Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.1380 – 1386

RESEARCH ARTICLE

Implementation of Hash Function Based On Neural Cryptography

Vineeta Soni
Computer Science and Engg
Mody University of Science and Technology
Laxmangarh (Rajasthan), India
vineeta.soni89@yahoo.com

Mrs. Sarvesh Tanwar
Assistant Professor,
Computer Science and Engineering
Mody University of science and Technology
Laxmangarh(Rajasthan),India
s.tanwar1521@gmail.com

Prof. Prema K.V.
Head of Department,
Computer Science and Engineering,
Mody University of Science and Technology
Laxmangarh (Rajasthan),India
Drprema.mits@gmail.com

Abstract— In this paper a new hash function is constructed based on multilayer feed forward network with piecewise linear chaotic map. Chaos has been used in data protection because of the features of initial value sensitivity, random similarity and ergodicity. We have used three neuronal layers to prove confusion, diffusion and compression respectively. This hash function takes input of arbitrary length and generate a fixed length hash value.(128 bit, 256 bit or 512 bit). By performance analysis and results we have shown that generated hash function is one way, collision resistant and secure against birthday attacks and man in middle attacks.

Keywords—Neural Network, Cryptography, Tree parity machines, Piecewise linear chaotic map, hash function, plain text sensitivity, data signature

I. Introduction

The wide use of computer networks and wireless devices in communication results a greater need for protection of transmitted information by using cryptography. Cryptography is the ability to send information between participants in a way that prevents others from reading it. So verification of integrity and authentication is a prime necessity in computer systems and networks [11]. In our work we are combining neural networks and cryptography to achieve a robust system against Man -in –the-middle attack and birthday attacks.

Neural Network Artificial neural networks are parallel adaptive networks of consisting of simple nonlinear computing elements called neurons which are intended to abstract and model some of the functionalities of human nervous system in an attempt to partially capture some of its computational strength. They can be used to model complex relationships between inputs and outputs or to find patterns in data. A trained neural network can be a thought of as an “expert” in the category of information it has been given to analyze [16].

So by combining both technologies we can generate better results using less complex functions and providing better security.

A one way hash function is also an important element of cryptography which encodes a variable length message into a hash value with fixed length and will be used in authentication and signature. Secure hash function must satisfy following requirement: one way, secure against birthday attacks and meet-in the middle attack [11].

So nonlinearity property of neural network can be used as alternative to hash algorithms. Neural network has one way property also. For example a neural network has many input and single output. It is easy to generate output from inputs but hard to recorded inputs from output. Recently it was reported that MD5 and SHA-1 are no longer secure [1] [3]. So new hash functions should be required to meet practical applications. The hash function generated by neural network satisfies the security requirements as well as can be effectively implemented with less complexity. Hash Function encodes a plaintext with variable length into a fixed length hash value which is often used in data authentication and signature. In this paper we use neural networks one way property with chaos maps random similarity property.

II. Neural Cryptography

Two identical systems, starting from different initial conditions can be synchronized by a common external signal which is coupled to the two systems. Both of the networks will be trained on their mutual output and can synchronize to a time independent state of identical synaptic weights. This rarity is applied to cryptography. In this case two partners in communication does not have a common secret key but they use their identical weights as a secret key for communication. This common weight can be used to form a key needed for encryption and decryption and other applications.

For this we have used a different type of neural network called Tree parity machine [13] [14] [15].

III. Tree Parity Machine

For this work we have used a simple neural network which is called tree parity machine (TPM). These are special type of neural network.

