Advances in Computer, Signals and Systems (2021) 5: 98-106
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2021.050116
ISSN 2371-8838

Parallel HAIFA Hashing Algorithm Based on Lorenz Chaos

Yuanyi Liu[1,*], Xin Zhou [2], Ge Liu [2]

[1]*School of Electronic and Electrical Engineering, Lingnan Normal University, Zhanjiang 524048, China*
[2]*School of Electronic and Information Engineering, Heilongjiang University of Science and Technology, Harbin 150022, China*
**corresponding author*

Keywords: hash function, Lorenz chaotic system, Parallel structure, HAIFA iterative structure

Abstract: Aiming at the inefficiency under parallel environment or large data computation, HAIFA hash function based on Lorenz chaos is constructed in parallel, and a parallel hash function based on Lorenz chaos is proposed. The algorithm compresses each message block independently and can be executed concurrently. After the hash value of each message block is obtained, every two hash values are combined. The odd-numbered rounds are combined with modular addition and right loop operation, while the even-numbered rounds are combined with XOR and left loop operation. The difference of each round of operation further enhances the anti-collision and anti-forgery attacks of the algorithm. The new parallel algorithm is tested for safety analysis and efficiency. The results show that the parallel modified algorithm has good performance and high efficiency, which has certain reference significance for the safety construction of parallel chaotic hash algorithm.

1. Introduction

Hash function changes data of any length to obtain a hash value of fixed length. It is widely used in cryptographic protocols, digital signatures, integrity authentication and other fields. Chaos is sensitive to initial value, pseudo-random, and difficult to predict, which is valued and favored by researchers. It has become a new method and technical means of constructing hashing algorithm, and has been gradually applied to information safety. In reference [1, 2], a new Hash function is designed based on chaotic neural network. In literature [3], a one-way Hash function is designed with spatio-temporal chaos and double disturbances based on serial. In literature [4], a Hash function based on cyclic shift is designed with variable parameters. These serial modes lead to the failure of CPU's parallel advantages and greatly reduce computing speed. In view of the inefficiency of the serial structure of the current Hash function algorithms, the method of constructing parallel hash functions by using chaotic systems began to rise with the aim of improving the running efficiency.

Li et al. [5] put forward a chaotic Hash function algorithm with variable parallel parameters based on piecewise linear mapping, which enhanced the execution efficiency of the parallel algorithm. However, when the chaotic parameter value is greater than 0.25, its iterative value degenerates to 0,

which becomes a fixed point. In this case, the algorithm does not have good confusion and diffusion. In literature [6], a scheme with key hash based on parallel structure is proposed. The parallel structure is designed by PMAC algorithm, and the XOR value is recompressed again. Hash collision can easily occur by using equal-length forgery attack. In this paper, the HAIFA hash function based on Lorenz chaos is constructed in parallel. The algorithm compresses each message block independently, and every two adjacent message blocks are calculated as a whole according to the calculated values. Under the premise of ensuring good confusion and diffusion effects and safety, the efficiency of the algorithm is improved to meet various performance requirements for the hash function.

2. Preliminary Knowledge

2.1. Lorenz Chaotic Mapping

Lorenz chaotic mapping is a three-dimensional chaotic mapping, and its mathematical expression in the form of differential equation is as follows:

$$\begin{cases} \frac{dx}{dt} = a(y-x), \\ \frac{dy}{dt} = (bx - y - xz), \\ \frac{dz}{dt} = (xy - cz)。 \end{cases} \tag{1}$$

In formula (1), a, b and c are the control parameters of Lorenz chaotic system. The typical values generally adopted are as follows $a=10$, $b=28$ and $c=8/3$. On the premise of maintaining a and c unchanged, when $b>27.74$, Lorenz chaotic system is in chaotic motion.

