Vol 3 Issue 3 Dec 2013

Impact Factor : 1.6772 (UIF)

ISSN No : 2249-894X

*Monthly Multidisciplinary Research Journal*

# *Review Of Research Journal*

## Chief Editors

**Welcome to Review Of Research**

**RNI MAHMUL/2011/38595** **ISSN No.2249-894X**

Review Of Research Journal is a multidisciplinary research journal, published monthly in English, Hindi & Marathi Language. All research papers submitted to the journal will be double - blind peer reviewed referred by members of the editorial Board readers will include investigator in universities, research institutes government and industry with research interest in the general subjects.

## *Advisory Board*



More.....

**Address:-Ashok Yakkaldevi 258/34, Raviwar Peth, Solapur - 413 005 Maharashtra, India**
**Cell : 9595 359 435, Ph No: 02172372010 Email: ayisrj@yahoo.in Website: www.isrj.net**

*Review Of Research*
Vol. 3, Issue. 3, Dec. 2013
**ISSN:-2249-894X**
Available online at www.ror.isrj.net

ORIGINAL ARTICLE

# DATA ENCRIPTION USING ARTIFICIAL NEURAL NETWORKS

**WALTER TAKASHI NAKAMURA , ROGÉRIO PEREIRA DOS SANTOS AND FABRÍCIO MORAES DE ALMEIDA**

Bachelor of Information Systems – FAAR, Brazil.
Bachelor of Information Systems (IESUR). MBA in Development of Internet Applications (IESUR), Brazil.
PhD in Physics (UFC). Program Researcher Doctoral and Master of Regional Development and Environment - PGDRA / UNIR, Brazil.

**Abstract:**

*Humans always had a need to communicate with each other. Privileged information needed to be hidden or coded so the communication could be accomplished without interception by third parties. This need arose the concept of data encryption. Today, with increased competition and the ease and speed with which information is transmitted, data encryption has become a priority for businesses and large corporations. More than ever the information has immeasurable value. Companies spend millions to obtain privileged information, just as invest millions in security to keep them in their possession. This unceasing search for information that can provide competitive advantages makes it necessary to develop more robust methods to ensure information security, including the use of Artificial Neural Networks, one of the areas of Artificial Intelligence.*

**KEY WORDS:**

Data Encryption, Information Security, Neural Network, Artificial Intelligence.

## INTRODUCTION

Data encryption aims to make it impossible to get a code and reproducing the original text without the corresponding key, using extremely large keys that prevents the use of brute force to decrypt the encrypted data (VOLNA et al., 2013), avoiding unauthorized persons from gaining access to certain information. The use and sharing of these keys for encryption and decryption is one of the major vulnerabilities that currently exist in data encryption. Recent studies by VOLNA et al. (2013) and RUTTOR (2006) show that the Artificial Neural Networks (ANNs) provide a new dimension in the development of systems for information security.

## 2. ENCRYPTION AND SYMMETRIC

Treat about encryption, symmetry and neural networks below.

### 2.1 Symmetric key encryption

Also called private key encryption, is the use of a single key which is shared between the two parties to perform communication. The sender performs the encryption of a given text 'T', using the shared key 'K', generating a ciphertext 'C', which is transmitted to the recipient (NOAMAN and JALAB, 2005):

$C = encode(T, K)$ (1)

**Title: DATA ENCRIPTION USING ARTIFICIAL NEURAL NETWORKS Source: Review of Research [2249-894X] WALTER TAKASHI NAKAMURA , ROGÉRIO PEREIRA DOS SANTOS AND FABRÍCIO MORAES DE ALMEIDA yr:2013 vol:3 iss:3**

The recipient receives the ciphertext (C) and, by the same key (K) used in the encoding process, do the reverse procedure, decoding it:

T = decode(C, K) (2)

The major issue in this methodology is the use of a single key for encryption and decryption processes, which makes it necessary to pre-share these keys before starting an encrypted channel (OLIVEIRA, 2013). If the key is discovered by third parties, the communication between the two parties will be compromised, necessitating periodic replacement of this key to mitigate this vulnerability. However, due to the simplicity of its algorithm, the runtime routines for encryption and decryption is smaller, and facilitate the implementation process, which can be attractive for use in systems with low processing power.

