Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, 2015, *9*, 573-580 573

Open Access

One-Way Hash Function based on Cascade Chaos

Fei Xiang[1,*], Changwei Zhao[2], Jian Wang[2] and Zhiyong Zhang[2]

[1]*Electrical Engineering College, Henan University of Science and Technology, Luoyang, Henan, 471023, P.R. China;*
[2]*Information Engineering College, Henan University of Science and Technology, Luoyang, Henan, 471023, P.R. China*

Abstract: This paper proposes a novel hash function construction based on cascade chaos. Firstly, the *L*-length original message is divided into a number of blocks according to every character, which are transformed into corresponding ASCII values. Then the ASCII values are normalized into the interval of (0, 1), which compose a big array with the L-length. Regarding the first normalized ASCII value with the algorithm parameter H as the initial input of the cascade chaos, a series of sequences are obtained, where the CBC mode is adopted. The 128-bit hash value is generated by transforming the sequences into binary sequence according to a rule. Simulation results and performance analysis show that the proposed hash function has high sensitivity to the original message and system parameters, strong confusion and diffusion, excellent statistical properties, strong collision resistance and flexible to generated 160, 256, 512-bit hash value after modifying the algorithm slightly.

Keywords: Cascade, chaotic systems, cryptography, hash function.

1. INTRODUCTION

Hash functions were proposed firstly by Knuth in the 1950's [1]. A hash function is a one-way function used for the compression of a larger block of data into a smaller fixed-sized representation of that data, which is looked as a message digest or a hash value. In recent years, with the development of E-commerce and internet applications, one-way hash function has become a research hot point [2-4] and is widely used in public-key cryptography, data confidentiality, verification of data integrity, authentication and non-repudiation of origin. Traditional hashing schemes, which have been accepted by government and industries as security standards [5], are based on hypothesis of complexity and need to carry out a large number of complex logical operations such as exclusive-OR and multi-round iterations. Moreover, recent research results show that these well-known hashing schemes such as MD5, SHA1 and RIPEMD reveal many undiscovered flaws on the collision frequencies [6-8]. So it is urgent to design new secure and efficient hashing schemes.

Chaos refers to a phenomenon, in which the long-term behavior of deterministic nonlinear dynamical system is extraordinarily sensitive to the initial values and parameters, but does not diverge, and can not be precisely repeatable. It is a general complex dynamical behavior in nonlinear systems. And it is very difficult to predict the long-term behavior correctly, because of the information loss in the iterations in chaotic systems, which leads to the amount of information of the chaotic sequences tending to zero. Chaotic sequences have some excellent properties such as extreme sensitivity on initial values and parameters, statistical properties like white-noise, ergodicity, confusion and diffusion, which make it possible to be used to construct hash functions. And it is actually attracting more and more interests of researchers. Literatures utilized chaotic changeable parameters [9], chaotic maps [3, 4], extended chaotic maps switch [10], 2D chaos [11] and chaotic neural network to construct hash functions [12] and got important results. These algorithms have two major drawbacks. Firstly, some algorithms are constructed based on low-dimensional chaotic system, the security of which can not be guaranteed because of adaptive synchronization forecasting phase space and all kinds of chaotic prediction techniques. Secondly, these chaotic sequences can degrade to be periodic sequences because of finite precision when digital implemented. Therefore the principle problems to improve one-way, weak collision, better security of chaotic hash functions are increasing complexity of chaotic signal and decreasing the influence of finite precision.

In this paper, we design a novel one-way hash scheme, whose structure is based on cascade chaotic systems. The scheme has strong collision resistance and fast performance efficiency. The rest of the paper is organized as follows: the hash function and the cascade chaotic systems are described in Section 2. The details of the proposed hash function are depicted in Section 3. The simulation results and performance are analyzed in Section 4. Finally, conclusions are drawn in Section 5.

2. PRELIMINARIES

2.1. Hash Function Properties

A hash function h should possess the following properties [13]:

1. Compression: a function h maps an input M with arbitrary finite bit length, to an output $h(M)$ of fixed bit length l.

