,}000$ times of these tests, and the corresponding distribution of changed bit number is illustrated in Fig. 4. As depicted in Fig. 4, the maximum changed bit number is 84 and the minimum is 45, which show a good diffusion effect of the proposed hash algorithm.

In addition, we perform the same experiment to capture the statistic characteristics of the diffusion and confusion on the proposed hash algorithm. First, we define six statistical metrics of minimum changed bit number $B_{\min}$, maximum changed bit number $B_{\max}$, mean changed bit number $\bar{B}$, mean changed probability P, standard deviation of the changed bit number ΔB, and standard deviation ΔP. They are computed as:

Minimum changed bit number: $B_{\min} = \min\{B_1, B_2,\ldots, B_N\}$,
Maximum changed bit number: $B_{\max} = \max\{B_1, B_2,\ldots, B_N\}$,
Mean changed bit number: $\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$,
Mean changed probability: $P = (\overline{B}/l) \times 100\,\%$,
Standard variance of the changed bit number: $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (B_i - \overline{B})^2}$,
Standard variance: $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} (B_i/l - P)^2} \times 100\,\%$, where N indicates the total number of tests, B_i denotes the changed bit number in the ith test, and l is the length of hash value. Based on the six statistics, we conduct the experiment on the hash algorithm N times, where $N = 256$, 512, 1024, 2048, and 10,000. The corresponding results of $B_{\min}$, $B_{\max}$, $\overline{B}$, P, ΔB, and ΔP are presented in Table 2. As depicted in Table 2, the mean changed bit number $\overline{B}$ is very close to the ideal value 64 bits (half of length of hash values), which is an empirical proof that the hash algorithm shows strong capability of confusion and diffusion [31], and the mean changed probability P is very close to the ideal value 50 % as well. Furthermore, the value of standard deviation of the changed bit number ΔB and standard deviation ΔP is very small, and with the increase of N, the two values are smaller and smaller, so the capability of diffusion and confusion is very stable.

Table 2 Statistics of number of changed bits

N	$N = 256$	$N = 512$	$N = 1024$	$N = 2048$	$N = 10{,}000$	Mean
$B_{\min}$	45	51	45	45	45	46
$B_{\max}$	80	84	84	84	84	83
$\overline{B}$	63.93	64.14	64.13	64.27	64.45	64.18
P (%)	49.94	50.11	50.10	50.21	50.35	50.14
ΔB	5.66	5.39	5.50	5.59	5.63	5.55
ΔP (%)	4.43	4.21	4.30	4.36	4.40	4.34

4.4 Analysis of birthday attack and collision resistance

In fact, collision resistance is similar to birthday attacks in theory. They are essentially a probability problem that two random input data are found such that they are hashed to the same output. In the following, we conduct qualitative and quantitative analysis of collision resistance on the proposed algorithm.

We first perform a qualitative analysis on the algorithm. In the proposed algorithm, the generalized chaotic mapping with variable parameters is introduced to ensure that the parameter p in each iteration is dynamically updated by the last iteration value $f(i)$ and the corresponding message bit in different positions. This inherent structure expedites the avalanche effect, which will ensure that each bit of the final hash value will be related to all the bits of message and even a single bit change in message or key will be diffused and result in great changes in the final hash value. Therefore, the proposed hash algorithm can resist the collision attack.

Then, we conduct the following experiment to make a quantitative analysis on collision resistance [32]: First, the hash value for a paragraph of message randomly chosen is generated and stored in ASCII format. Then, a bit in the message is selected randomly and toggled. A new hash value is then generated and stored in ASCII format as well. Two hash values are compared, and the number of ASCII character with the same value at the same location in the hash value is counted. We conduct the experiment 10,000 times. A plot of the distribution of the number of hits is demonstrated in Fig. 5. As illustrated in Fig. 5, there are 42 tests hitting twice, 404 tests hitting once, while in 9554 tests, no hit occurs, which are listed in Table 3.

Moreover, we introduce the absolute difference of two hash values d, which is calculated by the formula: $d = \sum_{i=1}^{N} \left| t(e_i) - t(e'_i) \right|$, where e_i and e'_i are the ith ASCII character of the original hash value and the new hash value, respectively, and the function $t(*)$ converts the entries to their equivalent decimal values. This kind of collision test is performed 10,000 times as well, with the secret key $X(0) = 0.2323$, and $f(0) = 0$. The maximum, mean, minimum values of d and mean/character are listed in Table 4.

