International Journal of Scientific & Engineering Research, Volume 5, Issue 10, October-2014
ISSN 2229-5518

Advanced Encryption Standard (AES) Enhancement Using Artificial Neural Networks

[1]Yasin Kh. Yasin, [2]Prof. Siddeeq Y. Ameen, [3]Dr Hassan Awheed Chiad

Abstract— The paper attempts to modify the conventional Advanced Encryption Standard (AES) to obtain a customized and more secure algorithm. This modification is based on the use of artificial neural networks for key expansion schedule processes. The latter is achived with use of symmetric key cipher based on Pseudo Random Number Generation PRNG in key generation processes for the initial weights for the neural network. hen train it Fast and low cost training algorithm based on Levenberg – Marquardt Algorithm have been used to get outputs that match the conventional key expansion used in the conventional AES algorithm.The proposed modification have been investigated using Matlab ver.8.0 simulation to verify the ability of ANN based key expansion to produce the same ciphertext as that achieved by the conventional AES for the same plaintext. The results of comparison between the two model show the ability of the proposed modification to produce the same secure system even with different key expansion technique. However, the use of neural network adds more security to the conventional AES, because the opponent need to know the topology of neural network and the number of iteration together with the initial value key that only required by the conventional AES.

Index Terms— AES, ANN, Key Expansion, LMA, PRNG.

1 Introduction

Cryptography has become an essential component to control the authenticity, integrity, confidentiality and non-repudiability of private data that flows through public networks. With increasing performance, requirements and improvements in technology, better-adapted ciphers are making their apparition to replace aging algorithms [1]. Block ciphers are mostly based on the idea by Shannon, that sequential application of confusion and diffusion [2]. One of the most important type of blocj cipher that achieves the above requirement is the Advanced Encryption Standard, AES.

AES is an iterated block cipher which was selected by NIST as an international standard. It is now the most widely deployed block cipher in both software and hardware applications. The three standardized versions of AES are called AES-128, AES-192, and AES-256 differ from each other in the key lengths: 128, 192, or 256 bits, encryption consists of 10 rounds of processing for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys [3]. The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks. The inclusion of a round-dependent round constant eliminates the symmetry, or similarity, between the ways in which round keys are generated in different rounds. The fewer bits one knows, the more difficult it is to do the reconstruction or to determine other bits in the key expansion [4]. 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.

This work was part Master thesis submitted to the University of Technology, computer Engineering Department, Baghdad, Iraq.

- *Yasin Kh. Yasin., He is now student in university of technology, computer Engineering Department, Bagdad, Iraq (e-mail:yaseen_albakry@yahoo.com).*
- *Dr. Siddeeq Y. Ameen, Professor with the University of Mosul, College of Computer and Information Engineering, Electronic Engineering Dept., Mosul, Iraq (e-mail: prof-siddeeqr@ieee.org).*
- *Dr. Hassan Awheed Chiad, Lecturer with the university of technology, computer Engineering Department , Bagdad, Iraq (e-mail: hsn_uot@yahoo.com).*

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The idea of Artificial Neural Network (ANN) came from the desire to produce a system that are capable to sophisticated computations in the same way that the human brain performs so that it can give that same understanding of problem as human brain[6]. Thus, the paper will attempt to implement Rijndael (Key Expansion in AES algorithm) using ANN, by using ANN for a purpose of Key generation. 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 key expansion schedule 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. Most of our discussion will assume that AES is a block cipher with a block length of 128 bits and the key length is 128 bits[11].

