Hash function construction using weighted complex dynamical networks*

Song Yu-Rong(宋玉蓉)† and Jiang Guo-Ping(蒋国平)

College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210003, *China*

(Received 11 August 2012; revised manuscript received 19 October 2012)

A novel scheme to construct a hash function based on a weighted complex dynamical network (WCDN) generated from an original message is proposed in this paper. First, the original message is divided into blocks. Then, each block is divided into components, and the nodes and weighted edges are well defined from these components and their relations. Namely, the WCDN closely related to the original message is established. Furthermore, the node dynamics of the WCDN are chosen as a chaotic map. After chaotic iterations, quantization and exclusive-or operations, the fixed-length hash value is obtained. This scheme has the property that any tiny change in message can be diffused rapidly through the WCDN, leading to very different hash values. Analysis and simulation show that the scheme possesses good statistical properties, excellent confusion and diffusion, strong collision resistance and high efficiency.

Keywords: hash function, weighted complex dynamical networks, chaotic map, cryptography

PACS: 05.45.−a, 89.75.−k **DOI:** 10.1088/1674-1056/22/4/040506

1. Introduction

Chaos is a kind of deterministic random-like process found in nonlinear dynamical systems and has been used in cryptography and the generation of pseudo-random numbers due to its attractive features such as sensitivity to initial value, random-like behavior and ergodicity.[1,2] In recent years there have been a tremendous number of reports on how to use chaotic systems to design cryptographic algorithms.[1,3–6] In particular, hash function construction based on chaos has attracted the attention of many researchers,[7–21] and a lot of hash functions based on chaos have been proposed and can solve the problem of massive operation in traditional hash algorithms. Based on chaotically coupled map lattices, several hash function schemes have been proposed.[8,12,14,15,20] Using chaotic tent maps, some algorithms for one-way hash functions were presented.[9,11,16] Akhshani *et al.*[10] proposed a hash function according to the hierarchy of two-dimensional (2D) piecewise nonlinear chaotic maps. Xiao *et al.*[17] proposed a one-way hash function algorithm on the basis of a piecewise linear chaotic map with changeable-parameters, and Luo *et al.*[20] proposed and analyzed a one-way hash function algorithm in line with the spatiotemporal chaotic system.

However, some drawbacks have been pointed out in chaos-based hash functions. For example, Li *et al.*[22] showed that the uniform distribution property in Xiao's scheme[17] will collapse in finite computing precision, even if it is theoretically uniformly distributed in continuous phase space. Wang *et al.*[23] pointed out that the schemes in Refs. [15] and [18] have a conflict problem caused by inappropriate parameter mapping, unsuitable plain-text grouping, or improper irritation processing. On the other hand, there are so many iterating times that the hashing speed is too slow for practical use in some chaos-based hash function algorithms. For example, in Ref. [19], the number of iterations for encrypting each message byte is at least 200. In Wang *et al.*'s scheme,[12] in order to maintain the independence of the distribution of each lattice value, the 4×8 two-dimensional (2D) coupled map lattice needs to be iterated 45 times in each step. A similar problem also exists in Refs. [10] and [11]. Therefore, a few key factors, such as choosing an appropriate map, enhancing the complexity of the chaotic map, and minimizing or eliminating the effect of finite precision must be considered carefully when a hash function is constructed.

It is well known that excellent confusion and diffusion are key properties for a secure cryptography algorithm. In some research results about complex dynamical networks,[24,25] the topology of a network often plays a crucial role in determining its dynamical behaviors, which include the synchronization of the network, the propagation of epidemic and rumor, and the diffusion of fads.[26–34] For example, the ability to achieve synchronization in a large-sized nearest-neighbor coupled network can be greatly enhanced by simply adding a tiny fraction of distant links, thereby making the network become a small-world model.[26] Compared with information propagation over other networks (e.g., random networks, small-world networks or scale-free networks), the spreading speed on the nearest neighbor coupled networks is slowest.[35,36] The concept of the efficiency of a network has been introduced as a measure of how efficiently it exchanges information.[37] Un-

*Project supported by the Natural Science Foundation of Jiangsu Province, China (Grant No. BK2010526), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20103223110003), and The Ministry of Education Research in the Humanities and Social Sciences Planning Fund, China (Grant No. 12YJAZH120).

