Vol 17 No 10, October 2008
1674-1056/2008/17(10)/3588-08

Chinese Physics B

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

# Constructing a one-way hash function based on the unified chaotic system*

Long Min(龙 敏)[a)†], Peng Fei(彭 飞)[b)], and Chen Guan-Rong(陈关荣)[c)]

[a)] *College of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410076, China*

[b)] *College of Computer and Communication, Hunan University, Changsha 410082, China*

[c)] *Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China*

(Received 17 September 2007; revised manuscript received 27 May 2008)

A new one-way hash function based on the unified chaotic system is constructed. With different values of a key parameter, the unified chaotic system represents different chaotic systems, based on which the one-way hash function algorithm is constructed with three round operations and an initial vector on an input message. In each round operation, the parameters are processed by three different chaotic systems generated from the unified chaotic system. Feed-forwards are used at the end of each round operation and at the end of each element of the message processing. Meanwhile, in each round operation, parameter-exchanging operations are implemented. Then, the hash value of length 160 bits is obtained from the last six parameters. Simulation and analysis both demonstrate that the algorithm has great flexibility, satisfactory hash performance, weak collision property, and high security.

**Keywords:** Hash function, unified chaotic system, cryptography
**PACC:** 0545

## 1. Introduction

Hash functions are most commonly used in cryptography.[1] They play an important role in protecting the authenticity of messages that are sent over an open channel, including such as data integrity and data-origin authentication.[2] They are used to compress inputs of arbitrary lengths into a result with a fixed number of output bits, for the hash result provides a unique imprint of a message and protecting a short imprint is easier than protecting the message itself.[3]

Differing from the conventional cryptography, chaos-based cryptography uses some chaotic systems to develop hash function algorithms.[4−10] The basic idea is to take advantage of the extreme complexity of the chaotic systems. In this endeavour, however, it has been observed that not all existing chaos-based hash function algorithms are as good as expected. The one suggested in Ref.[9] for example, although efficient for its linear characteristic, may be vulnerable to the collision attack. On the contrary, the one developed in Ref.[10] is based on a block cipher, for which the iteration time is long (at least 75 for each block) and the operational length $l$ of the binary fractions is also long (if the length of the hash value is 160 bit, then $l$=80), hence it is not very efficient. In the attempt of improving such defects, a new one-way hash function is designed in this paper, based on a now-well-known unified chaotic system.[11] Theoretical analysis and computer simulation both demonstrate that the new algorithm has sufficient flexibility, satisfactory hash performance, weak collision property, and high security, therefore has good potential as a candidate for practical applications such as e-commerce.

The rest of the paper is organized as follows. Section 2 describes the new one-way hash function algorithm. Section 3 shows its performance with security analysis, and the last section concludes the investigation.

## 2. Description of the new one-way hash function algorithm

### 2.1. The unified chaotic system

The unified chaotic system considered here is described as[12]

*Project supported by the Natural Science Foundation of Hunan Province, China (Grant No 08JJ4020).

†E-mail: Longm@tom.com

$$D_\lambda(x_1,x_2,x_3):\begin{cases}\dot{x}_1=(25\gamma+10)(x_2-x_1),\\ \dot{x}_2=(28-35\gamma)x_1-x_1x_3+(29\gamma-1)x_2,\\ \dot{x}_3=x_1x_2-(\lambda+8)x_3/3,\end{cases}\tag{1}$$

where $\gamma\in[0,1]$. While $\gamma$ is changed continuously in [0, 1], system (1) is chaotic. In particular, when $\gamma\in[0,0.8)$, system (1) is the Lorenz system (Fig.1(a)), with $\gamma=0.8$, it is the Lu system (Fig.1(c)), while $\gamma\in[0.8,1)$ corresponds to the Chen system (Fig.1(b)).