### 2.2 Asymmetric key encryption

The asymmetric key cryptography (or public key cryptography) is the use of a couple of different keys: a public key and private key. The public key is freely available from the network, allowing anyone to encrypt the information using it. However, this information can only be decrypted by using the corresponding private key, which is held by only one of the parts (PIAZENTIN and DUARTE, 2013).

This type of encryption is much more secure, since there is no sharing of the keys, in case the private key. Another advantage is the possibility to ensure the authenticity of a particular file (non-repudiation). If the user "X" wants to ensure its authenticity, it can encrypt the information with his private key and send it to the user "Y". User "Y" will only be able to read the message if he uses the Public Key of "X" and the message has been encrypted using the private key of "X", indicating that the user was actually "X" who wrote the message (SHIHAB, 2006 .)

Although ensures greater security, the complexity of their algorithm makes the encoding and decoding processes much more slow, as it will have to recognize the two keys and relate them at the right time to process the data (OLIVEIRA, 2013) by restricting its use to certain applications. To solve this problem, usually asymmetric encryptions are used for sending the keys and ensure the authenticity of the parties involved and symmetric ciphers for encoding and sending data.

### 2.3 Artificial Neural Networks (ANNs)

One of the features of the ANNs is their ability to learn from the environment in which they are inserted and improve performance after successive iterations by modifying the parameters, such as the synaptic weights and bias levels (HAYKIN, 2001).

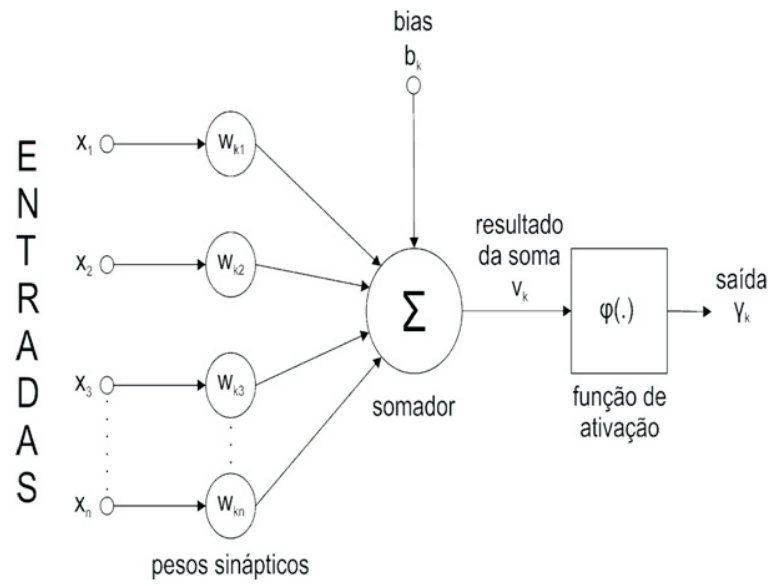


**Figure 1 - Model of an Artificial Neuron. Source (adapted from HAYKIN, 2001, p. 33).**

A neuron consists of a set of connections called synapses, an input signal adder, a bias and an activation function, which is responsible for limiting the output signal of the neuron (Figure 1). Each neuron is connected to other, thus forming an Artificial Neural Network.

### 2.4 Learning process

There are two distinct types of ANNs learning processes: supervised and unsupervised.

In supervised learning, the ANN is trained by a "teacher" who has knowledge about the environment in which it is located, providing vectors containing input data and the desired output. The ANN then compares the output value to the output desired by the "teacher" and modifies its parameters based on the error generated (HAYKIN, 2001).

In unsupervised learning, the ANN learns the patterns without the presence of a "teacher", using only the input data provided. An agent of a traffic monitoring system of a taxi might, for example, learn what are the best and worst days of movement without ever having been provided an example by a "teacher" (RUSSEL and NORVIG, 2010).

### 2.5 Learning by error correction

The process of learning by error correction consists of adjusting the weights of the connections according to the calculated error in the output of the ANN.

Assuming a neuron () as in Figure 1, the learning process begins with the provision of data in the input layer (). These values are multiplied by the corresponding weights (), which are initialized with random values. The result of the multiplication of all entries is added along with the bias () via an adder. The result of this sum () passes through an activation function ( ), which defines the result (output) of this process (HAYKIN, 2001).

