HASH FUNCTION IMPLEMENTATION USING ARTIFICIAL NEURAL NETWORK

V. R. Kulkarni[1,] Shaheen Mujawar[2] and Sulabha Apte[3]

[1] Department of Information Science and Engineering
Gogte Institute of Technology, VTU University
Belgaum, Karnataka, India
Vidyakulkarni21@rediffmail.com

[2]Department of Computer science and Engineering
Gogte Institute of Technology, VTU University
Belgaum, Karnataka, India
Shahu1985@gmail.com

[3]Department of Computer science and Engineering
Walchand Institute of Technology, Solapur
aptesulabha@gmail.com

ABSTRACT

In this paper an algorithm for one-way hash function construction based on a two layer feed forward neural network along with the piece-wise linear (pwl) chaotic map is proposed. Based on chaotic neural networks, a Hash function is constructed, which makes use of neural networks' diffusion property and chaos' confusion property. This function encodes the plaintext of arbitrary length into the hash value of fixed length (typically, 128-bit, 256-bit or 512-bit). Theoretical analysis and experimental results show that this hash function is one-way, with high key sensitivity and plaintext sensitivity, and secure against birthday attacks or meet-in-the-middle attacks. These properties make it a suitable choice for data signature or authentication.

KEYWORDS

One-way Hash function, Neural network, Chaotic map, Plaintext Sensitivity

1. INTRODUCTION

The wide use of computer network and wireless devices of digital communication results in a greater need for the protection of transmitted information by using Cryptography. Verifying the integrity and authenticity of information is a prime necessity in computer systems and networks. A one-way hash function is an important element of cryptography. A hash function encodes a plaintext with variable length into a hash value with fixed length, and it is often used in data signature or data authentication. As is known, a secure hash function should satisfy several requirements: one-way, secure against birthday attack and secure against meet-in-the-middle attack. The one-way property makes it impractical to find a plaintext with the required hash value. The hash function should be secure against birthday attack, which makes it difficult to find two plaintexts with the same hash value. It should also be secure against meet-in-the-middle attack, which makes it difficult to find a plaintext whose hash value is same as one of the given plaintexts. Recently, it was reported that widely used Hash functions such as MD5 or

SHA-1 are no longer secure. Thus, new Hash function should be studied in order to meet practical applications.

Nonlinearity is one of the inherent properties of the neural network, therefore, in this paper artificial neural network is introduced as an alternative to the hash algorithm like MD5, traditionally used in cryptographic applications. Neural networks' confusion and diffusion properties have been used to design encryption algorithms, such as the stream cipher or the block cipher. In fact, neural networks have also a one-way property. For example, if a neuron hash multi-inputs and single output, then it is easy to obtain the output from the inputs but difficult to recover the inputs from the output. These properties make them suitable for hash function [1, 2] design. Similarly, chaos has also been used in data protection because of the features of initial-value sensitivity, random similarity and ergodicity [3,4, 5]. It is also regarded as practical to combine chaos with neural networks. For example, Rachel [6] proposed a method for secure communication by combining chaotic systems with neural networks; S.G. Lian [12] presented an image encryption algorithm based on a chaotic neural network. Hash function encodes a plaintext with variable length into a hash value with fixed length, and it is often used in data signature or data authentication. In this paper we use neural networks one-way property together with chaos' random-similarity to construct a one-way Hash function. Hash function is a kind of one-way function, which often requires high security against statistical attack, birthday attack, meet-in-the-middle attack, etc. The multiplication between the weight and the plaintext diffuses the plaintext, and the chaotic function is iterated for multi-rounds in order to make the output depend on the input nonlinearly and complicatedly.

2. THE PROPOSED HASH FUNCTION

Structure of feed forward network for the proposed one way hashing is as follows:

The structure of feed forward network is made up of layers of neurons. For the purpose of the one-way hashing function, two layers of neurons are employed, the preceding layer is called the "hidden" layer, and the other called the "output" layer. Input bits are fed into the "hidden" layer of neurons. The output of the "hidden" layer becomes the input for the "output" layer. Due to the sigmoid function, the output of each neuron is a real number between 0 and 1.Since it will be more useful to have binary values of either 0 or 1, a threshold of 0.5 is taken. Real outputs less than 0.5 will be taken as 0, while those greater or equal to 0.5 will be taken as 1.The model of the feed forward network used in this paper has an output layer consisting of 128 nodes, with a hidden layer consisting of 64 nodes. The number of input bits is set to 640.The total number of weight values including bias weight values, n required is given in the equation below [7]

$$n= (128*65) + (64*641) =49344$$

The weights and biases of the neural network are generated by using random stream generator which makes use of Mersenne twister, keeping seed value constant. The value of the weights in the feed forward network is required to be the same every time the network is initialized for use as a one-way hash function.

