Journal of Computers Vol. 33 No. 3, June 2022, pp. 85-97
doi: 10.53106/199115992022063303007

Design and Analysis of Hash Function Based on Two-dimensional Integer Chaotic Map

Yu-Jie Liu*, Jian-Dong Liu, Ming Zhong, Bo Li, Hao-Qiang Xu

College of Information Engineering, Beijing Institute of Petrochemical Technology, Beijing, 102617, China
liuyujiechn@qq.com

Received 14 July 2021; Revised 14 November 2021; Accepted 14 December 2021

Abstract. Combining two-dimensional coupled map lattice and dynamic integer tent map, a hash function construction scheme with variable length output is proposed. The dynamic integer tent map is used as the non-linear function of two-dimensional coupled map lattice, and dynamic parameters are added to the two-dimensional coupled map lattice, the bit logic decision function of the dynamic integer tent map is used to control the change of the dynamic parameters. At the same time, the dynamic parameters are evolved by the cyclic displacement mode, which enhances the correlation between the plaintext difference and the parameters. The test results show that this hash function has strong security, simple implementation, and can be used as an ideal algorithm to replace the traditional hash function.

Keywords: hash function, chaos, two-dimensional coupled map, tent map, dynamic

1 Introduction

With the development of science and technology, the number of people using the Internet is increasing all over the world. Countless electronic devices are connected through the Internet, especially the rise of smart phones. The Mobile Internet has increased the daily activity of the Internet. The following information security problems have been unavoidable. Developers, both PC and mobile, are usually concerned with five main services when exchanging information over the Internet: confidentiality, authenticity, integrity, non-repudiation and availability. Hash function plays an important role in the scene of information transmission, such as encryption and web applications, and was mainly used in digital signature and message authentication to verify the integrity and authenticity of information [1].

The main design principle of traditional hash function is MD iterative structure. The mechanism of mixed operations is very similar in all rounds, using integer modulo addition and logical functions, etc. Traditional hash functions are represented by MD4, MD5, etc. Xiao-Yun Wang and others announced collision examples including MD4, MD5, HAVAL-128 and RIPEMD at Crypto'2004. Then lucks and Magnus constructed "meaningful collision" for MD5, which made the collision of MD5 not only in theory, but also in practical application fields have a huge impact. These hash functions are found collision examples and broken almost at the same time, which shows that there are defects in their design principle, so the cryptography community has been looking for alternative methods [2-4].

A new direction for building hash function based on chaos first appeared in 2002. The properties of chaos include: sensitivity to small changes of initial conditions and parameters, quasi random behavior, ergodicity, unstable periodic orbits with long period and expected confusion and diffusion [2, 4]. Researchers in related fields apply chaos system to the design of hash function, such as hash functions based on tent map or logistic map, but chaos system usually use floating-point operation, which is less efficient than bit operation, and the combination of complex design and floating-point representation also makes the security of hash functions difficult to verify. In reference [5], the tent map is designed in integer form, and combined with the bidirectional coupled map lattice model, proposed a hash function of integer coupled tent map, obtained a secure and efficient hash function. In reference [6], the two-dimensional logistic map method is applied to the encryption algorithm of AES to enhance its security, which has a significant optimization effect. In reference [7], a new image encryption scheme based on two-dimensional coupled map lattice map model is proposed to enhance the image encryption performance by exploiting the complex dynamical behavior of the two-dimensional coupled image lattice. It can be seen that chaos map is designed in integer form and dimensionality enhancement of chaos systems are effective methods for researchers to enhance the security of the algorithm. On this basis, this paper investigates the integer form

* Corresponding Author

and dynamic of tent map, constructing hash function with two-dimensional coupled map lattice model and analyzes its test indexes. In this scheme, the one-dimensional coupled map lattice is extended to two-dimensional, and the diffusion in two directions is extended to four. At the same time, the dynamic integer tent map is taken as the main nonlinear function, and the logical decision function of tent map is used to confuse message and shift dynamic parameters circularly, so as to enhance the correlation between plaintext difference and the parameters. The parallel iterative structure is used in the compression function, which makes the algorithm's realize faster in hardware and software. The existence of dynamic variables also makes the compressed functions have not only state variable links, but also the same number of dynamic link parameters, which improves the anti-collision ability of hash function without adding intermediate state variables. Finally, the hash values of 128 bits, 256 bits and 512 bits can be selected to improve the usability of the algorithm. Compared with other construction methods combining chaos systems and Hash functions, proposed method in this paper is innovative in that it converts the real domain calculation of traditional chaos systems to the integer domain, solves the error caused by the limited accuracy of computers, adds dynamic parameters in the tent map to solve its short-period problem, and then uses the dynamic integer tent map as the lattice point of the two-dimensional coupled map, combines the characteristics of two chaos models, increases the overall confusion and diffusion nature, improves the algorithm security, and the algorithm can choose different lengths of output values as needed, which is more flexible.

The rest of this paper is organized as follows: In the second section, dynamic integer tent map and two-dimensional coupled lattice map system are briefly introduced. In the third section, hash function construction process based on two-dimensional coupled dynamic integer tent map is introduced in detail. In the fourth section, various tests are carried out to evaluate the performance of the proposed algorithm. The work is summarized in the fifth section.

