EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

A novel parallel hash function based on 3D chaotic map

Amir Akhavan[1*], Azman Samsudin[1] and Afshin Akhshani[2]

Abstract

As the core of cryptography, hash function is one of the basic techniques for information security. During the last few years, considerable effort has been devoted to research on chaos-based hash functions. Nevertheless, the corresponding analysis of them lag far behind. In this paper, a new efficient scheme for parallel hash function based on high-dimensional chaotic map is proposed. In the proposed scheme, the confusion as well as the diffusion effect is enhanced significantly by utilizing two nonlinear coupling parameters. Theoretical and experimental results indicate that the proposed scheme can satisfy all performance requirements of hash function such as desired statistical properties and strong collision resistance. At the same time, the proposed scheme can keep the parallel merit and message sensitivity with high potential to be adopted for network security.

Keywords: Hash function; High-dimensional chaotic map; Chaos-based cryptography

1 Introduction

As a ubiquitous phenomenon in nature, chaos is a kind of deterministic random-like process generated by nonlinear dynamical systems. Chaotic dynamical systems possess the following main characteristics: sensitivity to tiny changes in initial conditions, random-like behavior, ergodicity, unstable periodic orbits, desired diffusion and confusion properties, and one-way property. Due to these properties, chaotic systems have become a very good candidate for use in the field of cryptography. The existing related research progress includes chaos-based symmetric encryptions [1-7], security protocols [8,9], asymmetric encryptions [10,11], and hash functions [12-18].

Hash function is one of the major tools in cryptography, which is usually used for integrity protection in conjunction with message authentication and digital signature schemes. There are many famous hash functions such as MD4, MD5, and SHA-1 [19-21] which are realized by complicated methods based on logical operations or multi-round iteration of some available ciphers [22]. Recently, some of the weaknesses of the famous and widely used hash functions MD5 and SHA-1 have been discovered. Apparently, it is very easy to carry out collision attack on MD5 [23-25], and at the same time, the collision attack on SHA-1 is close to practical [26]. Thus, the research on the design of secure and efficient hash functions attracts more and more attention.

In recent years, there has been a considerable advance in the construction of one-way hash functions based on chaotic maps, including 1D piecewise linear/nonlinear chaotic map [27,28], 2D nonlinear piecewise linear chaotic map [29], high-dimensional chaotic map [30], chaotic neural networks [31], hyper chaos [32], and chaos S-Box [33], which indicates extensive effort in constructing hash functions based on different new methods although some of these new algorithms have been proven to be insecure [22,34-37].

Recently, high-dimensional dynamical systems have been an attractive research field for many scientific areas particularly in computer sciences and have received considerable attention in secure communication field. This kind of chaotic map has better mixing nature and higher complexity. Moreover, it is more difficult or even impossible to predict the time series generated by the system. These properties granted them to be a good candidate for secure communications. To the best of our knowledge, only a few chaos-based hash functions with high-dimensional dynamical systems have been reported

*Correspondence: amir.akhavan@yahoo.com
[1] School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
Full list of author information is available at the end of the article

[30,32,38]. For instance, an algorithm for image encryption based on a high-dimensional chaotic map with high efficiency is proposed in [4]. The high-dimensional chaotic maps [4] possess remarkable dynamical properties such as invariant measure, ergodicity, the possibility of KS-entropy calculation, and bifurcation without any period doubling [4,39,40].

Hence, in this paper, a new scheme for constructing parallel hash function based on high-dimensional trigonometric map is suggested, and efficiency of the scheme is investigated in detail. The proposed algorithm fulfills the performance requirements of hash functions. Theoretical analysis and experimental results indicate that the proposed scheme demonstrated several interesting features, such as good statistical properties, strong collision resistance, and high flexibility. At the same time, it can keep the parallel structure and message sensitivity as required by practical hash functions.

2 A one-way hash function based on high dimensional chaotic map

2.1 High dimensional chaotic map

Hierarchy of the high-dimensional chaotic maps with ergodic behavior in the two intervals $[0,1]$ and $[0,\infty)$ is presented in [4,39,40]. One-parameter families of chaotic maps of the interval [0, 1] with an invariant measure can be defined as the ratio of polynomials of degree N

$$\Phi_N(x,\alpha) = \frac{\alpha^2 F}{1+(\alpha^2-1)F},$$

where α is the control parameter. F substitutes with the Chebyshev polynomial of the first kind $T_N(x)$, and N is the degree of Chebyshev polynomials. Hence,

$$\Phi_N(x,\alpha) = \frac{\alpha^2 (T_N(\sqrt{x}))^2}{1+(\alpha^2-1)(T_N(\sqrt{x}))^2}.$$

We used its conjugate or isomorphic map [39,41]. Conjugacy means that the invertible map $h(x) = \frac{(1-x)}{x}$ maps $I = [0,1]$ into $[0,\infty)$ [39,42]. Using the hierarchy of families of one-parameter chaotic maps, we can generate new hierarchy of tripled maps with an invariant measure. In this paper, one of the hierarchies of the chaotic map in the interval $[0,\infty)$ is adapted for constructing chaos-based hash function. Hence, this chaotic map can be defined as [4]

$$\Phi_N(x,\alpha)\begin{cases} x(n+1) = \frac{1}{\alpha_1^2}\tan^2(N_1 \arctan\sqrt{x(n)}), \\ y(n+1) = \frac{1}{\alpha_2^2(x(n))}\tan^2(N_2 \arctan\sqrt{y(n)}), \\ z(n+1) = \frac{1}{\alpha_3^2(x(n),y(n))}\tan^2(N_3 \arctan\sqrt{z(n)}), \end{cases} \quad (1)$$

where

$$\beta(x(n)) = \left(\sqrt{\frac{\alpha_2}{2-\alpha_2}} + \epsilon x(n)\right)^2,$$

$$\alpha_2(x(n)) = \frac{2\beta(x(n))}{1+\beta(x(n))}\sqrt{\frac{\beta(x(n+1))}{\beta(x(n))}},$$

$$\beta(x(n),y(n)) = \left(\sqrt{\frac{\alpha_3}{2-\alpha_3}} + \epsilon' y(n)\right)^2,$$

$$\alpha_3(x(n),y(n)) = \frac{2\beta(x(n),y(n))}{1+\beta(x(n),y(n))} \times \sqrt{\frac{\beta(x(n+1),y(n+1))}{\beta(x(n),y(n))}},$$

where α_1, $\alpha_2(x(n))$ and $\alpha_3(x(n),y(n))$ are control parameters of the chaotic map, and N_1, N_2, and N_3 are degrees of Chebyshev polynomials. In this paper, Equation 1 as a high-dimensional chaotic map is used in the process of generating a hash function.