The feed forward network will roughly require between 1.5 to 3 megabyte of memory in order to store the weights values.

2.1. Algorithm

Hash function accepts a variable size message M as input and produces a fixed size output, referred to as a hash code H (M).

- Read in entire file into uint32 vector readmessagefromfile();
- Since the input can be of variable number of bits long, padding is done in a similar fashion performed in MD5 .A single '1' bit is appended followed by '0'bits until the length of the message is congruent to 448 bits.
- This is followed by a 64-bit representation of the original message length. This technique is known as MD strengthening and overcomes security problems arising from messages of different lengths hashing to the same output. Padding is also done because 512 bits of data are manipulated at a time during the one-way hash process.
- Initial state of buffer (consisting of A, B, C and D)

```
A = uint32 (hex2dec ('67452301'));
B = uint32 (hex2dec ('efcdab89'));
C = uint32 (hex2dec ('98badcfe'));
D = uint32 (hex2dec ('10325476'));
```

- The hashing process proceeds by taking the 512 bits of data at a time, combine the input with the 128 bit output of the previous round, and obtain the 128 bit representation. A known initialization vector (IV) is used at the start of the process.
- The total number of input bits to the feed forward network is

  512 bit data +128 bit IV +1 bit bias = 641bits

  Hidden layer (h) consisting 64 nodes, Output layer (m) consisting of 128 nodes, input bits (n) are 640
- Initialize weights and bias

```
S=RandStream ('mt19937ar','seed', 9999);
RandStream.setDefaultStream(S);
w1=rand (n,h)-0.5;
b1=rand (1,h)-0.5;
b2=rand (1,m)-0.5;
w2=rand (h,m)-0.5;
```

- The feed forward network can be represented in matrix form as

$$\begin{bmatrix} W_{1,b} & W_{1,0} & \cdots & W_{1,639} \\ W_{2,b} & W_{2,1} & \cdots & W_{2,639} \\ \vdots & & \ddots & \\ W_{64,b} & W_{64,1} & \cdots & W_{64,639} \end{bmatrix} \times \begin{bmatrix} 1 \\ i_0 \\ \vdots \\ i_{639} \end{bmatrix} = \begin{bmatrix} o_0 \\ o_1 \\ \vdots \\ o_{63} \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} W_{1,b} & W_{1,0} & \cdots & W_{1,63} \\ W_{2,b} & W_{2,1} & \cdots & W_{2,63} \\ \vdots & & \ddots & \\ W_{128,b} & W_{128,1} & \cdots & W_{128,63} \end{bmatrix} \times \begin{bmatrix} 1 \\ o_0 \\ \vdots \\ o_{639} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{127} \end{bmatrix} \quad (2)$$

Where Y_0 to Y_{127} is the result of the output layer, O_0 to O_{63} is the result from the hidden layer and $W_{x,y}$ is the weight corresponding to neuron x in the layer and input y.

Take i, o, and y to be the array of input bits, hidden layer output bits and output layer output bits respectively.

$$O(i) = \text{sigmoid}(o(i), \alpha) \qquad (3)$$
$$Y(i) = \text{sigmoid}(y(i), \alpha) \qquad (4)$$

- Apply the piecewise linear chaotic map (pwl) for T times at each layer of each neuron of neural network. The chaotic map hash some properties suitable for constructing a cipher, such as initial-value sensitivity or parameter sensitivity. If the chaotic map iterated for T (T is big >50) times, slight difference in the initial value X(k) or the parameter Q causes large differences in the iterated value X(K+T). Generally, the chaotic function is iterated for T (T>50) times to keep the output randomness.

$$X(k+1) = f(X(k),Q) = \begin{cases} X(k)/Q, & 0 \le X(k) < Q \\ (X(k)-Q)/(0.5-Q), & Q \le X(k) < 0.5 \\ (1-Q-X(k))/(0.5-Q), & 0.5 \le X(k) < 1-Q \\ (1-X(k))/Q, & 1-Q \le X(k) \le 1 \end{cases} \qquad (5)$$

Where Q is the control parameter and satisfies $0<Q<0.5$

- In order to keep low cost, the map f() is iterated for only once for hidden layer and T times for output layer .

