ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse32942020.pdf
https://doi.org/10.30534/ijatcse/2020/32942020

Utilization of Artificial Neural Networks for Fast Encryption and Decryption of Messages

Ayman Haggag[1], Khaled Elsharkawy[2], Eman Said[3]
[1]Electronic Technology Department, Helwan University, Egypt, haggag@techedu.helwan.edu.eg
[2]Basic Science Department, October High Institute for Engineering & Technology, Egypt, khaled.elsharkawy68@gmail.com
[3]Electronic Technology Department, Helwan University, Egypt, emansaaid2@gmail.com

ABSTRACT

This paper aims to provide a fast method to decrypt short encrypted messages that consist of 10 letters by using the technique of trained Feed-Forward Artificial Neural Network (FF-ANN). The use of artificial neural network is one of the methods of artificial intelligence, a method that mainly depends on increasing the training of the network on many and many examples. These examples must take the same context of the main purpose of this research so that the results of the network are very close to the desired and desired realism. We give the feature of our trained FF-ANN and show how we train our network using 50 encrypted messages using our proposed method to achieve the best results. Our collection of test patterns of cryptographic messages took a shorter decryption time compared to the solution using mathematical methods. We can apply this method presented in this research in the large messages that the use of mathematics for encryption and decryption is very difficult and consumes a very long time, but the use of the neural network will be easy as a method and the expectation for time consumption to be little, especially with good network training which may be of interest in next researches.

Key words: Artificial Neural Network (ANN), Elliptic Curve Cryptography (ECC), Feed Forward Artificial Neural Networks (FF-ANN).

1. INTRODUCTION

The neural network is an artificial machine which is designed to do certain tasks in the same manner as the human brain does [1]. The neural network is simulated using software running on a personal computer. In the past few years the development of ANN greatly evolved. ANN has received much great attention and is now regarded as one of the greatest computation tools. Much of the interest in ANN is due to the obvious ability of neural networks to imitate the brain's ability in making decisions and drawing conclusions when presented with complex, noisy, and partial information. Generally, the ANN is a network with links, and the weights of these links are represented as the strength of connections. There is no standard definition of any of the popular implementations of ANN. As an example, the back-propagation neural network is the most common technique for solving practical problems which are designed for modeling how a particular task is performed by the brain.

The neural network may be physically implemented by the use of electronic components or it may be simulated using software running on a personal computer. A neural network can be seen as a parallel implementation of a distributed processor that is composed of a set of simple processing units. These units have an inherit natural tendency to store experimental knowledge, then make this knowledge available for use.

Neural networks have a noticeable ability to derive useful meanings from imprecise or complicated data. Neural networks can be used to detect trends and extract patterns that are far too complex for being identified by humans or even other simple computer techniques.

Cryptosystems are used for protecting important data, commonly protecting data integrity, data confidentiality, and the authenticity of information sources. Cryptosystems must meet the ever increasing strength of specifications concerning information security, as well as, meeting standard specifications relating to encryption and decryption.

Our goal is how to use the trained FF-ANN to decrypt a certain encrypted message in certain circumstances. We perform this task by training the artificial neural network with many examples that have the same behavior as our message. Also, the properties of the network are changed several times during training and after testing to get the best results in the end.

The rest of this paper is organized as follows. Section 2 is about the biological neuron, artificial neuron model, and feed-forward networks, feed-forward back-propagation networks, feed-forward operations, and back-propagation operations. Section 3 defines the cryptographic background, asymmetric key algorithms, the ECC algorithm and the advantages of using ECC. Section 4 presents simulation results of a numerical example of encryption and decryption of a message that consists of 10 letters and shows how we solve it algebraically and then solves it using our trained FF-ANN with all the parameters deduced from the network. Section V gives our evaluation and discussion and a comparison between the mathematical method and the proposed FF-ANN. Section 6 gives the conclusion.

2. NEURAL NETWORKS

The idea behind the use of Neural Networks in computation is to mimic the behavior of biological neurons in human brains. Neural Networks used in computation is a mathematical model for the simulation of biological neurons.

2.1 The Biological Neuron

The main terms used with a biological neuron [1] are defined in the following list of terms. Soma, which is the nucleus of the neuron as shown in Figure 1. Dendrites, which are the long irregularly, shaped filaments attached to the soma-input channels. Axon which is another link type that is attached to the soma output channels.