The result obtained is compared with a value supplied by a supervisor (supervised learning), setting the weights of synapses in accordance with the error, until the desired output is obtained.
The error () is calculated as the difference between the output () of neuron () and the desired value () in an iteration , according to the formula:

### 2.5 Backpropagation algorithm

The Backpropagation is a training algorithm of multilayer perceptron networks (MLP), which through supervised training uses pairs of input and output data for error checking and correction of synapses weights of each layer of neurons (PADUA, 2011).

The Backpropagation works in two phases, called forward phase and backward phase (Figure 2). In step forward, a signal is supplied to the neurons of the input layer. The output of this first layer is calculated and propagated to the neurons of the hidden layer, until get to the neurons of the output layer. The output obtained is compared with the desired result, and if the result is not satisfactory, occurs the opposite direction (backward phase), where the signal is propagated back, making the adjustment of the weights multiplying them with error rates thereof. This process continues iteratively until the desired value is obtained.

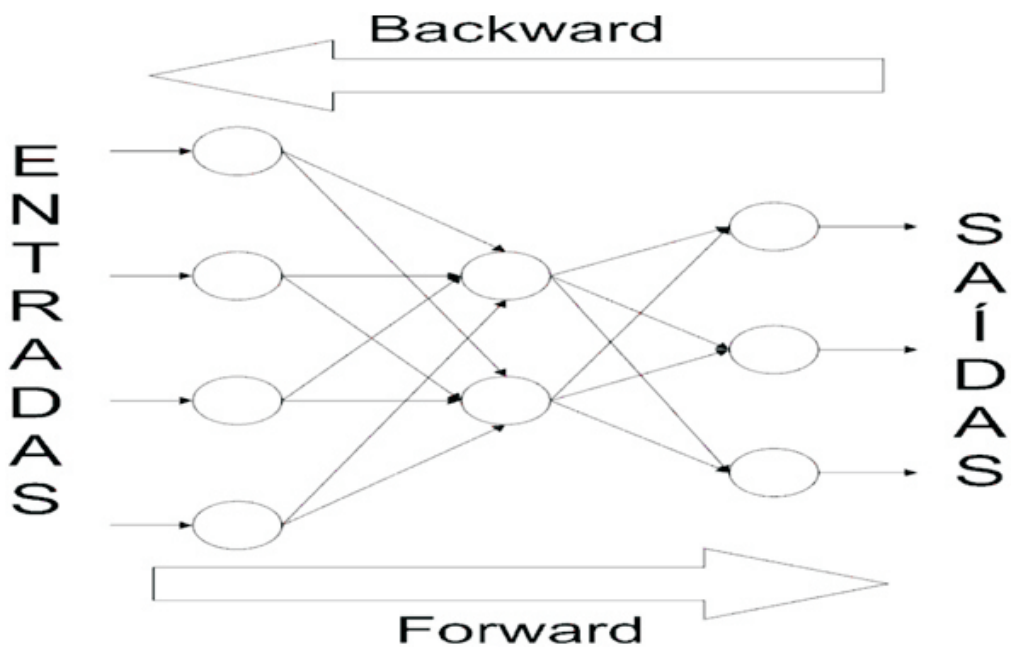


**Figure 2 - Model of the Backpropagation Algorithm.**

**2.6 Data encryption using Neural Networks**

A major advantage of the ANNs is the ability to identify patterns that have not been previously reported. To verify this capability was used Encog framework (RESEARCH HEATON, 2013).

We created a neural network consisting of three layers (input, hidden and output), each containing three neurons. This number of neurons was determined according to the standard that will be provided, in case a binary value of 3-bits, and the output pattern, also 3-bits. The backpropagation algorithm and the sigmoidal activation function was chosen, as the expected results are only positive values.
The values were provided as shown below:

| Input 1 | Input 2 | Input 3 | Ideal 1 | Ideal 2 | Ideal 3 | Significance |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |

**Table 1 – Values used in ANN training.**

The input values are variants of 3-bit binary numbers, reported in the field "Input". "Ideal" field values are the desired outputs for standard informed input fields. In this test the optimal values are the input values reversed.

Training was performed with a learning rate of 0.3, "momentum" 0 and a maximum error rate of 0.01% to obtain more accurate values.