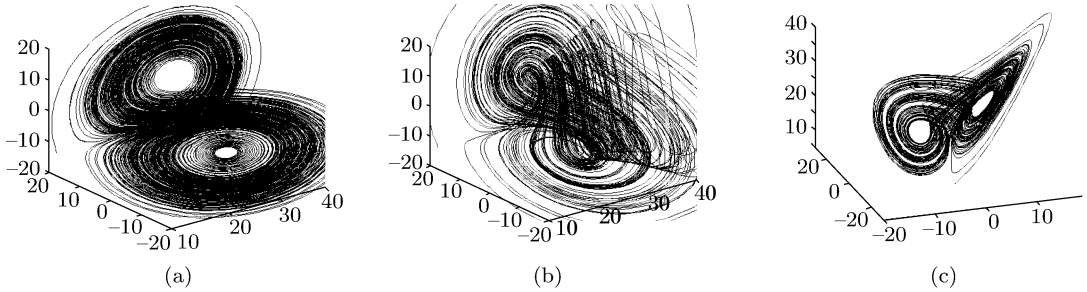



**Fig.1.** Attractors of three chaotic systems: (a) Lorenz attractor; (b) Chen attractor; (c) Lu attractor.

## 2.2. The new one-way hash function algorithm

Given a message $P$ of length $N$, and an initial vector (IV) $x$ ($0<x<1$), each character of $P$ is transformed into a real number in (0, 1), thereby generating

$$\boldsymbol{M}=[M_1,M_2,...,M_i,...,M_N],\tag{2}$$

where $\boldsymbol{M}$ denotes a set of values transformed from $P$ ($1\le i\le N$). Then, the new one-way hash function algorithm works as follows:

1) Iterate the logistic map several times

$$x(n+1)=4x(n)(1-x(n)),\tag{3}$$

with IV as the initial value, to obtain an array of values. Randomly choose six of these values and store them in six registers, denoted as $x_1$, $x_2$, $x_3$, $\gamma_1$, $\gamma_2$ and $\gamma_3$.

2) From $\boldsymbol{M}$, generate $\boldsymbol{M}1$ and $\boldsymbol{M}2$ in the following way:

$$\boldsymbol{M}1=[M_N,M_{N-1},...M_i...M_1],\tag{4}$$

$$\boldsymbol{M}2=[M_{\lfloor N/2\rfloor},M_{\lfloor N/2\rfloor-1},M_{\lfloor N/2\rfloor+1},...,M_{\lfloor N/2\rfloor-i},M_{\lfloor N/2\rfloor+i},...],\tag{5}$$

where $\lfloor a\rfloor$ is the floor integer of $a$, and $\boldsymbol{M}$, $\boldsymbol{M}1$ and $\boldsymbol{M}2$ have the same size.

3) Perform three round operations on each element of $\boldsymbol{M}$, $\boldsymbol{M}1$ and $\boldsymbol{M}2$, as shown in Fig.2.

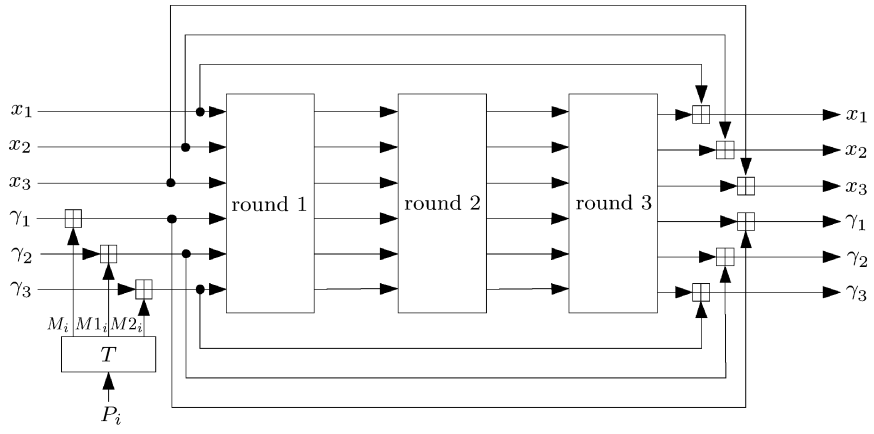



**Fig.2.** Operation on each element in the hash function.

Each single element of a message is processed as follows:

a) The values of registers$\gamma_1$, $\gamma_2$ and $\gamma_3$ are updated by operation $\boxplus$ between the registers and $M(i)$, $M1(i)$ and $M2(i)$:

$$\gamma_k = \gamma_k \boxplus A, \tag{6}$$

where $k$=1, 2, 3, $A = M$, $M1$, $M2$, and $a \boxplus b = 0.5 \cdot (a + b) \bmod 1$.

