International Journal of Bifurcation and Chaos
One-way Hash function based on delay-induced hyper-chaos
--Manuscript Draft--

Manuscript Number:	IJBC-D-17-00338R4
Full Title:	One-way Hash function based on delay-induced hyper-chaos
Article Type:	Research Paper
Keywords:	One-way Hash function; hyper-chaos induced by time delay; key-stream function; Cipher block chaining.
Corresponding Author:	Hai Peng Ren, Ph.D Xi'an University of Technology Xi'an, Shaanxi CHINA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Xi'an University of Technology
Corresponding Author's Secondary Institution:	
First Author:	Hai Peng Ren, Ph.D
First Author Secondary Information:	
Order of Authors:	Hai Peng Ren, Ph.D
	Zhao Feng Zhao
	Celso Grebogi, Ph.D
Order of Authors Secondary Information:	
Abstract:	A scheme for constructing one-way Hash function based on hyper-chaos induced by time delay and key-stream is proposed in this paper. In this scheme, the plain-text and secret key are used as the initial value in two hyper-chaotic Chen systems; these values are evolved in a hyper-chaotic way during a predened period. The results of the evolution are quantied and iterated using key-stream function iteration to confuse and diuse the plain-text and the secret key. The cipher block chaining method is used to generate a 128-bit Hash value for a plain-text of arbitrary length. Theoretical analysis and simulation results indicate that the proposed algorithm satises the required security performance, such as value ompression function, irreversibility, initial value sensitivity, forgery resistance and collision resistance.
Response to Reviewers:	see response file

Reply to reviewers of manuscript ID IJBC-D-17-00338R1 entitled by "One-way Hash function based on delay-induced hyper-chaos"

Dear Editor and reviewers,

After receiving the decision letter, the authors have revised the paper to address the comments of the reviewers. The replies are as follows:

Reviewer #1: I am satisfied with the revisions and responses. Meanwhile, I think the authors well address the comments of the Reviewer #4, so I suggest accept as it is.

Reply: Thank you for the comments, we fully agree!

Reviewer #4: Authors have made some minor improvements in the paper but unfortunately most significant issues are not addressed in the paper although authors gave extensive answers which are not satisfactory due to following reasons.

Comment 3: Authors correctly state in their response that birthday attack will require over 2^{64} hashes for hash value of 128 bits, but their proof of security is not sufficient. Authors claim that this attack will take about 60,000 years using a desk top computer, by hashing 10^6 times a second (the current cracking speed), but unfortunately cryptographic attack are not performed on single desk top computer.

There are many examples of attacks based on distributed computing which involve 100.000 computers, so even if mentioned speed is correct today, birthday attack could be performed in about half year. Also, there are many examples of attacks based on specialized computers which speed is much faster (for example computers used for attacks on DES). Therefore, authors cannot claim that m=128 is completely safe. As I said in my previous review, m=256 should be used.

Reply: as the reviewer wrote, the birthday attack explanation is correct, agreeing with a textbook. Furthermore, the desktop computer calculation is a reference. We can agree that for an infinite number of computers, no algorithm would be safe enough, but that is nonsense.

Comment 5: In their answer authors claim that " there is not a "fully" discrete space chaotic map to be readily used for Hash function". This is not correct because there are several fully discrete space chaotic maps which can be used in the same way as map used in this paper for generation of pseudo-random sequences. For example maps used in following papers are fully discrete, but there are other examples as well:

Pseudo-random number generator based on discrete-space chaotic map. Nonlinear Dynamics, 90, 223-232 (2018)

A New Pseudo-Random Number Generator Based on Two Chaotic Maps. Informatica, 24(2), 181-197 (2013)

I still believe that this problem can reduce resistance of this method to collision.

Reply: The idea was only to show the example in "Informatica", where the iteration (Eq.1 in it) is

$X_{n+1} = [[X_n^2 \bmod C] \times X_n + X_{init}] \bmod C.$

The reviewer is wrong by asserting that this iteration is fully discrete. There is digital round-off in the process of the calculation, which of course depends on the bit length used by the computer, as the authors have already replied in the last round of reviews.

Comment 6: Sensitivity to plain-text is directly related to attacks because attacker tries to find similar plaintexts which will give same hash value. Therefore, if proposed method is not sensitive enough to plain-text, it will be more vulnerable to attacks such as birthday attack. My point on this comment was that it would be good to perform more extensive test than just 6 examples. I know that 2^64 is too much, but testing on million examples would be sufficient and feasible.

Reply: First of all, the sensitivity of the plaintext is defined as one bit change in the plaintext leading to the bit change ratio of the cipher text. This has been shown in the paper. We agree that additional tests would increase the accuracy of the result. However, statistically, 30 Monte Carlo tests would demonstrate the statistical result, though it would not be safe.

Our tests is from small number to larger number (up to 10^4) of bit reverse. Statistics guarantees that it is safe. We can perform additional tests, of course, but there is no statistical reason for that.

Comment 7: please see comment 3.

See the reply above

For above mentioned reasons I cannot recommend this paper for publication in its current form. I would like to give authors another chance to improve this paper according to my previous comments. These comments should be addressed seriously in order to avoid publication of paper in which unsafe method is proposed.

Based on the recommendation of the first reviewer and our replies to the Reviewer #4, we hope that the paper can now be published in the IJBC.

Sincerely

Hai-Peng Ren, Chao-Feng Zhao, Celso Grebogi

International Journal of Bifurcation and Chaos
© World Scientific Publishing Company

One-way Hash function based on delay-induced hyper-chaos

Hai-Peng Ren [1,*], Chao-Feng Zhao [1,†], Celso Grebogi[1,2,‡]
1 *Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, Xi'an University of Technology, Xi'an 710048, P.R.China*
2 *Institute for Complex System and Mathematical Biology, University of Aberdeen, Aberdeen, AB24 3UE, UK*
* *renhaipeng@xaut.edu.cn*
† *tufei210@126.com*
‡ *grebogi@abdn.ac.uk*

Received (to be inserted by publisher)

A scheme for constructing one-way Hash function based on hyper-chaos induced by time delay and key-stream function iteration is proposed in this paper. In this scheme, the plain-text and secret key are used as the initial value in two hyper-chaotic Chen systems; these values are evolved in a hyper-chaotic way during a predefined period. The results of the evolution are quantified and iterated using key-stream function iteration to confuse and diffuse the plain-text and secret key. The cipher block chaining mode is used to generate a 128 bits Hash value for a plain-text of arbitrary length. Theoretical analysis and simulation results indicate that the proposed algorithm has satisfactory performance, such as value compression function, irreversibility, initial value sensitivity, forgery resistance and collision resistance.

Keywords: One-way Hash function; hyper-chaos induced by time delay; key-stream function; Cipher block chaining.

1. Introduction

Cryptographic Hash function is one of the most important cryptographic algorithms nowadays [Zhou *et al.*, 2012; Teh *et al.*, 2015; Jeng *et al.*, 2015; Kanso & Ghebleh, 2015; Ye *et al.*, 2015; Chenaghlu *et al.*, 2016; Li *et al.*, 2016]. There are some traditional one-way Hash algorithms, such as MD2, MD4, MD5, and SHA [Rivest, 1992; Dobbertin, 1996; Knudsen & Preneel, 2002; Mendel *et al.*, 2013], but most of them are based on the hypothesis of complexity that requires lots of complicated logic computing (such as XOR) or packet encryption method based on multiple iterations. Since Wang announced the decoding results of the encryption algorithms MD5, HAVAL-128, MD4, RIPEMD [Wang *et al.*, 2005a; Wang & Yu, 2005b], and the SHA-1 algorithm [Stevens, 2013] were cracked afterwards, the design of efficient and safe Hash algorithm has become the focus in information security research.

Because of the unique and distinct properties of chaotic dynamics, utilizing chaotic dynamics in cryptography has witnessed a promising development. Chaos possesses some special and relevant properties, such as ergodicity, sensitive dependency on initial condition, and dense set of unstable periodic orbits.