1874-110X/15 2015 Bentham Open

2. Irreversibility: given a function h and an input M, it is easy to compute $h(M)$. However, it is computationally infeasible to find any input which hashes to a specific output, *i.e.*, to find any pre-image M such that $h(M)=y$, given any y for which a corresponding input is not known.
3. Second-preimage resistance: it is computationally infeasible to find any second input which has the same output as any specified input, *i.e.*, given M, to find a second-preimage $M'=M$ such that $h(M)= h(M')$.
4. Collision resistance: it is hard to find any two distinct inputs M and M', which hash to the same output, *i.e.*, such that $h(M)= h(M')$.

It is well known that the message space is infinite, while the hash value is always fixed bit length. Maybe there is the same message with the same hash value, but when the hash value is big, such as 128-bits length, it is difficult to carry out exhaustive calculation, whose space is $2^{128}=3.4028\times10^{28}$.

2.2. Cascade Chaos

Continuous chaotic mathematical models are mostly multi-variable coupled differential equations, in which the system parameters and initial conditions are more, so the key space of the pseudorandom sequences are larger. But due to the complexity of the algorithms, the computational speed is slower and the rate of the sequence code is lower. Discrete chaotic maps possess the properties of faster computation speed, higher rate of sequence code and excellent complexity of sequences, due to the simple algorithms. But the shortcomings of the discrete systems are the smaller Lyapunov exponents, less initial conditions and system parameters and smaller key space, which lead to the lower security of sequences. In order to improve the randomness and security of the discrete chaos, that is, increase the Lyapunov exponents and the intervals of chaotic map parameters, cascade chaos were proposed [14].

Definition: For two different chaotic discrete sub-systems $f_1(x_n)$ and $f_2(x_n)$, where $x \in D$, $f_1(x) \in D_1$, $f_2(x) \in D_2$, and n=0, 1, 2, 3, …. If $D_1=D_2=D$, that is, the codomain of the two maps is equal, then the two sub-systems can compose a new cascade system,

$$x_{n+1} = f_s(x_n) = f_2(f_1(x_n)) \tag{1}$$

It is essential that the output of sub-system 1 by iterating with a certain initial value is regarded as the input of sub-system 2, and the output of sub-system 2 by iterating is regarded as the input of sub-system 1, forming a cyclic iteration between the two sub-systems, which is shown in Fig. (**1**).

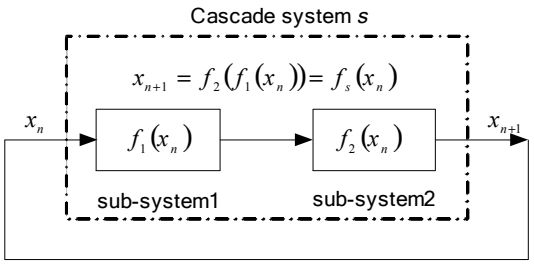

Fig. (1). Block diagram of cascade of two chaotic systems.

If the codomains of k chaotic systems are equal, then the cascade of two chaotic systems can extend the cascade of k chaotic systems,

$$x_{n+1} = f_s(x_n) = f_k(f_{k-1}(\cdots f_1(x_n))) \tag{2}$$

where, $f_1(x_n)$, $f_2(x_n)$, …, $f_k(x_n)$ are k sub-systems, $x \in D$, $f_1(x) \in D_1$, $f_2(x) \in D_2$,…,$f_k(x) \in D_k$ and $D_1=D_2=\ldots=D_k=D$. It can extend the m cascades of a chaotic system itself,

$$x_{n+1} = f(f(\cdots f(x_n))) = f^m(x_n) \tag{3}$$

Literature [14] pointed out that only when the codomains of sub-systems are equal, the sub-systems can be cascaded and proved that the Lyapunov exponent of cascade chaos approximately equal to the sum of that of every sub-systems. Therefore, the cascade of chaotic systems can increase the Lyapunov exponent, thus improve the sensitivity on the initial conditions and system parameters and randomness or complexity of the chaotic systems.

In this paper, we use two Logistic maps to compose a cascade chaos, referred to as L-L cascade system, which can be represented as

$$x_{n+1} = \mu_1[\mu_2 x_n(1-x_n)]\{1-[\mu_2 x_n(1-x_n)]\} \tag{4}$$

where μ_1, $\mu_2 \in [0, 4]$ and $x \in [0, 1]$. It is well known that for the Logistic map $x_{n+1}=\mu x_n(1-x_n)$, only when the range of parameter μ is in the interval [3.73, 4], the map shows the chaotic behavior. And when μ=4, it is a chaotic full map. But it has been proved in literature [14] that when μ_1=4, the range of μ_2 is expanded to [1.53, 4], the cascade system shows the chaotic behavior. Moreover the range of μ_2 is also expanded to 1.7, the cascade system meets chaotic full map. Additionally, the biggest Lyapunov exponent is increased to 1.3863 from 0.6930, which illustrates that the sensitivity to the initial conditions is enhanced and dynamical behavior is improved essentially.

