Neural Comput & Applic (2011) 20:133–141
DOI 10.1007/s00521-010-0432-2

ORIGINAL ARTICLE

A novel Hash algorithm construction based on chaotic neural network

Yantao Li · Shaojiang Deng · Di Xiao

Received: 15 July 2009 / Accepted: 21 July 2010 / Published online: 6 August 2010

Abstract An algorithm for constructing a one-way novel Hash function based on two-layer chaotic neural network structure is proposed. The piecewise linear chaotic map (PWLCM) is utilized as transfer function, and the 4-dimensional and one-way coupled map lattices (4D OWCML) is employed as key generator of the chaotic neural network. Theoretical analysis and computer simulation indicate that the proposed algorithm presents several interesting features, such as high message and key sensitivity, good statistical properties, collision resistance and secure against meet-in-the-middle attacks, which can satisfy the performance requirements of Hash function.

Keywords Cryptography · Chaotic neural network · Hash function · Chaotic map

1 Introduction

Hash function is a fundamental building block of information security and plays an important role in integrity protection, message authentication [1] and digital signature [2]. It is a one-way and compression function that maps an arbitrary length message to a fixed length value. It should satisfy the properties of one way and be against birthday attack and against meet-in-the-middle attack. The confusion and diffusion, one-way and compression properties of neural networks are suitable for constructing Hash function [3]. Neurons with multi-inputs are summed with the weight inputs and transfer function to generate the single-output that performs the properties of neural networks [4]. It is impossible to obtain the inputs from the output without related parameters through the neural network. Chaos has features of initial value sensitivity, pseudo-random and unpredictable orbit, which is also adapted to build Hash function [5–13]. So, it is practical to combine chaotic systems with neural networks named chaotic neural network. In 2003, Rachel et al. [14] proposed public channel cryptography based on neural networks and chaotic maps. Liu et al. [15] presented a one-way Hash function based on chaotic neural network in 2006. Yang et al. [16] developed a one-way Hash function construction based on chaotic map network in 2008. In 2009, Xiao et al. [17] created a keyed Hash function construction based on chaotic neural network, which worked in parallel computing environment. However, all of the Hash algorithms mentioned earlier employed simple chaotic dynamic systems. Compared to them, the piecewise linear chaotic map (PWLCM), the chaotic tent map (CTM), the 4-dimensional and one-way coupled map lattices (4D OWCML) and the chaotic neural network presented in our paper greatly improve the spatiotemporal complexity of nonlinear dynamics and mixture.

It follows that an algorithm for constructing a one-way novel Hash function based on chaotic neural network structure is proposed in this paper. The chaotic neural network is composed of input layer and output layer. The PWLCM is utilized as transfer function, and the 4D OWCML is employed as key generator of the chaotic neural network. Theoretical analysis and computer simulation verify the validity of the proposed algorithm.

The rest part of this paper is organized as follows. Section 2 introduces the preliminaries about the PWLCM and 4D OWCML used in the proposed algorithm. In Sect. 3, the novel Hash algorithm based on chaotic neural network

Y. Li (✉) · S. Deng · D. Xiao
College of Computer Science, Chongqing University, 400044 Chongqing, China
e-mail: yantaoli@foxmail.com; liyantao@live.com

is described in detail. Performance is analyzed in Sect. 4. Finally, conclusions are drawn in Sect. 5.

2 Preliminaries

2.1 Analysis of the piecewise linear chaotic map (PWLCM)

The PWLCM proposed in the algorithm is one-dimensional, and piecewise liner map defined as follows:

$$X(k+1) = f(X(k),P) = \begin{cases} X(k)/P, & 0 \le X(k) \le P, \\ (X(k)-P)/(0.5-P), & P \le X(k) < 0.5, \\ (1-P-X(k))/(0.5-P), & 0.5 \le X(k) < 1-P, \\ (1-X(k))/P, & 1-P \le X(k) \le 1, \end{cases} \quad (1)$$

where $X \in [0, 1]$ and P is the control parameter and satisfies $0 < P < 0.5$. The map is piecewise linear, and the parameter P ensures the map is running in a chaotic state when $0 < P < 0.5$. It transforms an interval [0, 1] onto itself and contains only one parameter P. The PWLCM has some properties suitable for constructing the Hash value, such as parameter sensitivity and initial value sensitivity.

Although the form of the tent map is simple and the equation involved is linear, for any parameter values in interval (0, 0.5), this map can display chaotic phenomenon with the initial value $X(0) = 0.232323$ shown in Fig. 1. Figure 2 shows the simulation of the PWLCM iterating 40 times with the initial value $X(0) = 0.323232$ and parameter $P = 0.231$. And we can confirm from Fig. 3 that the PWLCM is a completely chaotic map when $0 < P < 0.5$.

