International Journal of Computer and Electrical Engineering, Vol. 3, No. 2, April, 2011
1793-8163

AES Cryptosystem Development Using Neural Networks

Siddeeq. Y. Ameen and Ali H. Mahdi

Abstract— **Recently, there are some sorts of attacks that have been proven to be effective on the Advanced Encryption Standard AES. Thus the paper attempts to do some modification to stand against such sort of attacks by employing nonlinear Neural Network NN in the design and implementation of the AES. In the design of the NN performs encryption and decryption processes using of symmetric key cipher. The key used in both encryption and decryption processes is the initial weights for neural network and then train to its final weight with a fast and low cost algorithm, such as Levenberg – Marquardt Algorithm. The final weights of neural network represent the final key that can be used for encryption and decryption processes. The target from the network has been selected to match the output of the AES that have an efficient and recommended security. The proposed NN design has been modeled and computer simulated. Simulation results show the closeness of the results achieved by the proposed NN-based AES cryptosystem with that of the normal AES.**

Index Terms—**AES, Neural Networks, Symmetric Block Cipher, Levenberg – Marquardt Algorithm.**

I. Introduction

With increasing performance, requirements and improvements in technology, better-adapted ciphers are making their apparition to replace aging algorithms that have proven to be too weak or too slow for the current applications [1]. Encryption algorithms may be symmetric or asymmetric. Symmetric encryption is fast. It's commonly used for e-commerce transactions because it is fast. The symmetric key algorithms 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 ciphertext. DES and Rijndael are examples of algorithms based on this idea [2]. 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. Rijndael is an iterated block cipher, the encryption or decryption of a block of data is accomplished by the iteration of a specific transformation which is called (a round function). Further details about round structure and operation were found elsewhere [1].

The three standardized versions of AES are called AES128, AES-192, and AES-256 differ from each other in the key length (128, 192, and 256 bits) and the number of rounds (10, 12, and 14) [3]. Their security was thoroughly analyzed by the NSA [4]. However, recently, there are some sorts of attacks that have been proven to be effective on the AES [5]. Thus the paper attempts to do some modification to stand against such sort of attacks by employing nonlinear neural network in the design and implementation of the AES.

Artificial neural networks (ANNs) are models of intelligent systems are designed to simulate biological neural networks that exist in the human brain. Neural networks are used for classification and function approximation/mapping problems which are tolerant of some imprecision, which have lots of training data available, but to which hard and fast rules cannot easily be applied [6].

Finally, the paper will attempt to implement Rijndael cryptosystem using ANN. This approach is less complex design than the one with normal AES design and nonlinear in operation. The nonlinear must be reversible neural network that can make encryption/decryption process on plaintext/ciphertext with high performance and very low error rate. This will consider the architectural design of the ANN, the modification of ANN on software, testing the results of simulation, in order to provide NN-based AES. Further investigations are on the points of the significant effect on the implementation of NN, by keeping the internal structure of the NN components in mind, while making the design.

II. Design of NN-based AES Cryptosyst

The idea in this section is to design a non-linear reversible NN that can make encryption/decryption process on plaintext/ciphertext with high performance and very low error rate. The cryptographic algorithm which must be implemented by the neural network is the Rijndael algorithm. ANN provides a useful way of controlling nonlinear systems. The purpose of nonlinearity is to reduce the cracking ability on the system by using nonlinear activation function, and also nonlinear approximation property of the network makes it more useful for practical applications.

Multi Layer Perceptron (MLP) MLP is the most the most popular type of feed-forward NN, and the interesting

Manuscript received 30th September, 2010. This work was part Master thesis submitted to the University of Technology, Baghdad, Iraq.

S. Y. Ameen, Professor with the Gulf University, Kingdom of Bahrain, Dean of College of Engineering (phone: 973-39304338; fax: 973-17622230; e-mail: prof-siddeeqr@ieee.org).