3. ONE-WAY HASH FUNCTION BASED ON CASCADE CHAOS

3.1. Algorithm Description

The proposed one-way hash algorithm is detailed as follows:

Step 1: Divide the L-length original message into blocks, where each block corresponds to a character. Then transform the character of the original message into ASCII.

Step 2: Normalize the ASCII values. The range of normalized ASCII values is (0, 1). Thus we get a big array from the L-length original message, named S, whose length is L.

Step 3: Utilize $S_{1,1}=(S_1+H)/2$ as the initial input of the L-L cascade system (4), where S_1 is the first element of S and H is the parameter of the algorithm. In order to eliminate the effect of chaotic transition state, we cut the former 1000 iterative values. The sequence $\{S_{1,j}, j=1, 2, \ldots, 32\}$ is obtained after 1032 iterations.

Step 4: Utilize $S_{2,1}=(S_2+S_{1,32})/2$ as the initial input of the L-L cascade system (4), where S_2 is the second element

Table 1. Transform $S_{i,j}$ into binary sequences (i=1,..., L, j=1, ..., 32).

(1-1)

$S_{i,j}$	Binary	$S_{i,j}$	Binary
(0,1/16]	0000	(1/16,1/8]	0001
(1/8,3/16]	0010	(3/16,1/4]	0011
(1/4,5/16]	0100	(5/16,3/8]	0101
(3/8,7/16]	0110	(7/16,1/2]	0111

(1-2)

$S_{i,j}$	Binary	$S_{i,j}$	Binary
(1/2,9/16]	1000	(9/16,5/8]	1001
(5/8,11/16]	1010	(11/16,3/4]	1011
(3/4,13/16]	1100	(13/16,7/8]	1101
(7/8,15/16]	1110	(15/16,1)	1111

of S and $S_{1,32}$ is the last iterative value of Step 3. The sequence $\{S_{2,j}, j=1, 2, \ldots, 32\}$ is obtained after 1032 iterations.

Step 5: Analogously, we get the sequences $\{S_{3,j}\}$, $\{S_{4,j}\}$, …, $\{S_{L,j}\}$, where L is the length of the original message, and j=1, 2, …, 32.

Step 6: Transform the L sequences into 4×32=128-bit binary sequences according to the following Table **1**. Finally we obtain the 128-bit Hash value by modulo-2 adding the L binary sequences bit by bit.

3.2. Characteristic of the Proposed Algorithm

The algorithm has some characteristics: the L-L cascade system, the system parameter H and the cipher block chaining (CBC) mode.

1. The L-L cascade system has been proved that compared with Logistic map, it has larger key space, higher Lyapunov exponent and more sensitive to the initial conditions and system parameters. So the discrete chaotic pseudo random sequences generated by L-L cascade system must be more random and complex, which improve the degradation of dynamical behavior and expand the chaotic digital sequences. What's more, the parameters of μ1 and μ2 enlarge the key space of the algorithm.
2. In Step 3, we introduce the algorithm parameter H, the range of which is (0, 1). Firstly, the algorithm parameter H ensures the initial value cannot be the weak key of the L-L cascade system, such as 0, 0.25, 0.5, 0.75 and 1. Secondly, H magnifies the key space of the algorithm.
3. In Step 4, the last value of the first iteration is regarded as the part of the initial value of the second iteration, that is, we adopt the CBC mode, which assures that even if the two characters of the original message are same, the iterative sequences are not different.

4. PERFORMANCE ANALYSIS

In this section, we conduct several tests to investigate the performance of the proposed hash function. The parameters are selected as μ_1, μ_2=4 and H=0.0001.

4.1. Hash Sensitivity to the Original Message and Conditions Change

The aim of this subsection is to demonstrate the high sensitivity of the suggested hashing algorithm to the original message and condition modifications. For this purpose, a number of experiments have been performed. Each experiment targets a specific case. The obtained results show the high sensitivity to the original message and condition changes. Six of these experiments are listed below. In these experiments, C1 represents the original message, C2, C3, C4, C5 and C6 represent the original message with minor modifications.