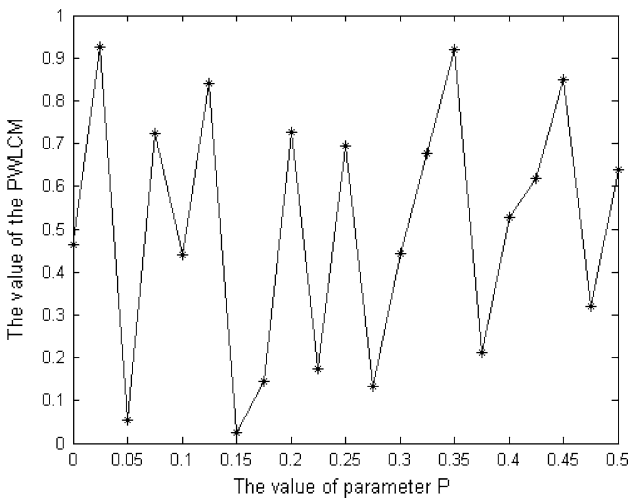

Fig. 1 Iteration property with changeable parameter P when $X(0) = 0.232323$

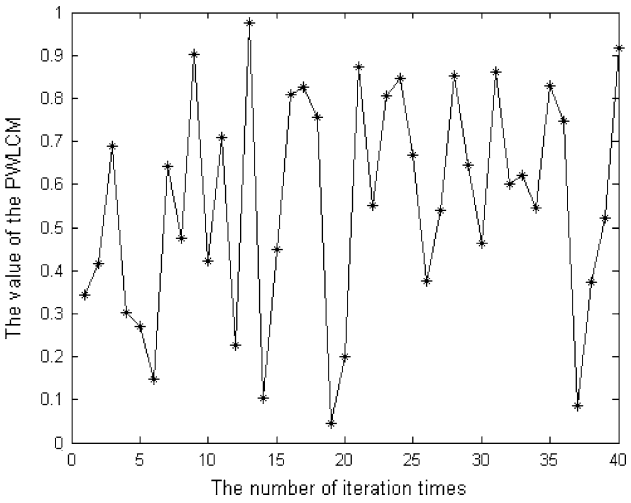

Fig. 2 Iteration property when $X(0) = 0.323232$ and $P = 0.231$

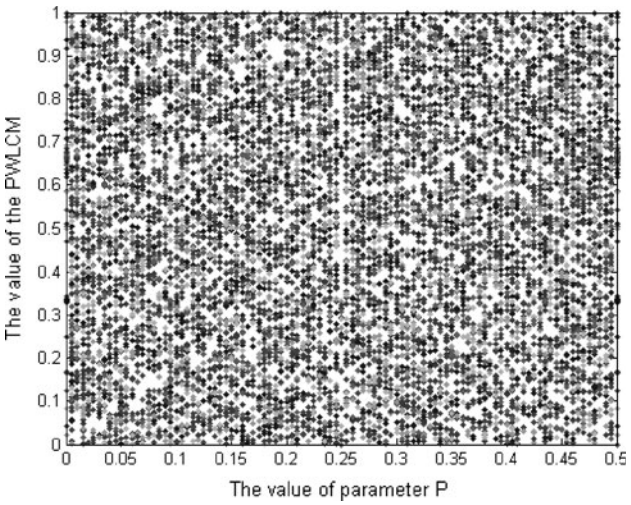

Fig. 3 The bifurcation diagram of the PWLCM iterated 100 times with $X(0) = 0.323232$

2.2 Analysis of the chaotic tent map (CTM)

The CTM utilized in the algorithm is one-dimensional, and discrete map defined as follows:

$$X(k+1) = g(X(k), Q) = \begin{cases} QX(k), & 0 \le X(k) \le 0.5, \\ Q(1-X(k)), & 0.5 \le X(k) \le 1. \end{cases} \quad (2)$$

where $X \in [0, 1]$ and $Q \in [\sqrt{2}, 2]$ are the iteration trajectory value and the parameter of the CTM. The map transforms an interval [0, 1] onto itself and contains only one parameter Q. The CTM has some properties suitable for constructing the Hash value, such as parameter sensitivity and initial value sensitivity. Figure 4 displays the chaotic iteration property with changeable parameter Q when $X(0) = 0.232323$. The simulation of the CTM iterating 40