b) Each round operation includes two parallel parts. In the first part, $x_1$, $x_2$ and $x_3$ are used as initial values, while the parameter of the unified chaotic system is denominated by $\gamma_1$, $\gamma_2$, and $\gamma_3$, respectively. In the second part, $\gamma_1$, $\gamma_2$, and $\gamma_3$ are used as initial values, while the parameter of the unified chaotic system is denominated by $x_1$, $x_2$ and $x_3$, respectively. After $n_1$ times of iterations on each specified chaotic system, the values of the outputs are exchanged according to $x_1 = x_3$, $x_2 = x_1$, $x_3 = x_2$ or $y_1 = y_3$, $y_2 = y_1$, $y_3 = y_2$. After these iteration operations have been completed, the output values of the six registers will be performed a feed-forward operation with the original input values of the corresponding registers and their results will be used as the initial values of the next round operation. The operations on each round of iteration are shown in Fig.3.

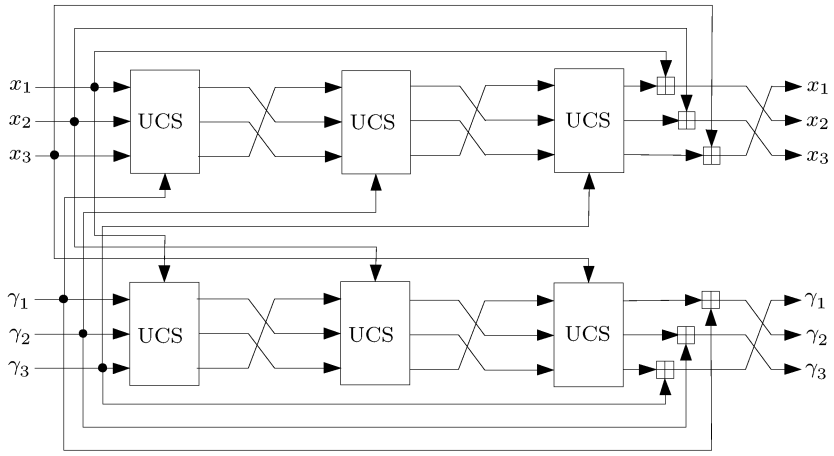



**Fig.3.** Iteration in each round based on the unified chaotic system.

c) After three rounds of operations, the output of the six registers will be performed a feed-forward operation with the original values of the corresponding registers, as shown in Fig.2.

d) The outputs of the six registers are taken as the inputs of the process for the next message element.

4) For 1 to $N$, repeat step 3 (a–d), until all elements of the message have been processed.

5) Transform the last values of $x_1$, $x_2$, $x_3$, $\gamma_1$, $\gamma_2$ and $\gamma_3$ in the corresponding binary format, then extract 27, 27, 26, 27, 27, 26 bits from them respectively, and finally concatenate them to form a 160-bit hash value.

# 3. Performance and security analyses

## 3.1. Hash results

The proposed algorithm has been evaluated for the following six cases:

C1: The original message is: 'The delegates on Monday began a three-day process to formally nominate Bush and Vice President Dick Cheney to a second term with a roll call full of home-state bragging and lots of praise for their beloved incumbents. Officials tried to keep secret which state would give Bush the 1,255 delegate votes to seal the nomination.'

C2: Change the first character $T$ in the original message into $t$.

C3: Change the word Cheney in the original message into ChEney.

C4: Change the full stop at the end of the original message into a comma.

