STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720
Volume 7, Issue 2
www.stmjournals.com

Using Artificial Neural Networking in Cryptography

Himani Dua, Abhay Shukla*
Student, Department of Information Technology, HMR Institute of Technology and Management, Delhi, India

Abstract

Cryptography system helps in transformation of information among authenticated users where there is no involvement of any third party using that information without that information with unauthorized access. This project implements encryption and decryption using neural network cryptography using AES key. The network construction will be generated solely on the parameters used in the training algorithm and the number of hidden neurons. In neural cryptography, both the communicating networks would receive an identical input vector(s), generate an output bit and are trained based on the output bit, the networks are to be synchronize to a state with identical time-dependent weights. This concept of synchronization by mutual learning can be applied to a secret key exchange protocol over a public channel which would be further used in cryptography. Earlier generation of secret key over a public channel used for encrypting and decrypting the given message using DES algorithm in neural cryptography was widely adopted. But, In this project we would using AES algorithm for key generation in neural cryptography. AES algorithm replaces the small key size generated by DES and also the Triple DES as it founds to be six times faster.

Keywords: *Neural networks, cryptography, key generation, Tree Parity machine, Security Attacks*

Author for Correspondence **E-mail:** shuklaabhay256@gmail.com

INTRODUCTION

With advancements in performance and requirements in technology, ciphers with better adaptation are proving to replace the weaker, slower and the aging algorithms for the current applications. Algorithm for encryption can be symmetric or asymmetric. Symmetric encryption is fast. Due to its fast performance it is commonly used in e-commerce for transactions. Symmetric key algorithm(s) can be further classified as: stream ciphers and block ciphers. Block ciphers are mostly based on the idea by Shannon, that sequential application of confusion and diffusion, will obscure redundancies in the plaintext, where confusion involves substitutions to conceal redundancies, and statistical patterns in the plaintext, and diffusion involves transformations (or permutations), to dissipate the redundancy of the plaintext, by spreading it out over the cipher text. DES and Rijndael are examples of algorithms based on this idea. AES (Advanced Encryption Standard) is an iterated block cipher which was selected by NIST as an international standard, and replaced DES. It is now the most widely deployed block cipher in both software and hardware applications. AES which is an iterative rather than a Feistel cipher which is based on 'substitution–permutation network' which refers to a comprised series of linked operations, some of which involves replacement of inputs by substitutions, while others involve shuffling of bits which is a process called permutations. AES does all its computations on bytes rather be dependent on bits. Hence, AES treats a 128 bit(s) of a plaintext block as 16 bytes. Now, these 16 bytes are arranged in four columns and four rows for processing which forms a matrix − Unlike DES, where number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds in AES uses a different 128-bit round key, which is calculated from the original AES key. The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details

Symmetric key Cryptography

A Symmetric-key cryptography is a method of encryption in which both the sender as well as the receiver shares the same key (or, keys which are different, but can be related in an easily computable way). As seen above, Alice and Bob share the same secret key. Bob can use the key to encrypt any message before passing it over a public channel. Here unauthorized person can only see the result of the encryption, but do not know the method to decrypt it. Once Alice receives the message it can be decrypted using Alice's copy of the key, transforming it back into plaintext.

IMPLEMENTATION OF ALGORITHM

Neural key Exchange Protocol

Diffie-Hellman is a protocol which is a most used protocol for key exchange between two clients/parties, lets say A and B. While Neural key exchange should be used for key exchange, which is based on the synchronization of two tree parity machines, which will be a secure replacement for this method. Synchronizing these two machines is similar to synchronizing two chaotic oscillators in chaos communications.

Tree Parity Machine

The tree parity machine is a special type of multi-layer feed-forward neural network. It consists of one output neuron, K hidden neurons and K*N input neurons. Inputs to the network are binary which *is {-1,1}*
The weights between input and hidden neurons take the values:*{-T1,….,0,…,Tn}*

Output value of each hidden neuron is calculated as a sum of all multiplications of input neurons and these weights which is X*i* = *sgn(∑j=1^{n}wijxij) ,where* Signum is a simple function, which returns -1,0 or 1:
Sgn(w) = {-1 if x<0,
0 if x=0,
1 if x>0.}

If the scalar product is 0, the output of the hidden neuron is mapped to -1 in order to ensure a binary output value. The output of neural network is then computed as the multiplication of all values produced by hidden elements. Output of the tree parity machine is binary.