C1: The original message is "This paper proposes a design method of true random bit generator (TRBG) based on cell phone recording and chaotic encryption. Firstly a recording from cell phone is collected, then is transformed the format operated by MATLAB. The recording is turned into a sequence in computer by MATLAB, which is encrypted by chaotic sequence from a chaotic system. Finally, through binarization a true random bit sequence is produced. Randomness and security analysis is carried out. Simulation results show that the properties of the bit sequence are excellent in randomness and security, and the method is safe, fast and easy to realize."

C2: Replace the first character of the original message "T" by "t".

C3: Replace the word "encryption" in the original message by "decryption".

C4: Replace the last character of the original message "." by ",".

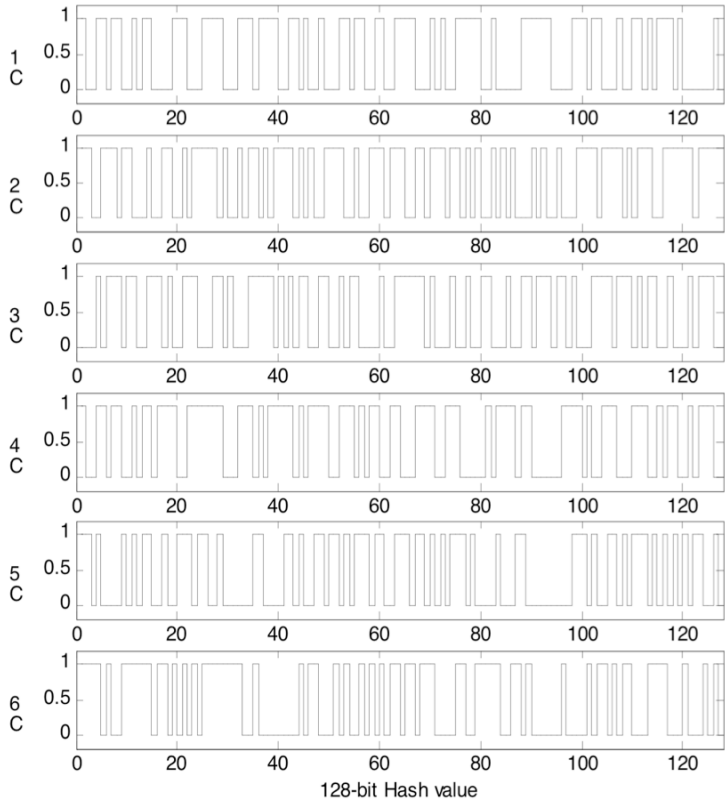

Fig. (2). Hash value of messages with tiny difference.

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

C6: Replace μ_1=4 of the L-L cascade system by μ_1=3.999999.

The 128-bit hash values in hexadecimal format are given as follows, followed by the corresponding different bits compared with the original hash value.

C1: 9B2C 38F1 DE69 1B33 C53E 41F8 7626 BA04;

C2: CEC4 CBE9 6BD4 F277 376A 5452 3DEB 9FBF (67 bits);

C3: 1767 4E34 7D59 9613 F449 89BB 47BA C8B9 (59 bits);

C4: 9B2D E7F1 D7D7 9D66 3CE0 BD81 E9C7 2CB9 (66 bits);

C5: D0AC 9D90 30D3 6BB3 B574 2300 74D7 5565 (57 bits);

C6: F4FD AAFF 2016 2956 DC33 E681 0B58 F115 (71 bits).

In Fig. (**2**), we plot the hash values of different messages with tiny difference. Simulation results indicate clearly that the proposed algorithm is so sensitive that any slight difference of the message or the system parameter will case huge changes in final hash value.

4.2. Statistical Distribution of Hash Value

The aim of this subsection is to demonstrate the uniform distribution of hash value, which is directly related to the security of hash function. For this purpose, we utilize 2-dimensional graphs to show the differences between the original message and the final hash value. In the left picture of Fig. (**3**), the ASCII values of the original message are localized within a small area; while in the right one of Fig. (**3**), the hexadecimal hash value spreads around very uniformly. The similar experiment has been done to a paragraph of all-'0' message with the same length as the above original message. The contrast between the message and hash value is pictured in Fig. (**4**). Even under this very extreme condition, the contrast is still distinct, and the distribution of hash value is also uniform. Simulation results indicate that no information of the original message can be left after confusion and diffusion.