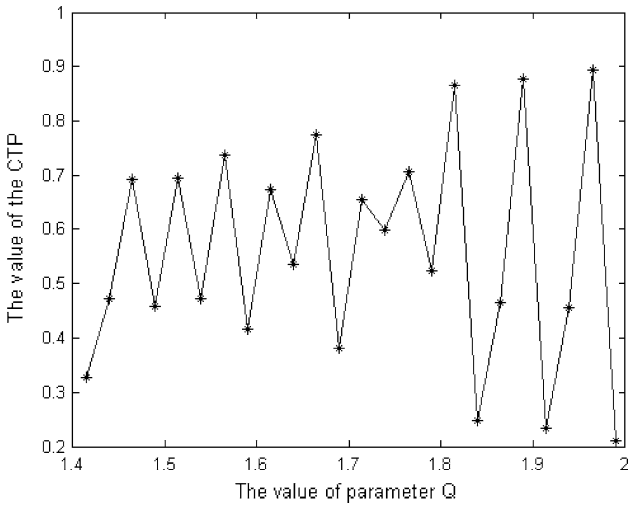

Fig. 4 Iteration property with changeable parameter Q when $X(0) = 0.232323$

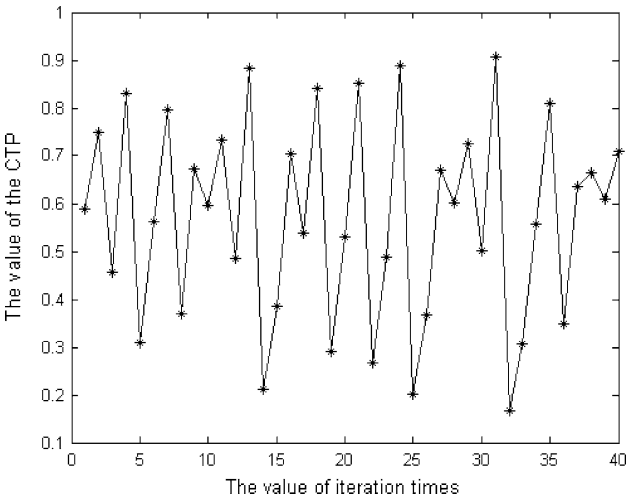

Fig. 5 Iteration property when $X(0) = 0.323232$ and $Q = 1.82$

times with the initial value $X(0) = 0.323232$ and parameter $Q = 1.82$ is shown in Fig. 5. We can get the conclusion from Fig. 6 that the chaotic property of the CTM is based on the parameter $Q \in [\sqrt{2}, 2]$.

2.3 Analysis of the key generator

The 4-dimensional and one-way coupled map lattices (4D OWCML) is described as:

$$\begin{cases} x_1(k+1) = (1-\varepsilon)g(x_4(k)) + \varepsilon g(x_3(k)) \\ x_2(k+1) = (1-\varepsilon)g(x_1(k)) + \varepsilon g(x_4(k)) \\ x_3(k+1) = (1-\varepsilon)g(x_2(k)) + \varepsilon g(x_1(k)) \\ x_4(k+1) = (1-\varepsilon)g(x_3(k)) + \varepsilon g(x_2(k)) \end{cases} \quad (3)$$

where $x_1(0), x_2(0), x_3(0), x_4(0) \in [0, 1]$ are the four initial values and the function $g()$ is the chaotic tent map (CTM) and $\varepsilon \in (0, 1)$ is a coupling constant.

The key generation function $gen()$ adopted in the algorithm based on 4D OWCML can be generalized as:

$$X(k+1) = gen(k+1) = (x_1(k+1) + x_2(k+1) + x_3(k+1) + x_4(k+1))/4. \quad (4)$$

3 Hash algorithm construction based on the chaotic neural network

3.1 The structure of the chaotic neural network

The chaotic neural network structure of the blocked Hash function proposed in the algorithm is shown in Fig. 7, which consists of both the input layer and output layer. The input layer has eight neurons, and the output layer has four neurons.

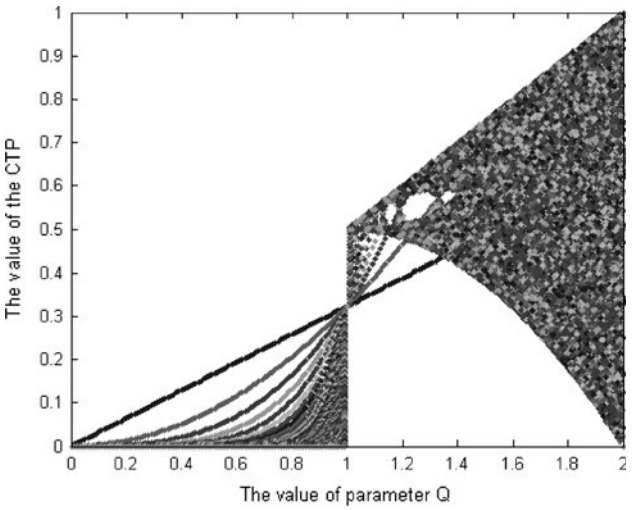

Fig. 6 The bifurcation diagram of the CTM iterated 10,000 times with $X(0) = 0.323232$

The input layer has eight neurons $W = (w_1, w_2, \ldots, w_8)$, and each neuron $w_i (i = 1, 2, \ldots, 8)$ has eight input data $m_i (i = 1, \ldots, 8; 9, \ldots, 16; \ldots; 57, \ldots, 64)$. Each m_i has a length of 8 bits. The structure parameters of each input neuron are the same, including the weight $\omega = [\omega_1, \omega_2, \ldots, \omega_8]$, the bias β and the transfer function $f()$ (PWLCM) with the parameter QI. For the input data $m_i (i = 1, \ldots, 8; 9, \ldots, 16; \ldots; 57, \ldots, 64)$, the corresponding eight neurons can be defined as $W = (w_1, w_2, \ldots, w_8)$.

$$w_i = f^{\tau}(\mathrm{mod}([\omega_1, \omega_2, \ldots, \omega_8] \times [m_{(i-1)\times 8+1}, m_{(i-1)\times 8+2}, \ldots, m_{(i-1)\times 8+8}]^T + \beta, 1), QI) \quad (5)$$

where τ is the iteration times of the PWLCM.