†Corresponding author. E-mail: songyr@njupt.edu.cn

© 2013 Chinese Physical Society and IOP Publishing Ltd

der the same conditions, the efficiency is higher in random networks than in nearest-neighbor coupled networks. According to the concept of a complex network, a coupled map lattice is a nearest-neighbor coupled network. In order to speed up the confusion and diffusion of an original message, a homogeneous mixing network is obviously more effective than a coupled map lattice.

In this paper, borrowing the idea of generating a weighted complex dynamical network, a novel scheme for constructing a hash function is proposed. In order to speed up the confusion and diffusion of an original message, we construct a weighted complex dynamical network from the original message. First, the original message is divided into blocks, and each block is divided into components. Second, we treat these components as the nodes of the network, and define the edges from the relations between the components (or nodes). Finally, we take the chaotic map as the node dynamics, and the divided message is chosen as the initial values of each node. Through chaotic iterations, quantization and exclusive-or operations, the scheme can construct a fixed-length hash value and lead to very different hash values due to the advantage that any tiny change in message can be diffused rapidly.

2. The weighted complex dynamical network and its generation from a message

2.1. The complex network generated from a message

We generate a weighted complex dynamical network (WCDN) from an original message. As shown in Fig. 1, the original message is divided into K blocks $M_1, M_2, \ldots, M_K$, then each block $M_k\,(k = 1, \ldots, K)$ with the L-bit is divided into N components $c_{(k,1)}, c_{(k,2)}, \ldots, c_{(k,N)}$, and each component $c_{(k,i)}\,(k = 1, \ldots, K; i = 1, \ldots, N)$ with m-byte (or $8m$-bit) can be regarded as a node in the network. Let $V_k = \{c_{(k,i)} | i = 1, 2, \ldots, N\}$ denote the node set of the network generated by the message block M_k, and $E_k = \{e_{(k,i,j)} | i, j = 1, 2, \ldots, N\}$ as the edge set of the network, where $e_{(k,i,j)}$ is an element of E_k describing the edge weight value between node i and node j. It is noted that the division must satisfy the condition $L = N \times 8m$. For example, we can set $L = 256$, $N = 8$, and $m = 4$.

2.2. The mathematical model of WCDN

For the network in Fig. 1, we consider a general weighted complex dynamical network consisting of N nodes with linearly diffusive couplings. Let each node of the network be assigned to a dynamical variable. The evolution of the dynamical variables can be described by[38,39]

$$x_{(k,i,t+1)} = f(x_{(k,i,t)}) + \sum_{j=1}^{N} e_{(k,i,j)} x_{(k,j,t)}, \quad i = 1, 2, \ldots, N, \quad (1)$$

where $x_{(k,i,t)}$ is the dynamical variable of the i-th node in the k-th network at the t-th time step. The topology of the network is introduced through the coupling matrix $\boldsymbol{E}_k$ with the elements $e_{(k,i,j)}$, and $f(x_{(k,i,t)})$ is a real function describing the local dynamics, which can be a chaotic map, such as a logarithmic map $f(x_{(k,i,t)}) = b + \ln|x_{(k,i,t)}|$, $b \in (-1, 1)$, logistic map $f(x_{(k,i,t)}) = \mu_{(k,i)} x_{(k,i,t)}(1 - x_{(k,i,t)})$, $\mu_{(k,i)} \in (3.57, 4)$ or other chaotic map. In our algorithm, the logistic map is taken as the local map.

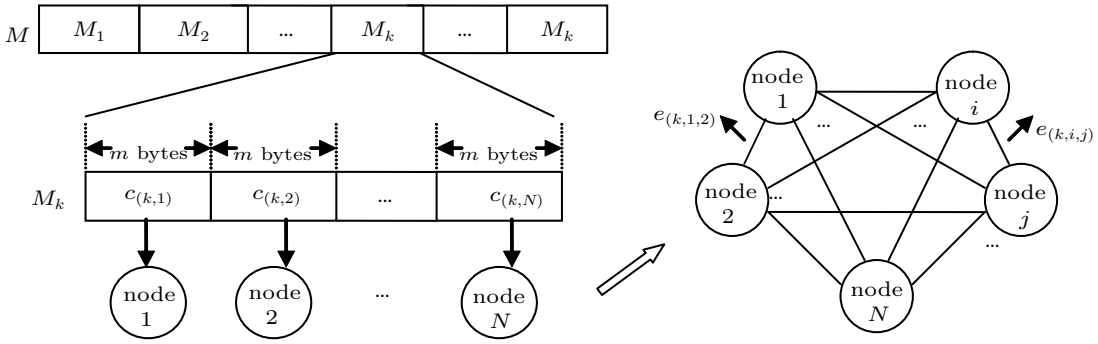


Fig. 1. Network generation from a message.

