Vol 17 No 7, July 2008
1674-1056/2008/17(07)/2388-06

Chinese Physics B

© 2008 Chin. Phys. Soc.
and IOP Publishing Ltd

One-way hash function based on hyper-chaotic cellular neural network*

Yang Qun-Ting(杨群亭) and Gao Tie-Gang(高铁杠)†

College of Software, Nankai University, Tianjin 300071, *China*

(Received 22 November 2007; revised manuscript received 31 December 2007)

The design of an efficient one-way hash function with good performance is a hot spot in modern cryptography researches. In this paper, a hash function construction method based on cell neural network with hyper-chaos characteristics is proposed. First, the chaos sequence is gotten by iterating cellular neural network with Runge–Kutta algorithm, and then the chaos sequence is iterated with the message. The hash code is obtained through the corresponding transform of the latter chaos sequence. Simulation and analysis demonstrate that the new method has the merit of convenience, high sensitivity to initial values, good hash performance, especially the strong stability.

Keywords: one-way hash function, hyper-chaos, cellular neural network, Runge–Kutta formula
PACC: 0545

1. Introduction

As the core of cryptography, hash is the basic technique for information security. The conventional hash function, such as MD5, SHA, is realized through the complicated method based on logical XOR operation or multi-round iteration of some available cipher. Each step of the former is simple, but the process round will also be huge even the message is very short, while the latter needs complicated computation because of its relying on the basic cipher. It was reported that Wang had deciphered widely by using hash function such as MD5 or SHA-1.[1] Thus, in order to meet practical application, it is desirable to study new hash generation algorithm.

Chaos is a kind of deterministic random-like process generated by nonlinear dynamic systems. The properties of chaos includes: sensitivity to tiny changes in initial conditions, random-like behaviour, ergodicity, unstable periodic orbits, desired diffusion and confusion properties, and one-way property. Neural network's confusion and diffusion properties have been used to design encryption algorithms, such as the stream ciphers[2,3] or block ciphers.[4] What is more, neural network has also the one-way property and is easy to design in circuit. The inherent merits of chaos and neural network form the theoretical foundation for hash function construction. The chaos[5−7] and neural network[8] have been used in the construction of one-way hash function. Recently, Liu *et al*[5] represented a hash scheme based on Heñon iteration; Yi[9] proposed a way to construct chaotic-maps-based hash function; Xiao *et al*[7] introduced a way to construct the Chebyshev chaos-based hash function; Wang and Zhang[10] presented a construction algorithm based on mixed chaotic dynamic systems; Peng *et al*[6] proposed a hash function construction method based on two-dimensional hyper-chaotic mapping; Lian *et al*[8] proposed a secure hash function based on the neural network.

The one-way hash function construction algorithms above are all based on chaos or neural network. However, the chaos has defects such as that the chaotic sequence generated by chaos may degenerate cycle sequence, while the neural network model may be too simple to defend attacks. Thus, the combination of them may generate a more complicated and secure algorithm. This paper presents a novel one-way hash function construction scheme based on a cell neural network with hyper-chaos characteristics. Simulations show that the proposed algorithm represents strong stability of diffusion and confusion in comparison with the algorithm[6,7] above. It improves the performance of chaos-based hash function with comparative simplicity.

*Project supported by Key Program of Natural Science Fund of Tianjin of China (Grant No 07JCZDJC06600).
†Corresponding author. E-mail: gaotiegang@nankai.edu.cn

2. One-way hash function construction based on chaotic neural network

2.1. The chaotic map model

In the proposed algorithm, hyper-chaotic cellular neural network (HCCNN) used in this paper is modelled by the following system:

$$\begin{aligned} x_i &= -c_i + \sum_{j=1}^{4} w_{ij} v_j, \\ v_i &= f_i(x_i), \quad i = 1, 2, 3, 4, \cdots, n, \end{aligned} \tag{1}$$

where $f_i(x_i) = \dfrac{1}{2}(|x_i + 1| - |x_i - 1|)$, $W = (w_{ij})$ is a 4×4 matrix and $w_{ij} = 0$ for $|i - j| > 1$.