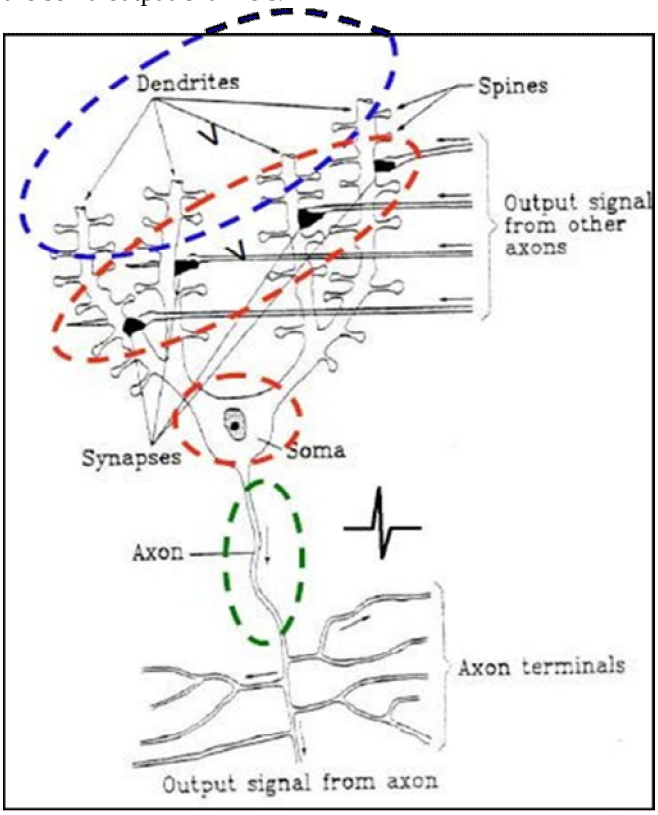

Figure 1: The biological neuron

The output of the axon is a voltage pulse (a spike) that lasts for a millisecond. Firing of the neuron, this involves the movement of ions into the cell. This occurs when neurotransmitters cause ion channels on the cell membrane. Synaptic junction: which is the electro-chemical contact between neurons in which axon terminates in, the size of synapses is linked with learning. Spine, which is a dendritic spine (or spine) that is a small membranous protrusion from a neuron's dendrite that typically receives an input from a single axon at the synapse.

2.2 Artificial Neuron Model

In artificial neurons, an activation function (AF) controls the firing as well as the strength of the exiting signal [2]. The activation function consists of many types but we will adapt the use the sigmoid type as shown in Figure 2.

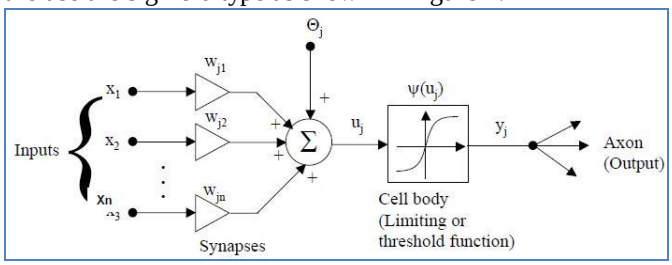

Figure 2: Artificial neuron model

θ_j : *external threshold , offset or bias*

w_{ji} : *synaptic weights*

x_j : *input*

y_j : *output*

$$y_j = \psi\left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right)$$

2.3 Feed-Forward Networks

Feed-forward networks [3] consist of an Input layer, a Hidden layer, and an Output layer. For the Input layer, this layer consists of a set of passive nodes that can only transmit the input signal to the following layer as shown in Figure 3. The number of inputs to the neuronal network corresponds to the number of neurons (n) in this layer.

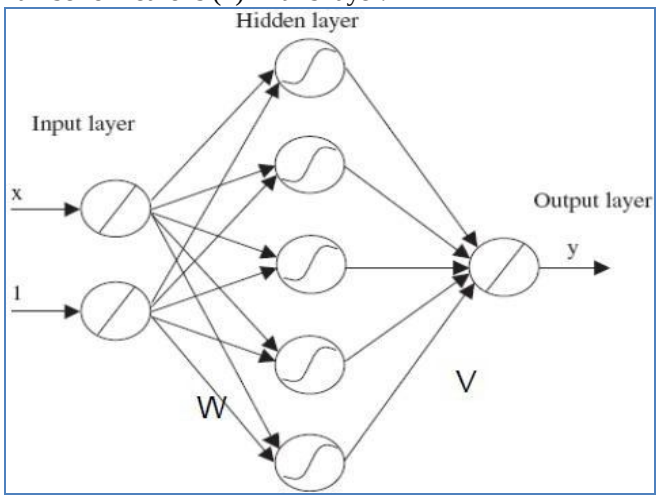

Figure 3: Feed-Forward Networks

For the Hidden layer, this layer has a variant number of layers with a variant number of neurons (m). The nodes in this layer are active as they take part in the signal modification. For the Output layer, the number of the output values of the neural network (r) corresponds to the number of neurons present in the output layer. The nodes in this layer are considered active nodes. For the example shown in figure 3, the Network size is calculated as follows:

$$n \times m \times r = 2 \times 5 \times 1$$

W_{mn} : *input weight matrix*

V_{rm} : *output weight matrix*

N.B: There is no feedback within this network

2.4 Feed-Forward Back-Propagation Networks

In Feed-forward back-propagation networks, Elman Recurrent Network, the output of a neuron is either directly or indirectly fed back to its input via some other linked neurons as shown in Figure 4.