Protocol

Each party (A and B) uses its own tree parity machine. Synchronization of the tree parity machines is achieved in these steps:

1. Initialize random weight values
2. Execute these steps until the full synchronization is achieved

I. Generate random input vector X
II. Compute the values of the hidden neuron

3. Compute the value of the output neuron
4. Compare the values of both tree parity machines

i. Outputs are others: go to 2.I
ii. Outputs are same: one of the suitable

Learning rules is applied to the weights after the full synchronization is achieved (the weights wij of both tree parity machines are same), A and B can use their weights as keys. This method is known as a bidirectional learning. One of the following learning rules can be used for the synchronization:

Hebbian Learning Rule

It is the rule which is used to alter weights between modal neurons.

Anti-Hebbian Learning Rule

This rule is based on reverse of Hebbians's rule, it is basically used in reduction of synaptic connectivity between neurons.

Random walk

It shows the walk series for further predictions and consistently predict the next value in the series.

Attacks and Security of this Protocol

In every attack it is considered, that the attacker E can eavesdrop messages between the parties A and B, but does not have an opportunity to change them.

Brute Force

To provide a brute force attack, an attacker has to test all possible keys (all possible values of

weights wij). By K hidden neurons, K*N input neurons and boundary of weights L, this gives(2L+1) KN possibilities. For example, the configuration K = 3, L = 3 and N = 100 gives us 3*10253 key possibilities, making the attack impossible with today's computer power.

Learning with own Tree Parity Machine

One of the basic attacks can be provided by an attacker, who owns the same tree parity machine as the parties A and B. He wants to synchronize his tree parity machine with these two parties. In each step there are three situations possible:

1. Output(A) ≠ Output(B): None of the parties updates its weights.
2. Output(A) = Output(B) = Output(E): All the three parties update weights in their tree parity machines.
3. Output(A) = Output(B) ≠ Output(E): Parties A and B update their tree parity machines, but the attacker cannot do that. Because of this situation his learning is slower than the synchronization of parties A and B. It has been proven, that the synchronization of two parties is faster than learning of an attacker. It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability.

Other Attacks

For conventional cryptographic systems, we can improve the security of the protocol by increasing of the key length. In the case of neural cryptography, we improve it by increasing of the synaptic depth L of the neural networks. Changing this parameter increases the cost of a successful attack exponentially, while the effort for the users grows polynomially. Therefore, breaking the security of neural key exchange belongs to the complexity class NP. Alexander Klimov, Anton Mityaguine, and Adi Shamir say that the original neural synchronization scheme can be broken by at least three different attacks—geometric, probabilistic analysis, and using genetic algorithms. Even though this particular implementation is insecure, the ideas behind chaotic synchronization could potentially lead to a secure implementation.

Permutation Parity Machine

The permutation parity machine is a binary variant of the tree parity machine. It consists of one input layer, one hidden layer and one output layer. The number of neurons in the output layer depends on the number of hidden units K. Each hidden neuron has N binary input neurons. The weights between input and hidden neurons are also binary. Output value of each hidden neuron is calculated as a sum of all exclusive disjunctions exclusive or) of input neurons and these weights. $N(x)$ is a threshold function, which returns 0 or 1. The output of neural network with two or more hidden neurons can be computed as the exclusive or of the values produced by hidden elements. Other configurations of the output layer for K>2 are also possible.

Security against Quantum Computers

A quantum computer is a device that uses quantum mechanisms for computation. In this device the data are stored as qubits (quantum binary digits). That gives a quantum computer in comparison with a conventional computer the opportunity to solve complicated problems in a short time, e.g. discrete logarithm problem or factorization. Algorithms that are not based on any of these number theory problems, are being searched because of this property.

Neural key exchange protocol is not based on any number theory. It is based on the difference between unidirectional and bidirectional synchronization of neural networks. Therefore, something like the neural key exchange protocol could give rise to potentially faster key exchange schemes.