4.3. Statistical Analysis of Confusion and Diffusion

There are two main principles in the design of hash function, which are confusion and diffusion. The aim of confusion is to make the relationship between the input bits and the output bits as complex and dependent as possible. The aim of diffusion is to make the output bits highly dependent on the input bits. The ideal one is that a one bit change in the input block results in 50% change in the output block. In this subsection, the following test were carried out to demonstrate the strong capability of the proposed hash function for confusion and diffusion. A paragraph of message is selected randomly and the corresponding hash value is generated; then a bit in the message is randomly changed and toggled and a new hash value is generated. Two hash values are compared and the number of different bits is counted. The following statistics are usually used. Here N is the number of tests, and B_i denotes the number of different bits between the hash values obtained in the i-th test.

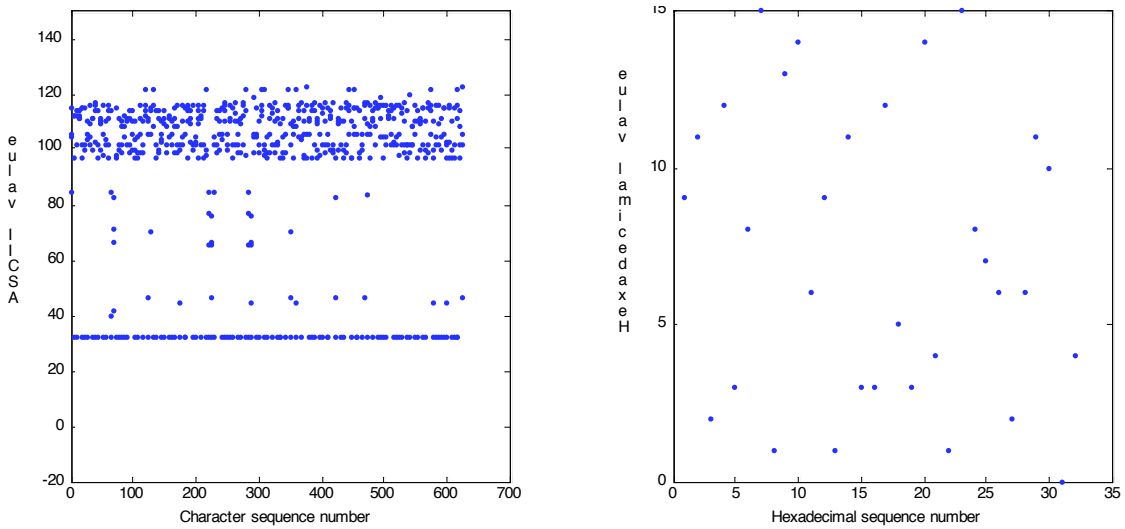

Fig. (3). Spread of hash value of the original message.

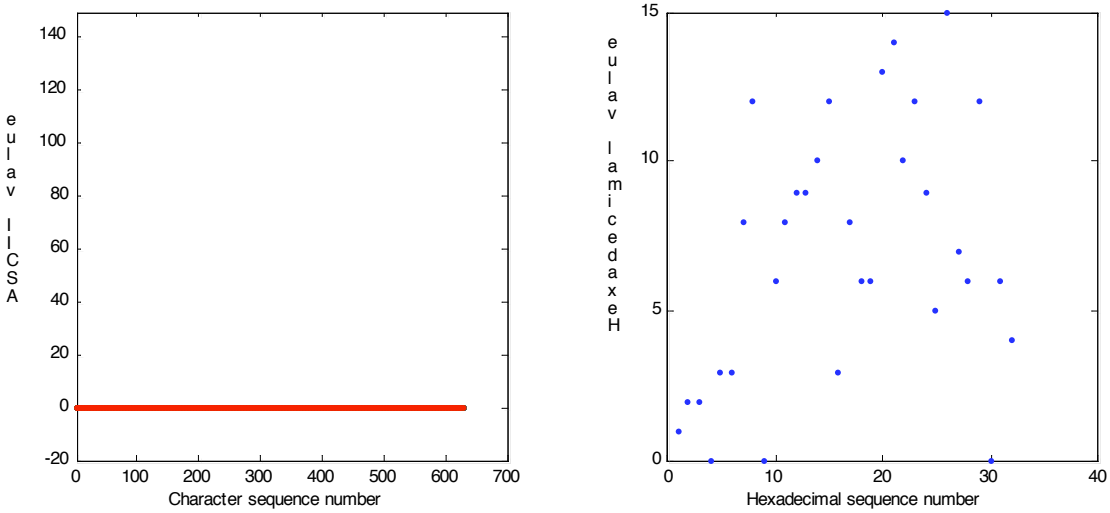

Fig. (4). Spread of hash value of all '0'-message.