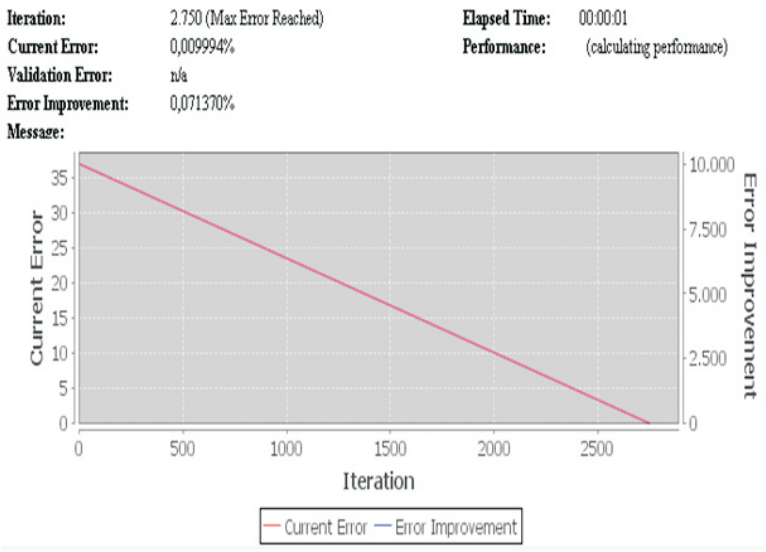


**Figure 4 - Graphic generated after processing the training data by the ANN.**

After 2,750 iterations with the given training pattern, the ANN could reach the maximum error rate of 0.009% in only 1 second (Figure 3).

The binary pattern 100 was given in ANN. This generated an output very close to the ideal value (011), as shown in table 2:

| Input | | Output | |
|---|---|---|---|
| Input 1: | 1 | Output 1: | 0.0128577231360... |
| Input 2: | 0 | Output 2: | 0.9915695988976... |
| Input 3: | 0 | Output 3: | 0.9997163519708... |

**Table 2 – Results of the test of ANN providing as input the binary pattern 100.**

Another test was performed, providing as input a pattern not previously reported for the ANN (101) during training. Even without knowledge of this standard, the ANN was able to perform the processing, resulting in the desired output (010), as follows:

| Input | | Output | |
|---|---|---|---|
| Input 1: | 1 | Output 1: | 0.0122387577786... |
| Input 2: | 0 | Output 2: | 0.9768195888977... |
| Input 3: | 1 | Output 3: | 0.0819583355082... |

**Table 3 – Results of the test of ANN providing as input the binary pattern 101.**

Increasing the learning rate to 1, the ANN learned the patterns in just 631 iterations, taking less than 1 second to complete the operation. Using these same rates and increasing the number of neurons in the hidden layer to 6, also resulted in a lower number of iterations (351) for this type of operation, without affecting the quality of the result (Figure 4), being possible to see even an improvement in identification of the unknown pattern (Table 4).

| Input | | Output | |
|---|---|---|---|
| Input 1: | 1 | Output 1: | 0.0136447093061... |
| Input 2: | 0 | Output 2: | 0.9228216109509... |
| Input 3: | 1 | Output 3: | 0.020526719353 5... |