Therefore, many chaos based cryptosystems were proposed, such as chaos based image encryption [Chen *et al.*, 2004; Wong *et al.*, 2009; Wang *et al.*, 2011a; Fu *et al.*, 2012; Ye *et al.*, 2015; Yang *et al.*, 2016], Hash functions [Xiao *et al.*, 2005, 2009; Ren & Zhuang, 2009; Zhou *et al.*, 2012; Jiteurtragool *et al.*, 2013; Jeng *et al.*, 2015; Li *et al.*, 2016; Choi *et al.*, 2017]. Wong developed a combined Hash scheme [Wong, 2003] based on the work of Baptista [Baptista, 1998], which used a dynamical look-up table. Lately, Alvarez showed that the algorithms in [Baptista, 1998] and [Wong, 2003] are insecure and inefficient [Alvarez *et al.*, 2004]. Compared with chaos in simple maps (such as the tent map [Yi, 2005]), hyper-chaos [Ren *et al.*, 2006; Cang *et al.*, 2010; Ren & Li, 2010; Ren *et al.*, 2017a] has more than one Lyapunov exponents and it has a more complex dynamics, which potentially promises better security. Gao and Chen offered an encryption based on hyper-chaos which was generated from Chen's chaotic system [Gao & Chen, 2008]. Although their algorithm had the advantage of large key spaces, the chosen-plaintext and chosen-ciphertext allows for possible attack [Rhouma & Belghith, 2008]. In 2015, the weakness of Gao and Chen's algorithm and an improved algorithm was reported in [Jeng *et al.*, 2015]. Four dimensional hyper-chaotic Chen system with two positive Lyapunov exponents was used to construct a Hash function in [Ren & Zhuang, 2009], demonstrating nice performance. Stable Chen system can be chaotified by linear time delay feedback [Ren *et al.*, 2006], compared to both Gao's chaos system [Gao & Chen, 2008] and four dimensional hyper-chaotic Chen system [Ren & Zhuang, 2009], possessing more complicated dynamics and having infinite dynamical dimension. Such a complex hyper-chaos might possess even better security potential. In this paper, the hyper-chaos with time delay is used to construct Hash function in order to obtain better performance.

Linear time delay feedback was proposed to generate hyper-chaos with infinite dimension[Ren *et al.*, 2006]in the sense that a continuum of initial conditions over the interval $-\tau \leq t < 0$ is required to specify the behavior, where τ is the time delay. It possesses more complicated dynamics, including single-scroll, double-scroll and multi-scroll attractors [Ren & Li, 2010; Ren *et al.*, 2017a,b]. To explore the potentialities of such complex attractors, in this paper, the attractors induced by time delay and key-stream iteration are proposed to construct one-way Hash function. Theoretical analysis and simulation results show that the proposed algorithm has an excellent performance and better anti-collision performance than the competing methods.

The remaining of this paper is organised as follows: the attractors induced by time delay feedback and the basic requirement of the Hash function is introduced in Sec. 2. In Sec. 3, a Hash algorithm is designed based on the hyper-chaos attractors and the key-stream iteration. The performance of the proposed Hash algorithm is evaluated and the comparison to existing methods is conducted in Sec. 4. Conclusions are given in Sec. 5.

2. Preliminary on Hash function

2.1. *Hash Function and its safety requirements*

A Hash function is a one-way function that is used to map a digital data of arbitrary length to another digital data of the fixed length. The image of the digital data returned by a Hash function is referred to as Hash value or message digest. Hash functions should possess properties like sensitivity to the plain-text, secure key, and collision resistance. Moreover, Hash function is required to satisfy the safety requirements:

1) Given a message m and a Hash function H, it should be easy and fast to compute Hash value $h = H(m)$.

2) Given h, it is very difficult to compute m, such that $h = H(m)$.

3) Given m, it is very difficult to find another message m', such that $H(m') = H(m)$.

2.2. *Hyper-chaotic Chen system with linear time delay feedback*

The Chen system with linear time delay feedback [Ren *et al.*, 2006] is described as follows:

$$\begin{aligned}\dot{x} &= a(y-x),\\ \dot{y} &= (c-a)x - xz + cy,\\ \dot{z} &= xy - bz + k(z - z(t-\tau)).\end{aligned}\tag{1}$$

where x, y, z are the state variables of the system, a, b, c are the parameters of the system, τ is the delay time, and k is the linear time delay feedback gain.

When the parameters $a = 35$, $b = 3$, $c = 18.5$, $k = 0$, the Chen system without time delay feedback is not chaotic. When the parameters $a = 35$, $b = 3$, $c = 18.5$, $k = 3.8$, and $\tau = 0.3$, the system (1) is chaotic [Ren *et al.*, 2017b], which demonstrates a composite multi-scroll attractor, as shown in Fig. 1(a). Using the method in [Ren *et al.*, 2017b], we calculate the positive Lyapunov exponents of the composite multi-scroll attractor, giving 1.5651, 0.4322, and 0.0684. When the parameters $a = 35$, $b = 3$, $c = 18.5$, $k = 2.85$, and $\tau = 0.3$, a double-scroll attractor is generated, as shown in Fig. 1(b). The positive Lyapunov exponents of the double-scroll attractor are given as 0.2219, 0.2216, and 0.0035. When the parameters $a = 35$, $b = 3$, $c = 18.35978$, $k = 2.85$, and $\tau = 0.3$, a single-scroll attractor is generated, as shown in Fig. 1(c). The positive Lyapunov exponents of the single-scroll attractor are given as 0.3963, and 0.1095.

As compared with ordinary chaotic systems, hyper-chaotic systems have more complex phase trajectory. The Hash function, using hyper-chaotic system, promises better performance than that using low dimensional chaotic system.

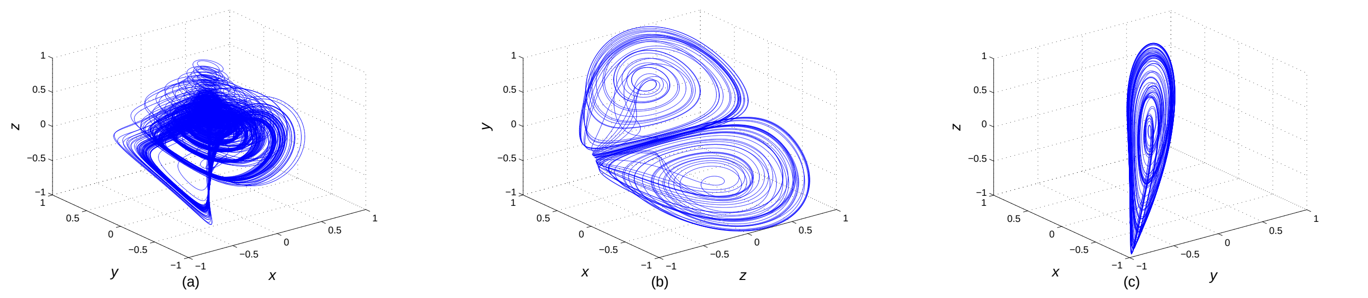

Fig. 1. The attractors induced by linear time delay feedback in the Chen system. (a) The composite multi-scroll attractor; (b) The double-scroll attractor, and (c) The single-scroll attractor.