2.2 Proposed algorithm

In this section, the framework of the proposed algorithm is described.

Step 1. First, the message $M : \{0,1\}^a$ is input and transformed into message blocks $M_{n\times 1}$.
Step 2. Keys of the algorithm (initial conditions, control parameters, and coupling parameters) are input.
Step 3. Creation and initiation of the threads T1 and T2. In this step, the two separate threads are generated, and initial conditions and control parameters are initiated (the threads are demonstrated in Figure 1 using two parallel vertical rectangles).
Step 4. In order to remove the transient effect and also initiate the initial conditions of the threads, Equation 1 is iterated 1,500 times, and the initial conditions for the threads are set ($x'_0 = x_{1,000}$, $y'_0 = y_{1,000}$, and $z'_0 = z_{1,000}$ for thread one (T1) and $x''_0 = x_{1,500}$, $y''_0 = y_{1,500}$, and $z''_0 = z_{1,500}$ for thread two (T2)).
Step 5. Each thread processes one chunk of the message using y_i and z_i of the map in Equation 1, and in each round, the control parameters are modified using the value of the processed chunk of the message and new initial conditions y_i and z_i. This step is repeated until the message is exhausted and the final z_n in each of the threads is used to generate matrices D' and D'' (see Figure 1).

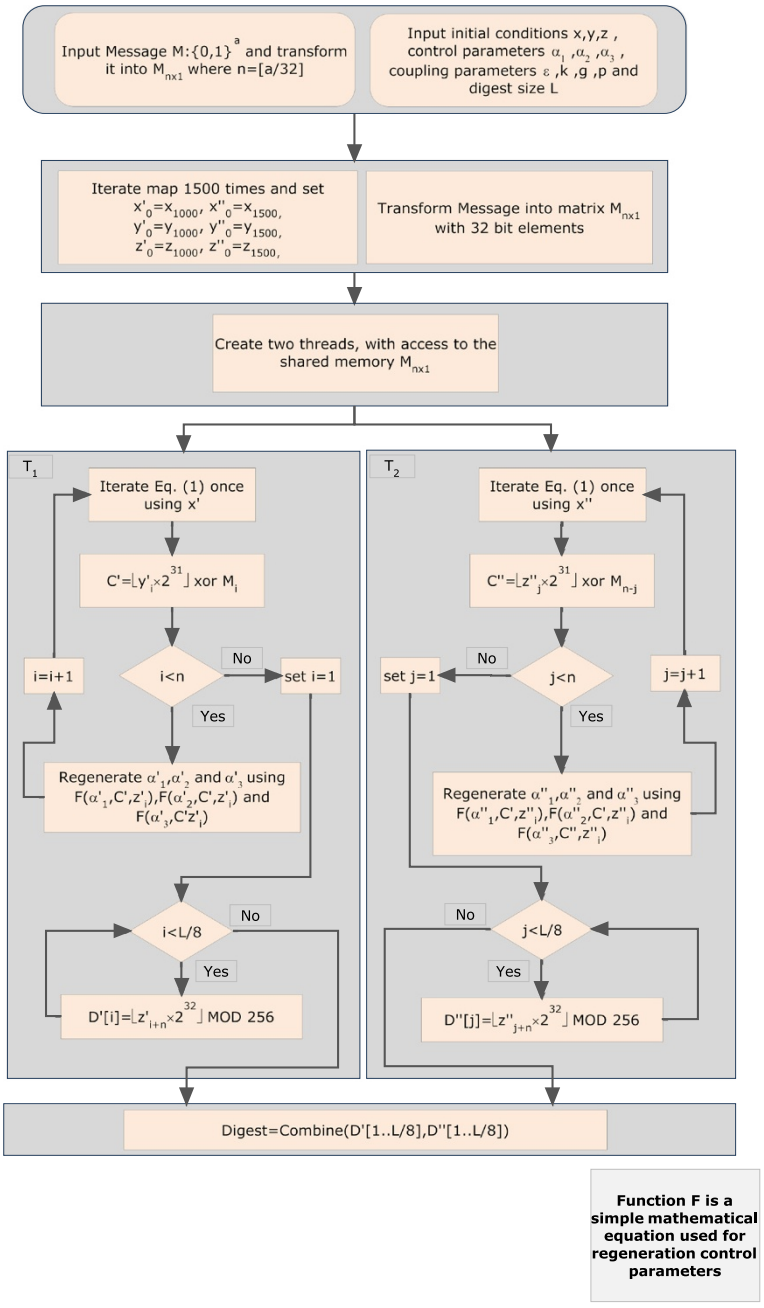

Figure 1 Block diagram.

Step 6. The matrices D' and D'' are generated using the final values of z_n in step 5 (see Figure 1). In this step, the chaotic map, Equation 1, is iterated $L/8$ times (L is the final size of the hash value which by default is set to 256). Finally, the matrices are combined using the function mentioned below.

Function Combine$(D'[1..k], D''[1..k])$
  For $i = 0$ *to* k
    If $D'[i] \ mod\ 2 = 0$
      $D[i] = D'[i]\ XOR\ D''[i]$;
    Else
      $D[i] = D'[i]\ mod\ 2^8\ XOR\ D''[i]$;
    End if
  end for
  Return D;
  Return Result;
End Function

Step 7. The final value of the hash function is output.

Figure 1 shows a block diagram that illustrates the steps in the algorithm. In this algorithm, the values of the parameters are as follows:

Initial conditions: $x = 3$, $y = 8$, and $z = 93$
Control parameters: $\alpha_1 = 12$, $\alpha_2 = 1.5$, and $\alpha_3 = 3$
Coupling parameters: $\epsilon = 0.2$ and $\epsilon' = 0.3$
$N_1 = 20$, $N_2 = 4$, and N_3=4

Note that all of the control parameters are selected in the chaotic region.

3 Performance analysis

3.1 Hash results of messages

We have chosen a message with the length of 1,024 characters, and all the characters are chosen equal to zero; in this way, it is easier to show the sensitivity of the hash function to the message.

Condition 1: The original message contains 1,024 null characters.
Condition 2: Change the first character to 1.
Condition 3: Change the 800th character to *Z*.
Condition 4: Add a space to the end of the message.
Condition 5: Change the hash key from key $\alpha_1 = 12$ to $\alpha_1 = 12.00000000000001$.
Condition 6: Change the hash key from key $x = 3$ to $x = 3.00000000000001$.