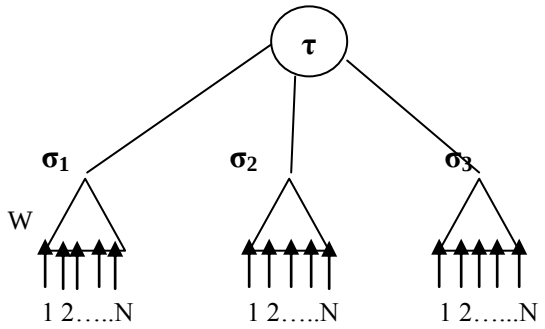

Figure 1: Tree Parity machine

This can be described by three parameters: K hidden layers, each of which can be seen as a single layer perception with N input neurons in each hidden layer, L is the maximum value for weight{-L…+L}. All input values are binary i.e.

$$x_{i,j} \in \{-1,+1\} \quad (1)$$

And the weights, which define the mapping from input to output, are discrete Numbers between −L and +L

$$W_{i,j} \in \{-L,-L+1,\ldots,+L\} \quad (2)$$

Output value of each hidden neuron is calculated as a sum of all multiplications of input neurons and these weights:

$$\sigma_i = \mathbf{sgn} \sum_{J=1}^{N} W_{ij} X_i \quad (3)$$

Two partners have the same neural machines and their output value is calculated by:

$$\tau = \prod_{i=1}^{k} \sigma i \quad (4)$$

Both networks receive common inputs vector X and select random initial weight vectors W. Both the networks trained by their output bits $\tau(A)=\tau(B)$.
The following learning rules can be applied:

1. If $\tau(A)\neq\tau(B)$ nothing is changed.
2. If $\tau(A)=\tau(B)=\tau$ only the hidden unit is changed for $\sigma k(A/B)=\tau(A/B)$
3. Three different rules can be considered for learning.

a)Hebbian learning rule :

$$w_i^+ = w_i + \sigma_i x_i \theta(\sigma_i \tau)\, \theta(\tau^A \tau^B) \quad (5)$$

b) Anti-Hebbian learning rule:

$$w_i^+ = w_i - \sigma_i x_i \theta(\sigma_i \tau)\, \theta(\tau^A \tau^B) \quad (6)$$

c) Random-walk learning rule:

$$w_i^+ = w_i + x_i \theta(\sigma_i \tau)\, \theta(\tau^A \tau^B) \quad (7)$$

Here, Theta is a special function. Theta (a, b) = 0 if a<>b; else Theta=1. The g(...) function keeps the weight in the range {- L..+L} [2].

IV. Secret Key Generation

1. First of all determine the neural network parameters i.e. k, the number of hidden layer units n, the input layer units for each hidden layer unit l, the range of synaptic weight values is done by the two machines A and B.
2. The network weights to be initialized randomly.
3. Repeat 4 to 7 until synchronization occurs.
4. The inputs of the hidden units are calculated.
5. The output bit is generated and exchanged between the two machines A and B.
6. If the output vectors of both the machines are same i.e. $\tau_A = \tau_B$ then the corresponding weights are modified using the Hebbian learning rule, Anti-Hebbian learning rule and Random-walk learning rule.
7. After complete synchronization, the synaptic weights are same for both the networks. And these weights are used as secret key [14].

In this technique we can increase the key size without increasing the weight range as a result we get maximum security with less synchronization time. This key can be utilized to encrypt a sensitive message transmitted over an insecure channel using any encryption algorithm..

We have used this key in sub key generations used as weight, bias and control parameter in Neural Networks to generate hash. While generating hash code value H(M) for a message M ,the hash code is different for same message because weights are different from the same values obtained at different synchronization session.

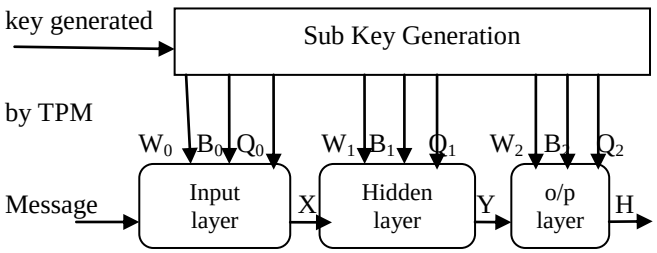

Figure 2: key generation from the key generated by TPM

The key generated from TPM is of 128 bit which is further divided in four parts 32 bit each. These four key will be quantized and used to generate all the nine sub keys which are composed of 151 data pixels for the uses of neural networks in hash code generation. The chaotic map is applied on subparts of keys to generate subkeys.The use of neural networks and this sub key generation is discussed in next section.