2.3. *Key-stream function*

A key-stream function is given by follows:

$$e(p) = \underbrace{f_1(\cdots f_1(f_1}_{n_1}(p, \underbrace{v), v), \cdots v)}_{n_1},\tag{2}$$

where p is one input to the iteration function f_1 and v is another input to the same function f_1. In the paper, p and v are derived by evolving the Chen system with time delay over a duration T using plain-text and the key as initial conditions. The parameter l must be chosen such that both p and v lie in $(-l, l)$, $l = 1$ and the number of iterations $n_1 = 30$ in this paper. And $f_1(*, *)$ is a piecewise linear function given by

$$f_1(a,b) = \begin{cases} (a+b) + 2l, -2l \le (a+b) \le -l,\\ (a+b), -l \le (a+b) < l,\\ (a+b) - 2l, l \le (a+b) \le 2l.\end{cases}\tag{3}$$

3. The proposed Hash algorithm

In this section, the proposed Hash algorithm based on the attractors in the Chen system with linear time delay feedback and key-stream function is described in detail. The input of Hash function is a message of arbitrary length, and the output of Hash function is m bits Hash value, where $m = 128$ in this paper. The reason for choosing the 128 bits Hash value is that it is long enough to ensure the desired performance of Hash function [Kanso & Ghebleh, 2015; Teh *et al.*, 2015]. Figure 2 is given to illustrate the proposed Hash algorithm. The proposed Hash algorithm is described by using the following steps:

Step 1: The plain-text is partitioned into blocks of 32 bits length. When the bit number in the last block is less than 32, compensating bits are used to compensate the bit length of the last block to 32 bits. (The way to get compensating bits is given as follows: Firstly, converting the plain-text into a value between $[0, 1]$ by dividing 2^{p_1}, where p_1 is the bit length of the plaint-text in the last block. Secondly, using the value within $[0, 1]$ as the initial value of the Logistic map, iterating the map with the time equal to the number of the compensating bits. Finally, if the iteration value is larger than 0.5 then converting it as "1", else converting it as "0". By this way, compensating codes are derived.) After this compensating code is used, we have N_p blocks with 32 bits length. Three consecutive blocks are rearranged into one group recorded as $M_i (i = 0, 1, \cdots, N_p - 1)$. If the remainder of the number of the plain-text blocks, N_p, divided by 3 is not zero, the last one (or two) block(s) is (are) grouped together with two (or one) blocks (block) before the last block.

Step 2: 128 bits key is divided into three numbers with 40 bits, 40 bits and 48 bits length, respectively. Then three numbers are converted into real numbers in $[0, 1]$ by dividing by 2^{40}, 2^{40}, and 2^{48}, respectively. For example, a key given by "2A86D71ECB063FAC589B74132C3874AB" can be divided into "2A86D71ECB","063FAC589B", and "74132C3874AB", then can be converted into decimals given by "251370348235", "26838063259", and "127625695098027", by dividing by 2^{40}, 2^{40}, and 2^{48}, respectively, we get the real numbers given by "0.228619999902548", "0.024409076339907" and "0.453417552741183".

Step 3: Each group of plain-text consisting of three 32 bits integers is converted into three values in $[0, 1]$ by dividing by 2^{32}. Three values are used as the initial states of the hyper-chaotic Chen system (with time delay) and the system is evolved for a period T (it is changeable and not a very long time). The final states of the hyper-chaotic Chen system are restricted into $[-1, 1]$, dividing it by the maximum absolute value of the corresponding state. These three values in $[-1, 1]$ are used as three p in the iteration (2) and the corresponding three v are obtained at the next step (step 4).

Step 4: Similar to the step 3, three real numbers derived in step 2 from 128 bits key are used as initial states of the hyper-chaotic Chen system and they are evolved for a time span T, restricting the final states to $[-1, 1]$. We get three v for iteration (2), use three p obtained in step 3 and three v in this step, iterate (2) for $n_1 = 30$ times, we get three iteration outputs.

Step 5: Transforming three iteration outputs obtained in step 4 into the corresponding binary integers with 40 bits, 40 bits and 48 bits length, then combine together these 128 bits integer. To this end, the Hash function output H for one group of (three) plain-text (with 32 bits) is obtained.

Step 6: Cipher block chaining (CBC) method is used to cascade the output of Hash function for every group into one Hash output. CBC is described by

$$k_0 = Key, h_i = H(M_i, k_i), k_i = h_{i-1} \oplus k_{i-1}, \cdots, h = h_{n-1} \oplus k_{n-1} \tag{4}$$

In this method, the former Hash output value and the former key are used to perform logical XOR operation to obtain the 128 bits key of the next Hash operation, this process ensures the further diffusion and confusion of the plain-text and key. A point about the digital deterioration should be noticed here is given as follows. In chaos-based cryptography applications and implementations on finite precision computers, the digital deterioration is a major problem, which causes the chaotic trajectory to differ drastically from the theoretical expectations [Li *et al.*, 2001, 2003; Lin *et al.*, 2017a,b; Murillo-Escobar *et al.*, 2017]. In the proposed algorithm, the digital deterioration will not be significant, because the proposed Hash function using the hyper-chaotic Chen system with time delay and key stream iterations evolves the chaotic dynamics for a time span T (it is changeable and not a very long time), and uses the final states as plain-text $p(t)$ and key $v(t)$ in the key-stream function to get the Hash value. T is relatively small for the computation precision of the commonly used commercial computers, at the same time, it will not have accumulative

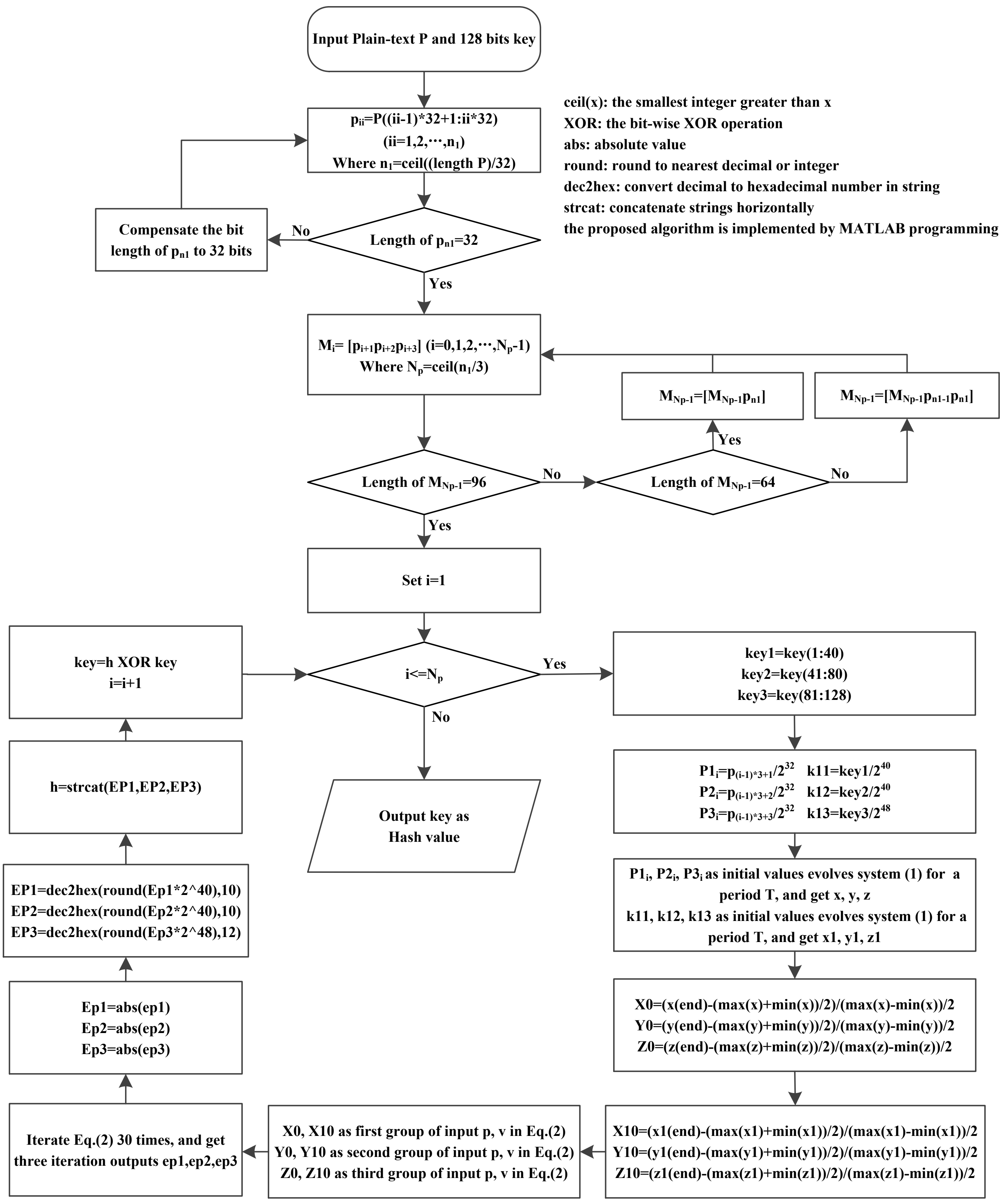
Input Plain-text P and 128 bits key
p_{ii}=P((ii-1)*32+1:ii*32) (ii=1,2,⋯,n_1) Where n_1=ceil((length P)/32)
ceil(x): the smallest integer greater than x
XOR: the bit-wise XOR operation
abs: absolute value
round: round to nearest decimal or integer
dec2hex: convert decimal to hexadecimal number in string
strcat: concatenate strings horizontally
the proposed algorithm is implemented by MATLAB programming
Compensate the bit length of p_{n1} to 32 bits
No
Length of p_{n1}=32
Yes
M_i= [$p_{i+1}p_{i+2}p_{i+3}$] (i=0,1,2,⋯,N_p-1) Where N_p=ceil(n_1/3)
M_{Np-1}=[$M_{Np-1}p_{n1}$]
M_{Np-1}=[$M_{Np-1}p_{n1-1}p_{n1}$]
Yes
Length of M_{Np-1}=96
No
Length of M_{Np-1}=64
No
Yes
Set i=1
key=h XOR key i=i+1
i<=N_p
Yes
key1=key(1:40) key2=key(41:80) key3=key(81:128)
No
h=strcat(EP1,EP2,EP3)
Output key as Hash value
$P1_i$=$p_{(i-1)*3+1}/2^{32}$ k11=key1/2^{40}
$P2_i$=$p_{(i-1)*3+2}/2^{32}$ k12=key2/2^{40}
$P3_i$=$p_{(i-1)*3+3}/2^{32}$ k13=key3/2^{48}
EP1=dec2hex(round(Ep1*2^40),10) EP2=dec2hex(round(Ep2*2^40),10) EP3=dec2hex(round(Ep3*2^48),12)
$P1_i$, $P2_i$, $P3_i$ as initial values evolves system (1) for a period T, and get x, y, z
k11, k12, k13 as initial values evolves system (1) for a period T, and get x1, y1, z1
Ep1=abs(ep1) Ep2=abs(ep2) Ep3=abs(ep3)
X0=(x(end)-(max(x)+min(x))/2)/(max(x)-min(x))/2
Y0=(y(end)-(max(y)+min(y))/2)/(max(y)-min(y))/2
Z0=(z(end)-(max(z)+min(z))/2)/(max(z)-min(z))/2
Iterate Eq.(2) 30 times, and get three iteration outputs ep1,ep2,ep3
X0, X10 as first group of input p, v in Eq.(2) Y0, Y10 as second group of input p, v in Eq.(2) Z0, Z10 as third group of input p, v in Eq.(2)
X10=(x1(end)-(max(x1)+min(x1))/2)/(max(x1)-min(x1))/2
Y10=(y1(end)-(max(y1)+min(y1))/2)/(max(y1)-min(y1))/2
Z10=(z1(end)-(max(z1)+min(z1))/2)/(max(z1)-min(z1))/2