It has been shown that when matrix W is

$$W = \begin{pmatrix} 2.1 & 2.5 & 0 & 0 \\ -2.6 & 1 & 3 & 0 \\ 0 & -2.8 & P & -1.1 \\ 0 & 0 & 100 & 160 \end{pmatrix}, \tag{2}$$

the HCCNN (1) exhibits hyper-chaotic dynamics with two positive Lyapunov exponents with the parameter $0.27 < P < 0.67$, for example, when $P = 0.4$, the four Lyapunov exponents are $\lambda_1 = 0.125$, $\lambda_2 = 0.022$, $\lambda_3 = 0$ and $\lambda_4 = -95.95$, respectively. The hyper-chaos attractors are shown in Fig.1. For more detailed analysis of the complex dynamics of the system, see Ref.[11].

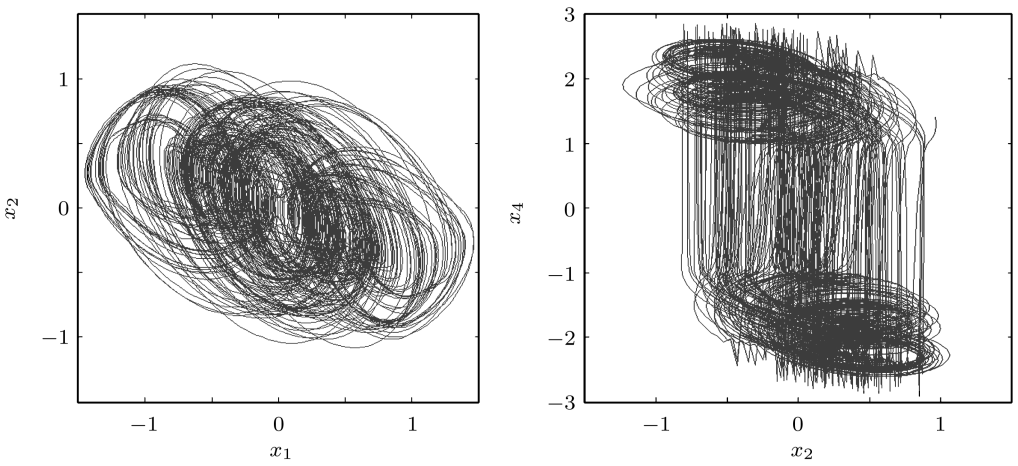


Fig.1. Phase portraits of the HCCNN.

2.2. The description of one-way hash function construction algorithm

The principle of the construction algorithm is as follows: select a sequence A_c of 128 bits and split it into four groups which are further mapped into the four initial parameters $x_1(0)$, $x_2(0)$, $x_3(0)$ and $x_4(0)$, and then use Runge–Kutta formula to iterate HCCNN for N_0 times (to avoid the harmful effect of transitional procedure, N_0 is a constant integer more then 300). To make all of the contents of multimedia participate in the process of hash generation, some iteration of HCCNN are needed, and the times of iteration depend on the length of multimedia dada, the detailed procedures are as follows:

Step 1 Select a key sequence A_c of 128 bits and split it into four groups, which are further mapped into the four initial parameters $x_1(0)$, $x_2(0)$, $x_3(0)$ and $x_4(0)$.

Step 2 Iterate CNN by using Runge–Kutta algorithm for N_0 times to avoid the harmful effect of transitional procedure, thus $x_1(N_0)$, $x_2(N_0)$, $x_3(N_0)$ and $x_4(N_0)$ are gotten.

Step 3 Then CNN is iterated continuously for $\dfrac{L}{4}$ times, where L is the length of multimedia data. Iteration formula can be described as

$$\begin{aligned} x_1(N_0 + i) = {} & g_1(x_1(N_0 + i - 1) + B_{4\times(i-1)+1} \times 10^{-5}, x_2(N_0 + i - 1) + B_{4\times(i-1)+2} \times 10^{-5}, \\ & x_3(N_0 + i - 1) + B_{4\times(i-1)+3} \times 10^{-5}, x_4(N_0 + i - 1) + B_{4\times(i-1)+4} \times 10^{-5}), \end{aligned}$$