Remark 1 The second term of Eq. (1) can be regarded as a disturbance generated by the network for the chaotic map $x_{(k,i,t+1)} = f(x_{(k,i,t)})$. If we take the logistic map, the disturbance might result in $x_{(k,i,t+1)} \notin [0, 1]$, namely, $x_{(k,i,t+1)}$ may be outside the domain of the map. To solve the problem, we can change the value $x_{(k,i,t+1)}$ through MOD operation, namely, $x_{(k,i,t+1)} = |x_{(k,i,t+1)}| \bmod 1$.

Thus, we can construct a modified weighted complex dynamical network as follows:

$$\begin{pmatrix} x_{(k,i,t+1)} \\ \vdots \\ x_{(k,N,t+1)} \end{pmatrix} = \left| \begin{pmatrix} \mu_1 x_{(k,1,t)}(1 - x_{(k,1,t)}) \\ \vdots \\ \mu_N x_{(k,N,t)}(1 - x_{(k,N,t)}) \end{pmatrix} + \begin{pmatrix} e_{(k,1,1)} & \cdots & e_{(k,1,N)} \\ \vdots & \ddots & \vdots \\ e_{(k,N,1)} & \cdots & e_{(k,N,N)} \end{pmatrix} \begin{pmatrix} x_{(k,1,t)} \\ \vdots \\ x_{(k,N,t)} \end{pmatrix} \right| \bmod 1, \quad (2)$$

or

$$\boldsymbol{x}_{(k,t+1)} = |\boldsymbol{f}(\boldsymbol{x}_{(k,t)}) + \boldsymbol{E}_k \boldsymbol{x}_{(k,t)}| \bmod 1. \quad (3)$$

Equation (3) is in the vector form of Eq. (2). In Eq. (3), $\boldsymbol{x}_{(k,t)} = (x_{(k,1,t)}, x_{(k,2,t)}, \ldots, x_{(k,N,t)})$, corresponding to the states of the nodes in the network, and $\boldsymbol{E}_k = \{e_{(k,i,j)} | i,j = 1,2,\ldots,N\}$.

Remark 2 A very useful characteristic of the above network as an encryption scheme is that it is highly sensitive to $x_{(k,i,t)}$ and $e_{(k,i,j)}$ in Eq. (2). Any tiny change in a node can rapidly diffuse and affect the evolution of the other nodes.

Remark 3 In the next section, we will transform $c_{(k,i)}$ into the network initial values $x_{(k,i,0)}$, and construct $\boldsymbol{E}_k$ in Eq. (3).

3. The construction of the hash function

We take a message with arbitrary length as an input signal and produce a hash value as output. The overall scheme of the hash function construction is shown in Fig. 2. The scheme or algorithm mainly consists of the following five steps.

Step 1 Pre-processing of an original message $\boldsymbol{M}$

Set the following transformation:

$$M' = M + \text{string}(|M|), \quad (4)$$

where '+' denotes the concatenation operation of a character string, $|M|$ is the length of the original message M, and string (x) represents a transformation from integer number x to the corresponding character string. Then there are padding characters to the last block so that the size of the expanded message consists of $K \times L$ bits.

Step 2 Determination of $\mu_{(k,i)}$ and $\boldsymbol{H}_0$

Choose suitable values of $\mu_{(k,i)}$, $i = 1,2,\ldots,N$ for computing Eq. (2), where $\mu_{(k,i)}$ is a real number and $\mu_{(k,i)} \in (3.57,4)$.

Let $\boldsymbol{H}_k$ $(k = 1,2,\ldots,K)$ with h-bit denoting the hash value related to the k-th message block. Choose a suitable $\boldsymbol{H}_0$ as the initial hash value with h bits. Note that $\boldsymbol{H}_0$ can be known to the authenticator and the attacker.