2.2. HAIFA Structure

HAIFA iterative framework is a hash function structure proposed by Biham and Dunkelman in 2006. The iterative structure of HAIFA is shown in Figure 1. It is a generalized MD structure, but it changes the input of message filling and compression functions of MD structure. The essential feature of HAIFA structure is that random Salt and Counter are added into the compression function to ensure the safety of iterative structure, so as to overcome the weak iterative ability of MD structure. In addition, HAIFA structure has good anti-collision, which can resist long message attack and fixed point attack.

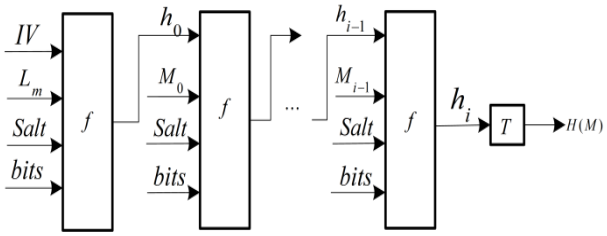

Figure 1: HAIFA structure

3. Construction of Hash Function with Lorenz Chaotic HAIFA Structure

Because XOR logic has the nature of exchange, it is not advisable to use simple XOR logic to merge hash values of each message block as it is vulnerable to equal-length forgery attack. Some improved schemes based on XOR logic can attack with specific messages, and it is easy to have hash collision.

Therefore, the original chaotic hash function is constructed in parallel with the improved parallel structure, and the improved parallel algorithm structure is shown in Figure 2.

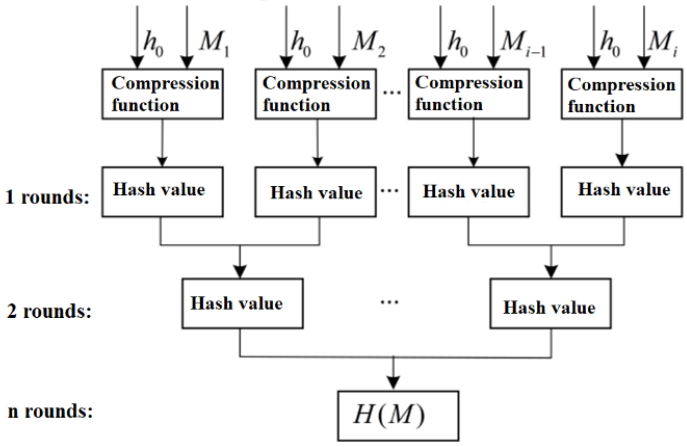

Figure 2: Algorithm structure; Compression function; Hash value

Main steps of improving parallel algorithm:

(1) Message filling and grouping

The length of the whole message is filled into an integer multiple of 512 bits, and each message is divided into i message blocks, which are represented by M_i. Then, each message M_i is divided into sixteen 32-bit messages, denoted as $M_i^0, M_i^1, \ldots, M_i^{15}$.

(2) Message initialization

Firstly, eight initial hash values, namely $H_0^0, H_1^0, \ldots, H_7^0$, are set, and each initial hash value consists of 32 bits, which are expressed in hexadecimal as follows

$H_0^0 =$ 5BE0CD19, $H_1^0 =$ A54FF53A, $H_2^0 =$ 1F83D9AB, $H_3^0 =$ BB67AE85, $H_{04}^0 =$ 3C61F372, $H_5^0 =$9BE0CD19, $H_6^0 =$510E527F, $H_{07}^0 =$6A09E667.

The above values are square roots of 8 prime numbers, and the hexadecimal representation of the first 32 digits of the decimal part is obtained. Four 32-bit random salt values are defined to store randomly assigned salt values s_0, s_1, s_2, s_3. Two counter values t_0, t_1 are defined, which are used to save the number of compressed bits of the whole message at this time. 16 intermediate variables are defined, and are respectively denoted as $v_0 \sim v_{15}$. It is obtained from the following formula that:

$$\begin{cases} v_i = h_{7-i} & 0 \le i \le 7 \\ v_i = t_0 \oplus s_{i-8} & 8 \le i \le 11 \\ v_i = t_1 \oplus s_{i-12} & 12 \le i \le 15 \text{。} \end{cases}$$

Wherein: i ——subscript of 16 intermediate variables;

$\oplus$ ——XOR operation.