The corresponding hash values in hexadecimal format are given as follows:

Condition 1:
FD97319170D87CB2F25CDCD8917ACA60
Condition 2:
E0BC3C61DFC69D6D40B7491BF88427EC
Condition 3:
9EEAFAAABFFE6539A011C180CA4E70B9
Condition 4: A164446E3387324D7E267991225B6C9
Condition 5: 3E5576B020A6095859217050F59E97F
Condition 6:
761B97931E7528308A4ED14F7ACA8028

The graphical display of binary sequences is shown in Figure 2. The simulation results indicate that the one-way property of the proposed algorithm is remarkably high, and any least difference of the message or key will cause huge changes in the final hash value.

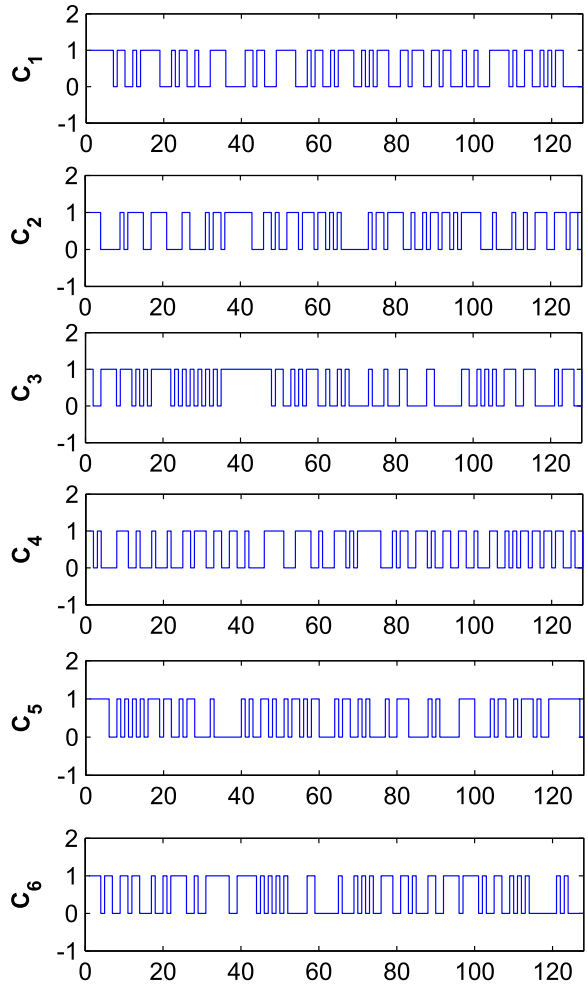

Figure 2 Hash values under different conditions.

3.2 Statistic analysis of diffusion and confusion

In order to hide message redundancy, Shannon pointed in his masterpiece [43] that 'It is possible to solve many kinds of ciphers by statistical analysis,' and therefore, he suggested two methods of diffusion and confusion for the purpose of frustrating the powerful statistical analysis. For the hash value in binary format, each bit is only 1 or 0, so the ideal diffusion effect should be that any tiny changes in initial conditions lead to the 50% changing probability of each bit. A message is selected and the hash value for the message is generated; then, 1 bit of the message is changed randomly and a new hash value is generated. The two hash values are compared with each other, and the changed bits are counted and called B_i. This test is performed N times, and the corresponding distribution of B_i is shown as Figure 3a,b, where $N = 10,000$.

Minimum changed bit number $B_{min} = min(\{B_i\}_1^N)$;
Maximum changed bit number $B_{max} = max(\{B_i\}_1^N)$;
Mean changed bit number $\bar{B} = \sum_{i=1}^{N} B_i/N$;
Mean changed probability $P = (\bar{B}/128) \times 100\%$;
Standard variance of the changed bit number
$\Delta B = \sqrt{\frac{1}{N-1}\Sigma_{i=1}^{N}(B_i - \bar{B})^2}$;
Standard variance
$\Delta P = \sqrt{\frac{1}{N-1}\Sigma_{i=1}^{N}(B_i/128 - P)^2} \times 100\%$;

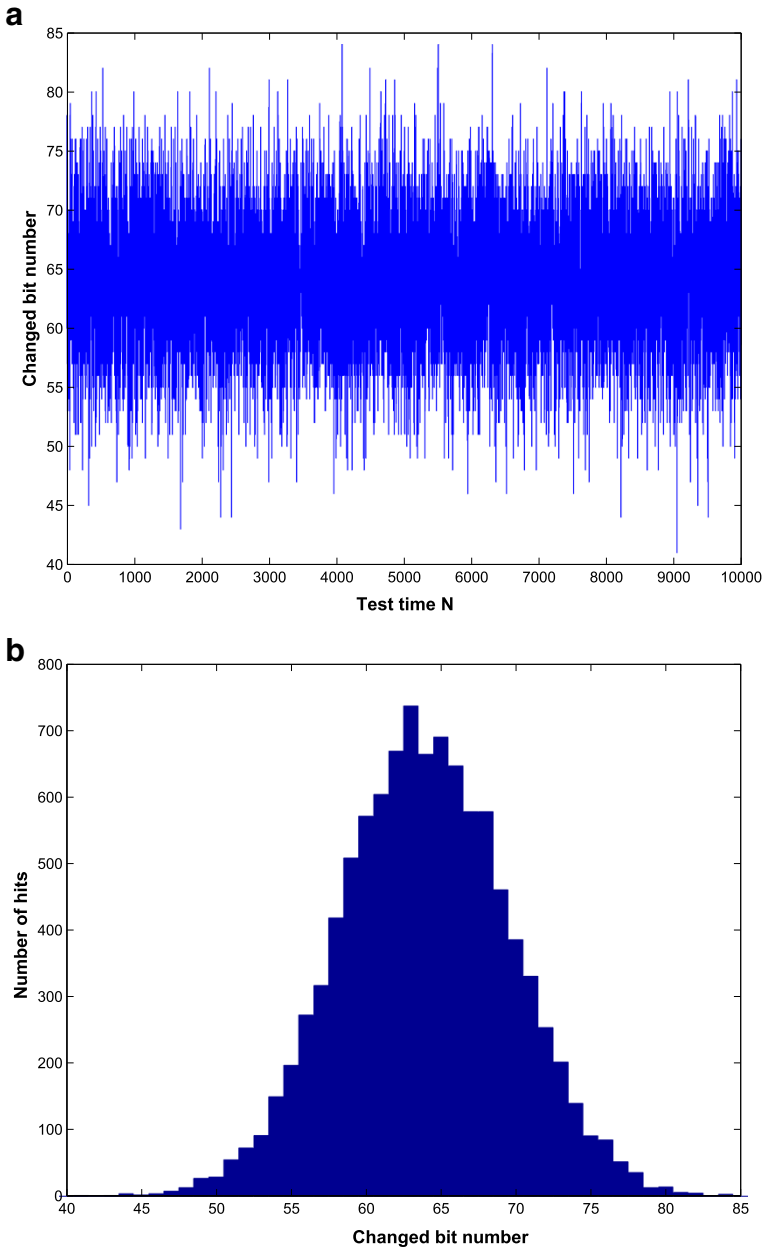

Figure 3 Distribution of changed bit number. (**a**) Plot of B_i and (**b**) histogram of B_i.