Step 3 Determination of $x_{(k,i,0)}$ and $\boldsymbol{E}_k$

For the first message block M_1, the initial value $x_{(k,i,0)}$ of node i can be obtained by the following transformation:

$$x_{(k,i,0)} = (k \times 2^{-7} + i \times 2^{-4} + c_{(k,i)} \times 2^{-8m}) \bmod 1, \quad (5)$$

where k denotes the order of message block M_k, and $c_{(k,i)} \times 2^{-8m}$ denotes that the integer $c_{(k,i)}$ with $8m$-bit is converted to the fixed point number with $8m$ bits precision. It is noted that the MOD operation can ensure $0 < x_{(k,i,0)} < 1$.

For the k-th message block M_k $(k \geq 2)$, the initial value of node i, $x_{(k,i,0)}$ can be obtained by the following transformation

$$x_{(k,i,0)} = (k \times 2^{-7} + i \times 2^{-4} + c_{(k,i)} \times 2^{-8m} + x_{(k-1,i,r)}) \bmod 1, \quad (6)$$

where $x_{(k-1,i,r)}$ is the output value corresponding to the previous message block M_{k-1} (see Fig. 2).

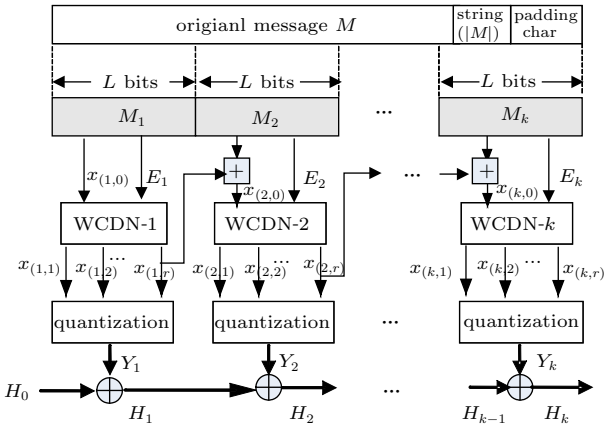


Fig. 2. The hash function generation scheme.

Then, we generate the coupling matrix $\boldsymbol{E}_k$.

a) Let $\varepsilon_{(k,i,j)} = c_{(k,i)} \oplus c_{(k,j)}, i,j = 1,2,\ldots,N$, where $\varepsilon_{(k,i,j)} = (\varepsilon_{(k,i,j)}[1], \varepsilon_{(k,i,j)}[2], \ldots, \varepsilon_{(k,i,j)}[m])$, and '$\oplus$' denotes the exclusive-or operation bit by bit.

b) Let $a_{(k,i,j)} = (\varepsilon_{(k,i,j)}[1] + \varepsilon_{(k,i,j)}[2] + , \ldots, + \varepsilon_{(k,i,j)}[m]) \bmod 256$.

c) Considering the position of each element in $\boldsymbol{E}_k$ and the order of message block k, we take the variable $p_{(k,i,j)}$ as

$$p_{(k,i,j)} = ((j-1) \times N + i + k) \bmod 256. \quad (7)$$

d) Generate the matrix $\boldsymbol{E}_k$ by the transformation,

$$e_{(k,i,j)} = p_{(k,i,j)} \oplus a_{(k,i,j)} - 128) \times 2^{-8}. \quad (8)$$

Step 4 Iterations and quantization

This step consists of r round iterations and quantization. After r round iterations by performing Eq. (2), $\boldsymbol{x}_{(k,1)}, \boldsymbol{x}_{(k,2)}, \ldots, \boldsymbol{x}_{(k,r)}$ are produced. The result of each iteration is quantized to N q-bit sequences by

$$Q_{(k,i,t)} = \text{int}(x_{(k,i,t)} \times 2^q), \quad (9)$$

where q is the quantized degree. Each $Q_{(k,i,t)}$ can be written as a q-bit binary sequence, $Q_{(k,i,t)} = b_{(k,i,t,1)} b_{(k,i,t,2)} \cdots b_{(k,i,t,q)}$ $(i = 1,2,\ldots,N$ and $t = 1,2,\ldots,r)$. Usually, N is the multiple of 8 and q has the form of 2^α $(\alpha = 0,1,2,\ldots)$. By rearranging the elements of $Q_{(k,i,t)} = b_{(k,i,t,1)} b_{(k,i,t,2)} \ldots b_{(k,i,t,q)}$, we can obtain the t-th round quantization output by

$$\begin{aligned} Y_t = {} & b_{(k,1,t,1)} b_{(k,2,t,1)} \cdots b_{(k,N,t,1)} b_{(k,1,t,2)} b_{(k,2,t,2)} \cdots \\ & b_{(k,N,t,2)} \cdots b_{(k,1,t,q)} b_{(k,2,t,q)} \cdots b_{(k,N,t,q)}. \end{aligned} \quad (10)$$