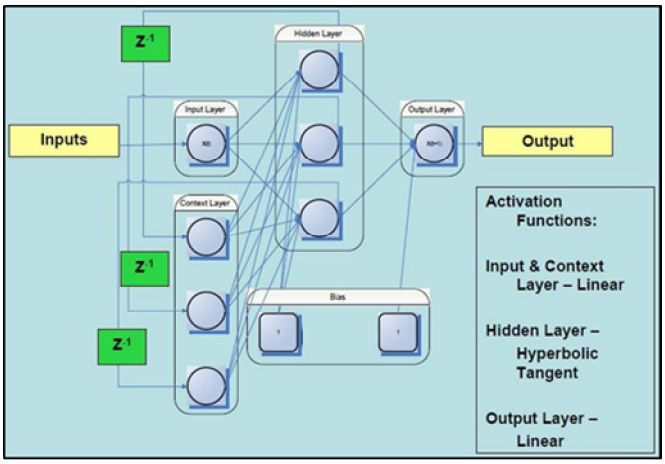

Figure 4: The Feed-forward back-propagation networks

A. *Learning Methods*

Artificial neural networks work through the optimized weight values, the method by which the optimized weight values are attained is called learning. During the learning process, we try to make the network learn how to produce a certain output when the corresponding input is given [4]. When learning is completed, the trained neural network, with updated optimal, weights must be able to produce the correct output corresponding to an input pattern, within a desired accuracy margin.

B. *Weight Adjustments and Updates*

There are two types of supervised learning algorithms that exist which is based on when or how weights are updated. Stochastic/delta/(online) learning: Where the neural network weights are adjusted after each input pattern. In this case, the next pattern to be input is randomly selected from the training set. This is to prevent any bias that may occur because of the sequences in which patterns are input in the training set. Batch (offline) learning: where changes in the neural network are accumulated and then used to adjust weights only at the end after all the training patterns have been input.

2.5 Feed-Forward Operation

Figure 5 shows Feed-forward and back-propagation operations [3, 4].

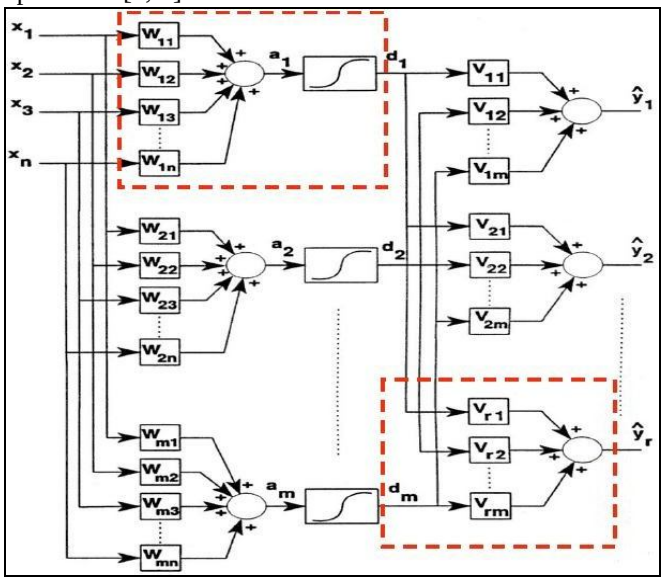

Figure 5: Feed-forward and back-propagation operations

Input weight matrix W_{ij} where i=1 $\rightarrow$ m (hidden neurons)

Step 1: Activation vector a_i : $a_i = \sum_{j=1}^{n} w_{ij} x_j \rightarrow \bar{a} = W \bar{X}$

Decision vector

$$d_i : d_i = sig(a_i), \ \ where \ \ sig(.) = \frac{1}{1+e^{-(.)}}$$

Step 2: Output vector y_i is given by (r is the number of outputs):

$$y_h = \sum_{i=1}^{m} v_{hi} d_i, \ \ h \in \{1,2,3,...,r\} \rightarrow \bar{y} = V \bar{d}$$

2.6 Back-Forward Operation

The back-propagation training algorithm is based on the principle of gradient descent and is given as half the square of the Euclidean norm of the output error vector

$$E(x,W,V,y)=\frac{1}{2}\left|\bar{e}_y\right|^2$$

Step 1: The output error vector $\bar{e}_y=\bar{y}-\bar{\hat{y}}$

Step 2: The decision error vector: $\bar{e}_d=V^T\bar{e}_y$,

The activation error vector: $\bar{e}_{ai}=d_i(1-d_i)\bar{e}_{di}$

Step 3: The weights changes:

$$\Delta V(k)=Y_g\ B_y(k)\,\bar{d}^T(k)+y_m\,\Delta V(k-1)$$

$$\Delta W(k)=Y_g\,\bar{e}_g(k)\,\bar{x}^T(k)+Y_m\Delta W(k-1)$$

g_g And g_m are learning and momentum rates, respectively.