C5: Add a blank space at the end of the original message.

C6: Change the initial condition from 0.562 to 0.561.

The corresponding hash values in hexadecimal are:
C1: 46C6FB394F43EBFDE6E112C7EB0DDB524C7BBCBD
C2: 6F4B067F98E2247645F0EE87516CB0D5D3AF7CE0
C3: A28419EB72CA40762BB4B65573FAB39CC6C0C8ED
C4: DAF96C28F14968CF241A34EFD039A1D612729EE2
C5: 8C0DD7B518250E09BAA742FF4D6183C94E707BF4
C6: C1AF41360E37FD166C1F1F35AA1A9FE9DD1A12D7

The graphical display of the above binary sequences is shown in Fig.4. It can be seen that the one-way property of the proposed hash function algorithm is fairly good, and a swift differentiation of the message or the initial vector causes some significant changes in the hash value.

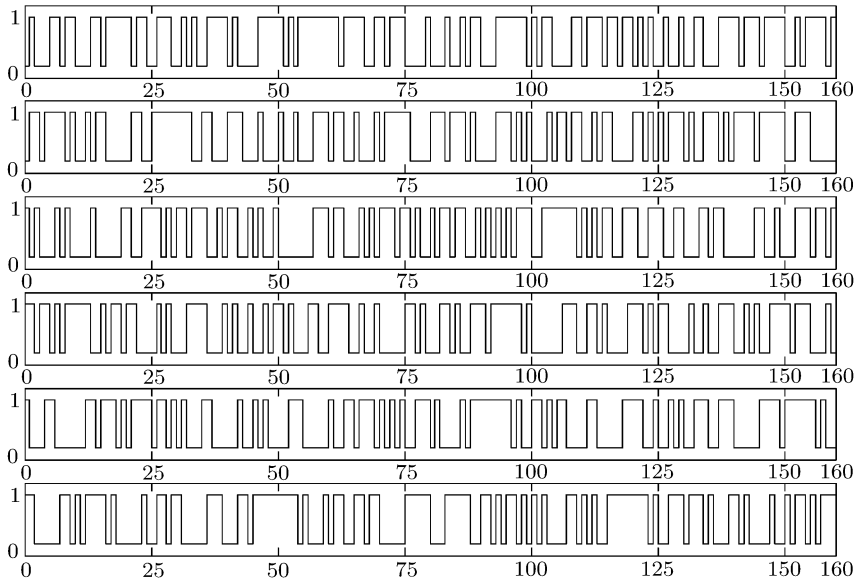



**Fig.4.** Hash values under different cases.

## 3.2. Statistic analysis

Shannon introduced the operations of diffusion and confusion for hiding the redundancy of a plaintext.[13] Diffusion and confusion are two important elements in the design of a modern cryptosystem, and so is a one-way hash function which requires the hash values be evenly distributed in the ciphertext space and be sensitive to the plaintext. The dependence between the plaintext and the hash value should be complicated enough. For a hash value in binary format, each bit is either 0 or 1, so the ideal diffusion effect should be that any tiny change in the initial condition results in about 50% change of each bit in probability.

Under the condition that the plaintext $P$ is changed by one bit each time, we observe the change of the corresponding bit number, $B$. Four statistics are defined as follows:

Mean changed bit number

$$\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i. \tag{7}$$

Mean changed bit number percentage

$$P = (\overline{B}/160) \times 100\%. \tag{8}$$

$B$'s unbiased variance

$$\Delta B = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(B_i - \overline{B})^2}. \tag{9}$$

$P$'s unbiased variance

$$\Delta P = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(B_i/160 - P)^2}. \tag{10}$$

Here, $N$ is the total statistic number and $B_i$ denotes the changed bit number in the $i$th test.

The test is performed as follows: A paragraph of message is randomly chosen and its corresponding hash value is generated. Then, a bit in the message is randomly selected and toggled, so a new hash value is obtained. These two hash values are compared and the number of changed bits is counted. The same process is performed for $N$ times. A corresponding distribution of changed bit numbers are shown in Fig.5, where $N$=512.