$$
\begin{aligned}
x_2(N_0+i) &= g_2(x_1(N_0+i-1)+B_{4\times(i-1)+1}\times 10^{-5}, x_2(N_0+i-1)+B_{4\times(i-1)+2}\times 10^{-5},\\
&\quad x_3(N_0+i-1)+B_{4\times(i-1)+3}\times 10^{-5}, x_4(N_0+i-1)+B_{4\times(i-1)+4}\times 10^{-5}),\\
x_3(N_0+i) &= g_3(x_1(N_0+i-1)+B_{4\times(i-1)+1}\times 10^{-5}, x_2(N_0+i-1)+B_{4\times(i-1)+2}\times 10^{-5},\\
&\quad x_3(N_0+i-1)+B_{4\times(i-1)+3}\times 10^{-5}, x_4(N_0+i-1)+B_{4\times(i-1)+4}\times 10^{-5}),\\
x_4(N_0+i) &= g_4(x_1(N_0+i-1)+B_{4\times(i-1)+1}\times 10^{-5}, x_2(N_0+i-1)+B_{4\times(i-1)+2}\times 10^{-5},\\
&\quad x_3(N_0+i-1)+B_{4\times(i-1)+3}\times 10^{-5}, x_4(N_0+i-1)+B_{4\times(i-1)+4}\times 10^{-5}),
\end{aligned}
\tag{3}
$$

where $i = 1, 2, \cdots, L/4$, $g_i(\bullet)$, $i = 1, 2, 3, 4$ stands for Runge–Kutta iteration functions of x_i, $i = 1, 2, 3, 4$. After finishing the iteration, four decimal fractions x_1, x_2, x_3, x_4 will be generated.

Step 4 These decimal values are preprocessed firstly as follows:

$$
\begin{aligned}
&x_i = \mathrm{de2bi}((\mathrm{Abs}(x_i) - \mathrm{Floor}(\mathrm{Abs}(x_i))) \times 10^{14}),\\
&i = 1, 2, 3, 4,
\end{aligned}
\tag{4}
$$

where, $\mathrm{Abs}(x)$ returns the absolute value of x. $\mathrm{Floor}(x)$ returns the value of x to the nearest integer less than or equal to x, and then converts them to binary value by using function $\mathrm{de2bi}(x)$.

Step 5 Extract 32-bit data from x_i and the binary sequences to perform the operation of XOR with A_c. Then 128-bits hash sequence will be generated.

3. Performance analysis and test

3.1. Hash results of message

We use the proposed algorithm to generate hash information on the following message under four kinds of condition.

The original message is '*The 21 leaders of the Asia Pacific Economic Cooperation forum ended their annual two-day summit in Australia with calls to bring to a quick resolution stalled global trade talks. The declaration calls on APEC members to set voluntary, nonbinding targets to cut greenhouse missions, thought to be a major contributor to global warming.*'

Condition 1 Change the first capital character 'T' in the original message into 'A'.

Condition 2 Change the word 'Asia' in the original message into 'Esia'.

Condition 3 Change the number '21' in the original message into '20'.

Condition 4 Change last blank space into '–'.

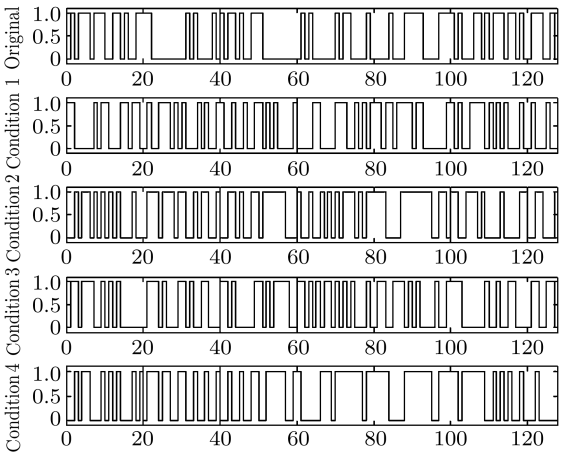


Fig.2. Hash values under different conditions.

The corresponding hash values in hexadecimal format are:

Original: 5DCD3C0142B4E00502C208F87A35DA72.
Condition 1: C16364EA29926A10C395DBD81A7AA244.
Condition 2: 2D5447765934DF892AEBE1FE6CE84331.
Condition 3: 6E54074698D06BCD5AA267509E049866.
Condition 4: 2C5457664924CF983BFBF0FE7DF95220.

The simulation results above indicate that the one-way property of the proposed algorithm is so perfect that any difference of the message will cause huge changes in the final hash value. The corresponding graphical display of binary sequences is shown in Fig.2.

3.2. Statistical analysis of diffusion and confusion

In order to hide message redundancy, Shannon introduced diffusion and confusion.[12] Diffusion means spreading out of the influence of a single plaintext bit so as to hide the statistical structure of the plaintext. Confusion means the use of transformations that complicate dependence of the statistics of ciphertext on the statistics of plaintext. They are two general principles to guide the design of practical cipher, including hash function. For the hash value in binary format, each bit is only 1 or 0. So the ideal diffusion effect should be that any tiny changes in initial conditions lead to the 50% changing probability of each bit.