METHODOLOGY

Cryptography using neural network would be working on the phenomenon of neural networks which would be generated by the help of inputs taken from the server and the client, these inputs would be decided by the server and client on a public channel. After taking these input values, synchronization of these values takes place which would check if the values inputted on both sides are same or not because these values would be

required in generation of a private key value. The inputs strength decides the key size that would be generated randomly but the values will decide the security strength of the key as in the key generated would be used in encryption as well as in decryption (Figure 1).The following phases describes the working of cryptography using neural network. It is listed below:

Input Phase

Input of the parameters required for the synchronization of two networks, input values taken is the number of hidden layers units, the input layer units for each hidden layer unit, the range of synaptic weight values is done by the two machine A and B. The weight of the networks are initialized randomly using the input value of weight range.

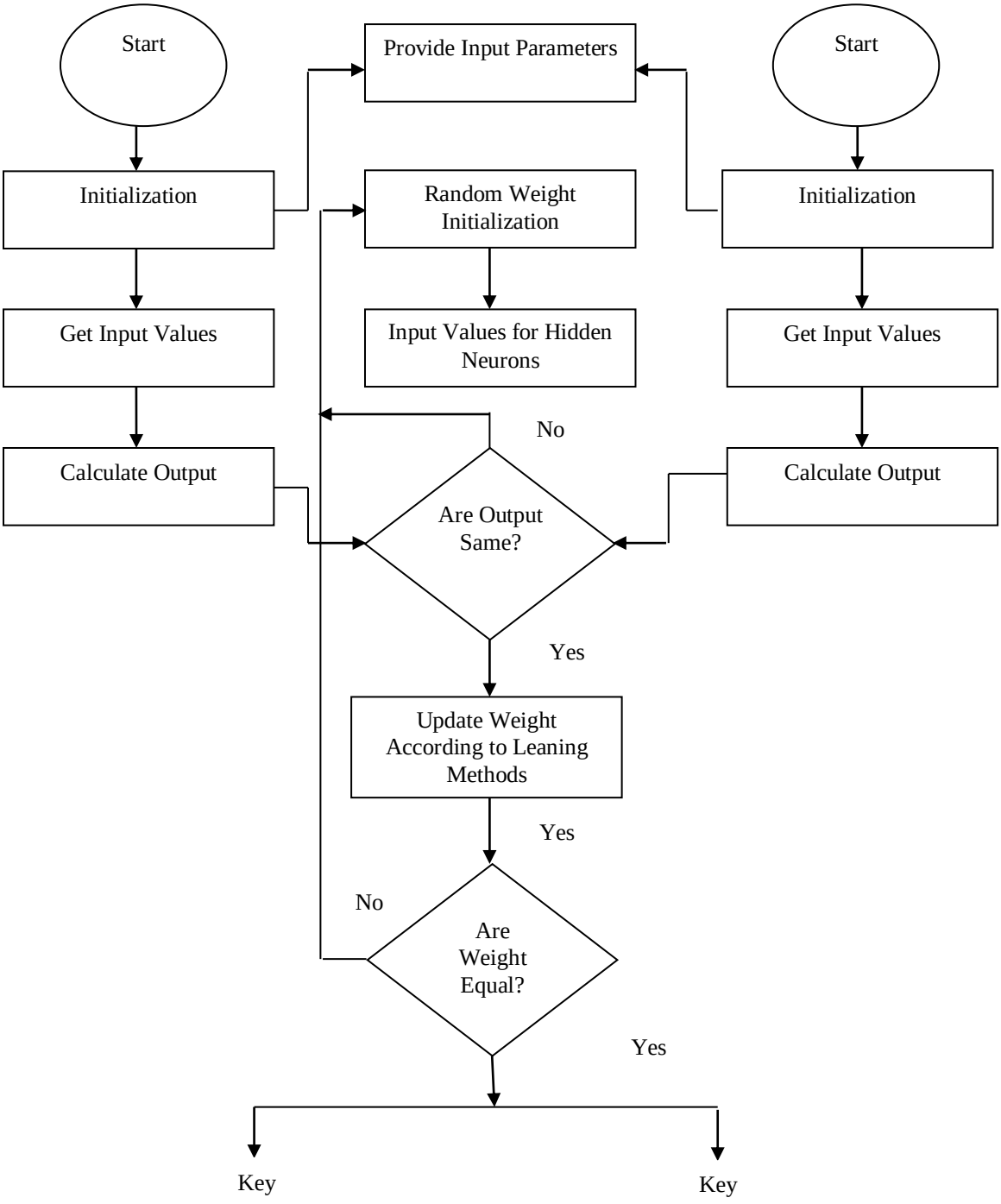

Fig. 1: *Methodology of Cryptography using Neural Network.*

These values will decide the size and time for the creation of the key that would be used in the encryption and decryption of the data and also the key would be generated randomly for every process which would make this whole process of randomly using the input value of weight range cryptography more secure.