2 . Conventional Advanced Encryption Standard Algorithm

Conventional AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called state. The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array is of dimensions 4x4. (In the Rijndael version with variable block size, the row size is fixed to four and the number of columns varies. The number of columns is the block size divided by 32 and denoted Nb). The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted Nk, is equal to the key length divided by 32 [4]. During each round, the following operations are applied on the state [4]:

1 .Sub Bytes: Every byte in the state is replaced by another one, using the Rijndael S-Box
2. Shift Row: A transposition step where each row of the state is shifted cyclically a certain number of steps.
3. Mix Column: A mixing operation which operates on the columns of the state, combining the four bytes in each column
4. AddRoundKey: Each byte of the state is combined with a round key, which is a Different key for each round and derived from the Rijndael key schedule
5. Final Round (No Mix Columns)
   - SubBytes
   - Shift Rows
   - AddRoundKey

In the Add Round operation, a round key is applied to the state by a simple bitwise XOR. The round key is derived from the cipher key by the means of the key expansion. The round key length is equal to the block key length (16 bytes). The key expansion involves the followings:

1) Takes as input a Nb word key and produces a linear array of Nb * (Nr+1) words.
2) Expanded key provide a Nb word round key for the initial AddRoundKey() stage and for each of the Nr rounds of the cipher.
3) The key is first copied into the first Nb words, the remainder of the expanded key is filled Nb words at a time.
4) Assuming a 128-bit key, the key is also arranged in the form of a matrix of 4 × 4 bytes. As with the input block, the first word from the key fills the first column of the matrix, and so on.
5) The four column words of the key matrix are expanded into a schedule of 44 words. Each round consumes four words from the key schedule.

Fig. 1 clearly show the arrangement of the encryption key in the form of 4-byte words and the expansion of the key into a key schedule consisting of 44 4-byte words.

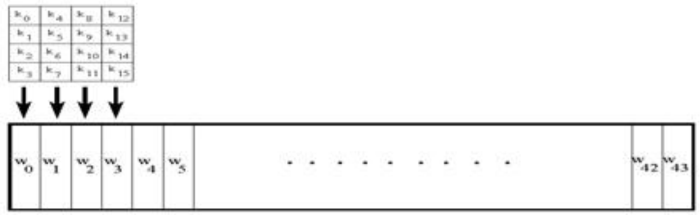

Figure.(1): Key expansion arrangement

3. NN-Based Key Expansion for AES Algorithm Design

To achieve the required AES operation, it is essential that the NN based key expansion should work for both encryption and decryption processes. In the design, the proposed key expansion schedule process has the following parameters:

i. The seed key (4 word) which represents the input vector to the NN.
ii. The key schedule which represents the target vector (desired output) of the NN.
iii. The PRNG which represents initial weights of the NN.
iv. The round keys will be extracted upon NN training achieved by both sender and receiver. The basic approach for training the model can be understood by the flow diagram shown in as shown in Fig. 2.

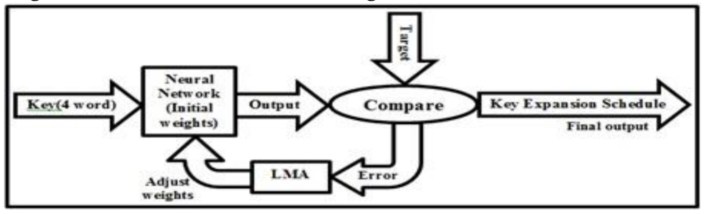

Fig. (2): Training operation foe NN key expansion for encryption/decryption processes

Initially, some random values are assigned for the weight and bias values and these values are optimized using Levenberg-Marquardt (LM) training algorithm [49]. Levenberg Marquardt algorithm, LM Agenerally is one of the most popular algorithms for non-linear minimum mean squares problems. Because of the characteristics of rapid convergence and stability, the method is used in many of the problems of modeling (Sakamoto et al, 2005).

The structural configuration of the model is selected as either 256,192 or 128. For the applied input pattern, some random numbers between +1 and −1 are assigned to the weights and biases and the output of the model is computed corresponding to that input pattern. Some arbitrary parameters required for training of the neural model like mean square error, learning rate, momentum coefficient, and spread value have also been taken as 5×10^{-7}[49], 0.1, 0.5, and 0.5, respectively.

The error between the calculated and the measured result is then computed and, according to this computed error, all the weights and biases are updated with the help of LM training algorithm. This updating process is carried out after presenting each set of input pattern, until the calculated accuracy of the model is estimated to be satisfactory.