A. H. Mahdi., He is now with the Department of Information Engineering, Baghdad University, Bagdad, Iraq (e-mail: aliinfo1980@yahoo.com).

property of it is its ability to emulate any input/output relationship [7]. The MLP designed here as a Batch training (Block – Adaptive) is a frequency update because it updates the weights after the entire block of training data is presented [7]. The block adaptation is more robust since the training step is averaged over all the training patterns. The MLP takes the following parameters:-

For the encryption process, the following settings can be used:-

1. The input vector is the plaintext.
2. The target from the neural network will be the ciphertext output from the AES algorithm.
3. The initial weights will be the key of the encryption process.
4. The non-linear activation function of each neuron will be (log sigmoid) that gives an output between (0-1).

For the decryption process, the following settings can be used:-

1. The input vector is the ciphertext.
2. The target (desired output) from the neural network will be the plaintext.
3. The initial weights will be the key of the decryption process.
4. The non-linear function of each neuron will be (sigmoid) that gives an output between (0-1). The input and output can be scaled by a factor to as to be compatible with the activation function.

For creating a successful NN, there are few tips to increase the likelihood of succeeding [8]:-

1. When the network design is too large for system that it implements, the network will "memorize" the training data, rather than "learn" from it. This will lead to a poor data performance for the testing data.
2. When initial weights are assigned to the network before training, randomize them with values between -1 and 1. This will usually result in faster convergence of the network.
3. A perfect error is ideal, it is not practical. In practical situations, strive for an acceptable error.

The designed NN-based cryptosystem must work for both encryption and decryption processes so as to achieve a symmetric system.

For the NN based cryptosystem to pass, two phases need to be verified, training and operation phases, as follows;

In the training phase, the plant (Rijndael algorithm) produces a ciphertext from specific input plaintext. The NN takes the input text as an input and the output text of the plant as the target output, and trains itself with the target output. After 5 to 10 iterations (according to the key used) with a group of plaintexts and same initial weights (the key)., the neural model structure is called "series-parallel model" where the plant (Rijndael algorithm) affects the dynamic behavior of the neural model [8]. The operation phase will be the feed-forward multi-layer NN with the final weights used to produce the output of encryption/ decryption processes. This phase will continue produces plaintext/ ciphertext until new key will used in the system, where NN will need to train it-self with the new key.

In the encryption and decryption processes, nonlinear activation function was used for each neuron which is the hyperbolic tangent (*tanh*). The property of Sigmoid produces an output in range of (-1 to 1). The data that are used in the AES are in range of (0-255). So, to make the NN working properly with the AES data (plaintext, ciphertext, cipher key) in presence of the Sigmoid activation function, there is a need to make the data in range of (0-1). This is achieved by using scaling factor (1/256) so as to be compatible with the output of activation function. The scaling factor can be used at the output of each neuron to rechange the output range back in (0-255).

III. System Implementation and Evaluation

In this paper and for simplicity only a 128-bit key length version of AES was considered for investigation. The AES uses a data block of length 128 bits for both of encryption/decryption process. Thus, the NN will use the data in bytes form, so each of the input and the output will be 16-bytes. The NN must have at least one hidden layer of 16 neurons so to achieve the input key length of 16 bytes.

The NN used in the training processes has a topology of [16-16-16-1]. It takes a vector of 16 bytes as an input and produces 16 bytes vector as a trained text. The weights of each layer are the key (16 bytes) vector. In this NN, the encryption process will not produce the desired text, because the minimum gradient will be reached without reaching the goal performance. Figs. 1 and 2 show the training with each of encryption and decryption processes with 16 bytes input NN.

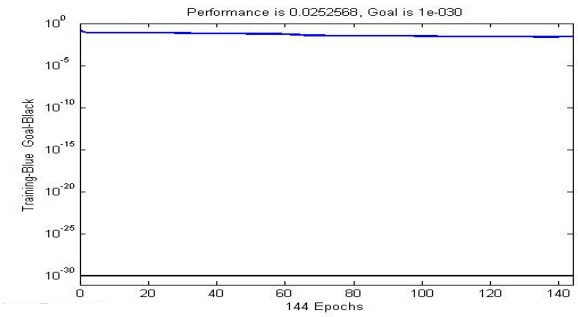


Fig. 1 Training of encryption process with 16 bytes input

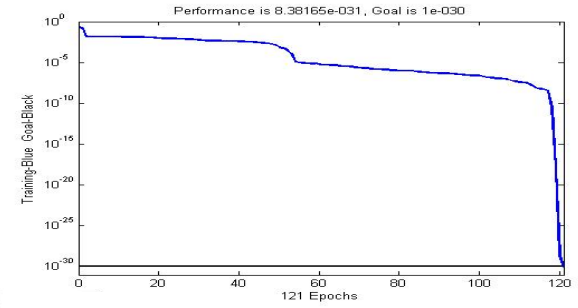


Fig. 2 Training of decryption process with 16 bytes input

Operation phase test shows that the output text on NN-Based AES in both encryption and decryption processes were not identical to that of the of encryption and decryption process of the normal AES.