Fig. 2. Block diagram

effect, unlike other pseudorandom number generation applications using chaos, which will be periodic over very long time evolution.

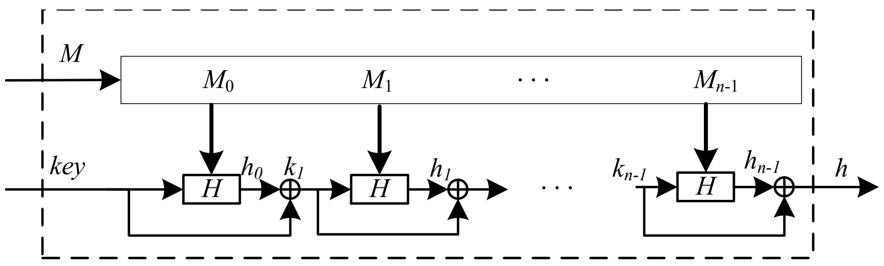

Fig. 3. Cipher Block Chaining (CBC) Hash function model

4. Performance analysis of the proposed Hash algorithm

4.1. *Sensitivity to plain-text and key*

The proposed Hash algorithm is performed for the following plain-texts (the initial secret key is "2A86D71ECB063FAC589B74132C3874AB"):

Plain-text 1: "Chaos is a deterministic process, which is ubiquitously present in the world. Because of its random like behavior, sensitivity to initial conditions and parameter values, ergodicity, and confusion and diffusion properties; chaotic cryptography has become an important branch of modern cryptography and has huge potential in protecting the assets."

Plain-text 2: Change the word "Chaos" in the Plain-text 1 into "chaos".

Plain-text 3: Change the word "values" in the Plain-text 1 into "value".

Plain-text 4: Change the full stop at the end of the Plain-text 1 into comma.

Plain-text 5: Add a blank space to the end of the Plain-text 1.

Plain-text 6: Replace the first "2" in the secret key with "3".

The corresponding Hash values in hexadecimal format using the composite multi-scroll attractor are given as:

"DCDBCFD54C4247EC4B9EBE3F04A98AC6" for Plain-text 1.

"19FDAABE34EB5C918D76A3510E770402" for Plain-text 2.

"07D9BB82659DDF07FDBF69CC0BE2014C" for Plain-text 3.

"E76E91CF3CC3E9CA7E17512B6E57FE43" for Plain-text 4.

"D1151DE14382409F70EC5B84BABDF06B" for Plain-text 5.

"FFD99F427133E4E93DC4E52CAB74DB35" for Plain-text 6.

The graphical display of the binary sequences for each plain-text using the composite multi-scroll attractor is shown in Fig. 4. Figure 4 indicates that the proposed algorithm using the composite multi-scroll attractor is so sensitive to the message and key that any tiny difference of the message or key cause significant changes in the final Hash value. For other attractors given by system (1) with different parameters, there are similar results.

4.2. *Analysis of one-way property of the proposed Hash function*

One-way property means that it is very easy to calculate the Hash value according to the plain-text and key, but it is very difficult to calculate the plain-text and key according to the final Hash value.

From mathematical point of view, the plain-text space is unlimited, but the resulting Hash is always a short string of fixed length. There are numerous plain-texts with the same Hash function value, but if the