The weight updates:

$$V(k+1)=V(k)+\Delta V(k)$$
$$W(k+1)=W(k)+\Delta W(k)$$

An epoch is one set of weight modifications, and many of these sets may be required before the desired level of accuracy of the approximation can be reached.

3. CRYPTOGRAPHIC BACKGROUND

In this section, we will describe and evaluate the efficiency of the most popular cryptographic primitives, both symmetric techniques, and asymmetric public-key techniques [5, 11,12,13,14, 16, 17].

This section is not intended as a rigorous and detailed explanation of the cryptographic primitives, but rather to help understand the differences in performance between these primitives. We will not define any Notation that is required to completely analyze any security aspects.

For example, there will use the terms "easy", "hard", "infeasible", etc. without mathematical definitions to describe the exact meaning of these terms. There are two primary types of cryptographic algorithms: symmetric key algorithms, which use the same key for encryption and decryption, and asymmetric public-key algorithms, which use two different keys for encryption and decryption.

In the following sections, these two algorithms will be discussed in addition to digital signature, digital certificate, Public Key Infrastructure (PKI), and Web of Trust (WoT) models.

3.1 Asymmetric key algorithms [6, 10]

The problems of key management in symmetric key algorithms [6] are solved by public-key cryptography (asymmetric key). This concept was introduced by Whitfield Diffie and Martin Hellman in 1976.

Public-key cryptography is a form of cryptography where the user has a pair of cryptographic keys, a public-key and a private-key. The private-key is kept secret, while the public-key may be distributed publicly. These keys are related mathematically, however, the private-key cannot practically be derived from the public-key. A message that is encrypted with the public-key can only be decrypted with the corresponding private-key.

The asymmetric public-key encryption scheme is illustrated in Figure 6. At the beginning, both Alice and Bob should have a pair of public and private-keys. In the case that Alice wants to send an encrypted message m to Bob, she first needs to get Bob's public-key (PK) and make sure that this key is authenticated. This public-key is used to encrypt the message m and convert it into cipher text c. Bob can then decrypts this cipher text using the corresponding private-key (SK) which is known only by him.

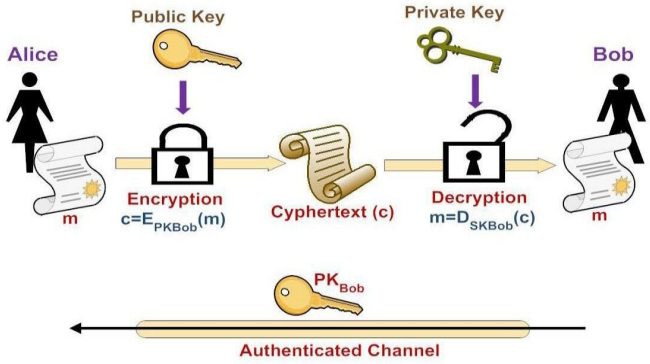

Figure 6: Public key encryption scheme

A popular problem with public-key cryptography is to proof that the public-key is authentic, and has not been replaced by an adversary or a malicious third party. The usual solution to this problem is to use the public-key infrastructure (PKI), where a trussed third party, known as Certificate Authority, certifies the ownership of public and private-key pair.

A very popular example of public-key cryptography is the RSA algorithm developed by Rivest, Shamir, and Adleman. This method is based around the integer factorization

problem. In RSA, to encrypt a message m or decrypt a cipher text c, the following calculations need to be performed:

$$c = m^e \bmod n$$
$$m = c^d \bmod n$$

A major benefit of this method is that it provides a tool for employing a digital signature. Digital signatures allow the recipient to verify the authenticity of the origin of the received information, and also that the information received has not been altered during transmission. Thus, public-key digital signatures provide authentication and data integrity.

The digital signature also provides non-repudiation. This means that it prevents the sender from denying the sending of this information. The basic manner in which digital signatures are created is illustrated in Figure 7. If the information is decrypted with the sender's public-key, then this is a proof that it must have originated from that sender.