(3) Compression function

The original BLAKE wheel function and Lorenz chaotic system are used to compress the cycle, and its output is 16 intermediate variables, which are denoted as $v_i'(0 \le i \le 15)$. The extraction formula for each v_i' value is:

$$h_i = H_i^0 \oplus v_i' \oplus s_j \oplus v_{i+8}' \tag{2}$$

In formula (2), $0 \le j \le 7$ and $j = i \bmod 4$. Then each $h_i(0 \le i \le 7)$ is combined according to formula (3) to obtain the hash value of each message block.

$$H_i(M) = h_0 \parallel h_1 \parallel h_2 \parallel h_3 \parallel h_4 \parallel h_5 \parallel h_6 \parallel h_7, \tag{3}$$

Wherein: $\|$—— tail addition operation;

$H_i(M)$—— hash value of this message block.

(4) Hash combination

The hash values obtained from each message block are combined. In each round of combination, it is necessary to judge the number of hash values in this round. If it is odd, it is necessary to complete the tail of the last hash value in this round. The completion rule is

$$H' = H_k \parallel H_{k+1} \quad k \bmod 2 = 1,$$

Wherein: k—— the number of hash values in this round;

H'——the overall hash value of this round after completion;

H_k——the last hash value;

H_{k-1}—— the hash value completed.

If the number of combinations in this round is even, remain unchanged, and then combination every two hash values. The combination mode is

$$\begin{cases} (H_p + H_{p+1}) >>> 7, & n \bmod 2 = 1 \\ (H_p \oplus H_{p+1}) <<< 11, & n \bmod 2 = 0 \end{cases}$$

Wherein: $+$—— modular addition operation;

$>>>$——Right cyclic shift operation;

$<<<$——Left cyclic shift operation;

n——The number of rounds of combination of every two hash values

H_p and H_{p+1}—— hash values to be combined.

It can be seen that modular addition and right cyclic shift are used in odd rounds, and XOR and left cyclic shift are used in even rounds. In this way, every hash value can be combined in each round. The last hash value will be obtained in the last round, which is the hash value of the whole algorithm.

4. Hash Function with Lorenz Chaotic HAIFA Structure

The performance of the proposed algorithm is analyzed and discussed from several aspects, such as hash value distribution, initial value sensitivity, characteristics of confusion and diffusion statistics between plaintext and abstract, and anti-collision test.

4.1. Hash Value Distribution

The algorithm selects a message composed of random characters, and counts the distribution of ASCII code values in the message and Hash values generated by the algorithm, as shown in Figure 3 and Figure 4 respectively. It can be seen that the message is basically scattered in a limited area, and the hash values are evenly distributed in the whole space, without exposing any statistical information related to plaintext messages, which shows that the improved parallel algorithm has good distribution.