After r round iterations, we combine each round output and produce a $32 \times r$-bit value, denoted by $\boldsymbol{Y}_k$. Thus, the quantization output of the k-th message block M_k is $\boldsymbol{Y}_k = (Y_1, Y_2, \ldots, Y_r)$.

We obtain $\boldsymbol{H}_k$ as follows:

$$\boldsymbol{H}_k = \boldsymbol{H}_{k-1} \oplus \boldsymbol{Y}, \ k = 1,2,\ldots,K. \quad (11)$$

Apparently, the number of rounds r, quantized degree q, and the size of network N determine the length of hash value h, namely $h = N \times q \times r$.

Step 5 Process the next message block following Step 3 and Step 4. After all K blocks are processed, the final hash value can be obtained from $\boldsymbol{H}_K$, namely $\boldsymbol{H} = \boldsymbol{H}_K$.

4. Security and performance analysis

4.1. The hash value of a message and parameter sensibility

We set the initial parameters as: $\boldsymbol{H}_0 = 0$, $\mu_1 = \mu_2 = \cdots = \mu_N = 3.95$, $N = 8$, $l = 256$ bits, $q = 4$, $r = 4$, and $h = 128$ bits.

The proposed algorithm will be separately performed in the following eight cases:

Case 0 The original message M is "In recent years, a large amount of work on chaos-based cryptosystems has been published. However, many of the proposed schemes fail to explain or do not possess a number of features that are fundamentally important to all type s cryptosystems.";

Case 1 Change the first letter 'I' into 'J' and only one bit is changed;

Case 2 Append a char '0' to the end of the message to study the influence of the padding char;

Case 3 Change the initial parameter $x_{(1,1,0)}$ by 2^{-32}, namely $x_{(1,1,0)} = x_{(1,1,0)} + 2^{-32}$;

Case 4 Change the coupling factor $e_{(1,1,1)}$ by 2^{-32} and 2^{-28}, respectively;

Case 5 Change the initial parameter μ_1 by 2^{-32}, 2^{-28}, and 2^{-26}, respectively.

The corresponding hash values in hexadecimal form are as follows.

Case 0 301F5E89DF016398A4659159A8B61008;

Case 1 912F7FDFD037EABCB0E7E30C92383527;

Case 2 1C889CE2CE5625FE6477CAD8ECCFB090;

Case 3 B1DE93F9FB7915361C296663FBD27D8F;

Case 4

301756815789EB913DDA24456CD452A0 (2^{-32});

9D97CD541E268753AB9590FA957EB671 (2^{-28});

Case 5

301F5E895F81E318309D53043030319A (2^{-32});

6F1F1C271C028DEA64D2D4DA4F36CF50 (2^{-28});

5095E03B9A9570FD767A10E432626242 (2^{-26}).

The simulation results indicate that a tiny change in the message or parameter will cause a substantial difference in the final hash value. Meanwhile, the sensibility of μ_1 is lower than that of $e_{(1,1,1)}$ and $x_{(1,1,0)}$.

4.2. Statistical analysis of diffusion and confusion

From the perspective of resisting attacks, any designed cryptosystem should have the following two basic cryptographic properties: confusion and diffusion.[1] For the hash value with a binary format, each bit is 0 or 1. Therefore, the ideal diffusion effect should be that any tiny change in the original message can cause a bit change in the hash value with a probability of 0.5, and that the change is stable. The following six performance indexes are usually defined and used:[8–12,20]

mean number of changed bits: $B = \frac{1}{S}\sum_{i=1}^{S} B_i$;

minimum number of changed bits: $B_{\min} = \min(B_i, i = 1, \ldots, S)$;

maximum number of changed bits: $B_{\max} = \max(B_i, i = 1, \ldots, S)$;

mean changed probability: $P = \frac{B}{h} \times 100\%$;

standard variance of B: $\Delta B = \sqrt{\frac{1}{S-1}\sum_{i=1}^{S}(B_i - B)^2}$;

standard variance of P: $\Delta P = \sqrt{\frac{1}{S-1}\sum_{i=1}^{S}(P_i - P)^2} \times 100\%$,

where S denotes the total number of tests, h the length of the hash value, and B_i the hamming distance for changing one bit of the message, with $B_{\max}$ and $B_{\min}$ being the maximal and minimal hamming distance.