One of the main solutions to this problem is to reduce the input and target vectors of the NN. This solution will tend to reduce the training time and to meet the goal performance. The idea of reduction is to make more than one NN operate in parallel. Each NN will operate with part of input and target

vectors, but with all of weight vector to achieve the normal AES operation.

The NN-based AES has been tested by building three NNs, each one has input and output vectors from the following:

16 bytes input and output vectors.

8 bytes input and output vectors.

4 bytes input and output vectors.

Each of these NNs has been tested with three plaintext vectors, three ciphertext vectors, and three key vectors. Each of these had length of 16 bytes. Figs. 3 and 4 show differences between the encryption and decryption processes for the three NNs. The idea of training NN for more than one data block (one block of data per iteration) is to train the weights of the NN for all (plaintext block/ ciphertext block) pairs so as to reduce the output bit error. This attempt will ensure that the NN will be trained with all the possibilities of input/target patterns which tend to train the weights with all these possibilities to get the final weights that represent the final keys for encryption and decryption processes. The increasing of iterations for each NN shows a reduction in bit error for all NNs.

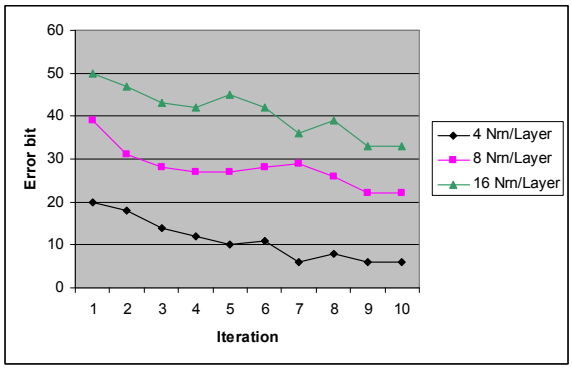


Fig. 3 Error bits/iteration for the three NN in the encryption process.

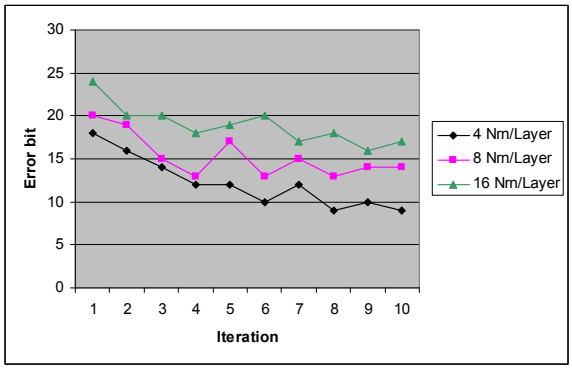


Fig. 4 Error bits/iteration for the three NN in the decryption process.

Fig. 3. shows the error bits at encryption process. The error bits range of 16 inputs NN are (33 - 40), and (22 – 28) for 8 inputs NN, these because the minimum gradient has been reached without meet the goal performance (which is 10^{-30} between the input and the desired output). The minimum reduction in the error bit is in the 4 input NN (6 – 10).

The decryption process shown in Fig. 4 has lower error bits rate than that in encryption process, for the three NNs, because the ciphertext produced by the NN in the encryption process will pass through the same NN topology with the same initial weights (key). Thus the NN will be trained without reaching the minimum or maximum gradient.

Several NN topologies have been tried and it has been found that the NN with topology [4-16-16-1] is the best one of these NNs. Each of encryption and decryption processes must have 4 NNs, from the chosen NNs; operated in parallel to achieve the AES with 16 bytes plaintext, ciphertext cryptosystem. The NN cryptosystem consists of 4 NN layers work in parallel. Each of the NN layer takes input vector of 4 bytes. The topology of the desired NN in the encryption/decryption process is as follows:-

Layer 1 with 4 neurons, where each neuron receives 1 byte from the input vector (plaintext/ciphertext).