4. Training and Operation Phase of the Proposed ANN Key Expansion

In the training phase of the ANN, the plant (AES algorithm) generating the output number from specific input number the ANN takes the seed key (4 Word) as an input and the random key (40 Word) of the platform as output target. Next, the NN will be trained with the output target itself. After 8 to 10 iterations or according to the acceptable error rate, the training period will be terminated. The NN which is used in the training process has topology [4-4-40-1]. It takes a vector of (16 bytes) as input and produces (160 bytes) vectors as text trainers. The NN topology desirable in the process key expansion has been found to be: -

i. Layer 1 with 4 neuron, where each neuron receives 4 byte from the input vector (key Word).
ii. Layer 2 (hidden layer) with 4 neuron that each neuron is a weighted sum of all the 4 neuron in the 1st layer.
iii. Layer 3 with 40 neuron that each neuron is a weighted sum of all the 4 neuron in the 2nd layer.
iv. The output layer with 1 neuron that is a weighted sum of all the 40 neurons in the 3rd layer.

In the operation phase, a three-layer feed-forward NN-with final weights will be created at the final output of the Pseudo Random Number Generator processes. This phase will continue to produce key-expansion until new key will be used in the system. As mentioned before, in the next step, the ANN need to be trained it-self with the new key.

It is worth to mention that the key expansion schedule processes uses a non-linear activation function to each neuron with hyperbolic tangent (tank). The property of sigmoid are used to produce an output in domain of (-1 to 1). The data that used in the AES are ranged between (0-255). Therefore, for the NN to operate properly with the key expansion within AES algorithm in presence of the sigmoid activation function, there is a need to make the data in the range of (0-1). This is achieved by scaling factor (1/256) to be compatible with the output activation function. A scalable factor at the output for each neuron to re-change the scope of production again at (0-255) can also be used.

The initial weights of the NN are produced by the Pseudo Random Number Generator, PRNG for the encryption/decryption process. These 16 bytes key are taken by the NN in the following way:

1. 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.
2. 4x4 matrix (represents 4 bytes from the key and repeated 4 times) to be tie initial weights from layer 1 to layer 2.
3. 40x4 matrix (represents the same 4 bytes in the layer 1, but repeated 40 times) to be the initial weights from layer2 to 3.
4. 1x40 matrix (represents the same in layer 1 from input, but transposed) to be initial weights from layer 3 to output layer.

The output from the NN will be:

$$output = \tanh(\sum_{k=1}^{40} w_{1k} \cdot \tanh(\sum_{k=1}^{40}\sum_{j=1}^{4} w_{kj} \cdot \tanh(\sum_{j=1}^{4}\sum_{i=1}^{4} w_{ji} \cdot \tanh(\sum_{i=1}^{4} w_{i1} \cdot input)))) \quad \text{-- (1)}$$

where the output is the key round for both of the encryption and decryption processes, w is the weights. The transfer function of tangent sigmoid in the hidden layers will be used to transfer an unlimited input range into a limited input range. The sigmoid function is characterized by their slope that should approximated to zero as the input increases. This give rise to a problem that when using steepest descent to learn a multilayer network with sigmoid functions, the regression can have a measure very small values. This will lead to small changes reasoned in weights and biases, even though the weights and biases are far from their perfect [50].

The learning algorithm adopted in this design is the LMA is as shown in Fig. 3. This algorithm was to approach second-order training quickly without having to calculate the Hessian matrix. The training parameters for LMA are η inc , η dec. The factor η is multiplied by η dec whenever been reduced the performance function 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 terminated. The parameter mem_reduc is used to control the size of memory used by the algorithm[49]. According to the AES type (128,192,256), the NN has been trained with various initial weights (PRNG) and number of rounds to check the its validity with the AES.