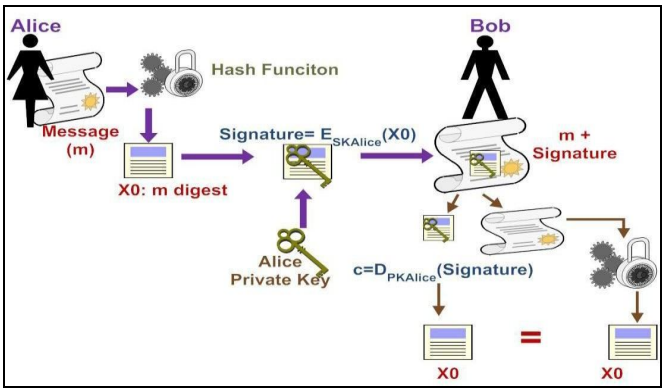

Figure 7: Digital signature

As can be seen in Figure 7, Alice wants to send a message m to Bob which is signed by her. Alice uses the hash digest of the message m and her private-key to create the signature. First, she uses a hash function on the message m and computes the hash digest. Then, she encrypts this digest using her private-key (SKAlice) and sends it with the message to Bob.

Bob after computing the digest on his side, by using exactly the same hash function applied to the message before transmission and applying it to the received message m and compares it with the message digest that resulted from decrypting the signature using the public-key of Alice (PKAlice). If both digests are the same, then this is a proof that the message m must have originated from Alice and also the message has not been modified by an attacker during transmission [18].

3.2 ECC algorithm

Elliptic Curve Cryptography (ECC) was proposed by Neal Koblitz and Victor Miller in 1985 [7, 8, and 9]. This method can be viewed as an Elliptic curve replica of the older discrete logarithm cryptosystem.

The mathematical basis of the security of the elliptic curve cryptography is the computational difficulty of the elliptic curve discrete logarithm problem. Elliptic curve digital signature algorithm is an alternative to the established public-key system such as the digital signature algorithm and RSA algorithm. This algorithm, which is one of the variants of the proposed elliptic curve cryptographic (ECC), have recently recognized as one of the important algorithms and gained a lot of attention in academia and many applications have been proposed for utilizing it in industry.

Some of the key concepts in ECC are: For a given key size, ECC offers considerably greater security. The fact of having smaller key size makes it much more compacted implementations for a level of security.

ECC is better suited for small devices like lot devices and sensors used to Wireless Sensor Networks that have very limited resources in terms of memory and processing power.

3.3 Benefits of Elliptic Curve Cryptography (ECC)

Elliptical curve cryptography is also a public-key encryption technique but rather based on the elliptic curve theory [14, 19]. Thus ECC can be used to create smaller, faster, and more efficient cryptographic keys. ECC generates keys by utilizing the inherit properties of the elliptic curve equations instead of using the legacy method of key generation as the result of the product of very large prime numbers as in RSA.

ECC allows the establishment of equivalent security with much lower computing resources and battery usage. This is the reason why it is becoming widely used for light mobile applications. Unlike other legacy algorithms such as RSA, ECC is based on several discrete logarithms that are much more difficult to be challenged at equivalent key lengths.

ECC has many advantages over RSA. ECC devices need much less storage space, less power consumption, less memory, and less bandwidth than the RSA system. This allows for the implementation of a cryptography system on constrained platforms and thin-clients, such as handheld computers, smart devices, wireless sensors and smart cards.

It is also very useful in situations where efficiency is of ultimate importance. For example, the key size currently recommended for RSA schemes is 2048 bits.

A much smaller ECC key of only 224 bits offers the same security level. Another example, 3072 bits RSA key and 256 bits ECC key are equivalent in the security level. This will be

more important as stronger security systems become mandatory while in the same time devices are getting smaller.

4. SIMULATION RESULTS [14, 15]

In this section, we will provide a numerical example and apply and compare hill cipher and trained FF-ANN to encrypt and decrypt messages.

4.1 Using hill cipher

We use hill cipher to encrypt and decrypt the following simple message algebraically. Let our message be [Direct year] which consists of 10 letters, we shall now encrypt this message by the following steps:

Step 1: construct the table of the message using hill cipher mode 26 [10, 11]

D	I	R	E	C	T	Y	E	A	R
3	8	17	4	2	19	24	4	0	17

Step 2: divide the original letter into blocks, every block contains 2 letters (2 * 2) and put

These blocks in the matrix P $\rightarrow P = \begin{pmatrix} 3\,17 & 2\,24 & 0 \\ 8 & 4\,19 & 4\,17 \end{pmatrix}$

Step 3: Choose the encryption matrix E (public key) [12] to have an inverse and can be multiplied by P

Let E= $\begin{pmatrix} 3 & 6 \\ 1 & 5 \end{pmatrix}$

Step 4: get the matrix Q the result of multiplication of (E * P) and then transfer it to mode 26

$$\rightarrow Q = \begin{pmatrix} 57 & 75 & 120 & 96 & 102 \\ 43 & 37 & 97 & 44 & 85 \end{pmatrix} \rightarrow Q = \begin{pmatrix} 5 & 23 & 16 & 18 & 24 \\ 17 & 11 & 19 & 18 & 7 \end{pmatrix}_{\mathrm{mod}\, e26}$$