Layer 2 with 16 neurons that each neuron is a weighted sum of all the 8 neurons in the 1st layer.

Layer 3 with 16 neurons that each neuron is a weighted sum of all the 8 neurons in the 2nd layer.

Layer 4 (output layer) with 1 neuron that is a weighted sum of all the 8 neurons in the 3rd layer.

The initial weights of the NN are the key for the encryption/decryption process. Thus the 16 bytes key in the NN were taken in the following way:

4x1 matrix (represents the 4 bytes from the key according to the 4 bytes positions of input vector) to be the initial weights from input to layer 1.

16x4 matrix (represents 16 bytes from the key and repeated 4 times) to be the initial weights from layer 1 to layer 2.

16x16 matrix (represents the same 16 bytes in the layer 1, but repeated 16 times) to be the initial weights from layer2 to 3.

1x16 matrix (represents the same in layer 1 from input, but transposed) to be initial weights from layer 3 to output layer.

The equation of the output of this NN is as follows:

$$output = \tanh\left(\sum_{k=1}^{16} w_{1k} \cdot \tanh\left(\sum_{k=1}^{16}\sum_{j=1}^{16} w_{kj} \cdot \tanh\left(\sum_{j=1}^{16}\sum_{i=1}^{4} w_{ji} \cdot \tanh\left(\sum_{i=1}^{4} w_{i1} \cdot input\right)\right)\right)\right) \quad (1)$$

where; output: plaintext/ciphertext (according to the process), w is the symmetric key. The tangent sigmoid transfer functions in the hidden layers compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slope must approach zero as the input gets large [9]. This causes a problem when using steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude; and therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal [10]. The learning algorithm suitable in this design is the LMA. This algorithm was designed to approach second-order training speed without having to compute the Hessian matrix [10].

Levenberg - Marquardt Algorithm (LMA) is an advanced non-linear optimization algorithm. It can be used to train the weights in a neural network just as Back Propagation Algorithm would be. It is reputably the fastest algorithm available for such training. LMA had been introduced to feed-forward networks for a number of years. The primary objective of this learning algorithm of multi-layer feedforward networks is based on gradient descent which is usually slow and the convergence is of linear order. In addition, the speed usually depends on some parameters, for example, the step size. Alternatively, various second order

learning methods had been proposed. Among these second order methods, LMA method, a modification to the Gauss-Newton method, is one of the popular and the effective methods. One characteristic of this method is that it incorporates an extra term to stabilize the system and to deal with the small residual problems in the learning [10].

LMA based on nonlinear optimization technique by minimizing the sum of squares of errors (SSE). The LMA represents a simplified version of Newton's method applied to the prob1em of training Multi Layer Perceptron NNs. The LMA is often superior to other training algorithms in off-line application. This motivates the proposal of using the algorithm to train the NN. Furthermore, LMA represents a simple heuristic strategy to increase the convergence speed of the back propagation algorithm with a batch update. The algorithm can be summarized as follows [10]:

If the error function over the entire training set has decreased, increase the learning rate (η) by multiplying it by a number η_increment.

If the error function has increased more than previous error, decrease the learning rate (η) by multiplying it by a number η_decrement.

The training parameters for LMA are η-inc, η-dec. The η is multiplied by η - dec whenever the performance function is reduced by a step. It is multiplied by -η inc whenever a step would increase the performance function. If η becomes larger than -η max, the algorithm is stopped. The parameter mem_reduc is used to control the amount of memory used by the algorithm [10].

The chosen NN has been trained with different initial weights (keys) to check the validity of the proposed cryptosystem. The same evaluation were carried out through the comparison of the proposed NN-AES with the AES as shown in Fig. 5. shows the encryption and decryption with different key.