We choose a paragraph of a text message (about 10 Kbytes), and set $S = 256$, 512, 1024, 2048, 10000. The simulation results are listed in Table 1 for $h = 128$ and Table 2 for $h = 160$, respectively.

Table 1. The statistical performance of the proposed hash function when the hash length is 128 bits.

	$S = 256$	$S = 512$	$S = 1024$	$S = 2048$	$S = 10000$
B	64.03	63.70	63.71	64.01	64.00
$P(\%)$	50.02	49.76	49.78	50.01	50.00
ΔB	5.49	5.71	5.87	5.69	5.67
$\Delta P(\%)$	4.29	4.47	4.59	4.44	4.43
$B_{\max}$	78	83	88	83	85
$B_{\min}$	50	45	45	45	45

Table 2. The statistical performance of the proposed hash function when the hash length is 160 bits.

	$S = 256$	$S = 512$	$S = 1024$	$S = 2048$	$S = 10000$
B	79.88	80.03	79.90	79.93	79.93
$P(\%)$	49.92	50.02	49.94	49.96	49.96
ΔB	5.71	6.14	6.30	6.10	6.32
$\Delta P(\%)$	3.57	3.84	3.93	3.81	3.95
$B_{\max}$	99	96	101	100	104
$B_{\min}$	60	57	59	61	58

According to the analyses of the data in Table 1 and Table 2, we can draw the conclusion that the mean changed bit number B and the mean changed probability P are very close to the ideal values of $h/2$ and 50%, respectively. Furthermore, ΔB and ΔP are both very small, indicating that the capabilities for diffusion and confusion are very stable. The results show that the hash function is secure against statistic attacks.

The corresponding distribution of changed probability is also plotted in Fig. 3, with $S = 2048$, $h = 160$.

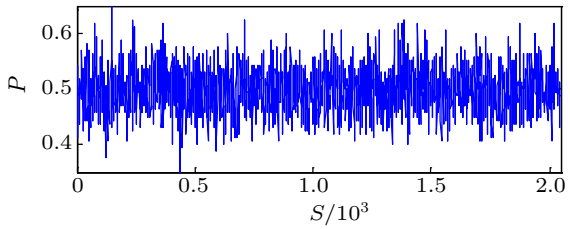


Fig. 3. Distribution of changed probability.

4.3. Collision test

First, we choose a message with a length of 10K bytes and produce the hash value in ASCII format using the proposed algorithm. Then a bit is randomly chosen in the message and negated, and a new hash value is generated for the new message and stored in ASCII format. Two hash values are compared and the number of ASCII characters with the same value at the same locations is counted by

$$\omega = \sum_{i=1}^{S} f(t(e_i), t(e_i')), \text{ where } f(x,y) = \begin{cases} 1, x = y, \\ 0, x \neq y, \end{cases} \quad (12)$$

where e_i and e_i' are the i-th entry of the original and the new hash value, respectively, and function $t(\cdot)$ converts the entries into their equivalent decimal value. This kind of collision test is performed S times.

The absolute difference between two hash values is calculated in S-independent collision tests by

$$d = \sum_{i=1}^{S} |t(e_i) - t(e_i')|. \quad (13)$$

Obviously, smaller ω or bigger d means stronger collision resistance.

Under the assumption of uniform and random distributions of hash values, the theoretical number of ω with different values during S-independent tests, denoted by $W_N(\omega)$, is then given by

$$\begin{aligned} W_S(\omega) &= S \times \text{prob}\{\omega\} \\ &= N \frac{h!}{\omega!(h-\omega)!} \left(\frac{1}{2^\kappa}\right)^\omega \left(1 - \frac{1}{2^\kappa}\right)^{h-\omega}, \end{aligned} \quad (14)$$

where κ is the number of bits in a byte.