Step 5: reverse the matrix Q into blocks ($2*2$) to get the encrypted message

5	17	23	11	16	19	18	18	24	7
F	R	X	L	Q	T	S	S	Y	H

So the encrypted message Q is (FRXLQTSSYH)

Step 6: start the decryption process, get the inverse of the matrix E call it E^{-1} and multiply it by the matrix Q so the result is matrix called D where $D = (E^{-1} * Q)_{\mathrm{mod}\, e26}$

$$E^{-1} = (\frac{1}{|E|} * adjo\,\mathrm{int}\, E)_{\mathrm{mod}\, e26} =$$

$$3 * \begin{pmatrix} 5 & -6 \\ -1 & 3 \end{pmatrix}_{\mathrm{mod}\, e26} = \begin{pmatrix} 15 & -18 \\ -3 & 9 \end{pmatrix}_{\mathrm{mod}\, e26} = \begin{pmatrix} 15 & 8 \\ 23 & 9 \end{pmatrix}$$

$$\therefore D = \begin{pmatrix} 15 & 8 \\ 23 & 9 \end{pmatrix} * \begin{pmatrix} 5 & 23 & 16 & 18 & 24 \\ 17 & 11 & 19 & 18 & 7 \end{pmatrix}_{\mathrm{mod}\, e26} = \begin{pmatrix} 3 & 17 & 2 & 24 & 0 \\ 8 & 4 & 19 & 4 & 17 \end{pmatrix}$$

Step 7: transfer matrix D into blocks ($2*2$) to get the decrypted message

3	8	17	4	2	19	24	4	0	17
D	I	R	E	C	T	Y	E	A	R

P= Direct Year

Now we can imagine if the number of letters increases to be 20 or more and we still use hill cipher (mode 26) or any mode in the asymmetric algorithms, the calculations will be more difficult so we shall try how we can decrypt the previous example by using the trained FF-ANN.

4.2 Using FF-ANN algorithm to decrypt messages [17]

Step 1: start training our neural network by using 15 message each consists of 10 letters.
Step 2: transform the letters to normalized values for both the encrypted and decrypted messages.
Step 3: our network consists of three layers the input layer which contains 10 neurons and one hidden layer and the output layer. We start to train and show the obtained data in Figures 8, 9 and 10.

```
>> inputs

inputs =

  Columns 1 through 11

    0.0385    0.3077    0.4231    0.3077    0.1538    0.2692    0.6923    0.4615    0.4231    0.5000    0.2308
    0.1538    0.4231         0    0.4615    0.6923         0    0.7692    0.1538    0.5385    0.3077    0.5385
         0    0.5385    0.4615    0.5000    0.6538    0.7308    0.5769    0.6538    0.8077    0.0769    0.8077
    0.7692    0.8077    0.3077    0.5384         0    0.1538    0.1538         0    0.3077    0.1538    0.1538
    0.7308    0.1538         0    0.6538         0    0.7308    0.6538    0.4231    0.5000    0.3846    0.6538
    0.9231    0.3077         0    0.7308    0.6923    0.7692    0.4615         0    0.2308    0.2692    0.5000
    0.1538         0    0.2692         0         0    0.6538         0    0.1154    0.8462         0    0.4615
    0.4615    0.4615    0.4615    0.5000    0.1538    0.3846    0.6538    0.2692    0.1538    0.4231    0.1538
    0.0385         0    0.1538    0.0769    0.1538    0.1538    0.3846         0    0.4231    0.1538    0.5000
    0.1154    0.2692    0.1154    0.1538    0.1154    0.9231    0.7308    0.4615    0.4231    0.1154    0.7308

  Columns 12 through 15

    0.0385    0.6923    0.0769    0.9231
    0.4231    0.7308         0    0.1538
         0    0.6538    0.4231    0.4231
    0.0769         0    0.0769    0.4231
    0.3846    0.8462    0.7692    0.5385
    0.2692    0.0385    0.4231    0.8462
    0.5385    0.1538         0    0.0385
    0.6538    0.6538    0.7308    0.3077
    0.6923    0.6538    0.5385    0.6538
    0.1538    0.9200    0.6538    0.1154
```