Minimum number of different bits: $B_{\min} = \min\left(\left\{B_i\right\}, i = 1,2,\cdots,N\right)$.

Maximum number of different bits: $B_{\max} = \max\left(\left\{B_i\right\}, i = 1,2,\cdots,N\right)$.

Mean number of different bits: $\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$.

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

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

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

The corresponding distribution of the number of different bits is plotted in Fig. (**5**), where N=10000. Obviously, the number of different bits to the message with 1 bit changed concentrates around the ideal number which is 64 bits. It indicates that the algorithm has very strong capability for diffusion and confusion.

Through the tests with N= 256, 512, 1024, 2048, 10000 respectively, the corresponding data are listed in Table **2**. The data show that both the mean number of different bits $\overline{B}$ and the mean probability P are very close to the ideal value 64-bit and 50%. Furthermore, ΔB and ΔP are very small, which indicates that the confusion and diffusion property is stable. A slight difference in the message causes great changes in the hash value, which contributes to the high sensitivity of the proposed hash function to the original message. This property is very important in keeping the resistance against statistical attack.

4.4. Collision Analysis

Collision resistance means that it is almost impossible to find two different input $x' \neq x$ whose hash values $h(x')=h(x)$.

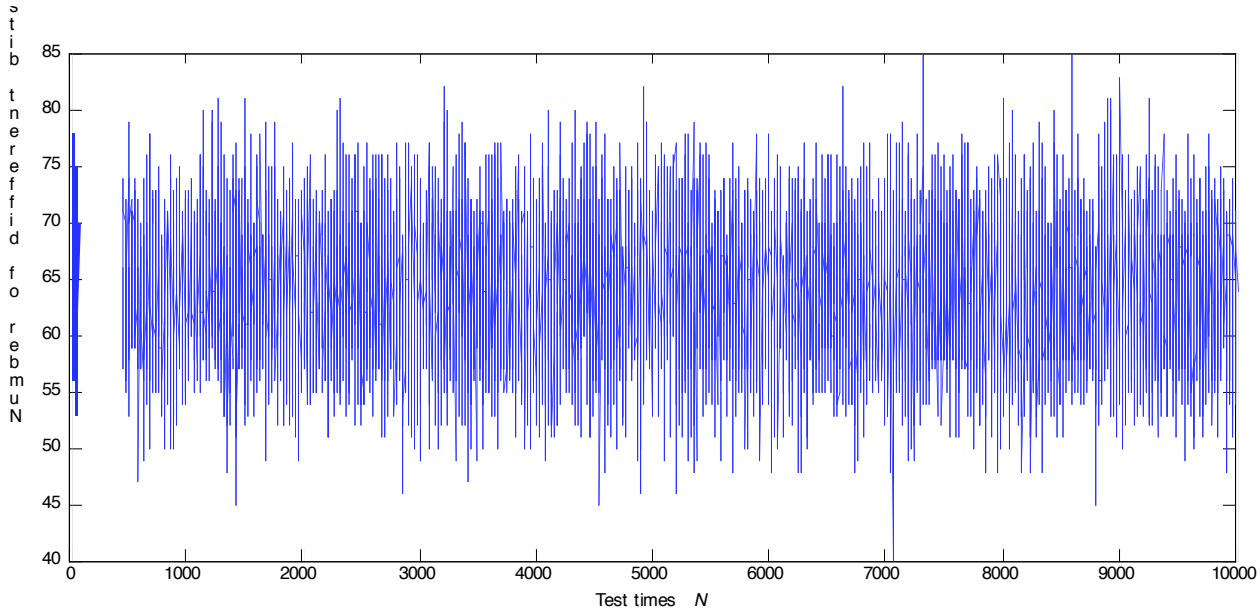

Fig. (5). Distribution of the number of different bits.

Table 2. Statistical results for 128-bit hash values generated under tests.