V. Proposed Hash Algorithm

Three layer neural network is used for generating the hash function which is composed for three layers- input layer hidden layer and output layer. These layers are used to realize data confusion, diffusion and compression respectively.

Let the layers inputs and output be $M=[M_1M_2\ldots M_{31}]$, $X=[X_0\ X_1\ldots..X_8]$, and $Y=[Y_0Y_1\ldots Y_8]$ and $H=[H_0H_1H_2H_3]$ so functioning of neural network is defined as-

$$H= f_2(W_2 Y + B_2) = f_2(W_2 f_1(W_2 X + B_1)) = f_2(W_2 f_1(W_1 f_0(W_0 M+B_0)+B_1+B_2))) \quad (7)$$

Where f_i, W_i and B_i (i=0,1,2) are the piecewise linear chaotic map as transfer function, weight and bias of the ith neuron respectively.

And f is defined as: Z(k+1)=f(Z(k),Q) =

$$f(Z(k),Q)=\begin{cases}\dfrac{Z(k)}{Q}, 0\le Z(k)<0.5\\ \dfrac{Z(k)-Q}{0.5-Q}, Q\le Z(k)<0.5\\ \dfrac{1-Q-Z(k)}{0.5-Q}, 0.5\le Z(k)<1-Q\\ \dfrac{1-Z(k)}{Q}, 1-Q\le Z(k)\le 1\\ Z(k)-1, \quad Z(k)\ge 1\end{cases}$$

Where Q is control parameter and its value in [0, .5].Here map is piecewise linear and it is in chaotic state when 0<Q<0.5.This chaotic map has parameter sensitivity means a small change in initial values Z(k),Q causes drastic change in iterated values Z(k+T) which is suitable for constructing a cipher.

Generally chaotic map is iterated for (T >=50) to keep the randomness of output for each layer. Chaotic map f() is applied on each layer and used as an activation function. Input and output layers are iterated T times in order to improve the randomness of output of hidden layer and Y and H. By improving the randomness the strength of cryptosystem is strengthened.

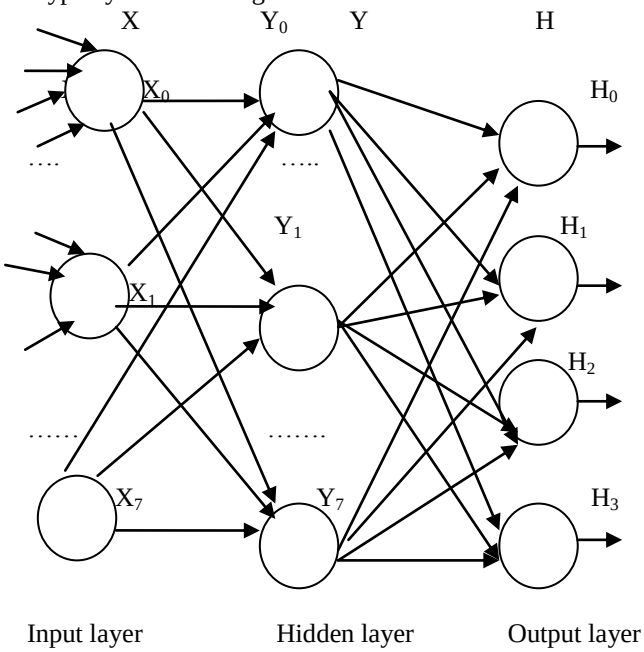

Figure 3: Three layer Neural network

First of all the message M is divided in n number of blocks each of 32 bits. Then XORED them together before applying on neural network.