2 Two-dimensional Coupled Dynamic Integer Tent Map System

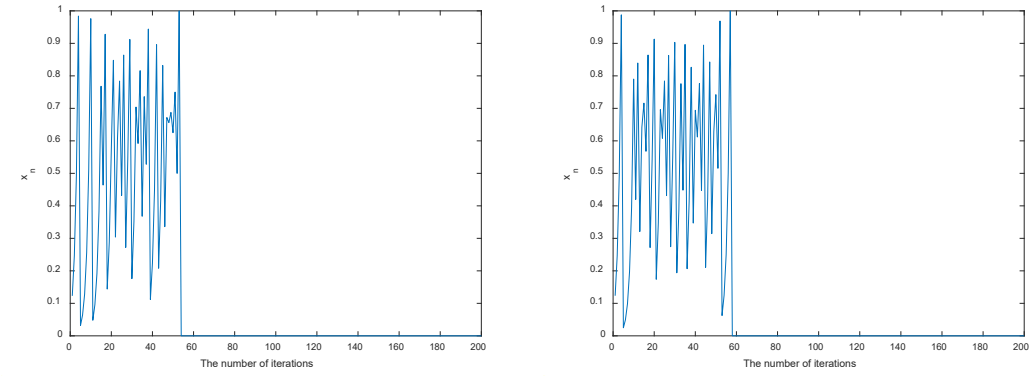
2.1 Dynamic Integer Tent Map

Tent map is a kind of nonlinear map essentially, which has a general mechanism of generating chaos, so it is often used as a simple chaos model. Its expression is as follows [8].

$$F_\alpha : x_{n+1} = \begin{cases} \dfrac{x_n}{\alpha}, & 0 \le x_n < \alpha \\ \dfrac{1 - x_n}{1 - \alpha}, & \alpha \le x_n \le 1 \end{cases}. \tag{1}$$

However, the chaos map is defined in the real number domain. In the finite precision implementation, the computer cannot deal with irrational numbers, and an aperiodic orbit cannot be generated.

In the case of double precision, the simulation experiment is carried out in equation (1). Due to the error caused by the limited accuracy, the obtained result is slightly different from the theoretical analysis. However, no matter what the initial value is, the final result is still 0 after more than 50 iterations (as shown in Fig. 1(a) and Fig. 1. (b)).



(a) α=0.5, x0=123 (b) α=0.5, x0=123456789

Fig. 1. Tent map time series

The tent map is equivalent to an integer field operation from a real number field, and its specific expression is as follows.

$$\mathrm{F}: x_{n+1} = \begin{cases} 2x_n, & 0 \le x_n < 2^{k-1} \\ 2(I - x_n) + 1, & 2^{k-1} \le x_n \le I \end{cases}. \tag{2}$$

Where $I=2^k-1$, the multiplication (division) operation of equation (1) is transformed into the shift operation of equation (2) in the finite integer field. However, as it is still defined in a finite domain, the iterative sequence generated by it is bound to enter the periodic state, there are even some cycle points with very small cycle lengths. For example, when the initial value $x_0=1$ and k=4, its iteration is shown in the Fig. 2. It can be seen that its value circulates in a relatively short period.

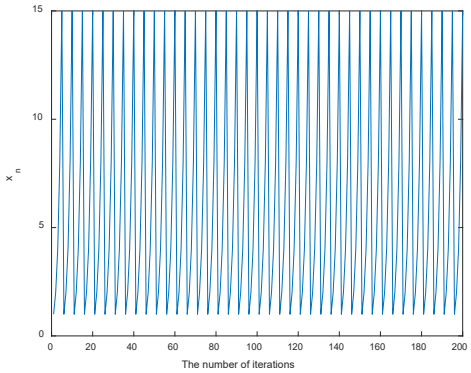


Fig. 2. Integer tent map iterations

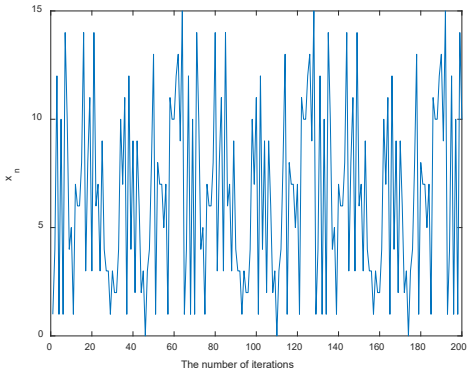


Fig. 3. Dynamic integer tent map iteration

Then the dynamic parameters are introduced to obtain the dynamic integer tent map. It not only solves the problem of short period of integer tent map, but also has the characteristic of uniform distribution of tent map, as shown in Fig. 4. Its specific expression is as follows:

$$\mathrm{F}: \mathrm{x_{n+1}} = \begin{cases} 2\mathrm{g_n}, & 0 \le \mathrm{g_n} < 2^{\mathrm{k-1}} \\ 2\left(2^{\mathrm{k}} - \mathrm{g_n}\right) + 1, & 2^{\mathrm{k-1}} \le \mathrm{g_n} \le 2^{\mathrm{k}} - 1 \end{cases}. \tag{3}$$

The value of g_n is relate with x_n and iterative steps, formula is as follows.

$$\mathrm{g_n} = (\mathrm{x_n} + \mathrm{n}) \bmod 2^{\mathrm{k}}. \tag{4}$$

n is the dynamic parameter and the number of iterative steps, which represents the lateral movement distance of the tent and controls its horizontal movement. x_n represents the iteration result of the nth step. 2^k is the upper bound of x_n, this paper take the upper bound is 2^{32}. In the iterative operation, with the progress of the iteration, n takes different values, which not only ensures the dynamism but also ensures the stability of the algorithm, and the iteration value will not be changed due to the change of dynamic parameters. When the initial value x_0= 1, k = 4, its iteration is shown in the Fig. 3. It can be seen that cycle has greatly increased. Fig. 4. shows the theoretical case of the dynamic tent map with its iterative condition when there are dynamic parameters n for perturbation.