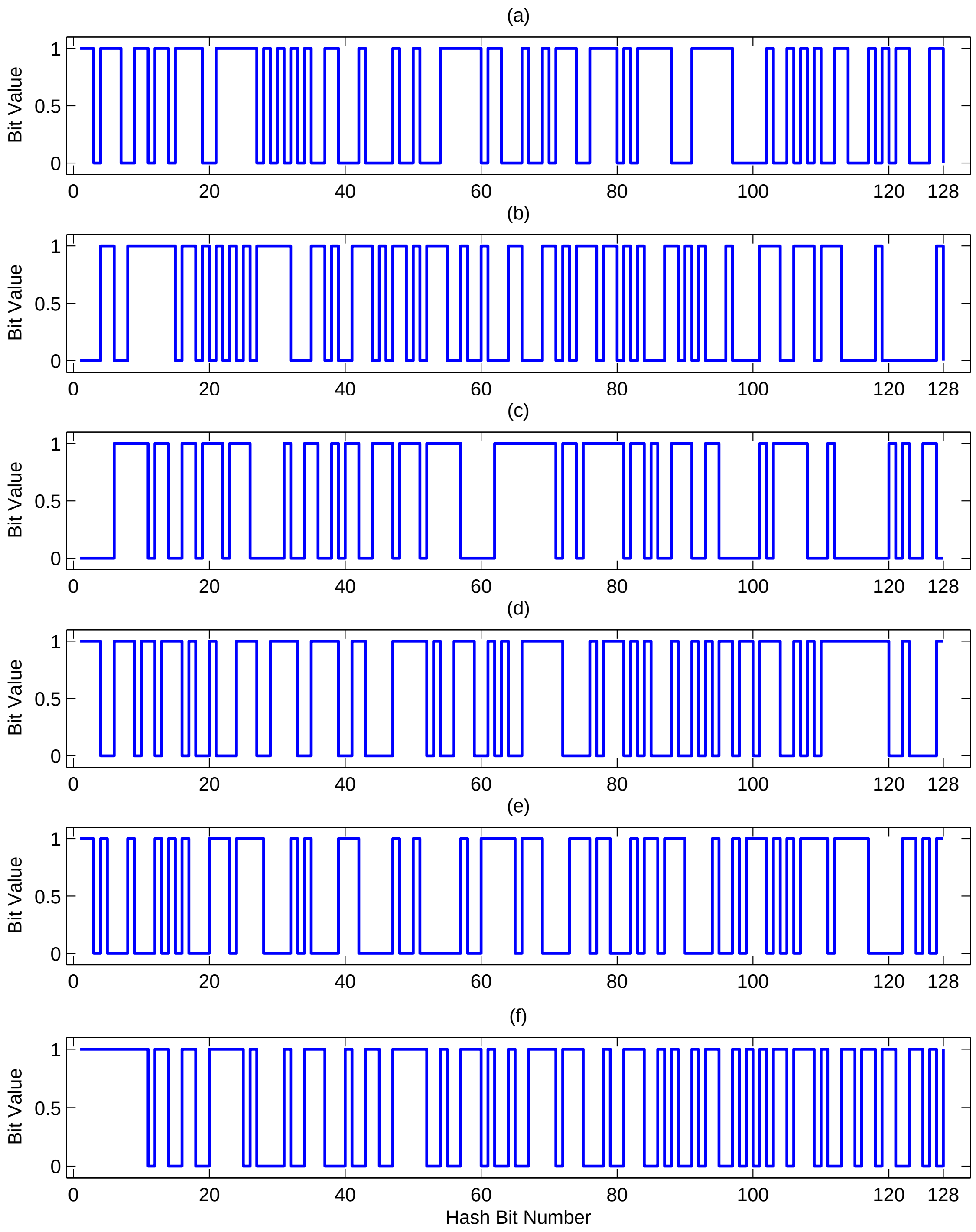

Fig. 4. Bit values of Hash function for Plain-texts 1 to 6 are shown in subplots (a) to (f), respectively.

Hash value reaches a certain bit length, for example, 128 bits, the resulting space has $2^{128} \approx 3.0428 \times 10^{28}$. It is difficult to calculate exhaustively in such a large space under the existing computing condition.

Generally, the key length should be no less than 128 bits, in order to prevent key exhaustive search attack; the length of the Hash value should not be less than 128 bits in order to prevent birthday attack.

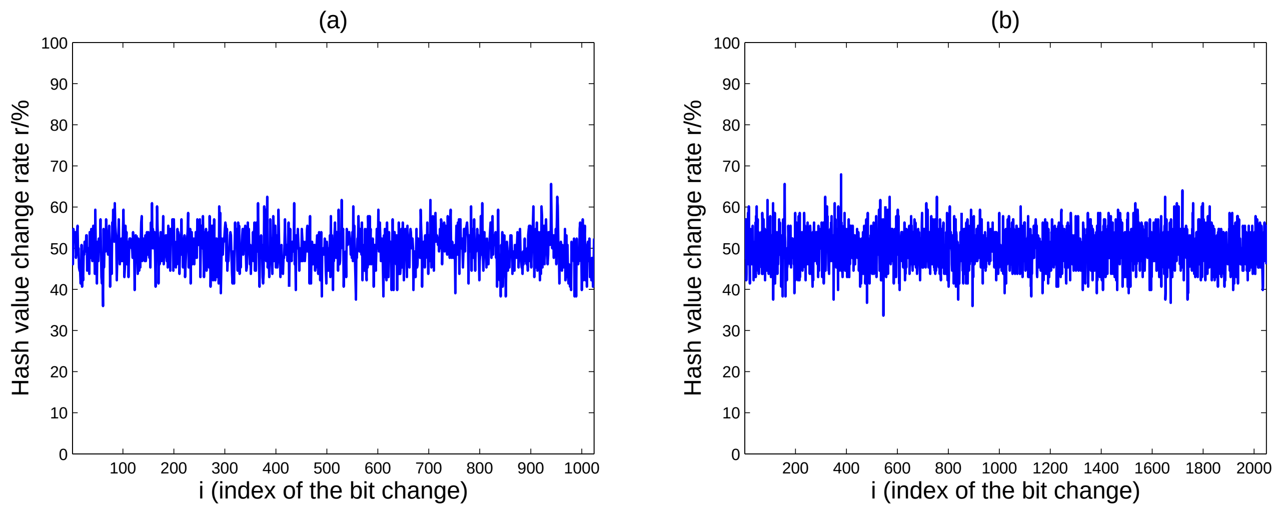

Fig. 5. The sensitivity analysis of the plain-text. (a) Bit change rates for the 1024 bits plain-text; (b) Bit change rates for the 2048 bits plain-text

4.3. *Statistical analysis of diffusion and confusion*

In this paper, the plain-text and secret key are confused and diffused by the evolution of the hyper-chaotic system with time delay feedback and repeated key-stream iteration. This mixed operation leads to better confusion and diffusion. Increased sensitivity to the plain-text and secret key are expected.

For the binary Hash function value, the ideal sensitivity is to ensure that any tiny changes, e.g., any "0" flip to "1" or any "1" flip to "0" in the plain-text or in the secret key will lead to about 50% bit change in the Hash function value. If the i-th bit is reversed, the corresponding Hash value is recorded as h_i, compared with the original Hash value h_0, and the number of different bits between h_i and h_0 is recorded as $D(h_0, h_i)$. Thus, the Hash values change rate for the i-th bit is defined by

$$r(i) = \frac{D(h_0, h_i)}{N_c} \times 100\%, \tag{5}$$

where N_c is the length of the Hash value; in this paper $N_c = 128$.

To measure the diffusion and confusion capabilities of the proposed Hashing scheme, we conduct the following experiment:

First, we obtain the Hash value of a randomly chosen message. Then, we randomly modify 1 bit in the original message and generate the corresponding Hash value. Finally, we compare the two Hash values and count the number of different bits in the Hash values denoted as B_i. Besides, we report the following statistical measures of the sequences B_i obtained in tests. $\overline{B} = \frac{1}{N}\sum_{i=1}^{N} B_i$ is the mean bit change number, $P = (\overline{B}/128) \times 100\%$ is the mean bit change rate, $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \overline{B})^2}$ is the standard variance of the bit change number, $\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i/128 - P)^2} \times 100\%$ is standard variance of P, where N indicates the test time of the experiments.

Employing the proposed algorithm using the composite multi-scroll attractor, the distribution of bit change number B_i is shown in Fig. 5 and Fig. 6. The Hash value change rate for 1024 test times, is shown in Fig. 5(a). For 2048 test times is shown in Fig. 5(b), the corresponding distribution of bit change number B_i with 2048 test times is shown in Fig. 6, the histogram of the bit change number in the Hash value when the i-th bit in the plain-text is reversed, B_i has a normal distribution centering at the ideal value of 64. These results show that the constructed Hash algorithm has a very good sensitivity to plain-text change.

Table 1 gives the statistic tests using the composite multi-scroll attractor for the different plain-texts with different length. The proposed algorithm has the mean bit change number $\overline{B} = 63.992$ and the mean bit change rate $P = 49.994\%$ that are close to the ideal values, i.e., 64 bits and 50%, respectively, which indicate that our algorithm has a strong capability of confusion and diffusion.