The output layer has four neurons $A = (a_1, a_2, a_3, a_4)$. The structure parameters of the output neurons are

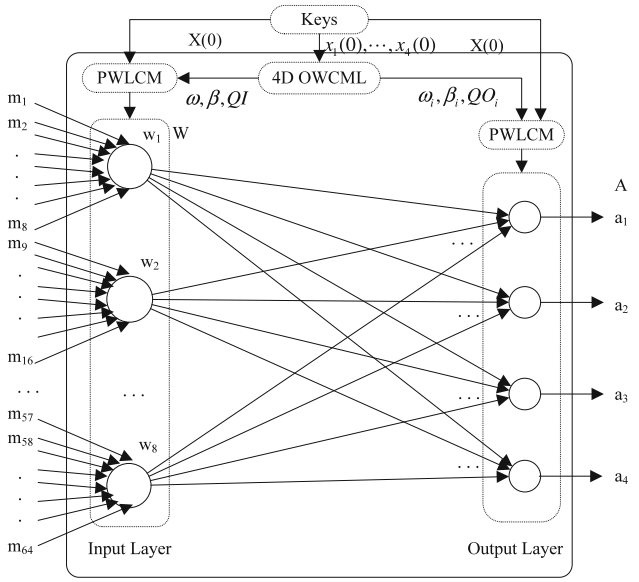

Fig. 7 The structure of two-layer neural network

composed of the weight of each neuron $\omega_i = [\omega_{i,1}, \omega_{i,2}, \ldots, \omega_{i,8}](i = 1, 2, 3, 4)$, the four corresponding biases $\beta_1, \beta_2, \beta_3, \beta_4$ and the transfer function $f()$ with the parameter $QO_i(i = 1, 2, 3, 4)$. For the eight neurons $W = (w_1, w_2, \ldots, w_8)$ of the input layer, the corresponding four output neurons can be described as $A = (a_1, a_2, a_3, a_4)$.

$$a_i = f^{\tau}(\mathrm{mod}([\omega_{i,1}, \omega_{i,2}, \ldots, \omega_{i,8}] \times [w_1, w_2, \ldots, w_8]^T + \beta_i, 1), QO_i) \quad (6)$$

where τ is the iteration times of the PWLCM.

3.2 Secret keys of the algorithm

The secret keys of the algorithm include the initial condition $X(0) \in [0, 1]$, initial parameter $P \in (0, 0.5)$ and iteration times τ of the PWLCM; the initial condition $X(0) \in [0, 1]$ and initial parameter $Q \in [\sqrt{2}, 2]$ of the CTM; the four initial values $x_1(0), x_2(0), x_3(0), x_4(0) \in [0, 1]$ and coupling constant $\varepsilon \in (0, 1)$ of 4D OWCML.

3.3 Description of the Hash algorithm based on chaotic neural network

The whole structure of the algorithm can be illustrated in Fig. 8. Let $N = 128$ be the bit-length of the Hash value. In advance, the original message M' with arbitrary length is padded with bits $(1010\ldots10)_2$ and the left 64-bit denoting the length of the original message M', such that its length is a multiple of 512. The Hash value with N-bit is generated as follows.

(1) Translate the appended message M into the corresponding ASCII code values, and then partition M into p blocks: $M_1, M_2, \ldots, M_p$, and each M_i $(i = 1,2,\ldots,p)$ has a length of 64-character (512-bit) denoted as $m'_{i,j}(i = 1, 2, \ldots, p; j = 1, 2, \ldots, 64)$. For each M_i, iterate the PWLCM $m'_{i,j}$ times with the initial value of last chaotic state $X(m'_{i,j-1})$ and the parameter $Q = (\frac{i}{p} + \frac{j}{64})/2$ to generate the corresponding decimal fraction $m_{i,j}(i = 1, 2, \ldots, p; j = 1, 2, \ldots, 64) \in [0, 1]$, which is short for $m_j(j = 1, 2, \ldots, 64)$in Fig. 7. And as shown in Fig. 7, $m_j(j = 1, 2, \ldots, 64)$ is divided into eight groups for further process.

(2) The secret keys utilized in the algorithm are set as the iteration times $\tau = 45$ of the PWLCM; the parameter $Q = 1.52$ of CMT; 128-bit secret key is partitioned into four keys with length of 32 bit each and then transform the four into corresponding decimal fractions by means of linear transform to generate $x_1(0), x_2(0), x_3(0), x_4(0)$ and the coupling constant $\varepsilon = 1/4$ of 4D OWCML; the initial Hash value $H(0) = \{0\}^N$.

At the input layer, iterate 10 times of key generation function $gen()$ Eq. 4 to generate the weight $\omega = [\omega_1, \omega_2, \ldots, \omega_8]$, the bias β and the parameter QI of the transfer function $f()$ Eq. 5, respectively.

At the output layer, continue to iterate key generation function $gen()$ 40 times to generate the weight $\omega_i = [\omega_{i,1}, \omega_{i,2}, \ldots, \omega_{i,8}](i = 1, 2, 3, 4)$, the biases $\beta_1, \beta_2, \beta_3, \beta_4$ and the parameter $QO_i(i = 1, 2, 3, 4)$ of the transfer function $f()$ Eq. 6, orderly and respectively.