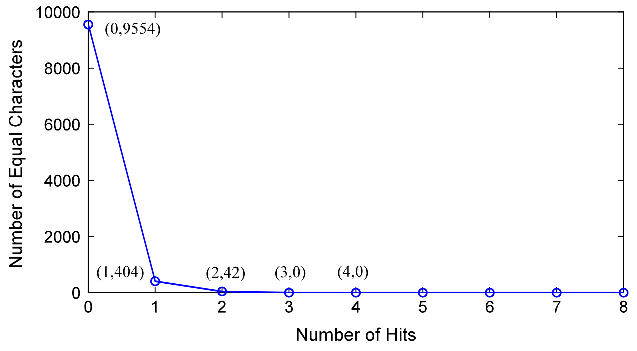


Fig. 5 Distribution of the number of ASCII characters with the same value at the same location in the hash value

Table 3 Number of hits (10,000-time tests)

Number of equal characters	0	1	2	3	4	5	6	7	8
The proposed algorithm	9554	404	42	0	0	0	0	0	0

Table 4 Absolute differences of two hash values (10,000-time tests)

Maximum	Minimum	Mean	Mean/character
2096	646	1429	89.29

Based on the calculation in Ref. [53], the theoretical mean/character value can be computed as $\frac{1}{3} \times 256 = 85.3333$. The mean/character value 89.29 is close the theoretical value.

Therefore, the qualitative and quantitative analysis on collision demonstrates that the proposed hash algorithm has good collision resistance.

4.5 Analysis of secret keys

In the proposed algorithm, the chaotic sensitivities to tiny modifications in initial values and parameters are fully utilized. The security of hash algorithm completely depends on message and secret keys. Therefore, the algorithm is immune from key recovery attack.

To investigate the key space, we conduct the following evaluation experiment: We first set the initial value $X(0)$ of

PWLCM to be larger than 10^{-16}, such as $X(0) = 0.2323200000000001$, and the corresponding changed bit number of the hash value is around 64 (85D8AD49506CD53A3A6AC15A15A49010 (62)). In contrast, if we set $X(0)$ to be 10^{-17} or less than 10^{-17}, the hexadecimal format of hash value is permanently shown as "7C3DB62518F8E15F3A350B766D5E2A47"; that is, no corresponding hash bit changes. Therefore, the sensitive precision of secret key $X(0)$ to hash value is 10^{-16}. Similarly, the sensitive precision of the parameter p is 10^{-16} as well. Considering both $X(0) \in (0, 1)$ and $p \in (0, 0.5)$, we can derive that the secret key space is approximately larger than 2^{106}. According to the ECRYPT II report on key lengths [33, 47–49], 106 bits provide sufficient security against brute force key-search attacks [50, 51].

4.6 Analysis of speed

First, the proposed hash algorithm does not need message padding as the preparation, while message padding is proportional to the length of original message; therefore, our algorithm without the message padding process improves the executive speed. Then, a generalized chaotic mapping and a one-dimensional piecewise linear chaotic map are utilized in our algorithm, the dynamical property of which is enough for the security of the algorithm, and the structures of which are simple that greatly reduce complexity of the algorithm, thereby providing *fast* computation speed and achieving the high *efficiency*.

Moreover, the proposed algorithm has the parallel property. Six iteration values $X(n)$, $X(2n)$, $X(3n)$, $X(4n)$, $X(5n)$, and $X(6n)$ contribute 20, 20, 20, 20, 24, and 24 bits to the final hash value, respectively, which corresponds to the parallel computation of 2.5, 2.5, 2.5, 2.5, 3, and 3 bytes.

4.7 Comparison with other 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 5 and 6 describe the comparison of statistical performance between the proposed algorithm and selected existing algorithms. Note that the results reported in Table 5 are based on $N = 2048$ random tests and 128-bit hash value, while the results of Table 6 focus on $N = 10{,}000$ random tests and 128-bit hash value. Based on the results, our algorithm shows better statistical performance.