The above collision test is performed $S = 10^4$ times using the proposed hash function, and the experimental maximum, minimum, and mean of d are listed in Table 3. The theoretical values and the statistical results of the $W_N(\omega)$ experiments are shown in Table 4.

Tables 3 and 4 show that the proposed algorithm possesses a stronger collision resistance capability than MD5, SHA1.

Table 3. Absolute differences between two hash values.

Absolute difference d	Minimum	Maximum	Mean
MD5	590	2074	1304
SHA-1	795	2730	1603
Ref. [12] (128-bit)	689	2295	1526
Ref. [12] (160-bit)	822	2745	1763
Ref. [11] (128-bit)	744	2431	1371
Ref. [10] (128-bits)	609	2329	1365
Ref. [10] (160-bits)	866	2731	1706
Ref. [9] (128-bits)	788	2273	1492
Ref. [8] (128-bits)	655	2064	1367
Ref. [20] (128-bits)	796	2418	1599
Our algorithm (128-bits)	562	2024	1242
Our algorithm (160-bits)	914	2827	1800

Table 4. Comparisons among the values of $W_N(\omega)$ obtained at different values of ω.

$W_N(\omega)$	$\omega = 0$	$\omega = 1$	$\omega = 2$	$\omega = 3$
Theoretical values (128-bits)	9393	589	17.33	0.32
Theoretical values (160-bits)	9247	725.26	27.02	0.64
Ref. [12] (128-bit)	9377	607	16	0
Ref. [11] (128-bit)	9434	533	13	0
Ref. [10] (128-bits)	9415	571	14	0
Ref. [9] (128-bits)	9381	605	14	0
Ref. [8] (128-bits)	9359	626	15	0
Our algorithm (128-bits) ($q = 4$)	9393	591	16	0
Our algorithm (160-bits) ($q = 4$)	9242	732	26	0

4.4. Further discussion

In comparison with traditional hash algorithms, such as MD5 with fixed 128-bit and SHA-1 with fixed 160-bit lengths, the proposed algorithm can produce a hash value with $n \times 32$ bits ($n \geq 1$). For a longer hash value, the computation time of the proposed algorithm will not increase so much. Furthermore, the matrix computation in WCDN can be easily designed to work in parallel to reduce the computation time.

When a chaotic map is realized using finite computation precision, the digital dynamical properties will be very different from those in the continuous domain. However, many research has shown that digital realizations using floating-point arithmetic are suitable approximations.[9,12,13]

Since the IEEE 754 floating-point standard was available in virtually all computers produced in 1980,[40] two hash values of a message with the same parameters and algorithms generated by two computers will be the same if the IEEE 754 floating-point standard is implemented in both computers, in spite of the different programming languages and platforms. In the proposed algorithm, the IEEE 754 double precision floating-point format possessing a 64-bit word length with a 52-bit fraction is adopted.

Furthermore, the fixed-point arithmetic can be adopted in the algorithm. In this case, assuming a finite precision of P-bit, equation (3) can be modified as follows:

$$\begin{aligned} \boldsymbol{x}_{(k,t+1)} &= |\boldsymbol{f}(\hat{\boldsymbol{x}}_{(k,t)}) + \boldsymbol{E}_k \hat{\boldsymbol{x}}_{(k,t)}| \bmod 1, \\ \hat{\boldsymbol{x}}_{(k,t)} &= g(\boldsymbol{x}_{(k,t)} 2^P).2^{-P}, \end{aligned} \quad (15)$$

where, $g(\cdot)$ is an approximation function, which may be the ceil, floor or round functions.

5. Conclusions

In this paper, we construct a weighted complex dynamical network from a message block and a hash function from the generated weighted complex dynamical network for the first time. Through chaotic iterations, quantization, and exclusive-or operations, the scheme can construct a fixed-length hash value. The analysis and simulation show that the scheme is extremely sensitive to the initial values and coupling matrix, and

also has good properties such as excellent one-way, confusion and diffusion, strong collision resistance, and good flexibility. It can be easily applied to data signature or authentication, and other areas.