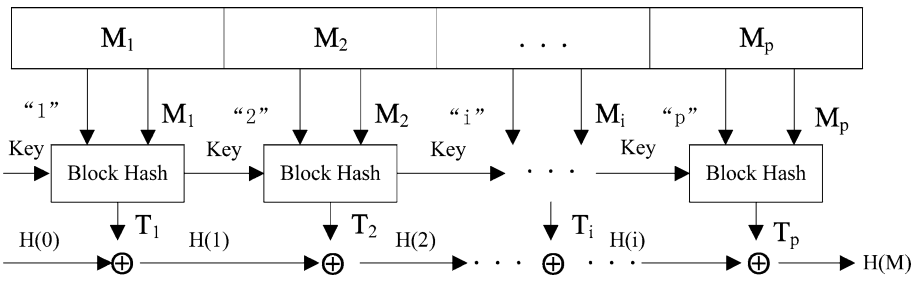

Fig. 8 The whole structure of the algorithm

(3) At the input layer, compute Eq. 5 with the obtained parameters in step (2) to generate $W = (w_1, w_2, \ldots, w_8)$. At the output layer, compute Eq. 6 with the obtained parameters in step (2) and the output data of input layer to generate $A = (a_1, a_2, a_3, a_4)$.
(4) Transform a_1, a_2, a_3, a_4 into the corresponding binary formats, then extract 32-bit of each and cascade these binary numbers orderly to generate the corresponding T_i.
(5) After all the blocks are processed, the final Hash value is generated by $H(M) = H(0) \oplus T_1 \oplus T_2 \oplus \cdots \oplus T_p$.

4 Performance analysis

4.1 Distribution of Hash value

The uniform distribution of Hash value is one of the most important properties of Hash function and is directly related to the security of Hash function. Simulation experiments have been done on the following paragraph of message:

> Chongqing University is a nationally famed comprehensive key university in China, directly under the State Ministry of Education, also a university listed among the first group of "211 Project" universities gaining preferential support in their construction and development from the Central Government of China. Currently, Chongqing University runs a graduate school and offers a wide range of undergraduate programs covering diverse branches of learning such as sciences, engineering, liberal arts, economics, management, law and education.

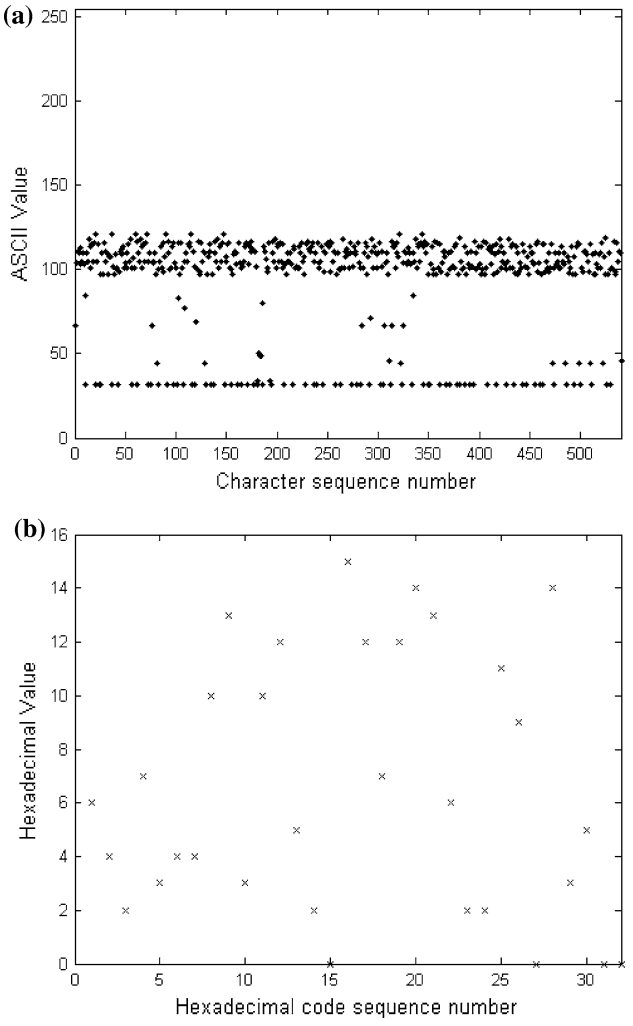

Fig. 9 Spread of messages and Hash values: **a** distribution of the original message in ASCII code; **b** distribution of the Hash values in hexadecimal format

Two graphs are used to demonstrate the differences between the original message and the final Hash value. In Fig. 9a, the ASCII codes of the original message are localized within a small and stable area and show certain information, while in Fig. 9b, the hexadecimal Hash values spread around very irregularly and show uncertain information. This simulation results indicate that there is no related information including the statistic information between the original message and the final Hash value.

4.2 Sensitivity of Hash value to the message and secret keys

In order to evaluate the sensitivity of Hash value to the message and secret keys, Hash simulation experiments have been conducted under the following different seven conditions:

C1: The original message is the same as the one in Sect. 4.1.
C2: Change the first character "*C*" in the original message to "*D*".
C3: Change the word "*directly*" in the original message to "*indirectly*".
C4: Add a blank space at the end of the original message.
C5: Change the parameter ε of the CMT from 1/4 to 1/3.
C6: Change τ of Eqs. 4 and 5 from 45 to 46.
C7: Exchange the first message block M_1-"*Chongqing University is a nationally famed comprehensive key uni*" with the second message block M_2-"*versity in China, directly under the State Ministry of Education*".

The corresponding Hash values in hexadecimal formats are gotten from simulation experiments as following, followed by the corresponding number of different bits compared with the Hash value obtained under Condition 1:

C1: 6427344AD3AC520FC7CED622B90E3500.
C2: 981C469ABDF32294AB95BC046693C2C0 (74).