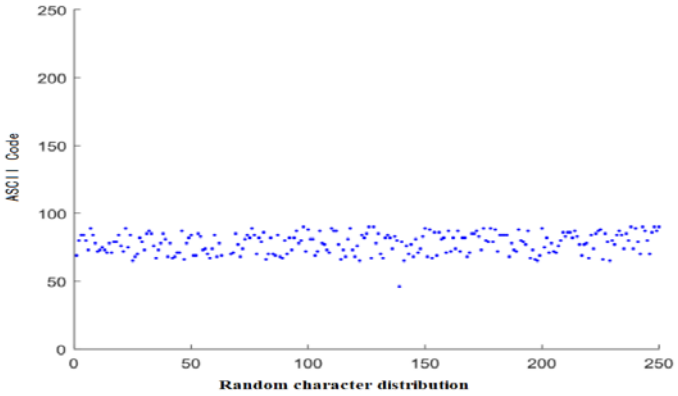

Figure 3: Random character distribution

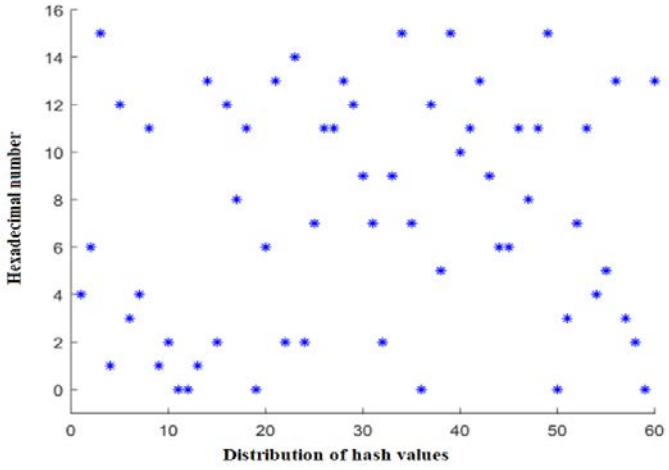

Figure 4: Distribution of hash values

4.2. Sensitivity of Initial Value

Hash algorithm should satisfy the sensitivity dependence on plaintext messages. To test the initial value sensitivity of the new parallel algorithm, select a text: “Research on Parallel Hash Function Based on Chaotic Map.”. Test is carried out under the following six conditions.

C1: Calculate the hash value of the given message.

C2: Change the first word "c" to "C".

C3: Change “.”at the end of the sentence to “,”.

C4: change the word "Hash" to "Hash-".

C5: add a "9" before the word "Based".

C6: Add a space at the end of the sentence.

The hash value obtained from C1~C6 is shown in Figure 5. It can be seen that slight changes in the message will also lead to great changes in the output hash value. This proves that the proposed new parallel algorithm has high initial value sensitivity.

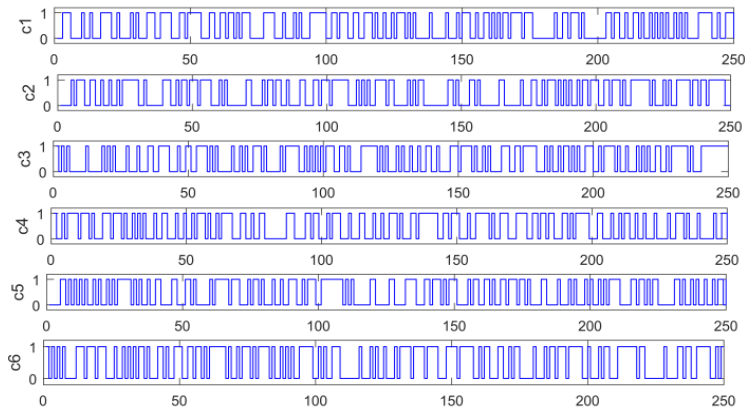

Figure 5: Hash values in different cases

4.3. Confusion and Diffusion Statistics

Confusion and diffusion are two important indexes of hash function, which are mainly used for avalanche effect test. From a statistical point of view, the relationship between plaintext and hash value becomes complicated, and every bit of plaintext can affect hash value. The test indexes are as follows:

Minimum number of changed bits:

$$B_{min} = \min(B_i),$$

Maximum number of changed bits:

$$B_{max} = \max(B_i),$$

Average change bits:

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

Average change probability:

$$P = (\bar{B}/n) \times 100\%,$$

Mean square deviation of B:

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

Mean square deviation of P:

$$\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{n} - P)^2 \times 100\%}$$

Wherein: N —— the number of tests;

n —— the length of the output hash value;

B_i ——the number of bits changed by the hash value obtained at i.