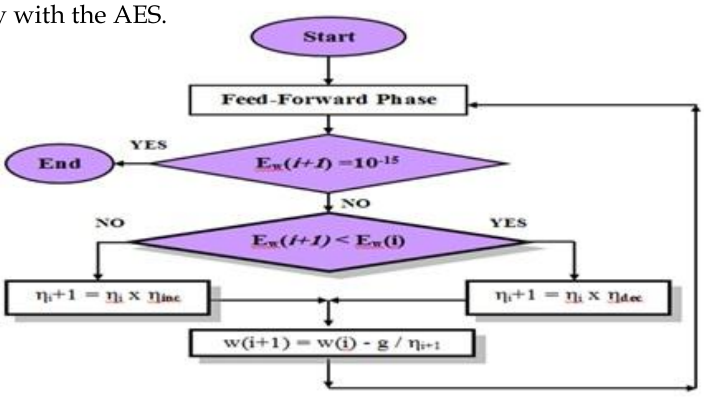
Start
Feed-Forward Phase
End
YES
$E_m(i+1) = 10^{-15}$
NO
NO
YES
$E_m(i+1) < E_m(i)$
$\eta_i+1 = \eta_i \times \eta_{inc}$
$\eta_i+1 = \eta_i \times \eta_{dec}$
$w(i+1) = w(i) - g / \eta_{i+1}$

Figure.(3): Flow chart of LMA

5 Simulation and Evaluation of NN Based AES Key Expansion

The performance of the NN based key expansion can be checked by an experiment that can test the NN for both encryption and decryption processes. The test should also compare their results and error ratio with the AES results. The first case in the simulation is the key expansion for AES_128. The results of such test is as shown in Table (1) and figure (5) shows result of value approach zero at each iterations.

Table(1) Performance system through each iterations

Epochs	Time (Second)	Mean Square Error (MSE)
0	0.2809999999999	1.32860414799650e+04
1	0.3900000000001	5.42712026082221e+03
2	0.4519999999998	9.46660826171853e+02
3	0.5150000000001	5.78951105726962e+01
4	0.5619999999998	1.14737146278459e-01
5	0.6079999999997	3.04542477671008e-06
6	0.6709999999999	5.89287877837792e-13
7	0.7179999999996	6.92718419325661e-21
8	0.7800000000001	1.35240464698344e-27

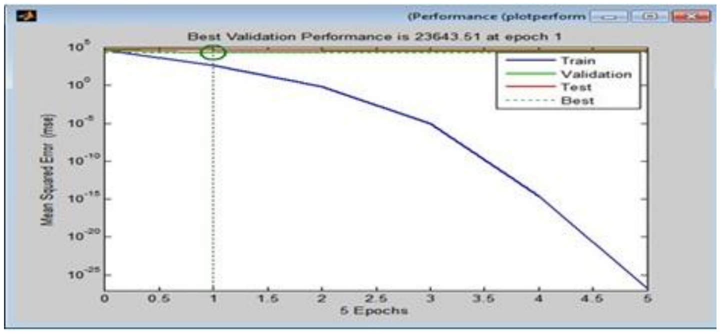

Figure (4) curve plots of ANN Lines represent the linear regression of observed and predicted value for AES_128 from Levenberg–Marquardt algorithm

It clear from the results presented in Table 1 and Fig. 4 that the average MSE keys training set were 1.32860414799650e+04 To 1.35240464698344e-27, respectively. Average number of epochs and CPU time elapsed at the end of training, respectively were 0.2809999999999 To 0.7800000000001s. The number of epoch (8) iteration 's .

The following can be observed from the results presented in Table 1 anf Fig. 4;

i. The system reduces the overall running time. This means that there is an improvement in performance when running in less overall system as shown in Table (1).
ii. The output number of the key-Expansion training algorithm is not identical to the desired target. Figure (6) shows the difference between Pseudo Random Number Generator (key word) of the AES and the NN-based AES.
iii. To adjust the output to the desired target as exactly as possible we should run some processes. We will put the entire data set through the network will perform a linear regression between the network outputs and the corresponding targets. We will need to un-normalize the network outputs ,this can be achieved via round instruction. In this case we will have on approximation of number to three digits output.

When the network completed the performance curve which determine the convergence of the training curve will have an error rate as shown in Fig. (5).