Through the tests with $N = 512$, 1,024, 2,048, and 10,000, respectively, the corresponding data are listed in Table 1. Based on the analysis of the data in Table 1, we can draw the conclusion: the mean changed bit number B and the mean changed probability P are both very close to the ideal value of 64 bits and 50%, while ΔB and ΔP are very little, which indicates that the capability for diffusion and confusion is very stable and strong.

3.3 Analysis of collision resistance and birthday attack resistance

Collision resistance and birthday attacks are similar in idea. Both are originated from the probability problem that two random input data are found to hash to the same value. In the proposed scheme, the state of the chaotic map is directly related to each message bit. Also, the control parameters and the initial conditions of the chaotic map are fully affected by the message bits. These opera-

Table 1 Static number of changed bit B_i

	$N = 512$	$N = 1,024$	$N = 2,048$	$N = 10,000$	Mean
$\bar{B}$	63.8808	63.8339	63.8945	63.9192	63.88212
P (%)	49.9069	49.8703	49.9176	49.9369	49.90793
ΔB	6.0032	5.8662	5.7711	5.6467	5.8218
ΔP(%)	4.6900	4.5830	4.5087	4.4115	4.5483
$B_{\min}$	45	45	43	41	43.5
$B_{\max}$	80	82	82	84	82

tions will ensure that each bit of the final hash value is related to all the bits of the message. Therefore, even 1-bit change in the message would lead to a completely different hash code. Moreover, the initial conditions and the control parameters of the chaotic map are related to the message and the state of the map, so the keyed hash function is so much sensitive to the keys.

3.4 Collision test

In order to investigate the collision resistance capability of the proposed algorithm, we have performed the following test [12,28,29]: first, a message is randomly chosen, and the hash value is generated for it and stored in ASCII format. Then, a bit from a random position in the message is chosen and its value is changed, and a new hash value is generated for the new changed message and is stored in ASCII format. Two hash values are compared, and the number of ASCII character with the same value at the same location in the hash value is counted. The absolute difference of two hash values and the theoretical number of ω with different values during N independent experiments which is denoted by $W_N(\omega)$ is calculated using the following formulas:

$$d = \sum_{i=1}^{N} |t(e_i) - t(e\prime_i)|, \tag{2}$$

$$\omega = \sum_{i=1}^{N} f(t(e_i) - t(e\prime_i)), \quad \text{where} \quad f(x,y) = \begin{cases} 1, & x = y, \\ 0, & x \neq y. \end{cases} \tag{3}$$

$$W_N(\omega) = N \times \text{Prob}\{\omega\} = N \frac{s!}{\omega!\,(s-\omega)!} \left(\frac{1}{2^k}\right)^{\omega} \left(1 - \frac{1}{2^k}\right)^{s-\omega}, \tag{4}$$

where e_i and $e\prime_i$ are the ith entry of the original and the new hash value, respectively, and function t(.) converts the entries to their equivalent decimal value. In Equation 4, $\omega = 0, 1, ..., s$. The theoretical values for Equation 4 are as follows [14]: $W_N(0) = 9,416$, $W_N(1) = 572$, $W_N(2) = 12$, and $W_N(\omega) = 0$ for $\omega = 3, 4, ..., 16$. Also, the experimental values of $W_N(\omega)$, the proposed algorithm, are as follows: $W_N(0) = 9,387$, $W_N(1) = 591$, $W_N(2) = 22$, and $W_N(\omega) = 0$ for $\omega = 3, 4, ..., 16$. A plot of the distribution of the number of ASCII characters with the same value at the same location in the hash value is given in Figure 4a,b. It seems that the proposed algorithm possesses a stronger collision resistance capability than the algorithms in [12,14,27,44] and can be compared with [45,46].

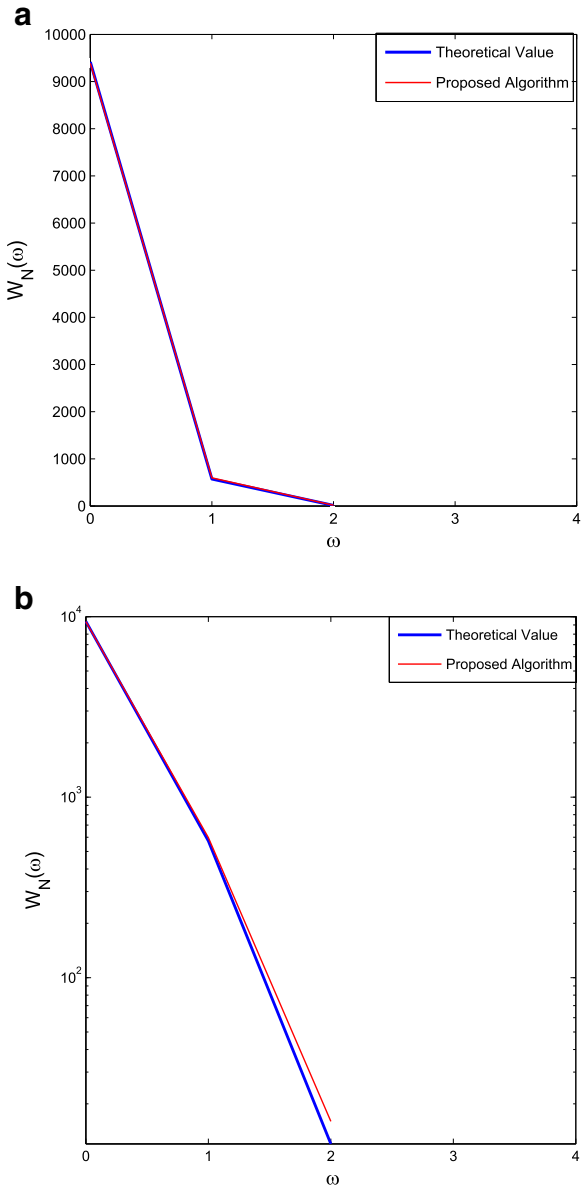

Figure 4 Distribution of *w*. *w* represents the number of 8-bit subblocks with the same value at the same location in hash value ($N = 10,000$). (**a**) Decimal scale of $W_N(\omega)$ and (**b**) logarithmic scale of $W_N(\omega)$.