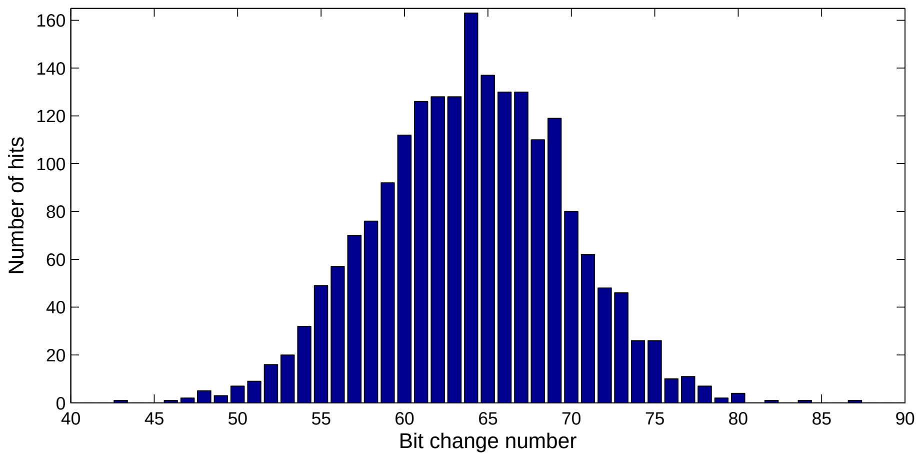

Fig. 6. Histogram of bit change number (2048 tests)

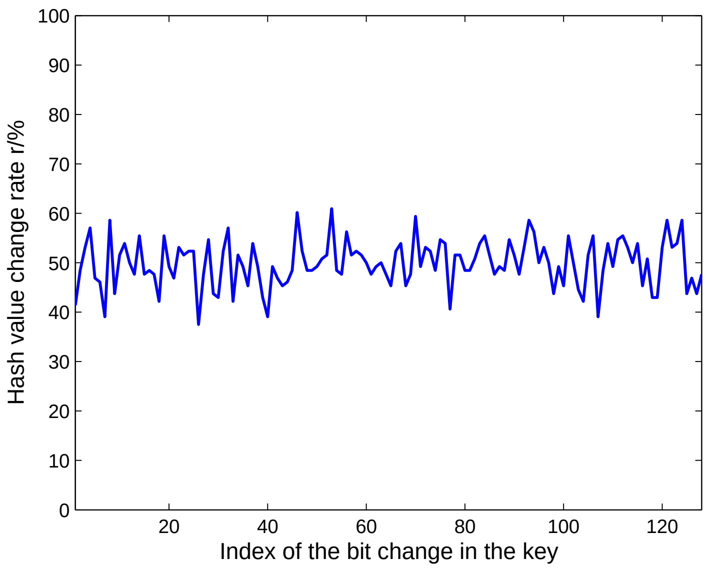

Fig. 7. The sensitivity analysis of the key (1024 tests)

Table 1. The sensitivity test of different plain-texts with different length

	N=200	N=512	N=1024	N=2048	N=10000	Mean
$\overline{B}$	63.96	64.19	63.88	63.97	63.96	63.992
ΔB	5.78	5.61	5.85	5.68	5.64	5.675
$P(\%)$	49.97	50.15	49.90	49.98	49.97	49.994
$\Delta P(\%)$	4.47	4.38	4.58	4.41	4.44	4.456

Table 2 gives the sensitivity test results of different length plain-texts using XOR operation to replace key-stream iteration in the proposed algorithm. By comparing Table 2 and Table 1, we see that the average bit change rate for 1 bit plain-text change, using the key-stream iteration operation, is 50%, while, that for the method using XOR is 49.94%. Therefore, the key-stream iteration operation is better than XOR

Table 2. The sensitivity of different length plain-texts using XOR operation to replace key stream iteration

	N=200	N=512	N=1024	N=2048	Mean
$\overline{B}$	63.17	63.60	64.35	64.11	63.81
ΔB	5.61	5.77	5.77	5.64	5.70
$P(\%)$	49.70	49.68	50.27	50.09	49.94
$\Delta P(\%)$	4.39	4.72	4.51	4.41	4.51

Table 3. The sensitivity test of the secret key for the plain–texts of different length

	N=150	N=512	N=1024	N=2048	Mean
$\overline{B}$	63.75	64.24	63.84	64.28	64.03
ΔB	5.841	5.801	6.101	5.17	5.73
$P(\%)$	49.81	50.19	49.88	50.23	50.02
$\Delta P(\%)$	4.57	4.53	4.47	4.04	4.40

operation in the sense of achieving better sensitivity to the plain-text.

Figure 7 is the sensitivity analysis of the secret key using the composite multi-scroll attractor for plain-text 1, the mean bit change rate of the Hash value is 49.8779% for 1 bit key reversing with 1024 tests. To test the key sensitivity in different plain-text lengths, Table 3 shows that 1 bit reversing secret key with the plain-text with different lengths brings the mean bit change number, $\overline{B}$, the mean bit change rate, P, the standard variance of the bit change number, ΔB, and the standard variance of P, ΔP.

From Figure 7 and Table 3, we conclude that the proposed method has very good sensitivity to the key.

4.4. ***Collision analysis***

Collision resistance and birthday attack are related to each other. Both reflect the probability that two input data are found to have the same Hash value. In what follows, we first perform qualitative analysis on birthday attack and then conduct quantitative analysis on collision resistance.

4.4.1. *Birthday attack*

When birthday attacks are considered, the Hash value length determines the security. The Hash value of 128 bits means that the birthday attack difficulty is in the order of 2^{64}, the difficulty magnitude of this attack is enough for general applications[Lin *et al.*, 2017c; Li & Li, 2016; Teh *et al.*, 2015; Kanso & Ghebleh, 2013].

4.4.2. *Collision test*

A collision occurs when two messages have the same Hash value. Collisions are unavoidable according to the pigeonhole principle. In a security application, it should be very difficult to find two messages with the same Hash value.

Since hyper-chaotic systems are usually defined for real numbers, it is not easy to provide a mathematical proof on the collision resistance of the chaotic Hash functions. Therefore, we perform the following test for the quantitative analysis on collision resistance:

Conventionally, 8 bits in the plain-text is selected, corresponding to the $ASCII$ value $0 \sim 255$, 8 bits in the Hash value is selected too, namely, the value is $0 \sim 255$. Therefore, the plain-text and Hash value share the same mapping space. Then, the number of the inverse image that corresponds to any value of the Hash value space is recorded as k, and the number of the point which has k inverse images is recorded as $N(k)$. The larger the number $N(1)$ and the smaller the $N(0)$, it means the better collision resistance.

n_k is used as a quantitative measure of collision resistance performance defined as:

$$n(k) = \frac{N(k)}{\sum_{k=0}^{K} N(k)}, \tag{6}$$

where K is the largest value of collision. The closer $n(1)$ value to 1, the lower the degree of the collision.

Using the aforementioned test method, Table 4 gives the collision resistance performance comparison for 8 bits length of the plain-text, which shows that the proposed algorithm has the best anti-collision performance.Meanwhile, in Table 4, the average time for generating Hash values for 30 Monte Carlo tests of plain-text 1 on the same computer platform, the average bit change number, $\overline{B}$, the average bit change rate, P, the average standard variance of the bit change number, ΔB, and the average standard variance of the change rate, ΔP, are also given. Table 4 is a comprehensive performance comparison of different methods, which demonstrates that the proposed algorithm has better comprehensive performance as compared with some existing algorithms.

As given in Table 4, the average time of the proposed method, i.e., 0.705s is faster than those of the methods in [Rivest, 1992; Wang, 2006; Li & Li, 2016; Lin *et al.*, 2017c] which are in the order of 1s. Although the methods in [Liu *et al.*, 2000; Peng *et al.*, 2005; Liu *et al.*, 2006a] have shorter average generating time of the order of 0.01s, the methods in [Liu *et al.*, 2000; Peng *et al.*, 2005; Liu *et al.*, 2006a] have weaker plain-texts sensitivity and/or collision resistance performance.

When 16 bits in the plain-text and Hash value are considered by the aforementioned way to test collision resistance, the proposed algorithm using the double-scroll attractor, $n(1) = 0.3694$, as shown by Table 5. The proposed algorithm has the best anti-collision performance when 16 bits Hash value and plain-text are considered as seen from Table 5.

In particularly, compared with the low dimensional chaotic behaviors generated by the original Chen system[Chen & Ueta, 1999] with one positive Lyapunov exponent, and the four dimensional hyper-chaotic Chen system[Wu & Wang, 2006] with two Lyapunov exponents, the chaos generated by Chen system with time delay[Ren *et al.*, 2006] has infinite dimension and multiple positive Lyapunov exponents, which has better diffusing and confusing ability. Table 4 and 5 show this point.