C3: 6619EC32E9B184450EDAF7C7D4DC525F (63).
C4: D9D8A52F9BF5219D436498F844E1BBF9 (74).
C5: 9B02BBF1BEA49F60D03E848CFEE5F8FF (78).
C6: 5314F93A608A33FA3C737DFB03F3C882 (78).
C7: 300EAAC153096563CC1740E8B2FB01EA (62).

The corresponding graphical display of binary sequences is shown in Fig. 10.

The simulation result indicates that sensitivity property of the proposed algorithm is so perfect that any tiny difference of the message or key will cause huge changes in the final Hash value. In addition, a careful analysis of sensitivity of Hash value to secret keys (C5, C6) reveals that tiny changes of secret keys cause corresponding 78-bit difference out of 128 bits. So, we can confirm that the key factor determines the quality of the proposed algorithm.

4.3 Statistical analysis of diffusion and confusion

Confusion and diffusion are two basic design criteria for encryption algorithm, including Hash algorithms. Diffusion means 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. Confusion means the use of transformations that complicate dependence of the statistics of cipher text on the statistics of plaintext. Hash function requires the message to diffuse its influence into the whole Hash space. This means that the correlation between the message and the corresponding Hash value should be as small as possible. If the Hash value is expressed in binary format, each bit can be only 0 or 1. Therefore, the ideal diffusion effect should be that any tiny change in the initial condition, control parameter or plaintext leads to a 50% changing probability for each bit of Hash value. Four statistics used here are as follows: mean changed bit number $\overline{B}$, mean changed probability P, standard deviation of the changed bit number ΔB and standard deviation ΔP. They are defined as follows:

Mean changed bit number: $\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$.

Mean changed probability: $P = (\overline{B}/128) \times 100\%$.

Standard deviation of the changed bit number: $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \overline{B})^2}$.

Standard deviation: $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i/128 - P)^2} \times 100\%$.

Here, N is the total number of tests and B_i denotes changed bit number in the ith test.

The diffusion and confusion test is performed as follows: a paragraph of message is randomly chosen, and the corresponding Hash value is generated. Then a bit in the message is randomly selected and toggled, and a new Hash value is obtained. Finally, two Hash values are compared, and the number of hanged bit is counted as B_i. This test is performed N times, and the corresponding distribution of changed bit number is shown as Fig. 11a, b, where $N = 2{,}048$.

It can be seen from Fig. 11a, b that the maximum changed bit number is 83, and the minimum is 42. It shows a good diffusion effect of the proposed Hash function algorithm.

The same tests on the algorithm with $N = 256$, 512, 1,024 and 2,048 have also been performed. Under the condition that 1 bit is changed at each time, the corresponding values of $\overline{B}$, P, ΔB, and ΔP are obtained as shown in Table 1.

Based on the analysis of the data in Table 1, we can draw the conclusion: the mean changed bit number $\overline{B}$ and the mean changed probability P are both very close to the ideal value 64-bit and 50%, while ΔB and ΔP are very little, which indicates the capability for diffusion and confusion is very stable. The statistical effect guarantees that attacker cannot carry out statistical attack.

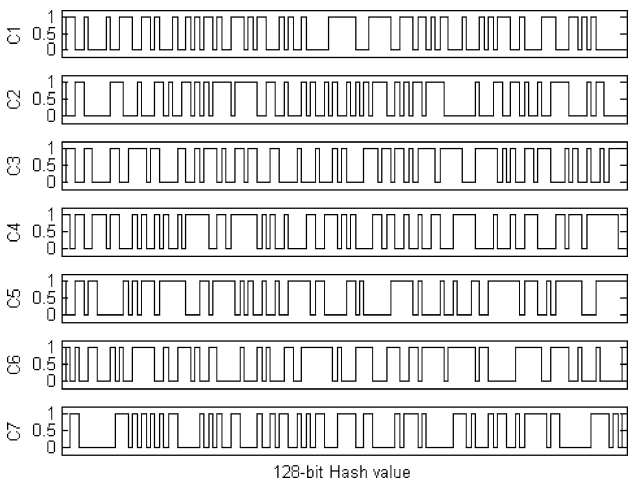

Fig. 10 Hash values under different conditions

4.4 Analysis of collision resistance

We have performed the following test to conduct quantitative analysis on collision resistance [6, 8, 12]: first, the Hash value for a paragraph of message randomly chosen is generated and stored in ASCII format. Then a bit in the paragraph is selected randomly and toggled, and thus a new Hash value is then generated and stored in the same format. Tow Hash values are compared with each other, and the number of character in this format with the same value at the same location in Hash value is counted. Moreover, the absolute difference of two Hash values is calculated using the formula: $d = \sum_{i=1}^{N} |t(e_i) - t(e_i')|$, where e_i and e_i' are the ith ASCII character of the original and the new Hash value, respectively, and the function $t(*)$ converts the entries to their equivalent decimal values. This kind of collision test is