3. Performance Analysis

3.1. Hash results of message

Plaintext chosen is: "Passage 2: American scientists have found that some birds are more intelligent than experts had believed. The scientists say birds have abilities that involve communication and different kinds of memory. In some unusual cases, their abilities seem better than those of humans."

We use the above mentioned algorithm to conduct simulation analysis under the following 4 kinds of condition:

C1: Change the number “2”in the original message into”3”.

C2: Change first letter “P” in the original message into “p”.

C3: Change the word “believed” in the original message into “thought”.

C4: Change the full stop at the end of the original message into comma.

The corresponding Hash values in Hexadecimal format are:

Original: A6CD7FF949F5B0B1CFF77DD2CA6F141D

H1:19745B51DEBA3C713074E73EE8A4F325
H2:D788A5A165919C6D5E9CC7396CE956EF
H3:E2FA9B888122A2656D74CEBD2067B473
H4:1A52A768E978A1F2CA018EEAAC35089C

The simulation result indicates that the one-way property is so perfect that small change in the message will cause huge changes in the final hash value. The corresponding graphical display of binary sequences is shown in Fig.1

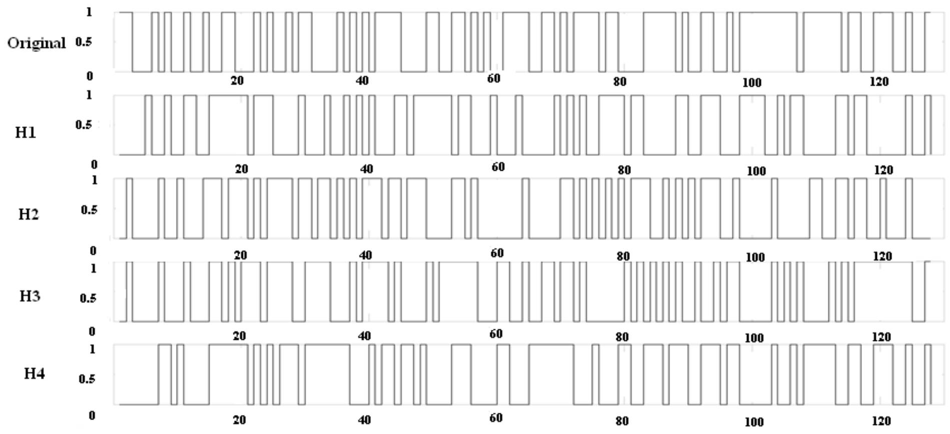

Figure 1: Hash value under different conditions

3.2. Plaintext Sensitivity

For a hash function, it is required that different plaintext produce different hash values. This property depends on the hash function's plaintext sensitivity. Slight difference in the plaintext will cause great changes in the Hash value, which makes the Hash function of high plaintext – sensitivity. This property is important to keep it secure against statistical attacks.

Experiments are done to test the hash function's plaintext sensitivity. As an example, plaintext chosen is: "Passage 2: American scientists have found that some birds are more intelligent than experts had believed. The scientists say birds have abilities that involve communication and different kinds of memory. In some unusual cases, their abilities seem better than those of humans." The according Hash value is H_o = A6CD7FF949F5B0B1CFF77DD2CA6F141D.Then only first bit of the plaintext is changed ,the according hash value is H_1= D788A5A165919C6D5E9CC7396CE956EF.If the second one is changed, the according Hash value is H_2= E5772AAB911F6EEAB39ED64E9369094B .And if the i-th one is changed ,then the according Hash value is H_i, Then the Hash value Change rate is computed by

$$r(i) = \frac{Dif(H_0, H_i)}{128} \times 100\%$$

Where Dif (H_o, H,) means the number of the different bits between H_o and H_i. For the proposed plaintext, the change rate is shown in Fig. 2 for different values of seed and number of inputs to hidden layer. As can be seen, the change rate is about 50% (64 bits), which shows that the Hash function is of high plaintext sensitivity.