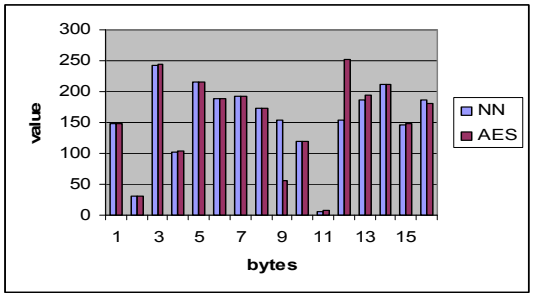
From the results of the execution, it's shown that some of the differences in the bits between the AES and NN cryptosystem, these error bits are reduced after a number of iterations, where the NN has been trained on all of the data that can be used in this cryptosystem.

IV. Conclusions

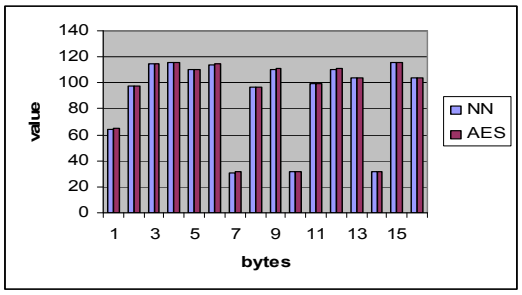
The designed NN-based cryptosystem is a good idea of building very complicated cryptosystem, where the crypto analyst or the cracker not just need the topology of the NN and the key to crack the system, but also need to know the number of adaptive iterations and the final weights for the encryption and decryption systems. Applying higher numbers of plaintext/ ciphertext to the NN-based cryptosystem so as to make the error rate as minimum as possible (which must be match 0 error).

The paper has shown that it is possible to implement such powerful algorithm with the use of NN. This will ensure that confusion and diffusion have been achieved in the ciphertext of the NN-based AES cryptosystem.

Finally, extra woks is now under investigation to implement the system with the FPGA devices and to investigate the confusion and diffusion achieved by both techniques.



a - encryption process



b- decryption process

Fig. 5 Comparison of AES and NN-AES for Encryption and decryption with different key

References

[1] W. Stallings, "Cryptography and Network Security: Principles and Practice", Prentice Hall, 3rd Edition, 2007.

[2] Seref SAGIROGLU, Necla ÖZKAYANeural Solutions for Information Security, Journal of Polytechnic, Vol: 10 No: 1 pp.21-25, 2007

[3] J. Daemen, V.Rijmen, "The Block Cipher Rijndael", NIST's AES home page 2001. http://www.nist.gov/aes.

[4] A. Samih, A. Aziz and N. Ikram, "An Efficient Software Implementation of AES-CCM for IEEE 802.11i Wireless St" 31st Annual International Computer Software and Applications Conference, Beijing, China, 2007.

[5] A. Biryukov, D. Khovratovich, and I. Nikoli, "Distinguisher and related-key attack on the full AES-256", CRYPTO'09, LNCS. Springer, 2009.

[6] B. Chandra, and P. Paul Varghese, Application of Cascade Correlation Neural Networks for Cipher System Identification World Academy of Science, Engineering and Technology 26 2007.

[7] J. Principe, N. Euliano, and W. Lefebvre, "Neural and Adaptive Systems: Fundamentals through Simulations", John Wiley& Sons, Inc., 2000.

[8] W. Sun, Y. Yuan, Optimization Theory and Methods: Nonlinear Program-ming, Springer Optimization and Its Applications, Springer, New York, 2006.

[9] K. Madsen, H.B. Nielsen, O. Tingleff, "Methods for Nonlinear Least Squares Problems", Informatics and Mathematical Modeling, Technical University of Denmark, 2nd Edition, April 2004.

[10] C. Kanzow, N. Yamashita and M. Fukushima, Levenberg-Marquardt Methods for Constrained Nonlinear Equations with Strong Local Convergence Properties", Journal of Computational and Applied Mathematics 172, pp. 375-397, 2004.

Siddeeq Y. Ameen born in 1960 in Mousol, Iraq. He awarded BSc in Electrical and Electronic Engineering from the University of Technology, Baghdad, Iraq in 1983. Next, he awarded the MSc and PhD from Loughborough University, UK. in 1986 and 1990, in the field of digital communication systems and data communication, respectively. From 1990-2006, Professor Siddeeq worked with the University of Technology in Baghdad with participation in most of Baghdad's universities. From 2006-present, he acts as a dean of college of Engineering at the Gulf University, in Bahrain. Through his academic life he published over 70 papers in field of data communication and information security.