In addition, Tables 7 and 8 present the comparison of the number of ASCII characters with the same value at the same location and absolute difference in 128-bit hash values between our algorithm and selected existing algorithms. Note that the reported results of Table 7 are based on $N = 2048$ random tests, while the results of Table 8 focus on $N = 10{,}000$ random tests. Based on the results, the proposed algorithm shows better collision resistance.

As a discussion, the proposed hash algorithm illustrates better statistical performance and collision resistance capability. We present a fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters. We dedicate to the improvement of the efficiency with high execution speed. However, most of the

Table 5 Comparison on statistical performance with $N = 2048$ random tests and 128-bit hash value

Algorithms	Statistical performance of the algorithms			
	$\overline{B}$	P (%)	ΔB	ΔP (%)
MD5 [34]	64.03	50.02	5.66	4.42
Li's [14]	63.57	49.66	7.43	5.80
Xiao's [15]	63.92	49.94	5.62	4.39
Xiao's [16]	64.09	50.07	5.48	4.28
Kanso's [17]	63.94	49.95	5.69	4.44
Deng's [20]	63.84	49.88	5.88	4.59
Guo's [24]	63.40	49.53	7.13	6.35
Kanso's [35]	64.01	50.01	5.61	4.38
Li's [36]	63.81	49.85	5.76	4.50
Li's [37]	63.89	49.91	5.64	4.41
Ren's [38]	63.92	49.94	5.78	4.52
Teh's [39]	64.01	50.01	5.66	4.26
Wang's [40]	63.98	49.98	5.53	4.33
Wang's [41]	64.15	50.11	5.77	4.51
Xiao's [42]	64.01	50.01	5.72	4.47
Xiao's [43]	64.18	50.15	5.67	4.41
Zhang's [44]	63.91	49.92	5.58	4.36
Zhang's [45]	64.43	49.96	5.57	4.51
This scheme	64.27	50.21	5.59	4.36

Table 6 Comparison on statistical performance with $N = 10{,}000$ random tests and 128-bit hash value

Algorithms	Statistical performance of the algorithms			
	$\overline{B}$	P (%)	ΔB	ΔP (%)
Kanso's [17]	63.94	49.95	5.64	4.41
Ren's [38]	64.00	50.00	5.62	4.39
Teh's [39]	63.85	49.88	5.67	4.43
Wang's [40]	63.90	49.91	5.58	4.36
Wang's [41]	63.99	49.99	5.61	4.39
Zhang's [44]	63.96	49.97	5.52	4.32
This scheme	64.45	50.35	5.63	4.40

Table 7 Comparison on collision resistance with $N = 2048$ random tests and 128-bit hash value

Algorithms	Number of hits				Absolute difference			
	0	1	2	3	Min.	Max.	Mean	Mean/char.
Xiao's [15]	1924	120	4	0	658	2156	1431.1	89.46
Xiao's [16]	1915	132	1	0	812	2034	1349.1	84.32
Deng's [20]	1940	104	4	0	583	2206	1399.8	87.49
Kanso's [35]	1954	92	2	0	731	2230	1368	85.50
Li's [36]	1928	118	2	0	687	2220	1432.1	89.51
Xiao's [42]	1926	120	2	0	605	1952	1227.8	76.74
Xiao's [43]	1932	114	2	0	573	2224	1401.1	87.56
Luo's [46]	1928	117	3	0	796	2418	1598.6	99.91
This scheme	1957	82	9	0	646	2096	1425	89.07

Table 8 Comparison on collision resistance with $N = 10{,}000$ random tests and 128-bit hash value

Algorithms	Number of hits				Absolute difference			
	0	1	2	3	Min.	Max.	Mean	Mean/char.
Akhavan's [13]	9434	553	13	0	744	2431	1371	85.69
Kanso's [35]	9431	557	12	0	656	2391	1364	85.25
Ren's [38]	9373	614	13	0	599	2455	1439	89.94
Teh's [39]	9969	31	0	0	575	2390	1379	86.81
Wang's [40]	9377	607	16	0	689	2295	1526	95.38
Wang's [41]	9359	626	15	0	655	2046	1367	85.44
Zhang's [44]	9416	572	12	0	565	2022	1257	78.56
This scheme	9554	404	42	0	646	2096	1429	89.29

start-of-the-art literature focuses on complexity of the algorithm, ignoring the speed and efficiency.