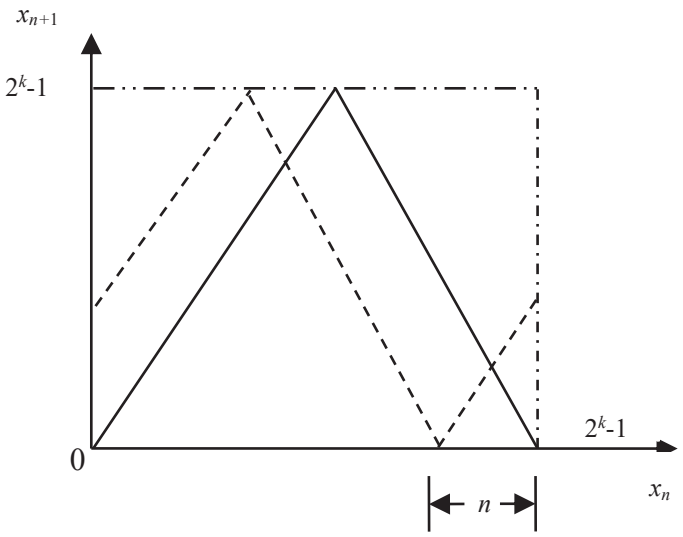


Fig. 4. Dynamic integer tent map

2.2 Two-dimensional Coupled Map Lattice Model

At present, in the research of spatiotemporal chaos system, the most popular model is coupled map lattice (CML). Due to the high efficiency of numerical diffusion and the direct use of the known results of chaos theory, this model has attracted much attention in the research of spatiotemporal chaos and is becoming an important branch in the field of nonlinear dynamics. The initial values and sensitivities of one-dimensional unidirectional coupled map lattice model and one-dimensional bidirectional coupled map lattice model are compared in detail in reference [9]. In the case of the same number of grids, the initial grid is perturbed by a very small order of magnitude, and the error of iterative sequence is observed, it is obvious that the bidirectional coupled map lattice model is much better than the unidirectional coupled map lattice model. This paper uses simplified two-dimensional coupled lattice map, a lattice can diffuse in four directions in space, and with the evolution of time iteration, each lattice will affect all other lattices, which greatly improves the confusion and diffusion degree of the model. Compared with the one-dimensional bidirectional coupled map lattice model, though the two-dimensional coupled map lattice model increase the computational complexity by two times, but its information diffusion speed increases from 2n + 1 to $2n^2 + 2n + 1$, the diffusion speed is greatly improved. The dynamic integer tent map is regarded as a nonlinear function of two-dimensional coupled map lattice model:

$$x_{(n+1)}(i,j) = [f(x_{(n)}(i-1,j) + f(x_{(n)}(i+1,j) + f(x_{(n)}(i,j-1) + f\big(x_{(n)}(i,j+1)\big] mod\ 2^{32}\ . \quad \textbf{(5)}$$

Where n is the number of iteration steps, i, j = 1,2,…,L, denotes the coordinates of two-dimensional lattices, the values of boundary lattices are x(n)(0,j) = x(n)(L, j), x(n)(L+1,j) = x(n)(1,j), x(n)(i,0) = x(n)(i, L), x(n)(i,L+1) = x(n)(i,1), (i, j) represents the coordinates of the lattices of two-dimensional, which includes not only the local reaction process of two independent components i and j, but also the diffusion reaction process of these two independent components on all lattices of the system with length L. Fig. 5 shows the lattice diffusion process of two-dimensional coupled map when L = 5 (we chose L=5 for better demonstration, and chose L=4 for actual design). The white lattice is the virtual lattice. The figure shows the information diffusion of lattice points in equation. (5) after three iterations. It can be seen that when the summary point is 25, the information of each lattice point will diffuse to all lattices after about four iterations, which is much faster than the number of one-dimensional lattices of the same scale.

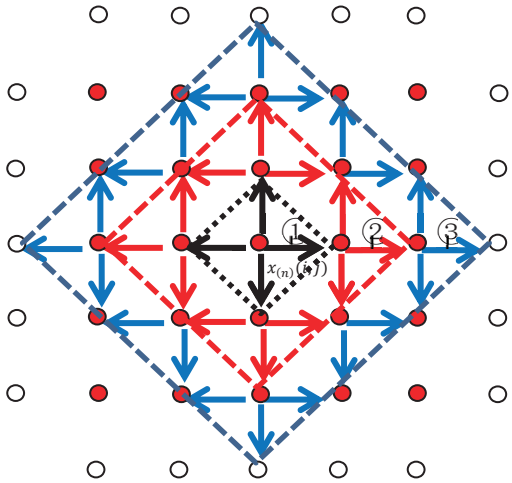


Fig. 5. Lattice diffusion process of two-dimensional coupled map

3 Implementation of the Algorithm

Symbol description: +: add operation of mod 2^{32}. ∼ : bit by bit logical inversion. ⊕ : bit by bit logical XOR. <<<: cyclic left shift operation. >>>: cyclic right shift operation, define $D = 2^{31}$, the dynamic integer tent map (equation. (3)) can use the ternary operator (? :) in C language described as:

$$x_{n+1} = g_n < D ? g_n \ll 1 : (\sim g_n \ll 1) + 1 . \quad \textbf{(6)}$$

This shows that dynamic integer tent map can be implemented by simple logic judgment, logic inversion and shift operation. If it is implemented by assembly language or hardware, its operation can be further simplified as: whether the highest bit of the test word is 0, if it is 0, it will shift to the left, otherwise, it will shift to the left and add 1 after the inversion. In the following hash algorithm design, the logic decision result of dynamic integer tent will also import parameter term k_i, unified use ternary operator (? :) description. Select the lattices size is 4×4. The general process of constructing hash function of two-dimensional coupled dynamic integer tent map is given below.