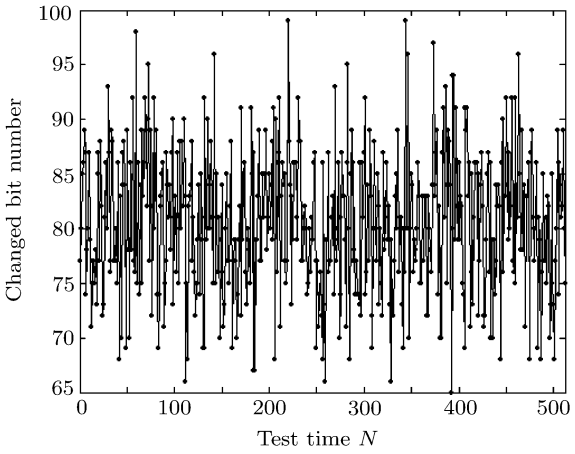



**Fig.5.** Distribution of changed bit number.

It can be seen from Fig.5 that the maximum changed bit number is 99 and the minimum changed bit number is 64. The percentage of changed bit number between 72–88 bit (with ±10% deviation from 80 bit) is about 86.57%. It shows a good diffusion effect of the proposed hash function algorithm.

Same tests on the algorithm with $N$=256, 512, 1024 and 2048 have also been performed. Under the condition that 1 bit is changed at each time, the corresponding values of $\overline{B}$, $\Delta B$, $P$ and $\Delta P$ are obtained as shown in Table 1. Apparently, every index is very near the ideal level, indicating the good performance of the proposed algorithm.

**Table 1.** Numbers of changed bit.

| $N$ | 256 | 512 | 1024 | 2048 | Mean |
|---|---|---|---|---|---|
| $\overline{B}$/bit | 80.625 | 80.5059 | 80.4141 | 79.7148 | 80.3150 |
| $\Delta B$/bit | 5.2695 | 5.2599 | 4.6581 | 4.4167 | 4.9011 |
| $P$/% | 50.39 | 50.32 | 50.26 | 49.82 | 50.20 |
| $\Delta P$/% | 5.64 | 5.65 | 5.47 | 5.34 | 5.53 |

## 3.3. Security analysis

As pointed out in Ref.[3], attacks to hash functions can be classified into generic attacks and shortcut attacks. The former can be applied to any algorithm and their complexity depends only on some generic parameters such as the size of the output space, while the latter can be used by any attacker who is able to find weakness of the algorithm so as to develop a specific attack on that particular algorithm and, in this case, the attack is more efficient than the former.

Assume that the output space of a hash function consists of all $n$-bit strings, that is, $R$=$\{0, 1\}^n$. Such an $n$-bit hash function is said to have the ideal security if

1. producing a preimage or a second preimage requires about $2^n$ operations;

2. producing a collision requires about $2^{n/2}$ operations.

These correspond to the complexity of the random preimaging and birthday attacks, respectively.

### 3.3.1. Analysis of the secure output length of the hash value

In Ref.[14], the cost of implementing a birthday attack to the popular MD5 hash function was discussed, which had an output length of $n$=128 bit (so, about $2^{64}$ operations are needed to produce a collision). The analysis, which dates back to 1995, leads to an estimate that collisions for MD5 can be found in 21 days using a customized 10 $M$\$ machine. Considering 'Moore's law', which states that the computing power available for a given cost doubles every 18 months, it is clear that an output length of $n$=128 bit is not sufficient for collision-resistance.

In the new algorithm developed in this paper, the output length $n$=160, thus it needs $2^{80}$ operations to produce a collision, implying that the output length of the proposed hash function algorithm is sufficient for collision-resistance; since if an attack requires $2^{75}$ operations, then it is considered impractical today.[15]

### 3.3.2. Analysis of preimage-attack resistance and collision-attack resistance

As described in Section 2.2, the hash result is extracted from the values of $x_1$, $x_2$, $x_3$, $\gamma_1$, $\gamma_2$ and $\gamma_3$. Most important of all, only part of each value was extracted to form the hash value, so the attacker cannot deduce the plaintext from the outputs. At the same time, there are three round operations in processing each element of the message, hence there are three different chaotic systems executing iteration operations in each round operation, which makes the algorithm fairly strong against attacks.