(2-1)

	***N*=256**	***N*=512**	***N*=1024**
B_{min}	46	48	47
B_{max}	78	78	79
$\overline{B}$	63.7383	63.5605	63.6494
P(%)	49.8	49.66	49.73
ΔB	5.0127	5.1121	4.9853
ΔP(%)	3.92	3.99	3.89

(2-2)

	***N*=2048**	***N*=1000**	***Mean***
B_{min}	45	40	45.2
B_{max}	81	85	80.2
$\overline{B}$	63.7765	63.9279	63.7305
P(%)	49.83	49.94	49.7895
ΔB	5.0676	5.3234	5.1002
ΔP(%)	3.96	4.16	3.984

Essentially, birthday attack is similar with the collision problem, which is a probability issue with two random input data hashing to the same values. The complexity of this attack is determined by the size of the hash value. For a 128-bit hash value, the complexity of the birthday attack is $2^{128/2}$. Thus the proposed scheme can resist against this type of attack. Besides, the CBC mode is adopted in the algorithm, where every iterative input of L-L cascade system is decided by the previous message character and the iterative output. The structure inherently has the ability to strengthen avalanche effect, which ensures the arbitrary bit of the hash value is related to all of the bits of the original message, that is, a slight change in the message or system parameter will lead to a totally different hash value through iterations.

To conduct a quantitative study on collision analysis, the following test is performed [15]. A random message, referred to as the original message, is randomly chosen and its

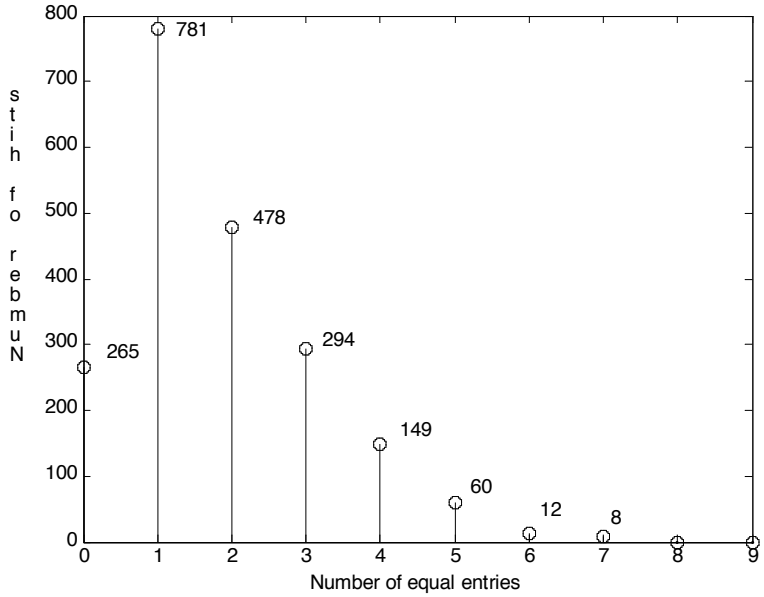

Fig. (6). Distribution of the identical hexadecimal on the same position.

corresponding 128-bit hash value is calculated and stored in hexadecimal. Then a bit is changed in the message and a new hash value is obtained. Comparing with the two hash values, the number of positions where the hexadecimal are identical is counted and recorded. We have run this test N=2048 times and record the results. In Fig. (**6**), the X-axis denotes the number of the same hexadecimal on the same position, and the Y-axis denotes the test times. It is shown that after N=2048 tests, the probability are 5.5588% appearing the same hexadecimal value on the same position. For the 128-bit hash value, in the 32 hexadecimals, there is only an average of 1.7882 hexadecimals are identical on the same position with every bit changed, which illustrates the low level of collision.

4.5. Key Space

The algorithm has three parameters to be the keys, μ_1, μ_2 and H. In the previous subsection, the analysis has proven that any slight change to the original message, the initial conditions or system parameters will lead to a significant change in the calculated hash value. It is shown that the hash function is high sensitive, which plays an important role in determining the key space. The key space of the proposed hash algorithm consists of μ_1, μ_2 in L-L cascade system and H in the algorithm. When μ_1=4, the changing interval of μ_2 is 2.17, which is 5.86 times of μ in single Logistic map [14]. The interval of H is (0, 1), which ensures the algorithm has enough large key space.