(1) Split and fill. The way of MD5 algorithm is used for splitting and filling. Divide the message M into 512bits plaintext with M_0, … , Mn,…, Mt and the last block is filled with: Mt = **10…0length (M), where length (M) represents the binary form of the length of M, the length of which is 64 bits. If the length is less than 64 bits, a character 1 is added to the high bit and then 0 is added.

(2) Each message block is divided into 16 bits message words, m_0, m_1,…, m_{15}.

(3) The initial vectors $x_{(0)}(i,j)$, $i, j = 1, ..., 4$

0x01234567UL; 0x89abcdefUL; 0x3210fedcUL; 0xba987654UL;
0x02468aceUL; 0x76543210UL; 0xfedcba98UL; 0xcdef0123UL;
0x456789abUL; 0xeca865420UL; 0x083b07fcUL; 0x192a26edL;
0x3a1945deUL; 0x3b0864cfUL; 0x6ed6c102UL; 0x7fc4e083UL;

(4) The initial values of 16 32-bits dynamic parameters $k_i^{(0)}$, $i = 0, ..., 15$.

0x5a827999UL; 0x6ed9eba1UL; 0x8f1bbcdcUL; 0xca62c1d6UL;
0x5793c62aUL; 0x66a6b778UL; 0x707b448dUL; 0xb7df971UL;
0x57d0f4dbUL; 0x4bde569UL; 0x6fbb8051UL; 0x2f41e8a8UL;
0x2f747580UL; 0xca62c1d6UL; 0x70f62baeUL; 0x4ada59cfUL;

(5) Embedded message:

$x_{(0)}(i, j) = m_t + x_0(i, j)$, $i, j = 1, ..., 4$, $t = 0, ..., 15$

(6) Compression function: a total of 16 iterations are carried out. In each iteration, dynamic transformation is carried out to combine the lattice value with the number of iterations, then map transformation is used in equation.(3), and finally diffusion iteration is carried out by using the coupled map model given in equation.(5).

1) $g_n(i, j) = x_n(i, j) + n$, n is the number of iterations, i, j = 1, ..., 4, n = 0, ..., 15.
2) $x_{(n)}(i, j) = g_{(n)}(i,j) < D? g_{(n)}(i,j) << 1: (k_n = k_n >>> 1, (\sim g_n(i,j) << 1) + 1)$
3) $x_{(n+1)}(i, j) = (x_{(n)}(i - 1, j) <<< 4) + (x_{(n)}(i + 1, j) <<< 8) + (x_{(n)}(i, j- 1) <<< 12) + (x_{(n)}(i, j + 1)$

(7) Add $x_{(0)}(i, j)$ into new value of $x(i, j)$:
$x(i, j) = x(i, j) + x_{(0)}(i, j)$, i, j = 1, ..., 4.
Add $k_i^{(0)}$ into new value of k_i:
$k_i = k_i + k_i^{(0)}$, i = 0, ..., 15.
(8) Do the process(5)-(7)with last message block，end with last one.
(9) Choose the length of output in 128,256,512 bits.
128bits hash value:
$(x(1,1) \oplus x(1,2) \oplus x(1,3) \oplus x(1,4))||$
$(x(2,1) \oplus x(2,2) \oplus x(2,3) \oplus x(2,4))||$
$(x(3,1) \oplus x(3,2) \oplus x(3,3) \oplus x(3,4))||$
$(x(4,1) \oplus x(4,2) \oplus x(4,3) \oplus x(4,4))$.
256bits hash value:
$(x(1,1) \oplus x(1,2)) \,||\, (x(1,3) \oplus x(1,4)) \,||\, (x(2,1) \oplus x(2,2)) \,||\, (x(2,3) \oplus x(2,4))||$
$(x(3,1) \oplus x(3,2)) \,||\, (x(3,3) \oplus x(3,4)) \,||\, (x(4,1) \oplus x(4,2)) \,||\, (x(4,3) \oplus x(4,4))$.
512bits hash value:
$x(1,1) \,||\, x(1,2) \,||\, x(1,3) \,||\, x(1,4) \,||\, x(2,1) \,||\, x(2,2) \,||\, x(2,3) \,||\, x(2,4)||$
$x(3,1) \,||\, x(3,2) \,||\, x(3,3) \,||\, x(3,4) \,||\, x(4,1) \,||\, x(4,2) \,||\, x(4,3) \,||\, x(4,4)$.

4 Experimental Results and Analysis

4.1 Numerical Distribution Analysis

The uniform distribution of hash value means that the hash value is evenly distributed in the bucket, and its result is closely related to the security of hash function [10]. Select the initial text as “Cryptographic hash functions play a fundamental role in modern” cryptography. While related to conventional hash functions commonly used in non-cryptographic computer applications - in both cases, larger domains are mapped to smaller ranges - they differ in several important aspects. Our focus is restricted to cryptographic hash Functions (here after, simply hash functions), and in particular to their use for data integrity and message authentication. “As shown in Fig. 6, the dispersion range of the initial text is (32-121), the printable character range in the ASCII code table. The corresponding hash value (hexadecimal) is 454bad739ef3bb3f75e51a2941e6f249. The corresponding distribution is shown in Fig. 7, which shows that it has a good uniform distribution. In addition, add an extreme case. When the initial text all is “space”, the ASCII value of hash operation is “48”, as shown in Fig. 8. The obtained hash value is 345a105e662238d1e18680168dcc99ea, and its distribution is shown in Fig. 9. It can be seen that the obtained hash value also presents a uniform distribution in extreme cases.