We have performed the following diffusion and confusion test. A paragraph of message is randomly chosen and hash value is generated, then a bit in the message is randomly selected and toggled and a new hash value is generated. Two hash values are compared with each other and the number of changed bit is counted as B_i. This kind of test is performed N (such as 128, 256, 512, 1024) times. The corresponding distribution of changed bit number, where $N = 1024$, is shown in Fig.3.

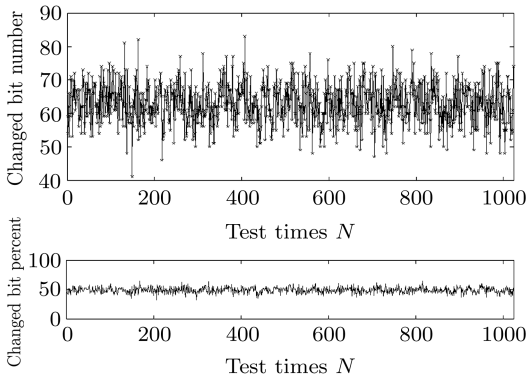


Fig.3. Distribution of changed bit number and percent.

Ideally, the changed bit number corresponding to 1 bit changed message should concentrate around 64. After simulating 1024 times, we can find from Fig.3 that the maximum and minimum changed bit number is 85 and 42 respectively. There are 912 numbers between 55 and 75, accounting for 89.06% nearly. There are 17 numbers not between 50 and 80, only accounting for 1.7% of the total. The under subplot shows that the actual changed bit number is round 50%. The statistical values above indicate that the algorithm proposed has very strong capability for diffusion and confusion.

Usually, four statistics are defined as follows:

$$\bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i, \tag{5}$$

$$P = (\bar{B}/128) \times 100\%, \tag{6}$$

$$\Delta B = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(B_i - \bar{B})^2}, \tag{7}$$

$$\Delta P = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(B_i/128 - P)^2} \times 100, \tag{8}$$

where, N is total statistic number, B_i is changed bit number when the ith simulation is performed, $\bar{B}$ and P represent mean changed bit number and mean changed probability respectively, ΔB and ΔP indicate the stability of diffusion and confusion. Through the tests with $N = 128, 256, 512, 1024$, respectively, the corresponding data are listed in Table 1.

Table 1. Number of changed bit B_i and corresponding index.

N=	128	256	512	1024	Even
$\bar{B}$	62.27	63.91	63.20	63.53	63.23
P/%	48.65	49.93	49.37	49.66	49.67
ΔB	0.59	0.42	0.30	0.21	0.38
ΔP/%	4.33	3.13	2.19	1.54	2.41

Table 1 indicates that the even changed bit number and changed percent is 63.23 and 49.67% respectively, which are very close to the ideal value 64 and 50%. While ΔB and ΔP, indicating the stability of diffusion and confusion, is very little, which represent that the capability for diffusion and confusion is stable.

3.3. Analysis of collision resistance

Collision resistance means that the hash results are identical to different random initial input. In order to investigate the collision resistance capability of the hashing approach, we have performed two collision tests.[13]

In the first experiment, 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. Two 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. The absolute difference of the two hash result is calculated by using the following formula:

$$d = \sum_{i=1}^{N} |t(e_i) - t(e_i')|. \tag{9}$$

Where e_i and e_i' are the ith ASCII character of the original and the new hash value, respectively, $t()$ converts the entries to their equivalent decimal values. This kind of collision test is performed 10000 times with $x_1(0) = 0.1234$, $x_2(0) = 0.5678$, $x_3(0) = 0.9012$ and $x_4(0) = 0.3455$. The maximum, minimum, mean values of d are listed in Table 2. A plot of the distribution of the number of ASCII characters with the same value at same location is given in Fig.4.