4.6. Flexibility

The proposed hash function is flexible in the sense because it is easily modified to produce 160, 256, 512-bit hash value. For example of 160-bit hash value, in the previous Section 3, as long as we extract 40 iterations in Step 3 and Step 4, we will get 160-bit hash value at the end. Analogously, when we extract the iterations 64 times and 128 times, we will get 256-bit and 512-bit hash value, respectively.

CONCLUSION

In this paper, we propose a novel hash function based on cascade chaos. The algorithm adopts the L-L cascade system, which enlarges the key space of the algorithm, improves the degradation of digital chaotic sequences and increases the complexity of the chaotic sequences. Moreover, it utilizes CBC mode, which strengthens the avalanche effect. Then the algorithm is flexible to modify to generate 160, 256 and 512-bit hash value. Simulation results and performance analysis show that the proposed hash function has extreme sensitivity to the original message, the L-L cascade parameter and the algorithm parameter, strong confusion and diffusion capability, strong collision resistance.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The work was sponsored by National Natural Science Foundation of China Grant No.61370220, Plan For Scientific Innovation Talent of Henan Province Grant No. 134100510006, Key Program for Basic Research of The Education Department of Henan Province Grant No. 13A520240 and No.14A520048, the Key Science and Technology Program of Henan Province, China Grant No. 142102210425, Training Foundation for Scientific Innovation Ability of Henan University of Science and Technology Grand No. 2013ZCX022.

REFERENCES

[1] D. Knuth, *The Art of Computer Programming, Sorting and Searching*, 2nd ed., American:Addison-Wesley, vol. 3, 1998, pp. 513-558.

[2] S. H. Wang, and G. Hu, "Coupled map lattice based hash function with collision resistance in single-iteration computation," *Information Science*, vol. 195, pp. 266-276, 2012.

[3] A. Kanso, H. Yahyaoui, and M. Almulla, "Keyed hash function based on a chaotic map," *Information Science*, vol. 186, pp. 240-264, 2012.

[4] A. Akhavan, A. Samsudin, and A. Akhshni, "Hash function based on piecewise nonlinear chaotic map," *Chaos, Solitons and Fractals*, vol. 42, pp. 1046-1053, 2009.

[5] NIST, *Secure Hash Standard,* 2001, Available from: http://csrc.nist. gov/Crypto Toolkit/tkhash.html.

[6] X. Y. Wang, D. G. Feng, X. J. Lai, and H. B. Yu, "Collision for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," *Rump Session of Crypto*'04, E-print, 2004.

[7] X. Y. Wang, X. J. Lai, D. G. Feng, H. Chen, and X. Y. Yu, "Cryptanalysis of the hash functions MD4 and RIPEMD," In: *Proceedings Eurocrypt'05,* Aarhus, Denmark, 2005, pp.1-18.

[8] X. Y. Wang, and H. B. Yu, "How to break MD5 and other hash functions," In: *Proceedings of Eurocrypt'05*, Aarhus, Denmark, 2005, pp.19-35.

[9] J. D. Liu , and Y. M. Yu, "A TCML-based spatiotemporal chaotic one-way hash function with changeable-parameter," *Acta Physica Sinica*, vol. 56, pp. 1297-1304, 2007.

[10] X. M. Wang, J. S. Zhang, and W. F. Zhang, "One way hash function construction based on the extended chaotic maps switch," *Acta Physica Sinica*, vol. 52, pp. 2737-2742, 2003.

[11] A. Akhshani, S. Behnia, A. Akhavan, M. A. Jafrizadeh, H. Abu Hassan, and Z. Hassan, "Hash function based on hierarchy of 2D piecewise nonlinear chaotic maps," *Chaos, Solitons and Fractals*, vol. 42, pp. 2405-2412, 2009.

[12] Z. Q. Huang, "A more secure parallel keyed hash function based on chaotic neural network," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, pp. 3245-3256, 2011.

[13] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanston, *Handbook of Applied Cryptography*, American: CRC Press, 1997, pp. 321-384.

[14] G. Y. Wang, and F. Yuan, "*Cascade Chaos and its Dynamic Characteristics*," *Acta Physica Sinica*, vol. 62, p. 020506, 2013.

[15] K. W. Wong, "*A Combined Chaotic Cryptographic and Hashing Scheme*," *Physics Letters A*, vol. 307, pp. 292-298, 2003.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014