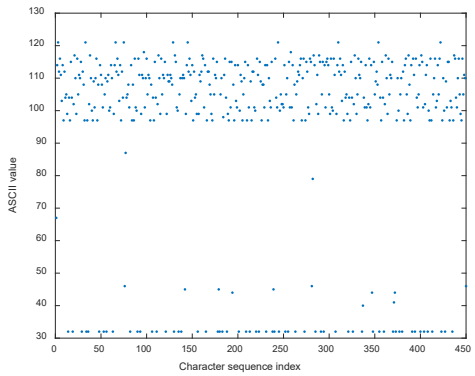


Fig. 6. Message distribution

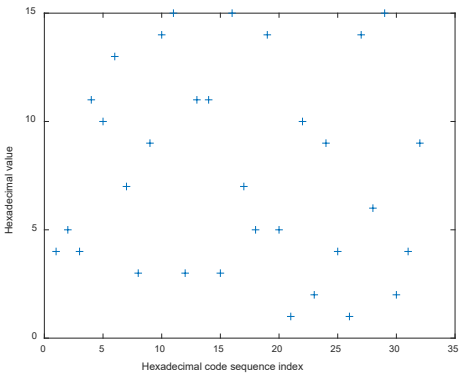


Fig. 7. Hash value distribution

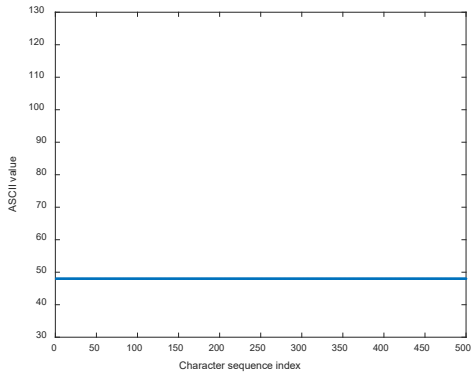


Fig. 8. Space message distribution

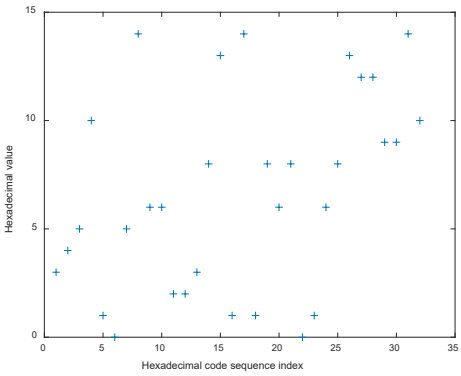


Fig. 9. Hash value distribution of space message

4.2 Sensitivity Test

In order to test the sensitivity of hash algorithm to the initial text, the initial text is slightly changed, and the hash values are compared and analyzed. The simulation experiments are carried out in the following five different situations.

The output length is 128 bits.

Condition 1: The initial text given in 4.1.

Condition 2: Change the first C in the initial text to B.

Condition 3: Change the first “functions” in the initial text to “function”.

Condition 4: Change the period at the end of the text to a comma.

Condition 5: Add a space at the end of the text.

The hash value expressed in hexadecimal is as follows. The results are expressed in binary and the output results are shown in the Fig. 10. It can be seen that small changes in the plaintext can have a great impact on the hash value.

Hash value of condition 1: 454bad739ef3bb3f75e51a2941e6f249.

Hash value of condition 2: e9b78c957ca26b44a0a9f0256ee927c.

Hash value of condition 3: 9bdea6eff82ea87739362f87754c8fe1.

Hash value of condition 4: 671682e351a70d5bf5cbb83fc7c7d59.

Hash value of condition 5: 4d9d58ae2e8e39bddcef3f739474a29.

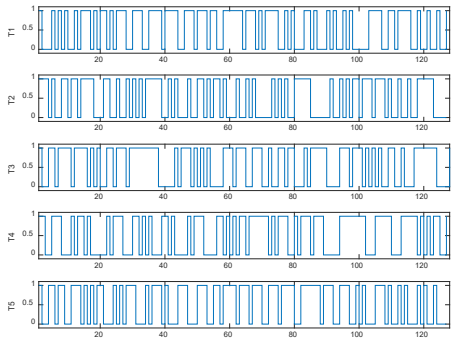


Fig. 10. Sensitivity test

4.3 Statistical Analysis of Confusion and Diffusion

Confusion and diffusion are two important properties of secure cryptographic operations in cryptography, which are recognized by Claude Shannon [11]. Confusion means that the relationship between the message and the hash value obtained by the algorithm must be complex and unpredictable, while diffusion means that the hash value is highly dependent on the message.

This paper uses the following method to detect the confusion and diffusion properties of the algorithm. Firstly,

a message is randomly selected to generate the corresponding hash value of the message, then a bit in the message is randomly modified to generate a new corresponding hash value. Compare the two hash values. Calculate the number of different bits in the same position of two hash values. The following six statistics are used to detect the confusion and diffusion properties of the algorithm.

Minimal number of bit changes:

Bmin= min{B1, B2, B3, …, B}.

Maximum number of bit changes:

Bmax= max{B1, B2, B3, …, BN}.