Select an arbitrary piece of plaintext message, and use the statistics and test scheme defined in literature [4] to make statistics on the confusion and diffusion characteristics of the new parallel algorithm. The number of tests is 10000. Compared with the original and other parallel chaotic hash algorithms, the results are shown in Table 2. The statistical results show that the index of the improved parallel algorithm is slightly worse than the original algorithm. Compared with other parallel hashing

algorithms, its average change bit number is closer to the ideal value of 128, and its average change bit rate is also closer to the ideal value of 50%, which is smaller than other parallel hashing algorithms. Therefore, the improved parallel algorithm has good diffusion and stability.

Table 2: Statistics of confusion and diffusion of the parallel algorithms

Index	Literature [5]	Literature [8]	Literature [9]	Original algorithm	The algorithm
$\overline{B}$	128.94	128.71	127.21	127.98	128.56
$P\%$	50.35	49.96	49.63	49.99	50.21
$\triangle B\%$	8.45	8.53	8.61	8.054	8.32
$\triangle P\%$	4.42	4.51	3.92	3.15	3.44

4.4. Anti-collision Analysis

The anti-collision test scheme of reference [7] is selected. The improved parallel hash algorithm has been tested for 10 000 times, and the distribution of hash values of the same characters is shown in Figure 6. It can be seen that there are not four or more hits, two are three hits, 72 are two hits, 1,213 are one hit, and there are no hits in 8,713 tests. The proposed new parallel algorithm also has good anti-collision performance.

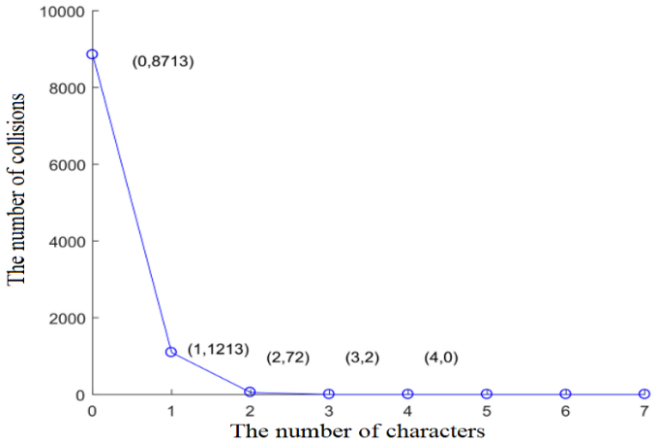

Figure 6: Distribution of the number of hash values with the same ASCII characters

In the anti-collision test, the collision resistance of hash function can be evaluated by calculating the distance between hash values of two messages. The calculation formula is

$$d_{hash} = \sum_{i=1}^{N}(|t(a_i) - t(b_i)),$$

Wherein: a_i, b_i —— the i th ASCII character of two hash values; d_{hash} ——the absolute difference of unit field between two hash values;

$t(x)$ ——converting the ASCII characters corresponding to x into decimal values.

The ideal value of d_{hash} is 85.33. After testing for 10 000 times and comparing with the original and other parallel hashing algorithms, the results are shown in Table 3. Seen from the results, the average absolute distance of the improved parallel algorithm is close to the theoretical value of 85.33, which indicates that the hash value of the new and improved parallel algorithm is evenly distributed, and it can be judged that its statistical characteristics meet the anti-collision requirements.

Table 3: Absolute distance comparison of hash values for the new parallel algorithm

Algorithm	Maximum	Minimum value	Average value	Average of absolute distances.
Literature [5]	4 317	1 208	2 656	84.53
Literature [8]	4 179	1 156	2 740	86.56
Literature [9]	4 472	1 467	2 730	85.21
Literature [10]	4 228	1 169	2 805	87.22
Literature [11]	4 383	1 465	2 638	83.67
Original algorithm	4 109	1 431	2 730	85.31
This algorithm	4 150	1 351	2 767	85.55