3.5 Theoretical value of mean of absolute difference

With the assumption that H is a discrete uniform distribution in the range of 0 to 255 (with 16 elements), the ideal mean value of this distribution would be equal to the $\frac{1}{2}$ maximum value of the distribution (in the maximum value condition, all elements are equal to 255): $\frac{255\times16}{2} = 2,040$. Using the ideal theoretical value of the hash digest H, we can calculate the ideal theoretical absolute difference of the two hash digests. With the assumption that the two hash digests H and H' are ideally uniform, the 'Sum of Absolute Difference' of these two digests has to be equal to $\frac{2}{3}$ of the mean value of a uniform distribution [47]. As the mean value of a uniform distribution with the characteristics of a hash digest with the size of 128 bits is equal to 2,040, therefore, the ideal mean value of the absolute difference for H and H' is equal to $\frac{2}{3} \times 2,040 = 1,360$. Minimum, maximum, and mean values of the absolute difference of two hash values are 547, 2,246, and 1,365.9641, respectively. It can be seen that the mean of absolute difference is 1,365.9641 and is very close to the ideal value.

3.6 Pseudo-collision resistance

Without loss of generality, normally in a collision attack, the attacker not only has access to the message (M) and can modify each block (M_i) of the message during the hashing process, but also can change the medial hash value (H_{i-1}). This type of collision attack is usually called pseudo-collision attack. The main goal of this attack is to use the weaknesses in the compression function so that the attacker can find a way to reconstruct a hash value for a different message [45,46]. In the proposed algorithm, the message is processed twice (two different threads), once from the beginning to the end and once from the end to the beginning of the message, so that there would be two medial hash values at the same time for each message block, which with regard to a small change in one or both of the medial values (in here, the control parameters and initial conditions) would lead to a completely different sequence (because of the nature of the proposed high-dimensional chaotic map), so it is almost impossible to find a pattern by changing these values.

3.7 Resistance of forgery attacks

Most of the parallel hash function algorithms have a mixing section in their structure which usually uses the XOR operation for this propose. This operation lets the attackers use the privilege of communicative law and practice forgery attack on the hash function. Some of these algorithms are broken by forgery attack [37]. Not only does the proposed algorithm process the whole message from two sides (which leads to high complexity in mixture), but also in the final step, instead of a simple XOR operation, a conditional method is used in order to resist forgery attack in any section of the algorithm.

3.8 Key space analysis

In this algorithm, besides the control parameter and initial condition of the piecewise nonlinear chaotic map, the parameters of the one-dimensional chaotic map are chosen as the secret keys. From the security point of view, the key space for a cryptographic algorithm should not be less than 2^{128} in order to resist brute force attacks [48,49]. If the precision 10^{-14} is used, the key space size for two coupling parameters, three initial conditions, and three control parameters of the chaotic map is over 2^{279}, so the key space is very large to resist all kinds of brute force attacks.

3.9 Analysis of speed

The proposed hash function is a multi-threaded algorithm which works fine on both single and multi-core processors, while the speed of the algorithm eventually doubles on a multi-core processor. Moreover, the process time needed for generating the hash value is in direct relation with the message length; in Figure 5, we can see that the computational complexity of the algorithm is linear and almost equal to $\Theta(n)$. Although the speed of the algorithms is lower than the existing traditional hash functions such as SHA-1 and MD5 [20], it is, however, acceptable for practical use; at the same time, the algorithm is so flexible since the digest size can be 128, 160, 256, or 512 bits in length. Finally, we present the average encryption speed comparison based on the specified platform with some hash algorithms [12,13,20,21, 28,29,50,51]. The average encryption speed of our algorithm (45 Mbps) is higher than that in [12,13,50,51]. In addition, the algorithms of [20,21,28,29] show higher encryption speed.

3.10 Uniform distribution on hash space

To check the distribution property in hash space, similar to that in [14], the following test is performed: two hash values are generated according to the method described in the previous subsection, and then, the number of the bits changed at each bit's location is counted. Figure 6 shows the statistical results for $N = 10,000$. The minimum, maximum, and mean of the changed bit number are 4,896, 5,124, and 5,063, respectively, for $N = 10,000$. It can be seen that the mean of the toggled bit numbers is 5,063 and is very close to the ideal value of 5,000 (half of the test times). All toggled bit numbers of each bit location in hash space also concentrate around the ideal value. Apparently, this shows the resistance against statistical attack.

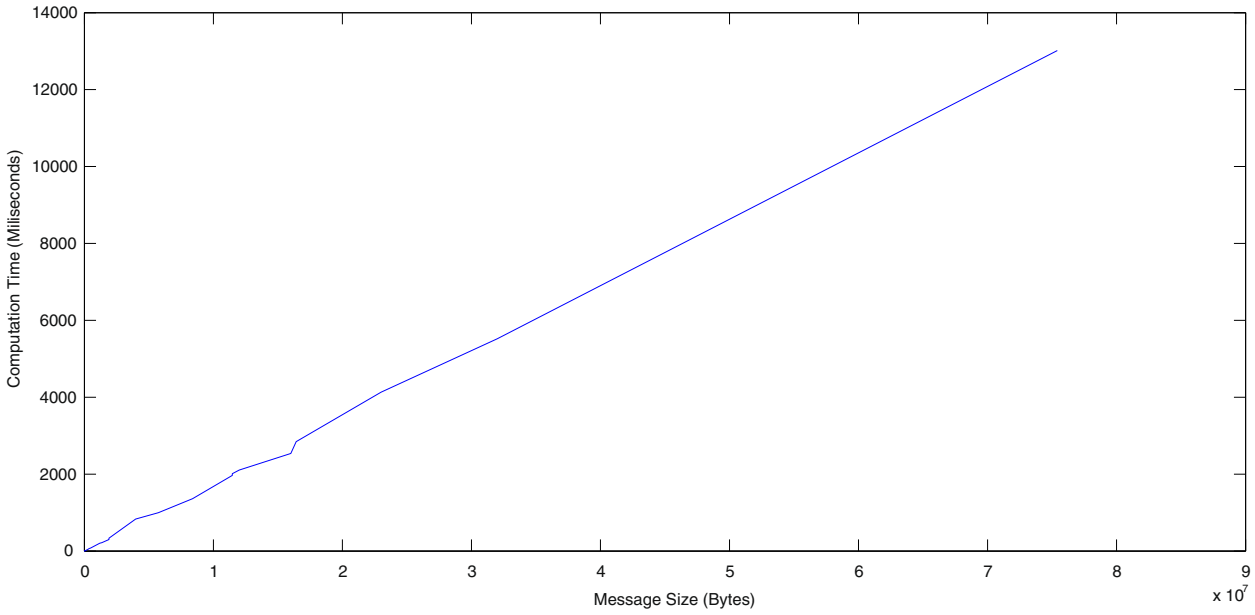

Figure 5 Computational time vs. message size.