Average number of bit change:

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

Average rate of bit change：

$$\overline{P} = \frac{\overline{B}}{8 \cdot L} \times 100\% \ . \tag{8}$$

Mean square error of bit change number:

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

Mean square error of bit change rate:

$$S_P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{8 \cdot L} - \overline{P})^2} \times 100\% \ . \tag{10}$$

Where B is the number of changed bits, N is the number of tests, and L is the length of hash value. When the output hash length is 128, 256, 512 bits, the initial text is the message given by 4.1. In the case of N= 256, 512,1024,2048 times, the above four statistics are counted respectively, and the statistical values when the plaintext 1 bit changes are obtained, as shown in Table 1 to Table 3.

Table 1. Statistical results of confusion and diffusion properties (128 bits)

N	256	512	1024	2048
$\overline{B}$	64.26	64.01	64.08	64.00
$\overline{P}$%	50.20	50.01	50.06	50.00
S_B	5.61	5.70	5.68	5.74
S_P%	4.38	4.45	4.44	4.48
Bmax	81	81	81	83
Bmin	51	48	43	43

Table 2. Statistical results of confusion and diffusion properties (256 bits)

N	256	512	1024	2048
$\overline{B}$	128.66	127.95	128.13	128.20
$\overline{P}$%	50.26	50.00	50.00	50.01
S_B	8.12	7.95	8.04	8.18
S_P%	3.17	3.10	3.14	3.20
Bmax	156	156	156	160
Bmin	103	103	103	103

Table 3. Statistical results of confusion and diffusion properties (512 bits)

N	256	512	1024	2048
$\overline{B}$	255.42	255.86	256.18	256.01
$\overline{P}$%	49.89	50.00	50.04	50.00
SB	10.75	11.31	11.08	11.09
SP%	2.10	2.21	2.16	2.16
Bmax	302	302	302	302
Bmin	224	220	220	220

We can see that, the average number of bit change and the average rate of bit change probability per bit of the algorithm are very close to the ideal state 64, 128 and 256 bits and 50%. Moreover, it is easy to observe that the number of bit change when 128 bits are output is around 64 through Fig. 11 and Fig. 12, which shows that the algorithm makes full and uniform use of cipher text space and ensures that the attacker hard to forge or reverse other plaintext cipher text when some plaintext cipher text pairs are known. In addition, S_B and S_P are very small, which indicates that the confusion and diffusion performance of the algorithm is stable.

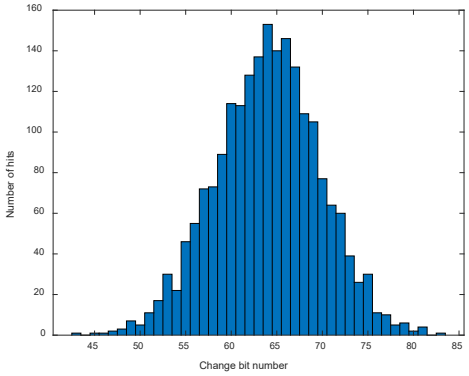


Fig. 11. 128 bit variance distribution

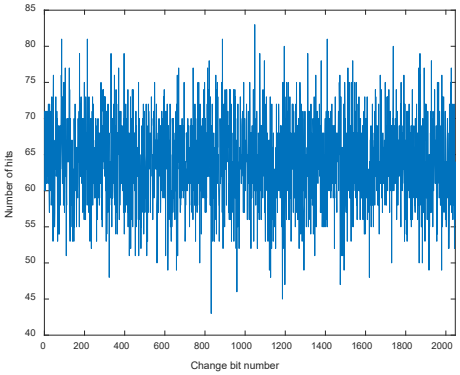


Fig. 12. Distribution of 128 bit variance

4.4 Collision Analysis and Birthday Attack

The anti-collision property of hash function is an important index to test its safety. If two different inputs X1≠X2 can be found and their hash value H1 = H2, it will be considered a collision. Birthday attack is essentially similar to collision problem. It discusses the probability problem that two random input data are processed by hash function to get the same hash value. For this algorithm, in the two-dimensional coupled map lattice model, the message is embedded in the corresponding lattice before the iteration, and the dynamic integer tent map is used as the nonlinear function of the model, which increases the confusion and diffusion of data processing, and the adjacent lattices receive mutual influence, and the diffusion degree of the whole model will increase one level every further iteration, Finally, all the lattices will be related to the initial lattices and the embedded message value. This will ensure that any bit of the final hash value will be related to all the bits of the message, any small change in the message, amplified by the continuous diffusion of the iterative process, will eventually lead to a completely different hash value.

In this paper, the collision resistance of the algorithm is tested by the method of reference [12]. That is, a section of plaintext is randomly selected in plaintext space to calculate and store its hash value in the form of ASCII code, and then the value of 1 bit in plaintext is randomly selected and changed following get another new hash result, and the number of ASCII characters with the same value in the same position is recorded. Repeat the above experiment 2048 times, the results are shown in the Fig. 13. As can be seen from the Fig. 13, the maximum number of the same characters is only 2, and the degree of collision is very low.