4.5. Efficiency Analysis

The speed of the algorithm is also an important index to evaluate the hash function. The new parallel algorithm uses parallel tree structure for calculation, which has more efficient computing speed. Setting the number of message packets of the algorithm as n and the computing time of the compression function as N, the computing time of this improved parallel algorithm is

$$T' = N \times log_2^n$$

The proposed hash function scheme is implemented by Java,100MB message length, the new parallel algorithm is implemented by Java on Intel (R) Core (TM) i5-8400 CPU @ 2.80ghz 16GB RAM and Windows 10. Message of 100MB message is used for computing speed test. The proposed new parallel algorithm is implemented by Java on Intel (R) Core (TM) i5-8400 CPU @ 2.80 GHz 16 GB RAM and Windows 10. After comparing with other parallel hash algorithms, the results are shown in Table 4. It can be seen from the table that compared with other parallel chaotic hash algorithms, the new parallel algorithm also has a highly efficient computing speed.

Table 4: Comparison of calculation speed for 100MB Messages

Algorithm	Speed /Mbps
Literature[5]	927.2
Literature[8]	1 517.4
Literature[9]	678.3
Literature[10]	589.1
Literature[11]	1 208.5
New parallel algorithm	1 669.4

5. Conclusion

In this paper, the hash function based on Lorenz chaos is constructed in parallel. Each message block is independently added to the compression process of each packet, so that it can be executed concurrently. In the hash combination part, different combination methods are adopted to further enhance the safety of the algorithm, and the final hash value can be obtained. The proposed algorithm meets the performance requirements of hash function. Therefore, the efficiency analysis shows that the efficiency of the algorithm proposed in this paper has been improved while the safety is guaranteed.

References

[1] *Wang, S.W. (2015) on information safety, network safety and cyberspace safety. Journal of Library Science in China, 41(2): 72-84.*

[2] *Wang, Y., Chen, Y., Zhao, Y. (2018) Parallel Hash Function Construction Algorithm Based on Piecewise Logistic Mapping. Computer Engineering and Applications, 54(15): 38-43.*

[3] *Li, H.J., Long, M. Construction of one-way hash function with spatio-temporal chaos and double disturbances. Journal of Chinese Computer Systems, 36(3): 539-543.*

[4] *Feng, Y.R., Li, Y.T., Xiao, D. (2011) Construction and performance analysis of chaotic hash function based on parallel and variable parameters. Computer Applied Research, 28(11): 4307-4310.*

[5] *Li, Y., Li, X. (2016) Chaotic hash function based on circular shifts with variable parameters. Chaos Solitons & Fractals the Interdisciplinary Journal of Nonlinear Science & Nonequilibrium & Complex Phenomena, 2016, 91: 639 - 648.*

[6] *Albertini, A., Aumasson, J.P., Eichlseder, M., et al. (2014) Malicious hashing: eve's variant of SHA-1 //International Conference on Selected Areas in Cryptography. Springer, Cham, 1 - 19.*

[7] *Wang, Y., Wong, K.W., Xiao, D. (2011) Parallel hash function construction based on coupled map lattices. Communications in Nonlinear Science and Numerical Simulation, 16(7): 2810 - 2821.*

[8] *Lv, A.P., he, Y. (2011) parallel hash function based on chaotic neural network. Modern Science & Technology of Telecommunications, 41(3): 38-41.*

[9] *Zhou, H., Wang, S. (2012) Collision analysis of a parallel keyed hash function based on chaotic neural network. Neurocomputing, 97: 108-114.*

[10] *Cui, H.S., Tian, Y., Deng, S.J. (2016) Improved parallel chaotic Hash function using bi-directional coupled map lattice. Computer Engineering and Applications, 52(4): 88-93.*

[11] *Zhao, G., Xu, G., Min, L.Q. (2011) Construction of a parallel spatiotemporal chaotic one-way Hash function. Microcomputer Information, 27(5): 232-235.*