3.11 Randomness tests for the hash

Cryptographic hash functions are an important tool of cryptography and play a fundamental role in efficient and secure information processing. Cryptographic hash functions can be used to construct pseudo-random number generators. To ensure the security of a cryptosystem, the cipher must have some properties such as good distribution, long period, high complexity, and efficiency. In this algorithm, several tests of randomness are performed on a stream of random numbers generated by the main dynamical system (Equation 1) of the hash function in order to test the randomness of the presented algorithm. These

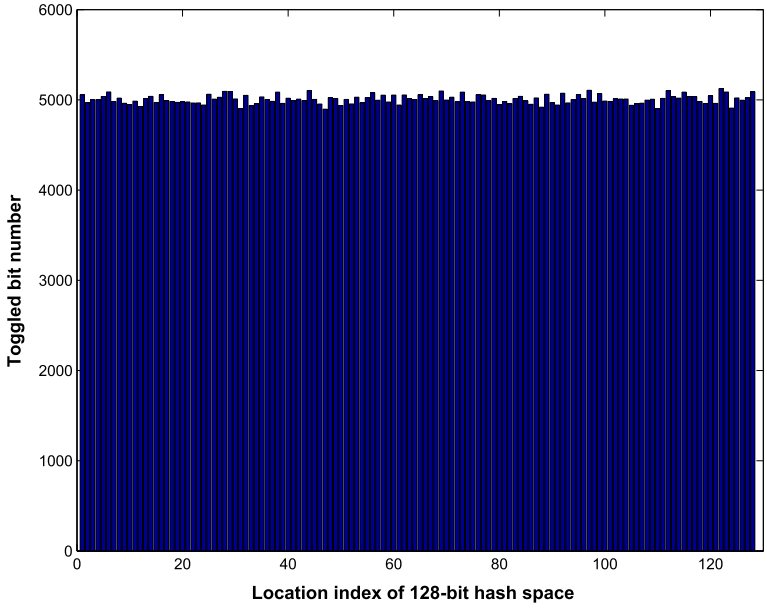

Figure 6 Distribution of hash value in hash space. The minimum, maximum, and mean of the toggled bit number are 4,896, 5,124, and 50,063, respectively, for $N = 10,000$.

tests include DIEHARD [52], NIST statistical test suite [53], and ENT test suite [54]. According to Tables 2 and 3, 4, and 5 which present NIST, DIEHARD, and ENT test results, respectively, of the sample, it can be concluded that the stream has a highly random behavior.

3.12 Resistance to birthday attack

The birthday attack is one of the common classic attacks on cryptographic hash functions, which is independent of the algorithm, and can be applied on any algorithm. The main goal of this attack is to find two messages with identical hash values with less than $2^{n/2}$ trials (n is the size of hash value) [46,51,55]. In fact, the algorithm is flexible so that the length of the hash value can be tuned. If the hash value size is set to 128, the difficulty of the attack is 2^{64} which is huge enough to resist brute force attack. Also in the proposed algorithm, in order to analyze the security and resistance against such an attack, several tests including the random number batteries and collision tests are applied. It can be seen from the results in Table 5 that the entropy of the random numbers generated by the hash function is very close to the ideal value ($7.999994 \approx 8$) and that the serial correlation coefficient in the same table is very close to zero, which demonstrates the collision resistance nature of the algorithm. The proposed algorithm, which is visually discussed using the block diagram (Figure 1), is very complex, and the initial conditions and control parameters are modified in each round. According to [56], 'if an appropriate padding scheme is used and the compression function is collision-resistant, then the hash function will also be collision resistant'. Therefore, the results of the tests, size of the hash value for the presented algorithm, and collision resistance of the proposed algorithm suggest that the effectiveness of the birthday attack is almost impossible and that the algorithm is resistant against this type of attack.

3.13 Flexibility

In this paper, an algorithm based on high-dimensional chaotic map is proposed in order to solve the problems such as the size of hash functions, padding of the message, resistance against differential attacks, and capability of efficiently working on multi-core CPUs using parallel processing. Although the algorithm is designed as an

Table 2 Results of the SP800-22 test suite for the 32-bit proposed high-dimensional map

Test name	Test parameters	*P* value	Result
Frequency		0.671779	Success
Block-frequency ($m = 128$)		0.911413	Success
Runs ($M = 10,000$)		0.739918	Success
Long runs of ones		0. 66882	Success
Rank		0.253551	Success
Spectral DFT		0.5622515	Success
No overlapping templates ($m = 9, B = 101001100$)		0.739918	Success
Overlapping templates ($m = 9, M = 1,032, N = 968$)		0.911413	Success
Universal ($L = 7, Q = 1,280, K = 141,577$)		0.602458	Success
Lempel-Ziv complexity		0.662344	Success
Linear complexity		0.407091	Success
Serial	*P* value 1	0.602458	Success
Serial	*P* value 2	0.02624	Success
Approximate entropy ($m = 10$)		0.299251	Success
Cumulative sum forward		0.534146	Success
Cumulative sum reverse		0.035174	Success
Random excursions	$X = -4$	0.637119	Success
Random excursions	$X = -3$	0.834308	Success
Random excursions	$X = -2$	0.991468	Success
Random excursions	$X = -1$	0.066882	Success
Random excursions	$X = 1$	0.275709	Success
Random excursions	$X = 2$	0.213309	Success
Random excursions	$X = 3$	0.122325	Success
Random excursions	$X = 4$	0.437274	Success

Table 3 Results of the SP800-22 test suite for the 32-bit proposed high-dimensional map

Random excursion variant (state x)	Test parameters	P value	Result
Random excursions	$X = -9$	0.964295	Success
Random excursions	$X = -8$	0.213309	Success
Random excursions	$X = -7$	0.911413	Success
Random excursions	$X = -6$	0.350485	Success
Random excursions	$X = -5$	0.637119	Success
Random excursions	$X = -4$	0.834308	Success
Random excursions	$X = -3$	0.964295	Success
Random excursions	$X = -2$	0.637119	Success
Random excursions	$X = -1$	0.999438	Success
Random excursions	$X = 1$	0.834308	Success
Random excursions	$X = 2$	0.534146	Success
Random excursions	$X = 3$	0.834308	Success
Random excursions	$X = 4$	0.739918	Success
Random excursions	$X = 5$	0.834308	Success
Random excursions	$X = 6$	0.964295	Success
Random excursions	$X = 7$	0.275709	Success
Random excursions	$X = 8$	0.122325	Success
Random excursions	$X = 9$	0.066882	Success