$M=M_1$ XOR M_2 XOR……M_N (8)

Here Input layer is defined as:

$X=f^T(\sum_{i=0}^{32} W_{0,i}\, M_i + B_0\ , Q_0)$ (9)

Weight vector is $W_0=[w_{0,0}\ w_{0,1}\ ....\ W_{0,31}]$for input layer because input M of 32 bit is divided into 8 data pixels each consists of 4 input neurons $\{M_0M_1M_2M_3]$ and $B_{0=}8\times1$ and chaotic map is applied on input layer is T iteration times to complicate the relationship between input and output layers. X This strengthens the property of diffusion.

Hidden layer is defined as:

$Y=f1\ (W_1X+B_1,Q_1)$ (10)

Where W_1 is of 8×8, $B_1=8\times1$ and $Y=8\times1$.The aim of hidden layer is to diffuse the change of X in Y. In order to keep cost low the map f() is iterated once for hidden layers.

And final hash value is generated as

$H=f_2(W_2Y+B_2\ ,Q_2)=f^T(W_2Y+B_2\ ,Q_2)$ (11)

It means the transfer function is iterated T(where T >=50) times in the output layer. So the statistical relationship between generated hash value and K generated by TPM is very difficult to compute. So the new Hash function proves the confusion as well as diffusion properties which are required to construct any cryptosystem.

VI. Security Analysis

A) Irreversible Property In the proposed hash function is easy to computed from M and K according to the equation, but infeasible to compute M and K if only H is known. So it is impossible for attacker to compute M and K when H(M) is known. This proves the one way property of the hash function.

$$\boldsymbol{X_j = f^T\left(\sum_{i=4j}^{4j+3} W_{0,i}M_i + B_{0,j} + Q_0\right)} \quad (12)$$

Where j=0 to 7

To compute M_i from the above equation here weight and bias are not known coz it is generated by the key generated from synchronization of Tree parity machine. Two methods can be tried: selected plain text attack and brute force attack. For the brute force attack 8 data pixels need 2^8*32=2^256 which is impractical for today's computers.

Selected plaintext attack can be practical if bias values and weight values are known. But here we compute weight values and bias values randomly after some permutation from key generated by tree parity machines and also according to chaotic map It needs 4^T (If T>=50 then 2^{100}) times to compute weight values and bias from X(input of hidden layer) which seems impractical if T is big enough. The piecewise linear chaotic map always keep the hidden layer and output layer one way.

B) Resistance to Man in the middle attack

Man in the middle attack is to find contradiction through looking for a suitable substitution of plain text block. But here we have divided our message in n number of message blocks and then XORed them together and after some permutation the message is applied to the neural network for hash code Generation. If any attacker tries to substitute plaintext before applying neural network needs a key, but here the used key is generated by Tree parity machine for each message at a time and the Key is used for the generation of weight ,bias, and control parameters for all the three layers.

VII. Implementation and Results

All work has been done in MATLAB and some data sets are obtained for synchronization time by varying number of input units.

S.No.	Different issues	Without NN	With NN
1.	Randomness	no	More
2.	Security	less	More
3.	Synchronization time	Not required	required

Table 1: Comparison between cryptography and Neural Cryptography

The number of iterations required for synchronization by varying number of input units. If the value of n increases, the synchronization time and number of iteration also increases.

S.No.	No. of Input units	No. of hidden units	No. of iterations	Weight range	Key
1.	5	3	764	7	@BHCCIF CFEFA GG@
2.	7	3	1156	7	IIBC@IFA FB CAHIG
3.	10	3	1201	7	KHIJKML HK OFGOJG

Table 2: Results of implementation of TPM when input units varying.

Hash function is generated by given a plain text block composed of 32 data pixels encoded into hash value H composed of 4 data pixels. Sub graph (a) shows change in 32 bits of plaintext with respect to binary sequence 0 and 1.subgraph (b) shows the Hash code of 128 bits according to given plaintext of 32 bits in binary sequence 0 and 1.