Figure 8: The normalized inputs that are given to the network

```
>> outputs

outputs =

 Columns 1 through 11

   0.0385    0.4615    0.2692    0.7795    0.6154    0.8077    0.8188    0.3077    0.5000    0.3462    0.9231
   0.8077    0.4231    0.4231    0.6811    0.6154    0.2692    0.5669    0.2308    0.1154    0.0385    0.9231
   0.6154    0.4615    0.2308    0.9164    0.9615    0.1154    0.9243    0.9615    0.2692    0.1538    0.3462
   0.8462    0.5769         0    0.6525    0.6538    0.5000    0.5663    0.6538    0.3462    0.8462    0.5769
   0.7308    0.6154         0    0.9236    0.1538    0.8077    0.8757    0.2692    0.8846    0.7692    0.9615
   0.3462    0.6154         0    0.4891    0.4615    0.5769    0.1952    0.4231    0.6538    0.7308    0.1538
   0.2308    0.7692    0.5769    0.4943    0.9231    0.2692    0.6948    0.9615    0.4615    0.5385    0.3077
   0.4615    0.3077    0.5769    0.2204    0.7692    0.5769    0.1673    0.4615    0.6154    0.1154    0.2308
   0.1154    0.6154    0.1538    0.4335    0.1538         0    0.3892    0.7692    0.8077    0.1538    0.8846
   0.5385    0.3462    0.7308    0.3760    0.7308    0.7692    0.3611    0.3077    0.5385    0.7308    0.1538

 Columns 12 through 15

   0.6538    0.6933    0.2308    0.6923
   0.1538    0.2745    0.0769    0.6923
   0.4615    0.8426    0.7308    0.8077
   0.3846    0.3836    0.8077    0.5385
   0.7692    0.8866    0.8462    0.6923
   0.7308    0.1200    0.8846    0.7692
   0.5385    0.5232    0.3846    0.9615
   0.8077    0.2139    0.6538    0.5769
        0    0.3711    0.5385    0.6538
   0.4615    0.3579    0.8077    0.2308
```

Figure 9: The normalized outputs that are given to the network

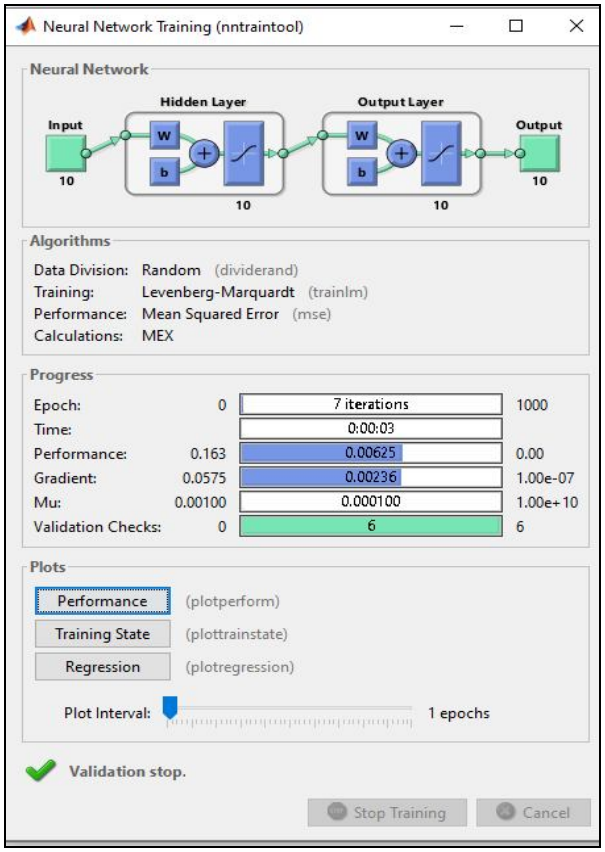

Figure 10: the statistics of error in our training

Step 4: We now make our test to decrypt several encrypted messages where one of them is the message given in the algebraic example (DIRECT YEAR).

Step 5: we see if the decryption is correct we stop.

Step 6: if the decryption is not completely right we make two corrections, the first correction is to in-crease the number of hidden layers and the second is increasing the number of messages stage by stage and see the result every time.

Step 8: We discover that when we use about 50 messages to train our FF-ANN we reach excellent results in a very small time (see test error curves in Figure 11.

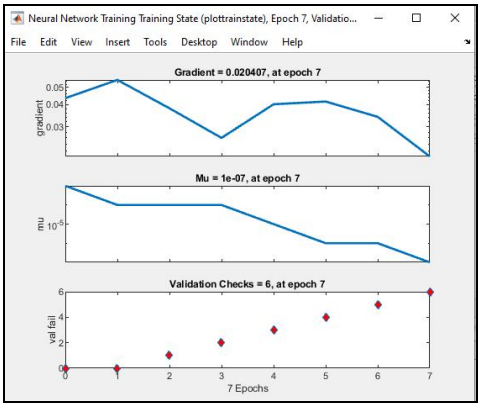

Figure 11: Test error curves

Step 9: Finally, we give the network three test patterns as shown in Figure 12 and the results are correct, one of these messages is (FRXLQTSSYH) which is encrypted we give its result in our trained FF-ANN which is (DIRECTYEAR) exactly correct.

```
>> network1([0.1154, 0.3077, 0.6538, 0.1538, 0.0769, 0.7308, 0.9231, 0.1538, 0, 0.6538]')

ans =

    0.8454
    0.2631
    0.4725
    0.7724
    0.9352
    0.6786
    0.3001
    0.6306
    0.4055
    0.7237
```

Figure 12: test pattern messages to our trained FF-ANN

5. EVALUATION AND DISCUSSION

In this section, we will provide our evaluation and discussion and give a comparison between the mathematical and the neural proposed method using FF-ANN. Table 1 gives a comparison between the mathematical and the proposed neural network method.