Table 4 DIEHARD test suite for the 32-bit proposed high-dimensional map

Test name	Average value	Result
Birthday spacing	0.800832	Success
Overlapping permutation	0.693145	Success
Binary rank 31×31	0.974342	Success
Binary rank 32×32	0 .770208	Success
Binary rank 6×8	0.264802	Success
Bitstream	0 .45412	Success
OPSO	0.6788	Success
OQSO	0.3555	Success
DNA	0.5722	Success
Count the ones 01	0.104101	Success
Count the ones 02	0.651653	Success
Parking lot	0.926198	Success
Minimum distance	0 .748646	Success
3DS spheres	0 .137205	Success
Squeeze	0 .019340	Success
Overlapping sum	0 .964091	Success
Runs	0.645063	Success
Craps	0.480936	Success

Table 5 Max grade of ENT test suite

Test name	Average value	Result
Entropy	7.999994	Success
Arithmetic mean	127.5165	Success
Monte Carlo	3.140157946	Success
Chi square	55.73	Success
Serial correlation coefficient	0.000277	Success

unkeyed hash function, the control parameters and the initial conditions of the algorithm can be treated as the secrete key for a keyed hash function. In comparison to the conventional hash algorithm such as MD5, MAC-DES, and HMAC-MD5 with fixed length, the proposed algorithm is advantageous in terms of adaptation to 160, 256, or 512 bits, with a small change in the steps, and since the algorithm is designed in such a way that it can work in parallel, therefore in longer digest speed, it would not be affected so much. Furthermore, in the proposed algorithm, the chaotic map is iterated with a double-precision floating-point arithmetic. A double-precision floating-point number is represented according to the IEEE 754 floating-point standard found in virtually almost all computers produced since 1980 [57]. The program is implemented in Visual C++.NET, and the standard of the floating point which is used in this compiler is the IEEE 754 floating-point standard so that the hash values generated on any computer would be the same.

4 Conclusion

A novel parallel hash function based on a high-dimensional chaotic map is proposed. In the core of the presented dynamical system lies interesting features such as invariant measure, ergodicity, bifurcation without period doubling, and the possibility of KS-entropy calculation [4]. By using significant properties of chaos and structure of the chaotic map, in the proposed algorithm, the message is totally connected to all parameters, so this structure can ensure the uniform sensitivity of hash value to the message. Since the coupled architecture is a complex chaotic system, it is very difficult or even impossible to predict its time series. Therefore, the complexity and nonlinearity of the chaotic map yield strong bit confusion and diffusion with the low expense of floating-point computations. Theoretical analysis and numerical simulation indicate that the proposed new scheme achieves several desirable features such as high flexibility, good statistical properties, parallel implementation, high key sensitivity, and message sensitivity. Moreover, it resists linear analysis, exhaustive key search, and statistical attack. It can be concluded that the proposed algorithm fulfills the performance requirements of hash function and that it is practical and reliable, with high potential to be adopted in real-world applications.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work was supported by USM (no. 1001/PICOMP/817059).

Author details
[1]School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia. [2]School of Physics, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia.