The whole process is as follows.

In each round, a round operation $R_i$ ($i \in \{1, 2, 3\}$) includes:

$$[x_1^{n_1}, x_2^{n_1}, x_3^{n_1}] = D_{\gamma_1}^{n_1}(x_1^0, x_2^0, x_3^0), \tag{11}$$

$$[x_1^{2n_1}, x_2^{2n_1}, x_3^{2n_1}] = D_{\gamma_2}^{n_1}(x_3^{n_1}, x_1^{n_1}, x_2^{n_1}), \tag{12}$$

$$[x_1^{3n_1}, x_2^{3n_1}, x_3^{3n_1}] = D_{\gamma_3}^{n_1}(x_3^{2n_1}, x_1^{2n_1}, x_2^{2n_1}), \tag{13}$$

$$[x_1^{R_1}, x_2^{R_1}, x_3^{R_1}] = [x_3^{3n_1} \boxplus x_3^0, x_2^{3n_1} \boxplus x_2^0, x_2^{3n_1} \boxplus x_2^0], \tag{14}$$

$$[x_1^0, x_2^0, x_3^0] = [x_3^{R_1}, x_1^{R_1}, x_2^{R_1}], \tag{15}$$

$$[\gamma_1^0, \gamma_2^0, \gamma_3^0] = [\gamma_1^p \boxplus M_i, \gamma_2^p \boxplus M1_i, \gamma_3^p \boxplus M2_i], \tag{16}$$

$$[\gamma_1^{n_1}, \gamma_2^{n_1}, \gamma_3^{n_1}] = D_{x_1}^{n_1}(\gamma_1^0, \gamma_2^0, \gamma_3^0), \tag{17}$$

$$[\gamma_1^{2n_1}, \gamma_2^{2n_1}, \gamma_3^{2n_1}] = D_{x_2}^{n_1}(\gamma_3^{n_1}, \gamma_1^{n_1}, \gamma_2^{n_1}), \tag{18}$$

$$[\gamma_1^{3n_1}, \gamma_2^{3n_1}, \gamma_3^{3n_1}] = D_{x_3}^{n_1}(\gamma_3^{2n_1}, \gamma_1^{2n_1}, \gamma_2^{2n_1}), \tag{19}$$

$$[\gamma_1^{R_1}, \gamma_2^{R_1}, \gamma_3^{R_1}] = [\gamma_3^{3n_1} \boxplus \gamma_3^0, \gamma_1^{3n_1} \boxplus \gamma_1^0, \gamma_2^{3n_1} \boxplus \gamma_2^0], \tag{20}$$

$$[\gamma_1^0, \gamma_2^0, \gamma_3^0] = [\gamma_3^{R_1}, \gamma_1^{R_1}, \gamma_2^{R_1}], \tag{21}$$

where $x_a^b (a \in \{1, 2, 3\}, b \geq 1)$ is the $b$th value of the register $x_a$, $\gamma_a^b$ ($a \in \{1, 2, 3\}$, $b \geq 1$) is the $b$th value of the register $\gamma_a$, $D_a^b$ ($m$, $n$, $l$) ($a \in \{x_1, x_2, x_3, \gamma_1, \gamma_2, \gamma_3\}$, $b \geq 1$) denotes the unified chaotic system with parameter $\gamma$ being denominated by the value of register $a$, and initial values are $m$, $n$, $l$, iterated for $b$ times. Thus,

$$[x_1^{R_1}, x_2^{R_1}, x_3^{R_1}, \gamma_1^{R_1}, \gamma_2^{R_1}, \gamma_3^{R_1}] = R_1(x_1^0, x_2^0, x_3^0, \gamma_1^0, \gamma_2^0, \gamma_2^0), \tag{22}$$