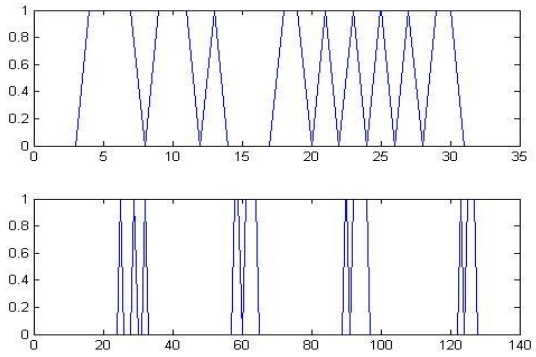

Figure 4: shows the variation of between plaintext and Hash Code

VIII. Conclusion and future work

The keyed hash function is proposed and analyzed in this paper. Key is generated by synchronization and mutual learning of tree parity machines. Then this secret key is used in generation of sub keys. These generated subkeys are used in three layer neural network to constructing a hash code based on neural network. So we are offering double security to generated hash code. In compare to cryptographic hash functions this neural hash function is easy to implement and provide best results due to synchronization and self adaptive behavior of neural networks. This hash function adopts the neural network's one way property, confusion and diffusion property suitably. Use of piecewise linear chaotic map adds more data protection Final hash of 128 bit is generated by XORing of intermediate hash values.

Our future work is to embedded harder mathematics in Neural cryptography to keep the system more secure and we can design an efficient Neural network based digital signature which uses the neural hash algorithm instead of using cryptographic hash functions.

References

1. Akhavan, Amir, Azman Samsudin, and Afshin Akhshani. "A novel parallel hash function based on 3D chaotic map.“ Springer *EURASIP Journal on Advances in Signal Processing* 2013.1 (2013): 1-12.
2. Allam, Ahmed M., Hazem M. Abbas, and M. Watheq El-Kharashi. "Authenticated key exchange protocol using neural cryptography with secret boundaries." *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013.
3. Dr. Ajit Singh, Aarti nandal **,**“Neural Cryptography for Secret Key Exchange and Encryption with AES”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013 ISSN: 2277 128X
4. Urbanovich, Pavel, Marcin Plonkowski, and Konstantsin Churikov. "The appearance of conflict when using the chaos function for calculating the hash code." *network* 3 (2012)
5. Li, Yantao, et al. "Parallel Hash function construction based on chaotic maps with changeable parameters." *Neural Computing and Applications* 20.8 (2011): 1305-1312 2011 – Springer
6. Arvandi, M., S. Wu, and A. Sadeghian. "On the use of recurrent neural networks to design symmetric ciphers." *Computational Intelligence Magazine, IEEE* 3.2 (2008): 42-
7. Lian, Shiguo, Jinsheng Sun, and Zhiquan Wang. "One-way hash function based on neural network." *arXiv preprint arXiv:0707.4032* (2009).
8. Hen, Tieming, and Rongrong Jiang. "Designing Security Protocols Using Novel Neural Network Model." *Natural Computation, 2007. ICNC 2007. Third International Conference on*. Vol. 1. IEEE, 2007.
9. Liu, Niansheng, and Donghui Guo. "Security analysis of public-key encryption scheme based on neural networks and its implementing”*Computational Intelligence and Security*. Springer Berlin Heidelberg, 2007. 443-450..
10. Arvandi, Maryam, and Alireza Sadeghian. "Chosen Plaintext Attack against Neural Network-Based Symmetric Cipher." *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. IEEE, 2007
11. William, S., & Stallings, W. (2006). “*Cryptography and Network Security, 4/E”*. Pearson Education India.
12. Godhavari, T., N. R. Alamelu, and R. Soundararajan. "Cryptography using neural network." *INDICON, 2005 Annual IEEE*. IEEE, 2005.
13. Mislovaty, R., et al. "Security of neural cryptography." *Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on*. IEEE, 2004.
14. Wolfgang, and Ido Kanter. "Interacting neural networks and cryptography, Springer Berlin Heidelberg, 2002. 383-391.
15. Klimov, Alexander, Anton Mityagin, and Adi Shamir. "Analysis of neural cryptography." *Advances in Cryptology—ASIACRYPT 2002*. Springer Berlin Heidelberg, 2002. 288-298.
16. Haykin, Simon. “*Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994”