Table 1: A comparison between the mathematical and the proposed neural network method

	Mathematical Method	Proposed FF-ANN Method
1	The process of the solution may take some time especially when the message contains a big number of letters, small messages take about 10 minutes while medium messages (about 20 letters) take about 20-25 minutes.	The process of decryption takes a small-time, our numerical example takes about 1 minute and its benefit will be clear when the message contains a big number of letters.
2	The results of decryption are exactly true regardless of the number of letters inside the	The results of decryption depend upon the quality of training our FF-ANN and how we change the features of

	message.	our network, we get excellent results when we made our training with 50 messages.
3	We must get the result of decryption message by message, i.e.: we do not have a general rule to decrypt all messages of the same number of letters.	We can deduce a utility function for a certain type of message to make a general rule of decryption for this type.

REFERENCES

1. O.S. Eluyode, Dipo Theophilus Akomolafe, "***ComparativeStudy of Biological and Artificial Neural Networks*",** European Journal of Applied Engineering and Scientific Research, 2 (1), 2013, pp.36-46.
2. K.Y. Lee, Y. T. Cha, J. H. Park, "***Short Term Load Forecasting Using an Artificial Neural Network*",** Transactions on Power Systems, Vol 1, No 7, 1992, pp. 124-132.
   https://doi.org/10.1109/59.141695
3. S. Tayal, N. Gupta, P. Gupta, D. Goyal and M. Goyal**, "A Review paper on Network Security and Cryptography,**" Advances in Computational Sciences and Technology , vol. 10, no. 5, 2017, pp. 763-770.
4. N. Sharma , Prabhjot and H. Kaur, **"A Review of Information Security using Cryptography Technique,"** International Journal of Advanced Research in Computer Science, vol. 8, no. Special Issue, 2017, pp 323-326**.**
5. N. Jirwan, A. Singh and S. Vijay , **"Review and Analysis of Cryptography Techniques,"** International Journal of Scientific and Engineering Research, vol. 3, no. 4, 2013, pp. 1-6.
6. Lyubashevsky, V., seiler, G: NTTRU**: "truly fast NTRU using NTT.IACR"**, Trans, cryptogr. Hardw. Embed.Sys. 2019(3), pp. 180-201.
7. Carlos Gershenson, **"Artificial Neural Networks for Beginners"**, arxiv.org, 2003.
8. Emil M Petriu, Professor, University of Ottawa, **"Neural Networks Basics"**, 2017.
9. Vishwa gupta, Gajendra Singh and Ravindra Gupta **"Advance cryptography algorithm for improving data security",** 2014.
10. Ritu Tripathi and Sanjay Agrawal **"Comparative Study of Symmetric and Asymmetric Cryptography Techniques‖".** 2014.
11. Ali Makhmali, Hajar Mat Jani **"Comparative Study On Encryption Algorithms And Proposing A Data Management Structure‖".** 2013.
12. Ms.Pallavi H.Dixit, Dr.Uttam L. Bombale and Mr. Vinayak B. **"Comparative Implementation of Cryptographic Algorithms on ARM Platform ‖".** 2013.
13. Er. Satish Kumar, Amritsar Mr. Amit Puri **"Comparative analysis of various cryptographic algorithm‖",** 2012.
14. Dr. Jitendra Sheetlani and Harsh Gupta **"Comparative Study of a New Variable Length Key Block Cipher Technique with DES for Network Security ‖",** 2014.
15. Sheetal Charbathia and Sandeep Sharma **" A Comparative Study of Rivest Cipher Algorithms‖".** 2014.
16. Alese, B. K., Philemon E. D., Falaki, **"Comparative Analysis of Public-Key Encryption Schemes ‖".** 2012.
17. M. Abdelrahman **"*Artificial neural networks based* steady state security analysis of power systems",** Thirty-Sixth South eastern Symposium on System Theory 2004 Proceedings, 2004.
18. Abdelbasit Mohammed **"A Review Paper on Cryptography",** 7th international symposium on digital forencis and Security, 2019.
19. B. Preneel**," Understanding Cryptography":** A Textbook for Students and Practitioners, London: Springer, 2010.