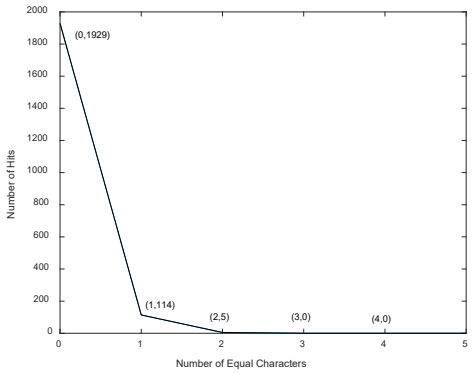


Fig. 13. Collision test

At the same time, character distance is a statistic used to test whether the hash values generated by two different plaintext messages are independent of each other. Formula defined in references [12-13]

$$\mathrm{d} = \frac{1}{\mathrm{s}} \sum_{\mathrm{i}=1}^{\mathrm{s}} | H_1[i] - H_2[\mathrm{i}] | \ . \tag{11}$$

Where d is the average character distance, H1 and H2 are the corresponding values of the ASCII character in the two hash values, and s is the byte length of the hash value. If the two tested hash values are independent and uniformly distributed, the average character distance should be 85.33 theoretically. The test results of the algorithm in this paper are shown in Table 4. The average character distance is 85.44, which is very close to the theoretical value. It can be considered that the Hash value obtained by this algorithm is independent and uniformly distributed.

Table 4. The result of average character distance

Absolute Difference	Max	Min	Mean	Mean/Character
This scheme	2149	541	1367	85.44

4.5 NIST Randomness Test

NIST test suite is a statistical software package composed of 15 tests, which are designed to test random (arbitrary length) binary sequences generated by cryptographic random or pseudo-random number generators based on hardware or software. Testing focuses on various types of existing nonrandom sequences. Some tests can be divided into sub tests. NIST tests mainly detect the ratio of P_value to the set value α is to judge whether the test is passed. The value of α is 0.01. If P_value upper than 0.01, it pass the test. The test results are shown in Table 5. The sequence generated by the hash function algorithm in this paper has passed all item tests and can be considered as an ideal random sequence.

Table 5. NIST random test results

Test	P_value
Frequency	0.729343
Block Frequency	0.954584
Cumulative sums	Success
Runs	0.872608
Longest Run of Ones	0.911362
Rank	0.657952

Discrete Fourier Transform	0.825685
Non-Overlapping Template Matchings	Success
Overlapping Template Matchings	0.537617
Universal Statistical	0.953532
Approximate Entropy	0.909159
Random Excursions	Success
Random Excursions Variant	Success
Serial	Success
Linear Complexity	0.569306

4.6 Perform Efficiency Test

Because of the hash function proposed in this paper based on complex chaos systems, compared with the traditional hash function and simple chaos map hash function, the calculation process is more complicated, its computing speed is relatively slow, but after testing, its running speed can meet the needs of practical application. We use a computer with Windows 10 (64 bit) operating system, which is configured with 3.2Ghz Intel (R) core (TM) i7-8700 CPU and 32GB RAM. We use C language to implement hash function and test samples of different sizes. The results are shown in Fig. 14. Its running time is linear with the processed messages, and the speed is about 50Mbps.

4.7 Overall Analysis

In this section, the hash function proposed in this paper is compared with other meaningful hash functions based on chaos. Table 6 describes the comparison of confusion and diffusion properties between the proposed algorithm and the selected literature algorithm, which are based on N = 2048 random tests and 128 bit hash values. In addition, Table 7 shows the comparison of the number of ASCII characters in the same position and the absolute difference of 128 bits hash value between the proposed algorithm and the selected literature algorithm based on N = 2048 random tests. Review previous tests, the first subsection tests the distribution of the algorithm's values, testing the general and extreme cases, and the results show that the Hash function proposed in this paper can make the Hash value evenly distributed into the "bucket" by processing any plaintext; the second subsection sensitivity test, five different cases are tested, and a comparative analysis of the results is given, which shows at a glance that a small change in the plaintext can make a big difference in hash value; the third subsection, the statistical analysis of the confusion and diffusion properties is carried out, and the four indicators are tested for different output lengths, and the results show that the test results are close to the ideal values regardless of the length of the output, indicating that the algorithm has ideal confusion and diffusion properties; the fourth subsection performs collision analysis, including collision test and average character distance test, and the test results have at most two collisions, and the average character distance is very close to the ideal value, indicating that the algorithm has high security; the fifth subsection performs NIST randomness test, and the results pass all 15 tests, and each result is much larger than the set value, indicating that the sequence generated by the algorithm is a more ideal random; the efficiency test is conducted in the sixth subsection, and the results show that the processing speed of the algorithm cannot reach the effect of traditional hash, but it can meet the practical application requirements. These test results show that the proposed algorithm meets the requirements of hash function.

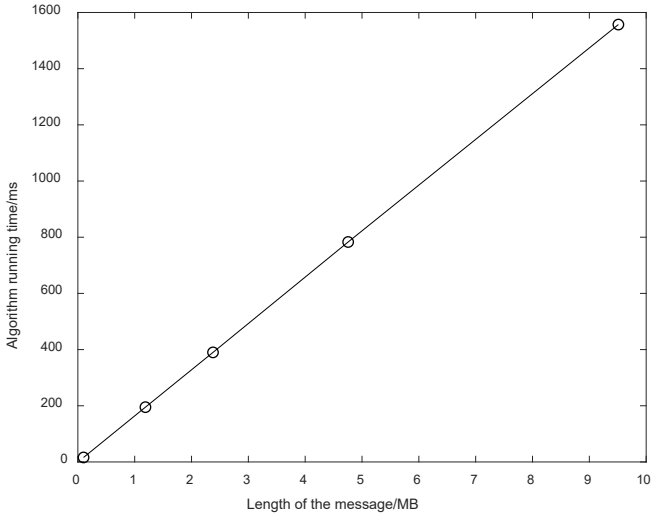


Fig. 14. Speed test

Table 6. Comparison on statistical performance with N = 2048 random tests and 128 bit hash value