5 Conclusion

In this paper, we present a fast and efficient hash algorithm based on a generalized chaotic mapping with variable parameters. We first define a generalized chaotic mapping by utilizing piecewise linear chaotic map and trigonometric functions. Then, we convert the arbitrary length of message into the corresponding ASCII values and perform 6-unit iterations with variable parameters and message values based on the generalized chaotic mapping. The final hash value is obtained by cascading extracted bits from iteration state values. We excessively evaluate the proposed algorithm in terms of distribution of hash value, sensitivity of hash value to the message and secret keys, statistical analysis of diffusion and confusion, analysis of birthday attacks and collision resistance, analysis of secret keys, analysis of speed, and comparison with other algorithms, and the results illustrate that the suggested algorithm is fast, efficient, and enough simple and has good confusion and diffusion capabilities, strong collision resistance, and a high level of security.

Acknowledgments Our sincere thanks go to the anonymous reviewers for their valuable comments. This work is supported in part by the National Natural Science Foundation of China (Grant Nos. 61402380 and 61528206), the Natural Science Foundation of CQ CSTC (Grant No. cstc2015jcyjA40044), and the Fundamental Research Funds for the Central Universities (Grant No. XDJK2015B030).

References

1. Rompel J (1990) One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the 22th annual ACM symposium on theory of computing, pp 387–394
2. Sklavos N, Alexopoulos E, Koufopavlou O (2003) Networking data integrity: high speed architectures and hardware implementations. Int Arab J Inf Technol 1:54–59
3. Tsudik G (1992) Message authentication with one-way hash functions. ACM SIGCOMM Comput Commun Rev 22:29–38
4. Pieprzyk J, Sadeghiyan B (1993) Design of hashing algorithms, Lecture Notes in Computer Science. Springer, Berlin
5. Knudsen L, Preneel B (2002) Construction of secure and fast hash functions using nonbinary error-correcting codes. IEEE Trans Inf Theory 48:2524–2539
6. Amin M, Faragallah OS, El-Latif AAA (2009) Chaos based hash function (CBHF) for cryptographic applications. Chaos Solitons Fractals 42(2):767–772
7. 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

