International Journal of the Physical Sciences Vol. 6(23), pp. 5571-5574, 9 October, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.239
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

A collision-based fast hash function

Zhou Qing

College of Computer Science, Chongqing University, Chongqing, 400044, China. E-mail: tzhou@cqu.edu.cn. Tel: +86 – 2365103199.

Accepted 25 May, 2011

A cryptographic hash function is proposed based on a simple physical model. The model contains two balls and a ring. Each ball moves and collides with the fixed ring. Message is modeled as perturbations, which influence the motions of balls. The final positions and velocities of the two balls are converted to hash value. Simulations demonstrate that the proposed simple physical model has excellent randomness and sensitivity properties. Moreover, it is more efficient than most well-known hash functions.

Key words: Collision, hash function, computation model, cryptographic application.

INTRODUCTION

Communication via internet, mobile and sensor networks is changing our lives. It is important to verify whether the received message is trustable or not during communication. Hash functions help to achieve this goal efficiently.

A hash function takes a variable-sized message and converts it into a fixed-size bit string called hash value. Secure hash function must be resistant to collision. Here, collision means the undesirable event that two different messages are found to share the same hash value. To avoid collision, a hash function should have randomness and sensitivity properties.

Wang and Yu (2005) proposed a method to construct collisions for some popular hash functions, and the conventional hash functions, such as MD5 (Rivest, 1992) and SHA (U.S. NIST, 2008) are facing challenges. Some new techniques for the design of cryptographic hash function become interested. Most of them are based on nonlinear maps. For example, Wong (2003) constructed a hash function using chaotic logistic map. Yi (2005) designed a hash function based on 1-D piecewise-linear map. Recently, some new structures of hash functions were proposed, such as coupled map lattice (Yang et al., 2009), chaotic map network (Wang et al., 2008) and neural network (Xiao et al., 2009). In all these schemes, message block is mapped to initial states or system parameters of the maps. It usually requires dozens of or even hundreds of iterations for them to fulfill security requirements. Hence, they are inefficient in processing messages.

In physics, the concept of collision is in contrast to that in hash functions. It means the situation whereby two or more objects strike or come together. Many computation models are designed based on the concept of collision (Margolus, 2002).

For example, HPP model, the first lattice gas model proposed to simulate fluid flows, contains a collision process (Hard et al., 1976). In this process, if two particles collide, both of them turn right to their original direction. Some cellular automata are also motivated from the collision of particles (Morita, 2008).

In this paper, we try to design a hash function from a collision based physical model that consists of a ring and a ball. The ring is fixed and the ball moves and collides with the ring. This model has three advantages as a prototype of a hash function:

1. The final system states are extremely sensitive to the initial state of the ball;
2. The phase space of the system presents excellent randomness property;
3. The model is simple so that the hash function can be implemented efficiently.

Some problems need to be solved to construct a hash function based on this physical model, that is, how to influence the motion of the ball using message blocks? How to transform the phase space of the system into hash value? How to enhance the security of the hash

function by a slight change of the model? Our proposed hash function presents excellent randomness and sensitivity properties after we overcome these problems. Moreover, it is more efficient than many known hash functions.

HASH FUNCTION BASED ON A COLLISION MODEL

Ring-ball collision model

Starting with the simplest model which contains a fixed ring and only one ball; the collisions between them are elastic, and there is no friction. After each collision, the direction of the velocity of the ball is changed. The orbit of the ball is determined by Equation 1 to 3.

$$x_n = \begin{cases} \dfrac{-kt - \sqrt{k^2 - t^2 + 1}}{1 + k^2}, & \dfrac{\pi}{2} < \alpha_{n-1} < \dfrac{3\pi}{2} \\ \dfrac{-kt + \sqrt{k^2 - t^2 + 1}}{1 + k^2}, & \text{otherwise} \end{cases}, \tag{1}$$

$$y_n = k(x_n - x_{n-1}) + y_{n-1}, \tag{2}$$

$$\alpha_n = \left[2f(x_n + jy_{n-1}) - \alpha_{n-1}\right] \bmod 2\pi, \tag{3}$$

where (x_n, y_n) denotes the position where *n*th collision occurs, α_n denotes the angle of the velocity of the ball after *n*th collision, $f(Z)$ denotes the phase angle of the complex number *Z* and *k* and *t* are calculated by Equation 4.

$$\begin{cases} k = \tan(\alpha_{n-1}) \\ t = y_{n-1} - kx_{n-1} \end{cases}. \tag{4}$$

The model is sensitive to the initial angle of the ball. The collision positions and the angles of the ball are significantly different even if the initial angle is changed slightly.