Table 2. Absolute differences of two hash values.

absolute difference d	maximum	minimum	mean
experiment	2343	256	1314

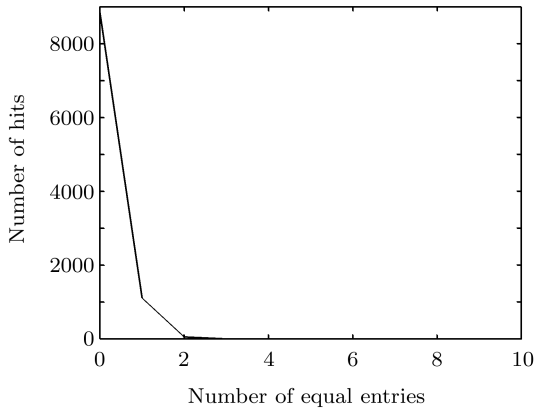


Fig.4. Distribution of the number of equal entries with the same value at the same location in the hash value.

In the second experiment, the hash value for a paragraph of message randomly chosen is generated and stored in ASCII format similarly. What is focused in this experiment is the possibility of colliding between every two hash results, thus every two hash results should be compared. The simulation is performed 10000 times equally in the same initial condition. The plot of the distribution of the number of ASCII characters with the same value at same location is given in Fig.5.

Notice that the maximum number of equal entries in the Figs.4 and 5 are both 3. Besides, most of the entries are different in ASCII format. It shows that the hash algorithm proposed possesses a strong collision resistance capability.

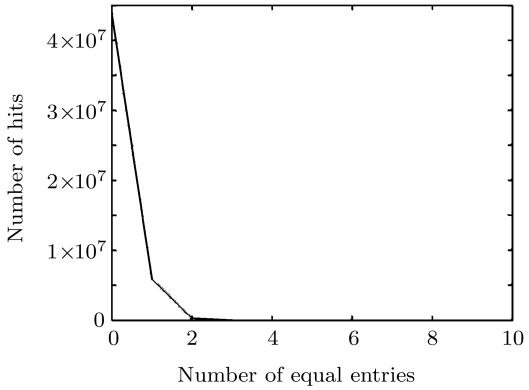


Fig.5. Distribution of the number of equal entries with the same value at the same location in the hash value.

3.4. Security of key

In the algorithm proposed, $x_i(0)$ are chosen as the secret keys, where $i = 1, 2, 3$ and 4 and $x_i(0) \in [0, 1]$ are decimal fractions. The key space is huge enough to resist any exhaustive key search.

3.5. Analysis of speed and flexibility

The executive speed of the algorithm proposed is proportional to the length of the plaintext nearly. The total iterative times is $N_0 + L/4$, while N_0 is the initial iterative times, L indicates the length of the plaintext. So the iterative time required is very little when the plaintext is short.

Besides, the length of the hash value could be changed easily through modifying the way to process $x_1(0)$, $x_2(0)$, $x_3(0)$ and $x_4(0)$. For example, 256 bits hash code could be constructed by concatenate the hash result of $N_0 - 1$ and N_0 steps.

4. Conclusion

In this paper, a hash function construction method based on cell neural network with hyper-chaos characteristics is proposed and analysed. The hash function adopts the inherent merits of chaos and neural network, such as diffusion property and confusion property. The stimulation indicates that the algorithm proposed satisfies all of the performance requirements of the hash function, especially the stability.

References

[1] Wang X Y and Yu H B 2005 *Lecture Notes in Computer Science* Vol. 3494 p19

[2] Chan C K and Cheng L M 2001 *IEEE Trans. Neural Networks* **12** 340

[3] Karras D A and Xorkadis V 2003 *Neural Networks* **16** 899

[4] Yee L P and Silva L C 2002 *Proceedings of the 2002 International Joint Conference on Neural Networks* **2** 1455

[5] Liu J N, Xie J C and Wang P 2000 *J. Tsinghua University (Sci. & Tech.)* **40** 55 (in Chinese)

[6] Peng F, Qiu S S and Long M 2005 *Chin. Phys. Soc.* **54** 4562 (in Chinese)

[7] Xiao D, Liao X F and Deng S J 2005 *Chaos, Solitons and Fractals* **24** 56

[8] Lian S G, Sun J S and Wang Z Q 2006 *Neurocomputing* **69** 2346

[9] Yi X 2005 *IEEE Trans. Circuits and Systems* II: *Express Briefs* **52** 354

[10] Wang X M and Zhang W F 2003 *Acta Phys. Sin.* **42** 2737 (in Chinese)

[11] Li Q D, Yang X and Yang F Y 2006 *Int. J. Bifur. Chaos* **16** 2453

[12] Shannon C E 1949 *Bell System Technical J.* **28** 656

[13] Kwok Wo W 2003 *Phys. Lett.* A **307** 292