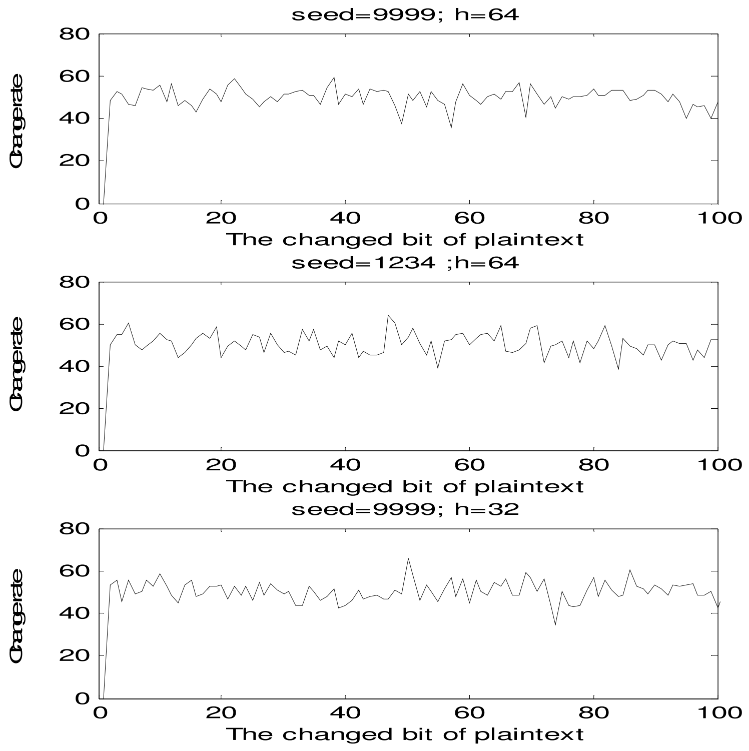

Figure 2: Plaintext sensitivity of the Hash function

3.3. Comparison of MD5 and Neural Hash

One of the properties such as plaintext sensitivity of neural network implemented hash function is compared against traditional hash function MD5 for the above plaintext. Consider above mentioned plaintext. The plaintext sensitivity graph for both MD5 and Neural Hash is plotted as shown in Fig 3:

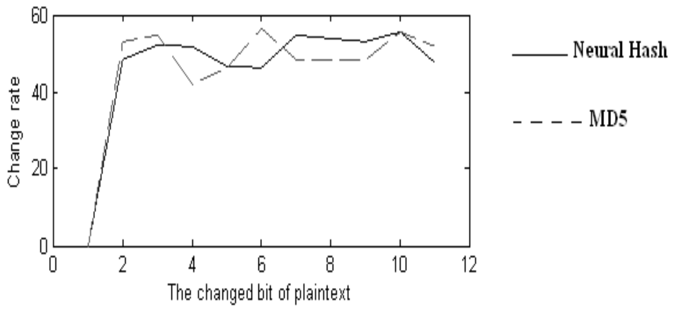

Figure 3: Plaintext sensitivity comparison between MD5 and Neural Hash

From above simulation results, both MD5 and Feed forward neural network with chaotic map implemented Hash function gives on average same plaintext sensitivity.

4. ADVANTAGE OF PROPOSED APPLICATION

Applying a feed forward network as a hash algorithm presents several useful attributes. Firstly, simply changing the initial weight values of the feed forward network employed results in a totally different hash output. Many hash algorithms can be implemented for different purposes by setting unique initial weight values, for each purpose. Secondly, the feed forward network structure can be modified, allowing for a hashed key of more than 128 bits, simply by adding more neurons into each layer. This hash function adopts the neural network's one-way property, diffusion property and confusion property suitably.

5. CONCLUSION

In this paper, an algorithm for one-way Hash function construction based on two layer feed forward neural network along with piece-wise linear chaotic map is presented. This hash function adopts the neural network's one-way property, diffusion property, and chaotic map confusion property. Applying a feed forward network as a hash algorithm presents several useful attributes. Firstly, simply changing the initial weight values of the feed forward network employed results in a totally different hash output. Many hash algorithms can be implemented for different purposes by setting unique initial weight values, for each purpose. Secondly, the feed forward network structure can be modified, allowing for a hashed key of more than 128 bits, simply by adding more neurons into each layer. The analysis and experiments shows that the change rate of plaintext sensitivity remains constant on an average 50% for different values of number of neurons in hidden layer and seed value which is used for generation of weights and biases at each layer of neural network.