Adding perturbations to the model

The hash value should depend on each block of message data. How each block of data influences the collision between the ball and the ring? Conventionally, each message block is converted to initial states or parameters of the function. In our approach, we treated each block of data as a perturbation of the collision, that is, the direction of the velocity of the ball changes with an angle determined by a block of data of every several collisions between the ball and the ring. When $n \bmod 5 \equiv 0$, we use Equation 5 to calculate α_n, otherwise, we use Equation 3.

$$\alpha_n = (\alpha_{n-1} + 2\pi m_i / 256) \bmod 2\pi. \tag{5}$$

Now, the collision model can work as a hash function: the message is modeled as a disturbance to the collision, and the x_n (Note that y_n depends on x_n) and α_n are taken as the hash value. Nevertheless, hash value contains not more than 104 bits if we adopt the double-precision floating-point numbers denoted by the IEEE Standard 754, where each number contains at most 52-bit information. Meanwhile, a practical hash function is expected to produce a hash value with 128 or more bits.

There are two approaches to solve this problem. The first one is to use other numeric format other than the IEEE Standard 754. Thus, both x_n and α_n can be represented with a 64-bit precision. Nevertheless, such format may not be supported by most computers, or it requires large quantity of extra computations. The second approach is to increase the number of the balls contained in the model.

Hash function based on two-ball collision model

If a model contains two or more balls, we must ascertain that the two balls influence each other. Otherwise, its hash value corresponds to the concatenation of hash values generated by the two one-ball collision models. And attackers can break the model easily using divide-and-conquer method. To solve this problem, we demand that the change of the direction of one ball depend on both the message and the direction of another ball (Equation 6).

$$\begin{cases} \alpha_n^{(1)} = (\alpha_n^{(2)} + 2\pi m_i / 256) \bmod 2\pi \\ \alpha_n^{(2)} = (\alpha_n^{(1)} + 2\pi m_i / 256) \bmod 2\pi \end{cases}, \tag{6}$$

where $\alpha_n^{(1)}$ and $\alpha_n^{(2)}$ are the angle of the velocity of the first and second ball after *n*th collision.

The last several byte of the message has relatively weaker impact on the final state of the balls. So after the states of the two balls are interfered by each byte of the message, we repeat the process again. In other words, the motions of the two balls will be affected by the message for twice. Then, we convert the final state of the two balls, $x_{2n}^{(1)}, x_{2n}^{(2)}, \alpha_{2n}^{(1)}, \alpha_{2n}^{(2)}$, into 128-bit hash value. Equation 7 describes the relationship between a floating-point number *b* and the 32-bit data [b_1, ..., b_{32}]. Concatenating four 32-bit data together, we obtain a 128-bit hash value.

$$b = \sum_{i=1}^{32} b_i 2^{-i} + \varepsilon, \tag{7}$$

where ε is a number smaller than 2^{-32}.

The proposed hash function can be described as follows:

Step 1. $n = 1$, $i = 1$, set the initial horizontal coordinates and angles of the two balls;
Step 2. Change the positions and directions of the two balls according to Equation 1 to 3; set $n = n + 1$;
Step 3. If n mod 5 = 0, change the directions of the two balls using *i*th message block according to Equation 6; set $i = i + 1$;
Step 4. If all message blocks have been processed, go to step 5; otherwise, go back to Step 2;
Step 5. Convert the final state of the two balls into 128-bit hash

Table 1. Statistics of the proportion of bit one of 1000 hash values.

Maximum (%)	Minimum (%)	Mean (%)	Standard deviation
63.28	36.72	49.88	0.0439

value according to Equation 7.

SIMULATION RESULTS AND PERFORMANCE ANALYSIS

We simulated the proposed hash function to check its randomness and sensitivity properties as well as its ability to resist collision. Then, we compare its proceeding speed with several well-known hash functions.

The initial horizontal coordinates and angles of the two balls are set as $x_0^{(1)}=0.1$, $x_0^{(2)}=-0.3$, $\theta_0^{(1)}=3\pi/2$, $\theta_0^{(2)}=\pi/2$. The message to be hashed is the text selected from the first paragraph of the paper: "A hash function takes a variable-sized message and converts it into a fixed-size bit string called hash value. Hash functions are important in cryptographic applications such as the message authentication and the digital signature schemes." The following messages are derived from it.

M_1: The original message;
M_2: Changes the first character A in the original message into B;
M_3: Deletes the hyphen in 'variable-sized';
M_4: Changes the full stop at the end of the original message into comma;
M_5: Adds a blank space to the end of the original message.

The hash values of the aforementioned messages are listed as follows:

H_1: A8136E6301376B3DA217BBFE3DE944A6
H_2: 0FF6B0A1A6932537C41D2995B37DE0FE
H_3: E65BA589190123E4613F9488F9406691
H_4: 1BA7609FDE2D2EDCE5FDB727CD38CD5C
H_5: 810EC9D1B6A72ACB9A8F2B738BFE5FA2

Figure 1 depicts the aforementioned hash values. It seems that all of them are random-like and a slightest change of the message leads to significant difference between the hash values. In the following, we present more detailed results about the secure properties of the proposed hash function.

Evaluation of randomness property

The most important measurement of randomness of hash value is the proportion of 1's of all bits of hash value, which is expected to be closed to 0.5. To investigate the randomness of the proposed hash function carefully, we computed the hash values of 1000 randomly generated messages. The statistics of the proportion of 1's of the 1000 hash values are presented in Table 1.