Table 4. The comprehensive performance comparison of the proposed method and some existing methods

Algorithms	$\sum_{k=0}^{K}$	$N(1)$	$n(1)$	Average time	$\overline{B}$	$P(\%)$	ΔB	$\Delta P(\%)$
Composite multi-scroll attractor	256	100	0.3906	0.705s	63.99	49.99	5.68	4.46
Double-scroll attractor	256	104	0.4063	0.705s	63.89	49.91	5.66	4.42
Single-scroll attractor	256	99	0.3867	0.705s	63.94	49.95	5.67	4.43
[Rivest, 1992]	256	60	0.2344	3.456s	63.99	49.99	5.58	4.36
[Liu *et al.*, 2000]	256	84	0.3281	0.029s	63.73	49.79	5.64	4.41
[Xiao *et al.*, 2005]	256	42	0.1641	0.231s	63.78	49.84	5.67	4.43
[Peng *et al.*, 2005]	256	87	0.3398	0.039s	64.11	50.09	5.57	4.33
[Yi, 2005]	256	26	0.1016	0.451s	63.59	49.68	6.47	4.15
[Liu *et al.*, 2006a]	256	101	0.3945	0.063s	61.24	47.84	5.23	4.09
[Liu *et al.*, 2006b]	256	87	0.3398	0.393s	63.76	49.66	5.69	4.45
[Wang, 2006]	256	44	0.1718	2.912s	63.56	49.26	5.64	5.34
[Ren & Zhuang, 2009]	256	95	0.3711	0.335s	63.79	49.83	5.43	4.24
[Kanso & Ghebleh, 2013]	256	80	0.3125	0.730s	64.01	50.03	5.60	4.42
[Li *et al.*, 2016]	256	63	0.2461	0.195s	63.92	49.94	5.80	4.54
[Li & Li, 2016]	256	76	0.2969	2.329s	64.19	50.14	5.64	4.44
[Lin *et al.*, 2017c]	256	88	0.3438	1.803s	63.88	49.76	5.78	4.51

Table 5. Collision resistance performance comparison among the proposed method and some existing methods (the plain-text with 16 bits length)

Algorithms	$\sum_{k=0}^{K} N(k)$	$N(1)$	$n(1)$
Composite multi-scroll attractor	65536	24080	0.3674
Double-scroll attractor	65536	24211	0.3694
Single-scroll attractor	65536	23933	0.3652
[Rivest, 1992]	65536	1468	0.0224
[Liu *et al.*, 2000]	65536	24094	0.3676
[Peng *et al.*, 2005]	65536	1506	0.0230
[Yi, 2005]	65536	116	0.0018
[Liu *et al.*, 2006a]	65536	23906	0.3648
[Liu *et al.*, 2006b]	65536	1916	0.0292
[Wang, 2006]	65536	6886	0.1051
[Ren & Zhuang, 2009]	65536	24109	0.3679
[Kanso & Ghebleh, 2013]	65536	23302	0.3556

Table 6. The collision resistance performance using XOR operation to replace key-stream iteration (for the double scroll attractor and 8 bits Hash value)

Algorithms	$\sum_{k=0}^{K}$	$N(1)$	$n(1)$
Key stream iteration	256	104	0.4063
XOR operation	256	80	0.3516

Table 7. The collision resistance performance using different compensating code methods (for the double scroll attractor and 8 bits Hash value)

Algorithms	$\sum_{k=0}^{K}$	$N(1)$	$n(1)$
Chaotic iteration	256	104	0.4063
Compensating 1 0	256	94	0.3672
Compensating 0 0	256	94	0.3672
Compensating 1 1	256	96	0.3750

In order to illustrate that the key-stream iteration method is being helpful to improve collision resistance performance, we give the collision resistance performance comparison in Table 6 for 8 bits Hash value and using XOR operation to replace the key-stream iteration in the proposed algorithms.

We can see from Table 6 that $n(1) = 0.3516$ for XOR operation, while, $n(1) = 0.4063$ for the key-stream iteration. This result shows that a better performance can be achieved by using the proposed key-stream iteration method as compared to the conventional XOR operation.

To show the performance of the proposed compensating code, we replace the compensating code generation algorithm of the proposed method with the conventional compensating code, i.e., using (1, 0) or (0, 0) or (1, 1) for compensation. The collision resistance performance comparison is given in Table 7. We can conclude from Table 7 that the proposed algorithm has better collision resistance performance.

Based on these results, the quantitative analysis of collision demonstrates that the proposed hyper-chaos based method has excellent collision resistance performance.

5. Conclusion

In this paper, we present a novel algorithm of one-way Hash function based on the hyper-chaotic attractors in the Chen system with linear time delay feedback. The advantage and innovation of this work lie in: Firstly, the theoretically infinite dimensional hyper-chaotic system with multiple positive Lyapunov exponents is used to construct the Hash function. Although only three states of the infinite dimensional system are used for constructing Hash function, they contain infinite dimensional dynamical system information, which yields the extremely complicated phase space and strong ability of diffusing and confusing. Secondly, other than commonly used XOR operation for further diffusing and confusing, key-stream iteration is used in the proposed method, which provides n times iteration for the plain-text and key to enhance the diffusing and confusing ability of the proposed method. Thirdly, compensating code generated by chaotic map in the message filling phase helps to improve the anti-collision performance of the proposed Hash algorithm. Fourthly, the chaotic evolution time T in the paper is relatively small for the computation precision of the commonly used commercial computers, the digital deterioration is not significant. The proposed algorithm achieves very competitive plain-text and the key sensitivity. Finally, the proposed algorithm has the best anti-collision performance with the reasonable computation cost.

References

Alvarez, G., Montoya, F. & Romera, M. [2004] “Cryptanalysis of dynamic look-up table based chaotic cryptosystems,” *Phys. Lett. A* **326**, 211-218.

Akhavan, A., Samsudin, A. & Akhshani, A. [2013] “A novel parallel Hash function based on 3D chaotic map,” *Eurasip. J. Adv. Sign. Process.* **2013**, 126.

Baptista, M. S. [1998] “Cryptography with chaos,” *Phys. Lett. A* **240**, 50-54.

Cang, S., Qi, G. & Chen, Z. [2010] “A four-wing hyper-chaotic attractor and transient chaos generated from a new 4-D quadratic autonomous system,” *Nonlinear Dyn.* **59**, 515-527.

Chen, G., Mao, Y. & Chui, C. K. [2004] “A symmetric image encryption scheme based on 3D chaotic cat maps,” *Chaos Solit. Fract.* **21**, 749-761

Chen, G. R. & Ueta, T. [1999] “Yet another chaotic attractor,” *Int. J. Bifurcation and Chaos* **9**, 1465-1466.

Chenaghlu, M. A., Jamali, S. & Khasmakhi, N. N. [2016] “A novel keyed parallel Hashing scheme based on a new chaotic system,” *Chaos Solit. Fract.* **87**, 216-225.

Choi, Y., Lee, Y., Moon, J. & Won, D. [2017] “Security enhanced multi-factor biometric authentication scheme using bio-Hash function,” *Plos One* **12**, e0176250.

Dobbertin, H. [1996] “Cryptanalysis of md4,” *Lect. Notes Comput. Sci.* **1039**, 53-69.

Fu, C., Chen, J., Zou, H. & Meng, W. H. [2012] “A chaos-based digital image encryption scheme with an improved diffusion strategy,” *Opt. Express.* **20**, 2363-2378.

Gao, T. & Chen, Z. [2008] “A new image encryption algorithm based on hyper-chaos,” *Phys. Lett. A* **372**, 394-400.

Guo, W., Wang, X. & He, D. [2009] “Cryptanalysis on a parallel keyed Hash function based on chaotic maps,” *Phys. Lett. A* **373**, 3201-3206.

Jeng, F. G., Huang, W. L. & Chen, T. H. [2015] “Cryptanalysis and improvement of two hyper-chaos-based image encryption schemes,” *Signal Process Image Commun.* **34**, 45-51.

Jiteurtragool, N., Ketthong, P., Wannaboon, C. & San-Um, W. [2013] “A topologically simple keyed Hash function based on circular chaotic sinusoidal map network,” *Int. Conf. Adv. Commun. Technol. ICACT.* 1089-1094.