$$[x_1^{R_2}, x_2^{R_2}, x_3^{R_2}, \gamma_1^{R_2}, \gamma_2^{R_2}, \gamma_3^{R_2}] = R_2(x_1^{R_1}, x_2^{R_1}, x_3^{R_1}, \gamma_1^{R_1}, \gamma_2^{R_1}, \gamma_3^{R_1}), \tag{23}$$

$$[x_1^{R_3}, x_2^{R_3}, x_3^{R_3}, \gamma_1^{R_3}, \gamma_2^{R_3}, \gamma_3^{R_3}] = R_3(x_1^{R_2}, x_2^{R_2}, x_3^{R_2}, \gamma_1^{R_2}, \gamma_2^{R_2}, \gamma_3^{R_2}), \tag{24}$$

where $R_i(\ )$ ($i \in \{1, 2, 3\}$) is the $i$th round operation.

Thus, the whole process can be described as follows:

$$[x_1, x_2, x_3, \gamma_1, \gamma_1, \gamma_1] = P(R_1(R_2(R_3(x_1^0, x_2^0, x_3^0, \gamma_1^0, \gamma_2^0, \gamma_3^0, M, M1, M2)))), \tag{25}$$

$$h = \text{Concat}(E(x_1, 27), E(x_2, 27), E(x_3, 26), E(\gamma_1, 27), E(\gamma_2, 27), E(\gamma_3, 26)), \tag{26}$$

where $P(\ )$ is an operation on all elements of the message of size $N$, $E(a, b)$ is an extraction operation which extracts $b$ bits to form $a$, Concat ($a$, $b$, $c$, $d$, $e$, $f$) is an operation which concatenates $a$, $b$, $c$, $d$, $e$, $f$ to form a large binary number and then turn it into a hexadecimal format, and $h$ is the hash result.

One can see that the whole process includes many nonlinear processes, and they are invertible. At the same time, the application of the unified chaotic system in the algorithm makes the whole process more complex and more difficult to predict than most traditional algorithms. As a result, the preimage-attack resistance ability is greatly enhanced.

The following test has been carried out along with a quantitative study on collision-resistance analysis:[16] firstly, the hash value for a randomly-chosen paragraph of a message is generated and stored in the ASCII format. Then, a bit in the message is randomly selected and toggled. A new hash value is also generated and stored, again in the ASCII format. These two hash values are then compared, and the number of ASCII characters with the same values at the same location in each hash value is counted. Furthermore, the absolute value of difference between the

two hash values is calculated using the following formula:

$$d = \sum_{i=1}^{N} |t(e_i) - t(e_i')|, \tag{27}$$

where $e_i$ and $e_i'$ are the $i$th ASCII characters of the original and the new hash values, respectively, and the function $t(\ )$ converts the entries to their equivalent decimal values. This kind of collision test has been performed for 2048 times, with the same initial vector $x$=0.562. The maximum, mean, minimum values of $d$ and the mean/character are listed in Table 2. A plot of the distribution of the numbers of the ASCII characters with the same value at the same location in the hash value is given in Fig.6. Notice that the maximum number of equal characters is only 4, showing that the occurrence rate of collision is very low.

**Table 2.** Absolute values of differences of the two hash values.

| | Maximum | Minimum | Mean | Mean/character |
|---|---|---|---|---|
| Absolute difference $d$ | 2786 | 977 | 1918 | 95.43 |

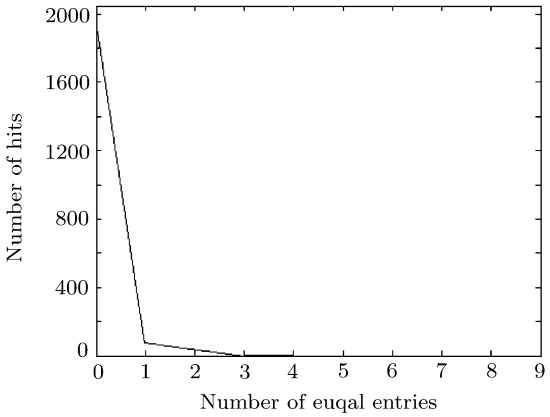



**Fig.6.** Distribution of the numbers of the ASCII characters with the same value at the same location in the hash value.