Synchronization Phase

The initialized weight and inputs are used to calculate the output of each hidden neuron. The final output of the network is calculated using the outputs of each hidden neuron. If the final output value of the two networks are same, then synchronization takes place. If the final output is different, then it will return back to random initialization of weight. This process will take place until the final outputs become same for both the networks and the synchronization takes place which would make sure that both the machines A and B have the same private key on the public channel.

Key Generation for Encryption & Decryption

After completion of synchronization phase, the synaptic weights are same for both the networks now. So these weights would be used as secret private key which would be known to both the machines A and B which would help in sharing information on a public channel with only authorized and known machine as the secret key generated and used for encryption & decryption of the file to be shared between two networks is only known to the defined and connected machines sharing same secret keyover a public network.

CONCLUSION

In conclusion of this project we have implemented cryptography using neural networks which would be using key using AES algorithm. It will help in sharing information over a public channel using a secret private key that would be generated randomly and would only be known to the machines that are authorized machines which makes the whole network of transferring sharing of data secured.

FUTURE WORK

This research paper represents one of the application that is applied but as there is still a huge number of applications to study which can be further be taken in contrast of study and analysis. As this paper includes only rough sketch of what can be done using Neural Network by generation of key using AES key to encrypt and decrypt the data using symmetric key or private key. One can study and work upon more beneficial ideas to explore and increase the usability & feasibility so that each and every citizen of nation can use it widely like as e-commerce websites for shopping, delivering, transferring data.

The future work that may be done in this regard includes:

1. Minimization of the error function by improved methods.
2. Implementation of better training algorithms and network architectures.
3. Increasing the efficiency of training for the generalized cryptosystems.

REFERENCES

1. Kinzel W., Kanter I. (2002) Interacting Neural Networks and Cryptography. In: Kramer B. (eds) Advances in Solid State Physics. Advances in Solid State Physics, vol 42. Springer, Berlin, Heidelberg.
2. Godhavari,T.,N.R.Alamelu, and R. Soundararajan. "Cryptography using neural network." 2005 Annual IEEE India Conference - Indicon, 11-13 Dec. 2005, Chennai, India.
3. ethesis.trkl.ac.in/7431/1/2015_Cryptography_Raj.pdf
4. VikasGujral, Satish Kumar Pradhan (2009). Cryptography Using Artificial Neural Networks. B.tech. project report submitted in the Natonal Institute of Technology (Rourkela), Orissa, India.
5. P S Revankar, DilipRathod (2009). Cryptography Using Neural network. International Conference on Sensor and Related Networks, At VIT University, Vellore, India.
6. Michal Rosen-Zvi, Einat Klein, IdoKanter, and Wolfgang Kinzel (2002). Mutual learning in a tree parity machine and its application to cryptography. Physical Review E, 66(6), 66-135p.

7. Andreas Ruttor, Wolfgang Kinzel, and IdoKanter (2007). Dynamics of neural cryptography. Physical Review E. 75(5), 56-104p.
8. N. Prabakaran, P. Loganathan, P. Vivekanandan (2008). Neural Cryptograpgy with Multiple Transfers Functions and Multiple Learning Rule. International Journal of Soft Computing, 3(3), 177-181p.
9. Biehl M., Caticha N., Riegler P. (2009) Statistical Mechanics of On-line Learning. In: Biehl M., Hammer B., Verleysen M., Villmann T. (eds) Similarity-Based Clustering. Lecture Notes in Computer Science, vol 5400. Springer, Berlin, Heidelberg
10. A. Engel, Otto-von-Guericke-Universität Magdeburg, Germany , C. Van den Broeck, Limburgs Universitair Centrum, Belgium.

Cite this Article
Himani Dua, Abhay Shukla. Using Artificial Neural Networking in Cryptography. *Journal of Artificial Intelligence Research & Advances*. 2020; 7(2): 56–61p.