The averaged proportion is almost equal to 0.5. The standard deviation equals to 0.0439, which is also closed to the theoretic value of 0.0442 determined by Equation 8, with n = 128 denoting the number of bits in the hash value. The results demonstrate that the proposed hash function has a satisfactory property of randomness.

$$\sigma(n)=\frac{1}{2\sqrt{n}}. \quad (8)$$

Evaluation of sensitivity property

The sensitivity of a hash function can be measured by bit change rate c, which is calculated by Equation 9.

$$c=\frac{1}{N}\sum_{i=1}^{N}H_1(i)\oplus H_2(i), \quad (9)$$

where $H_1(i)$ and $H_2(i)$ are the i-th bits of two hash values H_1 and H_2, respectively; N is the number of bits in hash value. For hash functions with good sensitivity, the bit change rate is close to 0.5 even if the messages differ in only one bit. To test the sensitivity of the proposed hash function, 1000 messages were generated randomly by program. For each message, we flipped one randomly chosen bit and obtained a new message. Then, we calculated the bit change rate for each pair of messages. Table 2 presented the statistics of change rates for 1000 pairs of messages, both the averaged value and the standard deviation of which are extremely closed to the theoretical value of 0.5 and 0.0442, respectively.

Evaluation of collision resistance ability

A secure hash function should be resistant to collision. In order to test the collision-free properties of the proposed hash function, we generated 1000 pairs of messages randomly; each pair of messages differs in only one bit. Then, the hash values of each pair of messages were computed and two hash values are compared byte by

Table 2. Statistics of change rates for 1000 images.

Maximum (%)	Minimum (%)	Mean (%)	Standard deviation
67.19	35.16	49.94	0.0429

Table 3. Performance comparisons with other hash function.

Hash function	Number of iterations	Output size (bits)	Number of iterations per output bit
Wong's scheme	200	8	25
Yi's scheme	75	104	0.72
MD5	64	128	0.50
SHA-512	80	512	0.16
Proposed scheme	10	128	0.08

byte. The number of identical bytes in the same position of two hash values is expected to be small. Simulation shows that none of these 1000 pairs shares more than two identical bytes in the same position.

Performance analysis

Most hash functions map message block to their initial states or function parameters, such as the classic MD5 (Rivest, 1992) or SHA-512 (U.S. NIST, 2008) algorithm, or new method proposed by Wong (2003) or Yi (2005). In order to achieve proper security level, they need a large number of iterations. By treating the message block as perturbations to the phase space of the model, our proposed hash function only requires 10 iterations to process each message block. Table 3 lists the number of bits in hash value versus the number of iterations required for several classic hash functions. It shows that our approach is the most efficient one in regards to the speed to generate one bit of hash value.

CONCLUSION

Collision is a common event occurring in physical world. Some computation models are constructed based on the concept of collision. In this paper, a hash function is proposed based on a simple physical model containing two balls and a ring. Simulation demonstrates that the proposed hash function has excellent randomness, sensitivity and collision-resistance properties. Moreover, it achieves high security level in much less number of iterations than many well-known hash functions. This simple physical model seems promising to fulfill the requirements of the complex cryptographic hash function.

ACKNOWLEDGEMENTS

The work is supported by the National Nature Science Foundation of China under Grant 61003246 and 61070246 and the National Science Foundation of Chongqing under Grant CSTC, 2009BB2208.

REFERENCES

Hard J, Pazzis O, Pomeau Y (1976). Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. Phys. Rev., A. 13: 1949-1961.

Margolus N (2002). Physics-like Models of Computation. In: Andrew (ed.) Collision-based computing. Springer-Verlag, London, pp 83-106.

Morita K (2008). Reversible computing and cellular automata - A Survey. 395: 101-131.

Rivest R (1992). The MD5 Message-Digest Algorithm, http://www.ietf.org/rfc/rfc1321.txt.

U.S. National Institute of Standards and Technology (2008). FIPS PUB 180-3: Secure Hash Standard, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

Wang XY, Yu HB (2005). How to break MD5 and other hash functions. In: Cramer R (ed.) Lecture Notes in Computer Science 3494. Springer-Verlag, German, pp 19-35.

Wang Y, Liao XF, Xiao D, Wong KW (2008). One-way hash function construction based on 2D coupled map lattices. Inf. Sci., 178(5): 1391-1406.

Wong KW (2003). A combined chaotic cryptographic and hashing scheme. Phys. Lett., A. 307:292–298.

Yang HQ, Wong KW, Liao XF, Wang Y, Yang DG (2009). One-way hash function construction based on chaotic map network. Chaos Solitons and Fractals. 41: 2566-2574.

Xiao D, Liao XF, Wang Y (2009). Parallel keyed hash function construction based on chaotic neural network. Neurocomput., 72: 2288-2296.

Yi X (2005). Hash Function Based on Chaotic Tent Maps. IEEE Trans. Circ. Sys., II. 52: 354-357.