### 3.4. Analysis of speed and flexibility

From Section 2.2, one can see that the operations of the proposed hash function algorithm are mainly the iteration operations on the unified chaotic system. There are about $18n_1$ times of iteration in processing each element of the message. The amount of the message with a length of $N$ is approximately $18Nn_1$ times of the iteration operations. So, it is reasonable to conclude that the time complexity of the algorithm is $O(N)$, at the same time, from Figs.2 and 3, the space complexity of the algorithm is also $O(N)$. Tests have shown that $n_1$=3 gives a good trade-off for the proposed hash function algorithm in terms of efficiency and performance.

Although the length of the proposed algorithm is fixed to 160 bit, it can be modified by extracting more bit from $x_1$, $x_2$, $x_3$, $\gamma_1$, $\gamma_2$ and $\gamma_3$ to form 192 bits or 256 bits in hash values, to meet the demand of high-level applications if needed.

## 4. Conclusion

A new one-way hash function has been constructed based on the unified chaotic system. With different parameter values, the unified chaotic system represents different chaotic systems. The hash function algorithm is mainly composed of three round operations with the initial vectors and the plaintexts, where in each operation all information is processed by three different chaotic systems generated from the same unified chaotic system. At the same time, feedforwards are used in the end of each round operation and the end of the message processing, where a parameter-exchanging operation is also introduced during the round operation. The hash value is finally obtained from the last six parameters so generated. Simulation and analysis results both have demonstrated that the new algorithm has great flexibility, satisfactory hash performance, weak collision property, and high security. Therefore, it possesses good potential in engineering and technological markets such as e-commerce. This paper therefore has also demonstrated the good potential of chaotic systems for data-security applications.

## References

[1] Preneel B, Rijmen V and Bosselears A 1998 *Lecturer Notes in Computer Science* (Berlin: Springer-Verlag) **1528** 105

[2] Menezes A J, Van Oorschot P C and Vanstone S A 1997 *Handbook of Applied Cryptography* (London: CRC Press)

[3] Van Rompay B 2004 Leuven: K U*Ph.D. Dissertation*

[4] Liu J N, Xie J C and Wang P 2000 *J. Tsinghua Univ.* (*Sci. & Tech.*) **40** 55
[5] Wang X M, Zhang J S and Zhang W F 2003 *Acta Phys. Sin.* **52** 2737 (in Chinese)
[6] Hayes S, Grebogi C and Ott E 1993 *Phys. Rev. Lett.* **70** 3031
[7] Heileman G L, Abdallah C, Hush D R and Baglio S 1993 *Proceedings of International Symposium on Nonlinear Theory and its Applications* (Waikiki, Hawaii) p1183
[8] Peng F, Qiu S and Long, M 2005 *Acta Phys. Sin.* **54** 4562 (in Chinese)
[9] Di X, Liao X F and Deng X J 2005 *Chaos, Solitons and Fractals* **24** 65
[10] Xun Y 2005 *IEEE Trans. Circuits and System*-II **52** 354
[11] Lu J, Chen G, Cheng D and Celikovski C 2002 *Int. J. Bifur. Chaos* **12** 2917
[12] Hua C C and Guan X P 2005 *Phys. Lett.* A available from http://www.elsevier.com/locate/pla
[13] Shannon C E 1949 *Bell Syst. Tech. J.* **28** 656
[14] Van Oorschot P C and Wiener M J 1999 *Journal of cryptology* **12** 1
[15] Blaze M, Diffie W, Rivest R L, Schneier B, Shimomura T and Thompson E 1996 Available from http://www.schneier.com/paper-keylength.html
[16] Wong K W 2003 *Phys. Lett.* A **307** 292