ACKNOWLEDGEMENTS

We would like to thank to computer centre, Gogte Institute of Technology, for providing laboratory facilities. We thank our esteemed Management for their financial support.

REFERENCES

[1] Shiguo Lian, Zhongxuan Liu, Zhen Ren, Haila Wang, "Hash Function Based on Chaotic Neural Networks" IEEE, 2006.

[2] Shiguo Lian,Jinsheng Sun,Zhiquan Wang, " One-way Hash Function Based on Neural Network" Journal of Information Assurance and Security,2006

[3] Yi Du, Detang Lu, Daolun Li,"An Effective Hash-based Method for Generating Synthetic Well Log" 2008 IEEE.

[4] Qun-ting Yang,Tie-gang Gao,Li Fan,Qiao-lun Gu, " Analysis of One-way Alterable Length Hash Function Based on Cell Neural Network" Fifth Intenational Conference on Information Assurance and Security,2009

[5] Qinghua Zhang,Han Zhang and Zhaohui Li,"One-way Hash Function Construction Based on Conservative Chaotic Systems" Journal of Information Assurance and Security, 5, pp.171-178, 2010

[6] M.K. Rachel, K. Einat, K. Ido, Wolfgang, "Public Channel 012. Cryptography by Synchronization of Neural Networks and ChaoticMaps," Physical Review Letters, Vol. 91, No. 11, Sep 12, 2003: 118701/1-118701/4.

[7] L.P. Yee, D. L.C. Silva, "Application of Multilayer Perceptron Network as a One-way Hash Function" International Joint Conference on Neural Networks, Vol. 2,2002.

[8] Li, C., S. Li, D. Zhang and G. Chen,"Cryptanalysis of a chaotic neural network based multimedia encryption scheme". Advances in Multimedia Information Processing PCM ,2004 .

[9] Khalil Shihab, "A Backpropagation Neural Network for Computer Network Security" Journal of Computer Science 2 (9): 710-715, 2006.

[10] C.-K. Chan and L.M. Cheng. The convergence properties of a clipped Hopfield network and its application in the design of key stream generator, IEEE Transactions on Neural Networks, Vol. 12, No. 2,pp. 340-348, March 2001.

[11] D.A. Karras and V. Zorkadis. On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators.Neural Networks, Vol. 16, No. 5-6, June - July, 2003: 899-905

[12] S.G. Lian, G.R. Chen, A. Cheung, Z.Q. Wang. A Chaotic-Neural-Network-Based Encryption Algorithm for JPEG2000 Encoded Images. In: Processing of 2004 IEEE Symposium on Neural Networks (ISNN2004), Dalian, China, Springer LNCS, 3174 (2004) 627-632.

[13] Liew Pol Yee and L.C. De Silva. Application of multilayer perception networks in symmetric block ciphers. Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, USA, Vol. 2, 12-17 May 2002: 1455 – 1458.

[14] N. Masuda, K. Aihara, "Cryptosystems With Discretized Chaotic Maps," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Vol. 49, No. 1, Jan 2002.

[15] H.P. Lu, S.H. Wang, G. Hu, "Pseudo-random number generator based on coupled map lattices," International Journal of Modern Physics B 2004; 18(17-19): 2409-2414.

[16] William Stallings. Cryptography and Network Security, Third Edition.

[17] NIST.Secure Hash Standard. Federal Information Processing Standard.FIPS-180-1.April 1995

[18] S Sivanandam .S Sumathi Introduction to Neural Networks Using MATLAB 6. 0 (Computer Science Series)

[19] Rudra Pratap Getting Started With MATLAB 7 - A Quick Introduction For Scientists And Engineers

[20] www.mathwork.com

Authors

Prof.V. R. Kulkarni is presently working as HOD in the department of information science and Engineering at Gogte Institute of Technology, VTU, University, Belgaum, Karnataka, India and Obtained graduate degree from Walchand College of Engineering, Sangli in the year 1984 and Master degree from Birla Institute of Technology, Pilani in the year 1994.Pursuing research in the area application of Neural Network in Cryptography under the guidance Prof.Dr.S.S.Apte in Solapur University.

Shaheen Mujawar is currently pursuing M.Tech. in Computer Science and Engineering at Gogte Institute of Technology, VTU University, Belgaum,Karnataka,India and completed B.E in Electronics and Communication Engineering at SDMCET, Dharwad, VTU University, Karnataka, India in the year 2006.