References

[1] Alvarez G and Li S 2006 *Int. J. Bifur. Chaos* **16** 2129
[2] Kocarev L and Jakimoski G 2003 *IEEE Trans. Circ. Sys.* I: *Regular Papers* **50** 123
[3] Chen F, Liao X, Xiang T and Zheng H 2011 *Inf. Sci.* **181** 5110
[4] Liao X, Chen F and Wong K W 2010 *IEEE Trans. Comput.* **59** 1392
[5] Chen F, Wong K W, Liao X and Xiang T 2012 *IEEE Trans. Inf. Theor.* **58** 445
[6] Li P, Li Z, Halang W A and Chen G 2006 *Phys. Lett. A* **349** 467
[7] Wang S H and Shan P Y 2011 *Chin. Phys. B* **20** 090504
[8] Wang Y, Wong K W and Xiao D 2011 *Commun. Nonlinear Sci. Numer. Simul.* **16** 2810
[9] Yang H, Wong K W, Liao X, Wang Y and Yang D 2009 *Chaos, Solitons and Fractals* **41** 2566
[10] Akhshani A, Behnia S, Akhavan A, Jafarizadeh M A, Abu Hassan H and Hassan Z 2009 *Chaos, Solitons and Fractals* **42** 2405
[11] Akhavan A, Samsudin A and Akhshani A 2009 *Chaos, Solitons and Fractals* **42** 1046
[12] Wang Y, Liao X, Xiao D and Wong K W 2008 *Inf. Sci.* **178** 1391
[13] Zhang J, Wang X and Zhang W 2007 *Phys. Lett. A* **362** 439
[14] Wang S and Hu G 2007 *Chaos* **17** 023119
[15] Zhang H, Wang X F, Li Z H and Liu D H 2005 *Acta Phys. Sin.* **54** 4006 (in Chinese)
[16] Yi X 2005 *IEEE Trans. Circ. Sys.* II **52** 354
[17] Xiao D, Liao X and Deng S 2005 *Chaos, Solitons and Fractals* **24** 65
[18] Peng F, Qiu S S and Long M 2005 *Acta Phys. Sin.* **54** 4562 (in Chinese)
[19] Wong K W 2003 *Phys. Lett. A* **307** 292
[20] Luo Y L and Du M H 2012 *Chin. Phys. B* **21** 060503
[21] Zheng F, Tian X J, Li X Y and Wu B 2008 *Chin. Phys. B* **17** 1685
[22] Li S, Chen G and Mou X 2005 *Int. J. Bifur. Chaos* **15** 3119
[23] Wang J Z, Wang Y L and Wang M Q 2006 *Acta Phys. Sin.* **55** 5048 (in Chinese)
[24] Barabási A L and Albert R 1999 *Science* **286** 509
[25] Watts D J and Strogatz S H 1998 *Nature* **393** 409
[26] Wang X F and Chen G 2003 *IEEE Circ. Sys. Mag.* **3** 6
[27] Yang C L and Tang K S 2011 *Chin. Phys. B* **20** 128901
[28] Yu W, Cao J, Chen G, Lu J, Han J and Wei W 2009 *IEEE Trans. Sys., Man, Cyb.,* Part B: *Cybernetics* **39** 230
[29] Lu J and Chen G 2005 *IEEE Trans. Autom. Control* **50** 841
[30] Lu J, Yu X, Chen G and Cheng D 2004 *IEEE Trans. Circ. Sys.* I: *Regular Papers* **51** 787
[31] Trpevski D, Tang W K S and Kocarev L 2010 *Phys. Rev. E* **81** 056102
[32] Nekovee M, Moreno Y, Bianconi G and Marsili M 2007 *Physica A: Statistical Mechanics and Its Applications* **374** 457
[33] Lü L, Chen D B and Zhou T 2011 *New J. Phys.* **13** 123005
[34] Ash J, Newth D 2007 *Physica A: Statistical Mechanics and Its Applications* **380** 673
[35] Song Y R, Jiang G P and Xu J G 2011 *Acta Phys. Sin.* **60** 120509 (in Chinese)
[36] Song Y R and Jiang G P 2009 *Acta Phys. Sin.* **58** 5911 (in Chinese)
[37] Latora V and Marchiori M 2001 *Phys. Rev. Lett.* **87** 198701
[38] García P, Parravano A, Cosenza M G, Jiménez J and Marcano A 2002 *Phys. Rev. E* **65** 045201
[39] Li X and Chen G 2003 *IEEE Trans. Circ. Sys. I*: *Fundamental Theory and Applications* **50** 1381
[40] Goldberg D 1991 *ACM Computing Surveys (CSUR)* **23** 5