Kanso, A. & Ghebleh, M. [2013] "A fast and efficient chaos-based keyed Hash function," *Commun. Nonlin. Sci. Numer. Simul.* **18**, 109-123.

Kanso, A. & Ghebleh, M. [2015] "A structure-based chaotic Hashing scheme," *Nonlinear Dyn.* **81**, 27-40.

Knudsen, L.& Preneel, B. [2002] "Construction of secure and fast Hash functions using non-binary error-correcting codes," *IEEE Trans. Inf. Theory.* **48**, 2524-2539.

Li, S., Li, Q., Li, W., Mou, X. & Cai, Y. [2001] "Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudorandom coding," *Proc. IMA Int. Conf. Cryptography and Coding*, pp. 205-221.

Li, S., Mou, X., Yang, B. L., Ji, Z. & Zhang, J. [2003] "Problems with a probabilistic encryption scheme based on chaotic systems," *Int. J. Bifurcation and Chaos* **13**, 3063-3077.

Li, Y., Li, X. & Liu, X. [2016] "A fast and efficient Hash function based on generalized chaotic mapping with variable parameters," *Neural Comput. Appl.* **28**, 1405-1415.

Li, Y. T. & Li, X. [2016] "Chaotic Hash function based on circular shifts with variable parameters," *Chaos Solit. Fract.* **91**, 639-648.

Lin, Z. S., Guyeuxy, C., Wang, Q. X. & Yu, S. M. [2017a] "Diffusion and Confusion of Chaotic Iteration Based Hash Functions," *Proc. - IEEE Int. Conf. Comput. Sci. Eng., IEEE Int. Conf. Embed. Ubiquitous Comput., Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci., CSE-EUC-DCABES* pp:444-447.

Lin, Z. S., Guyeux, C., Yu, S. M., Wang, Q. X. & Cai, S. T. [2017b] "On the use of chaotic iterations to design keyed hash function," *Cluster Comput* **2**, 1-15.

Lin, Z. S., Yu, S. M. & Lü, J. H. [2017c] "A novel approach for constructing one-way Hash function based on a message block controlled 8D hyperchaotic map," *Int. J. Bifurcation and Chaos* **27**, 1750106.

Liu, G. J. Shan, L. & Dai, Y. W. [2006a] "One-way Hash function based on chaotic neural network," *Acta Phys. Sin.* **55**, 5688-5693.

Liu, G. J., Liang, S., Sun, J. S., Dai, Y. W. & Wang, Z. Q. [2006b] "Construction of Hash function based on spatiotemporal chaotic systems," *Control Decis.* **21**, 1244-1248.

Liu, J. N., Xie, J. H. & Wang, P. [2000] "One way Hash function construction based on chaotic mappings," *J. Tsinghua Univ.* **40**, 55-58.

Luo, Y. L. & Du, M. H. [2012] "One-way Hash function construction based on the spatiotemporal chaotic system," *Chin. Phys. B* **21**, 84-93.

Mendel, F., Nad, T. & Schläffer, M. [2013] "Improving local collisions: new attacks on reduced SHA-256," *Lect. Notes Comput. Sci.* **7881**, 262-278.

Murillo-Escobar, M. A., Cruz-Hernández, C., Cardoza-Avendaño, L., & Méndez-Ramírez, R. [2017] "A novel pseudorandom number generator based on pseudorandomly enhanced logistic map," . *Nonlinear Dyn.* **87**, 407-425.

Peng, F., Qiu, S. S. & Long, M. [2005] "One-way Hash function construction based on two-dimensional hyper-chaotic mappings," *Acta Phys. Sin.* **54**, 4562-4568.

Ren, H. P., Liu, D. & Han, C. Z. [2006] "Anti-control of chaos via direct time delay feedback," *Acta Phys. Sin.* **55**, 2694-2701.

Ren, H. P. & Zhuang, Y. [2009] "One-way Hash function construction based on Chen-type hyper-chaotic system and key-stream," *J. Commun.* **30**, 100-113.

Ren, H. P. & Li, W. C. [2010] "Heteroclinic orbits in Chen circuit with time delay," *Commun. Nonlin. Sci. Numer. Simul.* **15**, 3058-3066.

Ren, H. P., Bai, C., Huang, Z. Z. & Grebogi, C. [2017a] "Secure Communication with Hyper-Chaotic Chen System," *Int. J. Bifurcation and Chaos* **14**, 1750076-1-15.

Ren, H. P., Bai, C., Tian, K. & Grebogi, C. [2017b] "Dynamics of delay induced composite multi-scroll attractor and its application in encryption," *Int. J. Non. Linear Mech.* **94**, 334-342.

Rhouma, R. & Belghith, S. [2008] "Cryptanalysis of a new image encryption algorithm based on hyper-chaos," *Phys. Lett. A* **372**, 5973-5978.

Rivest, R. [1992] "The MD5 Message-Digest Algorithm," *IETF Network Working Group,* **473**, 492.

Stevens, M. [2013] "New collision attacks on SHA-1 based on optimal joint local-collision analysis," *Lect. Notes Comput. Sci.* **7881**, 245-261.

Teh, J. S., Samsudin, A. & Akhavan, A. [2015] "Parallel chaotic Hash function based on the shuffle-exchange network," *Nonlinear Dyn.* **81**, 1067-1079.

Wang, X., Lai, X. & Feng, D. [2005a] "Cryptanalysis of the Hash Functions MD4 and RIPEMD," *Lect. Notes Comput. Sci.* **3494**, 1-18.

Wang, X. & Yu, H. [2005b] "How to break MD5 and other Hash functions," *Lect. Notes Comput. Sci.* **3494**, 19-35.

Wang, L. [2006] "Research of one way Hash function based on logistic mapping," *Comput. Eng. Des.* **27**, 774-776.

Wang, Y., Wong, K. W. & Liao, X. F. [2011a] "A new chaos-based fast image encryption algorithm," *Appl. Soft Comput. J.* **11**, 514-522.

Wang, Y., Wong, K. W. & Xiao, D. [2011b] "Parallel Hash function construction based on coupled map lattices," *Commun. Nonlin. Sci. Numer. Simul.* **16**, 2810-2821.

Wong, K. W. [2003] "A combined chaotic cryptographic and Hashing scheme," *Phys. Lett. A* **307**, 292-298.

Wong, K. W., Kwok, B. S. H. & Yuen, C. H. [2009] "An efficient diffusion approach for chaos-based image encryption," *Chaos Solit. Fract.* **41**, 2652-2663.

Wu, X. J., & Wang, X. Y. [2006] "Chaos synchronization of the new hyperchaotic Chen system via nonlinear control," *Acta. Phys. Sin.* **55**, 6261-6266.

Xiao, D., Liao, X. F. & Deng, S. J. [2005] "One-way Hash function construction based on the chaotic map with changeable-parameter," *Chaos Solit. Fract.* **24**, 65-71.

Xiao, D., Liao, X. & Wang, Y. [2009] "Improving the security of a parallel keyed Hash function based on chaotic maps," *Phys. Lett. A* **373**, 4346-4353.

Xiao, D., Shih, F., Y. & Liao, X. [2010] "A chaos-based Hash function with both modification detection and localization capabilities," *Commun. Nonlin. Sci. Numer. Simul.* **15**, 2254-2261.

Yang, Y. G., Xu, P. & Yang, R. [2016] "Quantum Hash function and its application to privacy amplification in quantum key distribution, pseudo-random number generation and image encryption," *Sci. Rep.* **6**, 19788.

Ye, G., Zhao, H., & Chai, H. [2015] "Chaotic image encryption algorithm using wave-line permutation and block diffusion," *Nonlinear Dyn.* **83**, 1-11.

Yi, X. [2005] "Hash function based on chaotic tent maps," *IEEE Trans. Circuits Syst. Express Briefs.* **52**, 354-357.

Zhou, Q., Liao, X. & Liu, J. [2012] "Design of image Hash functions based on fluid dynamics model," *Nonlinear Dyn.* **67**, 1837-1845.