Algorithm	Statistical performance of the algorithms			
	$\overline{B}$	$\overline{P}$%	SB	SP
This scheme	64.00	50.00	5.74	4.48
[1]	63.94	49.96	5.69	4.44
[10]	63.94	49.96	5.61	4.38
[14]	63.57	49.66	7.43	5.80
[15]	63.92	49.94	5.62	5.39
[16]	63.99	49.99	5.66	4.40
[17]	63.84	49.88	5.88	4.59
[18]	64.17	50.13	5.73	4.48
[19]	63.97	49.98	5.68	4.41

Table 7. Comparison on colony resistance with N = 2048 random tests and 128 bit hash value

Algorithm	Number of hits				Average byte distance
	0	1	2	3	
This scheme	1929	114	5	0	85.44
[10]	1930	117	2	0	84.38
[14]	1945	103	0	0	78.44
[15]	1924	120	4	0	89.46
[16]	1928	116	4	0	84.94
[17]	1940	104	4	0	87.49

5 Conclusion

This paper is based on dynamic integer tent map and two-dimensional coupled map lattice model, and adds another dynamic parameter to increase the confusion property. The main contents include integrating tent map and making it dynamic to increase chaos characteristics; combining with two-dimensional coupled map lattice model, adding dynamic parameters and using dynamic tent map to control the change of dynamic parameters. It can be said that the double dynamic parameters greatly enhance the confusion and diffusion properties of the system. Tests show that the algorithm has strong ability of confusion and diffusion, and can be applied to a variety of encryption occasions. The limitation is that the processing process is more complicated, which makes the processing speed slower and cannot achieve the same effect as the traditional hash function. In the future, we will consider designs that are more efficient, such as parallel, running on big data platforms, improve its operational efficiency, and try to apply it in block-chain technology to enhance overall security.

References

[1] A. Kanso, H. Yahyaoui, M. Almulla, Keyed hash function based on a chaotic map, Information Sciences 186(1)(2012) 249-264.

[2] N. Abdoun, S. Assad, O. Deforges, R. Assaf, M. Khalil, Design and security analysis of two robust keyed hash functions based on chaotic neural networks, Journal of Ambient Intelligence and Humanized Computing 11(5)(2020) 2137-2161.

[3] J.S. Teh, M. Alawida, J.J. Ho, Unkeyed hash function based on chaotic sponge construction and fixed-point arithmetic, Nonlinear Dynamics 100(1)(2020) 713-729.

[4] A. Akhavan, A. Samsudin, A. Akhshani, Hash function based on piecewise nonlinear chaotic map, Chaos, Solitons & Fractals 42(2)(2009) 1046-1053.

[5] J.D. Liu, One-way hash function based on integer coupled tent maps and its performance analysis, Journal of Computer research and development 45(3)(2008) 563-569.

[6] T. He, W.D. Feng, H.W. Wang, J. Tan, Research and improvement of AES encryption algorithm based on two-dimensional chaotic map, Computer and Digital Engineering 49(10)(2021) 1993-1997.

[7] Y. Wang, G.K. Jiang E.M. Yin, Image encryption based on two-dimensional coupled image lattice model, Journal of Southwest Jiaotong University, <http://kns.cnki.net/kcms/detail/51.1277.u.20201104.1140.002.html>, 2021.

[8] J.D. Liu, X.H. Wang, K. Yang, C. Zhao, A Fast New Cryptographic Hash Function Based on Integer Tent Map System, Journal of Computers 7(2012) 1671-1680.

[9] J.D. Liu, Y. Yu, A TCML-based spatiotemporal chaotic one-way Hash function with changeable-parameter, Acta Physica Sinica 56(3)(2007) 1297-1304.

[10] Y. Li, G. Ge, Cryptographic and parallel hash function based on cross coupled map lattices suitable for multimedia communication security, Multimedia Tools and Applications 78(13)(2019) 17973-17994.

[11] C.E. Shannon, Communication theory of secrecy systems, The Bell System Technical Journal 28(4)(1949) 656-715.

[12] K.W. Wong, A combined chaotic cryptographic and hashing scheme, Physics Letters A 307(5-6)(2003) 292-298.

[13] J. Feng, L.T. Yang, Q. Zhu, K.K. Choo, Privacy-Preserving Tensor Decomposition Over Encrypted Data in a Federated Cloud Environment, IEEE Transactions on Dependable and Secure Computing 17(4)(2020) 857-868.

[14] Y. Li, S. Deng, D. Xiao, A novel Hash algorithm construction based on chaotic neural network, Neural Computing and Applications 20(1)(2011) 133-141.

[15] D. Xiao, X.F. Liao, S. Deng, Parallel keyed hash function construction based on chaotic maps, Physics Letters A 372(26) (2008) 4682-4688.

[16] Y. Li, G. Ge, D. Xia, Chaotic hash function based on the dynamic S-Box with variable parameters, Nonlinear Dynamics 84(4)(2016) 2387-2402.

[17] S. Deng, Y. Li, D. Xiao, Analysis and improvement of a chaos-based hash function construction, Communications in Nonlinear Science and Numerical Simulation 15(5)(2010) 1338-1347.

[18] Z. Liu, Y. Wang, G. Jiang, L.Y. Zhang, Design and Analysis on a Parallel Chaos-Based Hash Function, International Journal of Bifurcation and Chaos 30(13)(2020) 2050188.

[19] H.P. Ren, C.F. Zhao, C. Grebogi, One-Way Hash Function Based on Delay-Induced Hyperchaos, International Journal of Bifurcation and Chaos 30(2)(2020) 2050020.