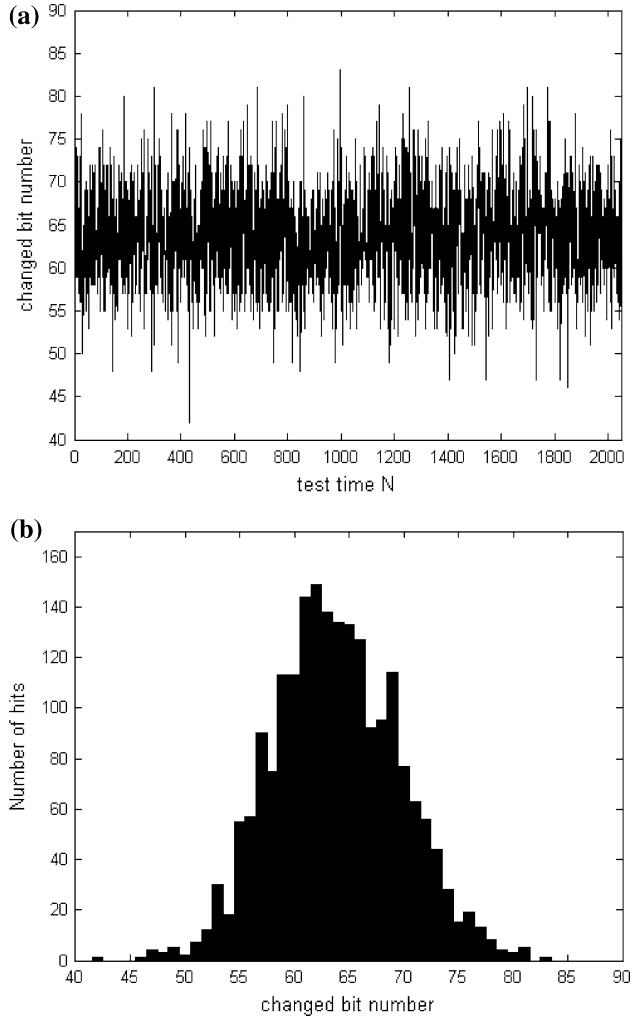

Fig. 11 Distribution of changed bit number: **a** Plot of B_i, **b** Histogram of B_i

Table 1 Statistics of number of changed bit

N	256	512	1,024	2,048	Mean
$\overline{B}$	63.7773	63.5605	63.5439	63.8076	63.6723
P (%)	49.8260	49.6567	49.6437	49.8497	49.7440
ΔB	5.3915	5.6264	5.7143	5.7563	5.6221
ΔP (%)	4.2121	4.3956	4.4643	4.4971	4.3923

performed 2,048 times. Two Hash values are compared, and the number of ASCII characters with the same value at the same location in the Hash value, namely the number of hits, is counted. A plot of the distribution of the number of hits is given in Fig. 12. Seen from Fig. 12, there are 2 tests to hit twice, and 118 tests to hit once, while in 1,928 tests, no hit occurs. The maximum number of equal characters at the same location in two hash values is only 2.

The maximum, minimum, mean and mean/character values of d are listed in Table 2. The simulation result

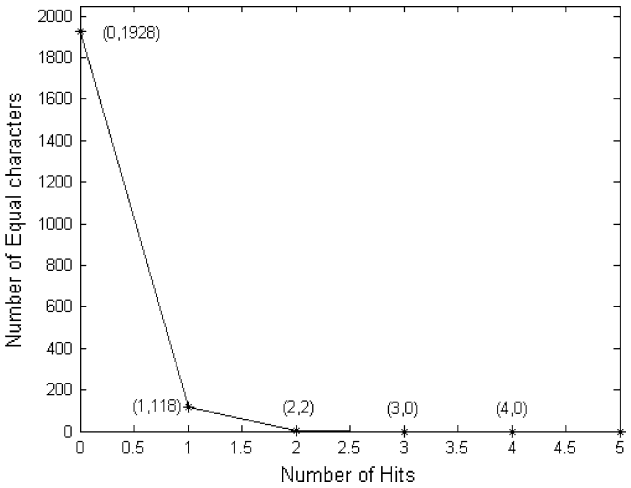

Fig. 12 Distribution of the number of ASCII characters with the same value at the same location in the Hash value

Table 2 Absolute difference of two hash values

Absolute difference	Maximum	Minimum	Mean	Mean/character
Values	2,220	687	1432.1	89.51

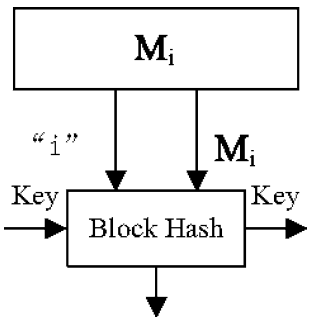

Fig. 13 Meet-in-the-middle attack

indicates that the sensitivity property of hash value is perfect that the absolute difference/character in the final hash value corresponding to any least difference of message or key will always wave around the theoretical value 85.3333 [13].

4.5 Analysis of meet-in-the-middle resistance

Meet-in-the-middle attack [18] means to find a contradiction through looking for a suitable substitution of the last plaintext block. For instance, $M = (M_1, \ldots, M_i, \ldots, M_j, \ldots, M_p)$, the expected contradicted one is $M' = (M_1, \ldots, M_j, \ldots, M_i, \ldots, M_p)$. The attack process is just to exchange the position of M_i with M_j but keep the final Hash value $H(M)$ unchanged. In our algorithm, shown in Fig. 13, each