Received: 26 July 2012 Accepted: 6 June 2013
Published: 3 July 2013

References

1. S Li, Analyses and new designs of digital chaotic ciphers. Ph.D. thesis, School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China (2003)
2. S Behnia, A Akhshani, S Ahadpour, H Mahmodi, A Akhavan, A fast chaotic encryption scheme based on piecewise nonlinear chaotic maps. Phys. Lett. A. **366**, 391–396 (2007)
3. A Akhshani, H Mahmodi, A Akhavan, A novel block cipher based on hierarchy of one dimensional composition chaotic maps IEEE Int. Conf. Image Process, 1993–1996. (IEEE, ICIP 2006)
4. S Behnia, A Akhshani, S Ahadpour, A Akhavan, H Mahmodi, Applications of tripled chaotic maps in cryptography. Chaos Soliton Fract. **40**, 505–519 (2009)
5. A Akhshani, S Behnia, A Akhavan, H Abu Hassan, Z Hassan, A novel scheme for image encryption based on 2D piecewise chaotic maps. Optics Comm. **283**, 3259–3266 (2010)
6. Z-L Zhu, W Zhang, K-W Wong, H Yu, A chaos-based symmetric image encryption scheme using a bit-level permutation. Inform. Sci. **181**, 1171–1186 (2011)
7. A Akhavan, H Mahmodi, A Akhshani, A new image encryption algorithm based on one dimensional polynomial chaotic maps. Lect. Notes Comput. Sci. **4263**, 963–971 (2006)
8. JW Byun, DH Lee, JI Lim, EC2C-PAKA: an efficient client-to-client password-authenticated key agreement. Inform. Sci. **177**, 3995–4013 (2007)
9. D Xiao, X Liao, S Deng, A novel key agreement protocol based on chaotic maps. Inform. Sci. **177**, 1136–1142 (2007)
10. K Wang, WJ Pei, LH Zou, YM Cheung, ZY Hea, Security of public key encryption technique based on multiple chaotic systems. Phys. Lett. A. **360**, 259–262 (2006)
11. B Wang, Q Wu, Y Hu, A knapsack-based probabilistic encryption scheme. Inform. Sci. **177**(19), 3981–3994 (2007)
12. Y Wang, X Liao, D Xiao, K Wong, One-way hash function construction based on 2D coupled map lattices. Inform. Sci. **178**, 1391–1406 (2008)
13. X Yi, Hash function based on chaotic tent maps. IEEE Trans. Circuits Syst. II. **52**(6), 354–357 (2005)
14. J Zhang, X Wang, W Zhang, Chaotic keyed hash function based on feedforwardŰfeedback nonlinear digital filter. Phys. Lett. A. **362**, 439–448 (2007)
15. Q Zhou, X Liao, K-W Wong, Y Hu, D Xiao, True random number generator based on mouse movement and chaotic hash function. Inform. Sci. **179**, 3442–3450 (2009)
16. Z Huang, A more secure parallel keyed hash function based on chaotic neural network. Commun. Nonlinear Sci. Numer. Simulat. **16**, 3245–3256 (2011)
17. S Deng, D Xiao, Y Li, W Peng, A novel combined cryptographic and hash algorithm based on chaotic control character. Commun. Nonlinear Sci. Numer. Simulat. **14**, 3889–3900 (2009)
18. M Amin, OS Faragallah, AA Abd El-Latif, Chaos-based hash function (CBHF) for cryptographic applications. Chaos Solitons Fractals. **42**, 767–772 (2009)
19. R Rivest, in *Advances in Cryptology-CRYPTO'90*. The MD4 message digest algorithm (Springer-Verlag, London, 1991), pp. 303–311
20. R Rivest, The MD5 Message-Digest Algorithm. RFC 1321 (1992)
21. NIST, FIPS PUB 180-1, Secure Hash Standard (1995)
22. S Deng, Y Li, D Xiao, Analysis and improvement of a chaos-based Hash function construction. Commun. Nonlinear Sci. Numer. Simulat. **15**(5), 1338–1347 (2010)
23. J Liang, X-J Lai, Improved collision attack on hash function MD5. J. Com. Sci. Tech. **22**, 79–87 (2007)
24. Y Sasaki, Y Naito, N Kunihiro, K Ohta, Improved collision attack on MD5. Cryptology ePrint Archive, Report 2005/400 (2005). http://eprint.iacr.org/2005/400 Accessed 23 June 2013
25. X Wang, D Feng, X Lai, H Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199 (2004). http://eprint.iacr.org/2004/199. Accessed 23 June 2013
26. X Wang, YL Yin, H Yu, Finding collisions in the full SHA-1. Lect. Notes Comp. Sci. **3621**, 17–36 (2005)
27. D Xiao, X Liao, S Deng, One-way hash function construction based on the chaotic map with changeable-parameter. Chaos Soliton Fract. **24**, 65–71 (2005)
28. A Akhavan, A Samsudin, A Akhshani, Hash function based on piecewise nonlinear chaotic map. Chaos Soliton Fract. **42**, 1046–1053 (2009)
29. A Akhshani, S Behnia, A Akhavan, MA Jafarizadeh, H Abu Hassan, Z Hassan, Hash function based on hierarchy of 2D piecewise nonlinear chaotic maps. Chaos Soliton Fract. **42**, 2405–2412 (2009)
30. HS Kwok, WKS Tang, A chaos-based cryptographic hash function for message authentication. Int. J. Bifurcation Chaos. **15**, 4043–4050 (2005)
31. S Lian, ZX Liu, Hash function based on chaotic neural networks. IEEE ISCAS, 237–240 (2006)
32. J Peng, S-Z Jin, H-L Liu, W Zhang, A novel hash function based on hyperchaotic Lorenz system. Fuzzy Inform. Eng. **2**, 1529–1536 (2009)
33. A Ghaemi Bafghi, R Safabakhsh, B Sadeghiyan, Finding the differential characteristics of block ciphers with neural networks. Inform. Sci. **178**, 3118–3132 (2008)
34. K Wong, A combined chaotic cryptographic and hashing scheme. Phys. Lett. A. **307**, 292–298 (2003)
35. G Alvarez, F Montoya, M Romera, G Pastor, Cryptanalysis of dynamic look-up table based chaotic cryptosystems. Phys. Lett. A. **326**, 211–218 (2004)
36. D Xiao, W Peng, X Liao, T Xiang, Collision analysis of one kind of chaos-based hash function. Phys. Lett. A. **374**, 1228–1231 (2010)
37. W Guo, W Xiaomin, H Dake, C Yang, Cryptanalysis on a parallel keyed hash function based on chaotic maps. Phys. Lett. A. **373**, 3201–3206 (2009)
38. F Peng, SS Qiu, M Long, One-way hash function construction based on two-dimensional hyper-chaotic mappings. Acta. Phys. Sin. **54**, 4562–607 (2005). [in Chinese]
39. MA Jafarizadeh, S Behnia, S Khorram, H Naghshara, Hierarchy of chaotic maps with an invariant measure. J. Stat. Phys. **104**, 1013–1028 (2001)
40. MA Jafarizadeh, S Behnia, Hierarchy of chaotic maps with an invariant measure and their coupling. Physica D. **159**, 1–21 (2001)
41. S Behnia, A Akhshani, H Mahmodi, A Akhavan, A novel algorithm for image encryption based on mixture of chaotic maps. Chaos Soliton Fract. **35**, 408–419 (2008)
42. MA Jafarizadeh, S Behnia, Hierarchy of chaotic maps with an invariant measure and their compositions. J. Nonlinear Math. Phys. **1**, 1–16 (2002)
43. CE Shannon, Communication theory of secrecy systems. Bell Sys. Tech. J. **28**, 656–715 (1949)
44. H Yang, K-W Wong, X Liao, Y Wang, D Yang, One-way hash function construction based on chaotic map network. Chaos Soliton Fract. **41**, 2566–2574 (2009)
45. D Xiao, X Liao, Y Wang, Parallel keyed hash function construction based on chaotic neural network. Neuralcomputing. **72**, 2288–2296 (2009)
46. Y Wang, K-W Wong, D Xiao, Parallel hash function construction based on coupled map lattices. Commun. Nonlinear Sci. Numer. Simulat. **16**, 2810–2821 (2011)
47. R Johnsonbaugh, *Discrete Mathematics*. (Macmillan Pub. Co./Collier Macmillan Publishers, New York/London, 1984)
48. B Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edn. (Wiley, New York, 1996)
49. ECRYPT I I, ECRYPT II yearly report on algorithms and keysizes (2010). http://www.ecrypt.eu.org/documents/D.SPA.13.pdf. Accessed 23 June 2013
50. H Zhang, X Wang, Z Li, D Liu, One way hash function construction based on spatiotemporal chaos. Acta. Physica Sinica. **54**, 4006–4011 (2005)
51. A Kanso, H Yahyaoui, M Almulla, Keyed hash function based on a chaotic map. Inform. Sci. **186**, 249–264 (2012)
52. G Marsaglia, Diehard, a battery of tests for random number generators (1997). http://stat.fsu.edu/pub/diehard/. Accessed 23 June 2013

53. A Rukhin, J Sotto, J Nechvatal, M Smid, E Barker, S Leigh, M Levenson, M Vangel, D Banks, A Heckert, J Dray, S Vo, A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22. NIST Gaithersburg 2001
54. J Walker, ENT, A pseudorandom number sequence test program (1998). http://www.fourmilab.ch/random/. Accessed 23 June 2013
55. A Menezes, P van Oorschot, S Vanstone, *Handbook of Applied Cryptography*. (CRC, Boca Raton, 1996)
56. IB Damgård, in *Advances in cryptology - CRYPTO '89 Proceedings*, Lect. Notes Comput. Sci. A design principle for hash functions (Springer, New York, 1989), pp. 416–427
57. D Goldberg, D Priest, What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**, 5–48 (1991)

doi:10.1186/1687-6180-2013-126
Cite this article as: Akhavan *et al.*: **A novel parallel hash function based on 3D chaotic map.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:126.