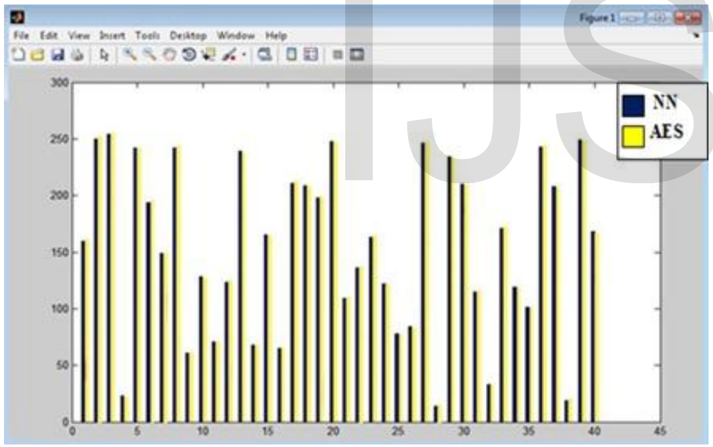

Fig. (5): 2-D diagram (PRNG) between AES algorithm typically and the NN-based AES

Often helpful to check the network response in further detail. One option is to carry out a regression analysis between the network response and the identical targets. The network output and the identical targets slope need to be checked. If we had a completely matching (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0. In Fig. (6) we can see that the numbers are very close. The correlation coefficient between the outputs and targets is a measurement of the extent of the variation at the output and by the targets. If this number is equal to 1, then there is an ideal relationship between the targets and outputs. The results shown in Fig. (6) shows that we have a number very close to 1, which indicates it good fit. Furthermore, it can be shown that the behavior is linear between network outputs versus the targets. The perfect fit (output equal to targets) is indicated by the solid line. In this problem it is difficult to distinguish the best linear fit line from the perfect fit line, because the fit is so good.

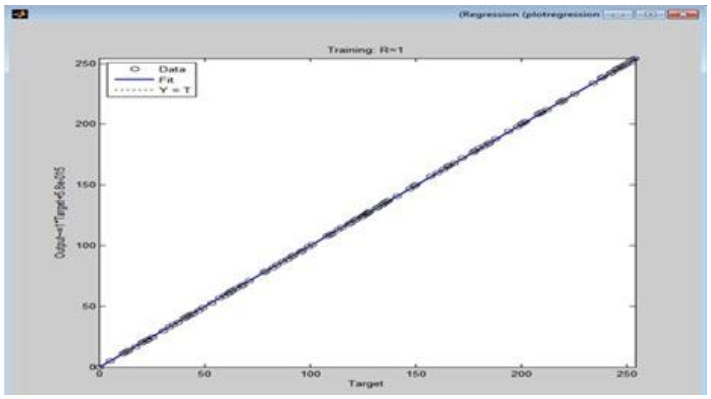

Fig. (6) The training processes for the three layer neural between Target and Output

The paper also show the implementation and evaluation of AES key expansion algorithm that based to LM-ANN for 128, 192 and 256-bits. The shown in Table 3 show that; key size has an almost-linear impact on number of epochs (iterations), time and system performance. However, the AES standard has key size variants. Therefore, longer keys will give more secure encrypted text output. This means that the increase in the size of the key offset by an increase over time with a decrease in system performance (incremental MSE). Some companies that employ ultra-high security in their systems may require longer key size than 128-bit. Finally, the results shown in Table 3 compare the performance of different AES and the time with respect to epochs used.

Table (3) The epochs , Time & MSE for the training algorithms.

Training algorithm	Epochs	Time(s)	Performance
AES Key expansion 128	8	0.7800000000001	1.352404646983 e -27
AES Key expansion 192	9	1.35700000000000	5.997673605920 e -24
AES Key expansion 256	10	1.46600000000000	2.082105121550 e-22

Having produced the round keys using the NN-based key expansion, it is essential to verify that against the conventional AES. Thus the test involve the execution of NN-based AES_128. Furthermore, the NN-based AES has been compared with the AES with the following case. When the encryption process begins, the first 128-bits plaintext matrix is the following:

"AES based on ANN"

which should be changed to Hexadecimal numbers matrix as:

41	62	64	20
45	61	20	41
53	73	6F	4E
20	65	6E	4E

The key vector used is:

00	04	08	0c
01	05	09	0d
02	06	0a	0e
03	07	0b	0f

The produced ciphertext in Hexadecimal matrix from is:

01	f3	16	58
fd	df	3c	0a
68	18	53	88
cc	21	27	b7

This will give the final ciphertext matrix:
"&#245;p&#181;&amp;&#185;p(¬&#228;&#191;z)&#205;h &#214;"

When the decryption process begins, the first 128-bits re-plaintext Hexadecimal matrix is:

41	62	64	20
45	61	20	41
53	73	6F	4E
20	65	6E	4E

where re-plaintext matrix which produced from the decryption process is:

"AES based on ANN"

It is clear from the results of the execution, that there is some of the little differences little in the bits between the conventional AES and ANN based AES. These error bits can be reduced and faded out as more number of iterations is achieved or the NN has been trained on all of the data that can be used in this Key Expansion algorithm. Furthermore, it has also been observed that there is a difference in one character between the trained text and the target text, this difference is shown in the following example:

The target decimal number is: 31 (011111). while the output decimal number from the neural network is: 32 (100000)

This error is occurred even if the target performance of the error reaches the 10-15. The NN makes the training with the target data in its decimal representation, not in the binary representation. Thus the NN considers the error between 31 and 32 is too small. This problem occurs because of the properties of the LMA.

6 Conclusions

The paper presented a NN-Based key expansion instead of conventional way used to produce the rounds key in the conventional AES. This will enhance the conventional AES security since more than the original 128, 192 or 256 bits key size need to be guessed to break the AES. The paper comes to the following conclusions;

i. The proposed key generation method reduces the threats of breaking the symmetric keys by taking artificial neural networks and from it the overall key will be generated. This strengthen the security since ANN training used to obtain the round keys only known by sender and receiver.
ii. The design NN-based Key expansion in AES algorithm will produce a very complicated structure. This will be difficult for the cryptanalyst or the cracker to guess. In this case not just the topology of the NN need to be guessed, but also need to know the number of adaptive iterations and the final weights for the Key expansion schedule.
iii. The design is completely modular, this will make it possible to redesign or modify any module without affecting the overall design.
iv. Finally, there is very close correlation between the conventional AES and the NN-based AES that uses NN for key expansion.

References

[1] W. Stallings, "Cryptography and Network Security Principles and Practices, Fourth Edition", Prentice Hall, November 16, 2005.

[2] D. Selent, "ADVANCED ENCRYPTION STANDARD", Rivier academic Journal , FALL 2010.

[3] F. Nekoogar, Amphion, "Digital Cryptography: Rijndael Encryption and AES Applications", TechOnLine Publication, Oct. 2001.

[4] S. ElAdib and N. Raissouni "AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key", International Journal of Reconfigurable and Embedded Systems (IJRES),2012.

[5] S. Soni, H. Agrawal, M. Sharma, "Analysis and comparison between AES and DES Cryptographic Algorithm", International Journal of Engineering and Innovative Technology, Vol 2, Issue 6, December 2012, pp.362-365.

[6] I. Kanter, W. Kinzel, "Neural Cryptography", Institute for Theoretical Physics University, Am Hubland, Minerva Center and Department of Physics Bar-Ilan University, 2002.

[7] David G. Luenberger, Yinyu Ye, " Linear and Nonlinear Programming ", Stanford University, 2010

[8] Sh. Pandey and Prof. M. Mishra, " Cryptanalysis of Feistel cipher using Back propagation Neural Network ", International Journal of Emerging Technology and Advanced Engineering , March 2012.

[9] D. Peacham and B. Thomas, "A DFA attack against the AES key schedule," SiVenture White Paper 001, 2008.

[10] Kh. Shihab ,"A back propagation neural network for computer network security". Journal of Computer Science 2:710–715, 2006.

[11] Siddeeq. Y. Ameen, and Ali H. Mahdi, " Modeling and Software Implementation of NN-Based AES Cryptosystem", University of Technology, Baghdad, Iraq,2010.

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

[13] Soukaena H. Hashem, " Developing a Block-Cipher-Key Generator Using Philosophy of Data Fusion Technique", Paper, Journal of Emerging Trends in Computing and Information Sciences, Volume2 No.5, MAY 2011.