Table 3 The performance comparison with 2,048 random tests

Schemes	Maximum numbers	Mean/character of absolute differences	Statistical performances of the schemes			
			$\overline{B}$	P (%)	ΔB	ΔP (%)
MD5 [9]	2	81.5	64.03	50.02	5.66	4.42
Xiao's [6]	3	94.125	63.8501	49.88	5.7889	4.52
Zhang's [8]	2	78.5625	63.91	49.92	5.58	4.36
Wang's [9]	2	95.375	63.98	49.98	5.53	4.33
Deng's [13]	2	87.49	63.84	49.88	5.88	4.59
Yang's [16]	2	93.25	64.14	50.11	5.55	4.33
Xiao's [17]	2	76.7375	64.0098	50.01	5.7236	4.47
Our algorithm	2	89.51	63.8076	49.8497	5.7563	4.4971

of the message blocks denoted by the sequence number will be divided into 64 sub-blocks, and the ASCII code values and the order "i, j" of every sub-block are set as the iteration times and the parameter $Q = (\frac{i}{p} + \frac{j}{64})/2$ of the PWLCM to generate the corresponding decimal fraction m_j. Thus, this keeps the algorithm secure against this kind of attack. The proposed algorithm is immune from meet-in-the-middle attack.

4.6 Comparison with other algorithms

Recently, some chaos-based Hash algorithms are proposed. We choose some excellent algorithms from [6, 8, 9, 13, 16, 17] as representatives and carry out the corresponding comparison tests. The results are listed in Table 3. (Note: since the corresponding data from [6, 8, 9, 13, 16, 17] are based on test time $N = 2{,}048$ in common, we set the same test time of our algorithm to them for convenient comparison).

Based on the comparison in Table 3, we can obtain the conclusions: our proposed algorithm shows a lower value in the maximum number of equal characters at the same location in two hash values than Xiao's algorithm [6], the closest mean/character of absolute difference of two hash values to theoretical value 85.3333 among all proposed schemes except Deng's algorithm [13], and some better statistical performances of diffusion and confusion. So our proposed algorithm outperforms all the proposed algorithms in some aspects.

5 Conclusion

Based on two-layer chaotic neural network structure, an algorithm for constructing a one-way novel Hash function has been proposed and analyzed in this paper. The chaotic neural network is composed of input layer and output layer. The piecewise linear chaotic map (PWLCM) is utilized as transfer function, and the 4-dimensional and one-way coupled map lattices (4D OWCML) is employed as key generator of the chaotic neural network. Simulation results show that the algorithm has extreme sensitivity to message and secret key, strong diffusion and confusion capability, good collision resistance and security against meet-in-the-middle attacks.

Acknowledgments Our sincere thanks go to the anonymous reviewers for their valuable comments. The work described in this paper was fully funded by Project No. CDJZR10180003 supported by the Fundamental Research Funds for the Central Universities.

References

1. Kwok HS, Tang WKS (2005) A chaos-based cryptographic Hash function for message authentication. Int J Bifur Chaos 15: 4043–4050
2. Gao W, Li F, Wang XL (2009) Chameleon Hash without key exposure based on schnorr signature. Comput Stand Interfaces 31:282–285
3. Lian SG, Sun JS, Wang ZQ (2006) Secure Hash function based on neural network. Neurocomputing 69:2346–2350
4. Lin IC, Ou HH, Hwang MS (2005) A user authentication system using back-propagation network. Neural Comput Appl 14:243–249
5. Wong KW (2003) A combined chaotic cryptographic and Hashing scheme. Phys Lett A 307:292–298
6. Xiao D, Liao XF, Deng SJ (2005) One-way Hash function construction based on the chaotic map with changeable-parameter. Chaos Solitons Fractals 24:65–71
7. Yi X (2005) Hash function based on chaotic tent maps. IEEE Trans Circuits Syst II-Express Briefs 52:354–357
8. Zhang JS, Wang XM, Zhang WF (2007) Chaotic keyed Hash function based on feedforward–feedback nonlinear digital filter. Phys Lett A 362:439–448
9. Wang Y, Liao XF, Xiao D, Wong KW (2008) One-way Hash function construction based on 2D coupled map lattices. Inform Sci 178:1391–1406
10. Xiao D, Liao XF, Deng SJ (2008) Parallel keyed Hash function construction based on chaotic maps. Phys Lett A 372:4682–4688
11. Liu XD, Xiu CB (2008) Hysteresis modeling based on the hysteretic chaotic neural network. Neural Comput Applic 17:579–583
12. Deng SJ, Xiao D, Li YT, Peng WB (2009) A novel combined cryptographic and Hash algorithm based on chaotic control character. Commun Nonlinear Sci Numer Simulat 14:3889–3900
13. Deng SJ, Li YT, Xiao D (2009) Analysis and improvement of a chaos-based Hash function construction. Commun Nonlinear Sci Numer Simulat 15:1338–1347

14. Rachel MK, Einat K, Ido K,Wolfgang (2003) Public channel cryptography by synchronization of neural networks and chaotic maps. Phys Rev Lett 91:118701/1-118701/4
15. Liu GL, Shan L, Dai YW et al (2006) One-way Hash function based on chaotic neural network. Acta Phys Sin 55:06–5688 (in Chinese)
16. Yang HQ, Wong KW, Liao XF et al (2009) One-way Hash function construction based on chaotic map network. Chaos Solitons Fractals 41:2566–2574
17. Xiao D, Liao XF, Wang Y (2009) Parallel keyed Hash function construction based on chaotic neural network. Neurocomputing 72:2288–2296
18. Vanstone SA, Menezes AJ, Oorschot PC (1996) Handbook of applied cryptography. CRC Press, New York