8. 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
9. Wang Y, Yang D, Du M, Yang H (2007) One-way hash function construction based on iterating a chaotic map. In: Proceedings-CIS Workshops 2007, 2007 international conference on computational intelligence and security workshops, pp 791–794
10. Maqableh M, Samsudin AB, Alia MA (2008) New hash function based on chaos theory (CHA-1). Int J Comput Sci Netw Secur 8(2):20–26
11. 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, ICACT, pp 1089–1094 (2013)
12. Zhang Q, Zhang H, Li Z (2009) One-way hash function construction based on conservative chaotic systems. In: 5th international conference on information assurance and security, IAS 2009, vol 2, pp 402–405
13. Akhavan A, Samsudin A, Akhshani A (2009) Hash function based on piecewise nonlinear chaotic map. Chaos Solitons Fractals 42:1046–1053
14. Li Y, Xiao D, Deng S, Han Q, Zhou G (2011) Parallel hash function construction based on chaotic maps with changeable parameters. Neural Comput Appl 20(8):1305–1312
15. Xiao D, Liao X, Deng S (2008) Parallel keyed hash function construction based on chaotic maps. Phys Lett A 372:4682–4688
16. 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
17. Kanso A, Ghebleh M (2013) A fast and efficient chaos-based keyed hash function. Commun Nonlinear Sci Numer Simul 18:109–123
18. Nouri M, Khezeli A, Ramezani A, Ebrahimi A (2012) A dynamic chaotic hash function based upon circle chord methods. In: 2012 6th international symposium on telecommunications, IST 2012, pp 1044–1049
19. Akhavan A, Samsudin A, Akshani A (2013) A novel parallel hash function based on 3D chaotic map. EURASIP J Adv Signal Process 2013(1):1–12
20. 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
21. Alvarez G, Montoya F, Romera M, Pastor G (2004) Cryptanalysis of dynamic look-up table based chaotic cryptosystems. Phys Lett A 326(3):211–218
22. Arumugam G, Lakshmi Praba V, Radhakrishnan S (2007) Study of chaos functions for their suitability in generating message authentication codes. Appl Soft Comput 7(3):1064–1071
23. Li C, Wang S (2007) A new one-time signature scheme based on improved chaos hash function. Comput Eng Appl 43(35):133–136
24. 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
25. 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
26. Wang S, Shan P (2011) Security analysis of a one-way hash function based on spatiotemporal chaos. Chin Phys B 20(9):090504–090507
27. 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
28. Bellare M, Ristenpart T, Multi-property-preserving hash domain extension: the EMD transform. In: Proceedings of 2nd NIST cryptographic hash workshop, Corwin Pavilion, UCSB Santa Barbara, CA
29. Zhang J, Xian X (2001) Nonlinear adaptive predictive targeting control of the continuous chaotic system. Acta Phys Sin 50(11):2092–2096
30. Zhang J, Wang X, Zhang W (2007) Chaotic keyed hash function based on feedforward–feedback nonlinear digital filter. Phys Lett A 362:439–448
31. Kanso A, Yahyaoui H, Almulla M (2012) Keyed hash function based on a chaotic map. Inf Sci 186:249–264
32. Wong KW (2003) A combined chaotic cryptographic and hashing scheme. Phys Lett A 307:292–298
33. Yearly Report on Algorithms and Keysizes, D.SPA.17 Rev. 1.0, ICT-2007-216676 ECRYPT II (2011)
34. Rivest R (1992) The MD5 message-digest algorithm. IETF Network Working Group, RFC 1321
35. Kanso A, Ghebleh M (2015) A structure-based chaotic hashing scheme. Nonlinear Dyn 81:27–40
36. Li Y, Deng S, Xiao D (2011) A novel Hash algorithm construction based on chaotic neural network. Neural Comput Appl 20:133–141
37. Li Y, Xiao D, Deng S (2012) Keyed hash function based on a dynamic lookup table of functions. Inf Sci 214:56–75
38. 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
39. Teh JS, Samsudin A, Akhavan A (2015) Parallel chaotic hash function based on the shuffle-exchange network. Nonlinear Dyn 81:1067–1079
40. Wang Y, Liao X, Xiao D, Wong K (2008) One-way hash function construction based on 2D coupled map lattices. Inf Sci 178(5):1391–1406
41. Wang Y, Wong KW, Xiao D (2011) Parallel hash function construction based on coupled map lattices. Commun Nonlinear Sci Number Simul 16:2810–2821
42. Xiao D, Liao X, Wang Y (2009) Parallel keyed hash function construction based on chaotic neural network. Neurocomputing 72:2288–2296
43. 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
44. Zhang J, Wang X, Zhang W (2007) Chaotic keyed hash function based on feedforward–feedback nonlinear digital filter. Phys Lett A 362:439–448
45. Zhang H, Wang X, Li Z, Liu D (2005) One way hash function construction based on spatiotemporal chaos. Acta Phys Sin 54:4006–4011
46. Luo Y, Du M (2012) One-way hash function construction based on the spatiotemporal chaotic system. Chin Phys B 21(6):060503
47. Li Y, Qi X, Ren Z, Zhou G, Xiao D, Deng S (2011) Energy modeling and optimization through joint packet size analysis of BSN and WiFi networks. IEEE IPCCC, Orlando
48. Qi X, Zhou G, Li Y, Peng G (2012) Radiosense: Exploiting wireless communication patterns for body sensor network activity recognition. IEEE RTSS, San Juan, Puerto Rico
49. Nguyen DT, Zhou G, Qi X, Peng G, Zhao J, Nguyen T, Le D (2013) Storage-aware smartphone energy savings. ACM Ubicomp, Zurich
50. Norouzi B, Seyedzadeh SM, Mirzakuchaki S, Mosavi MR (2014) A novel image encryption based on hash function with only two-round diffusion process. Multimedia Syst 20:45–64
51. Lo NW, Chiang MC, Hsu CY (2015) Hash-based anonymous secure routing protocol in mobile ad hoc networks. IEEE Asia JCIS
52. Kanguzhin BE, Nurakhmetov DB, Tokmagambetov NE (2014) Laplace operator with δ-like potentials. Russ Math 58:6–12
53. Li Y, Xiao D, Deng S (2011) Hash function construction based on the chaotic look-up table with changeable parameter. Int J Mod Phys B 25:3835–3851