**Table 4 – Results of the test after changing the parameters of the ANN.**

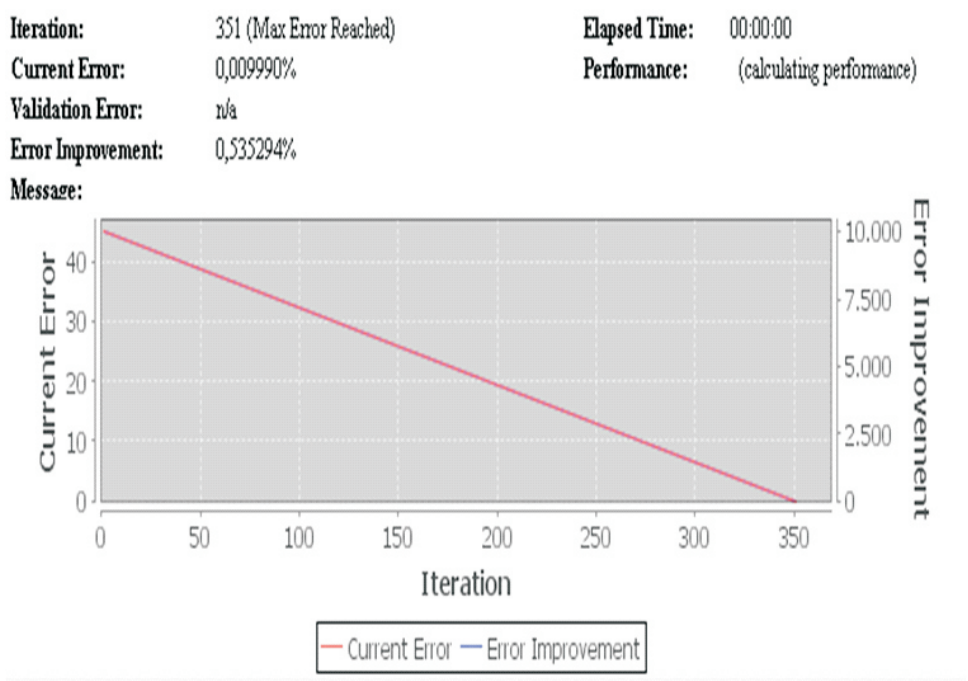


**Figure 5 – New graphic result of the ANN training after change of its parameters.**

Although the processing speed and identification of unknown patterns, the greatest advantage of the use of ANNs in the process of data encryption is its ability to synchronize with other ANNs, which enables the creation of different keys for each connection.

The neural synchronization is made through a type of multi-layered ANN called TPM (Tree Parity Machine), composed by an input layer containing neurons, one hidden layer formed by neurons and an output layer containing only one neuron (PIAZENTIN and DUARTE, 2013).

The neural synchronization process consists of two ANNs (A and B) of TPM type that are initialized with random weights of discrete values which vary in a range between -L and +L. At each iteration are supplied input binary data (+1 and -1) common to both. The sum of all inputs multiplied by the respective weight of the hidden layer neuron is done and the sign function (sgn) is applied to this result. If

the resulting value of the sum is positive, the neuron generates an output +1, indicating that it is active, otherwise, if a value less than or equal to zero, outputs a negative (-1), indicating that it is inactive.

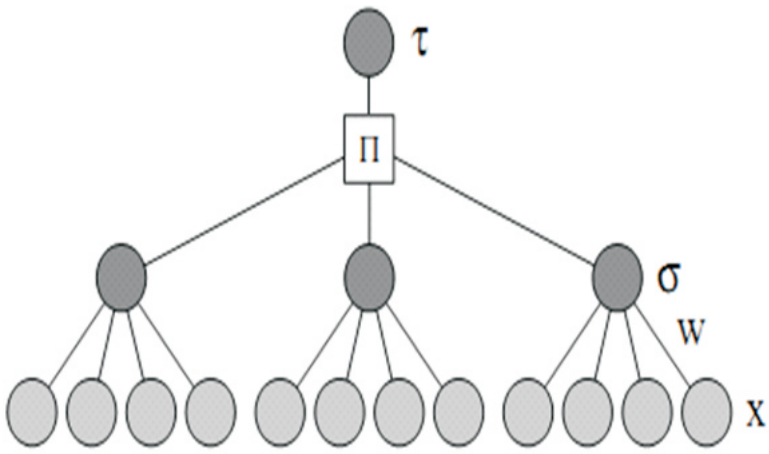


**Figure 6 - Model of a TPM type ANN. Source: RUTTOR, 2006, p. 14.**

The total output is then calculated as the product of all outputs of the hidden layer neurons, represented by . The neuron sends its output to the other, making the necessary adjustments to the synaptic weights, until both begin to produce the same values (RUTTOR, 2006), ie are synchronized. The synaptic weight vector used for the synchronization process to occur will result in the cryptographic key that will be used by the ANNs.

The weights are adjusted according to the output of each neuron in the hidden layer and the total output of the network. If the total output , no change is made. Weights are adjusted only when they satisfy the condition , so can apply one of the following rules (RUTTOR, 2006): Hebbian rule; Anti-Hebbian rule; Random-walk Learning Rule and where Θ is the step function.

To simulate this synchronization process, we used a prototype developed in Delphi by a user named Alexander Popovsky, aka "Cybertrone" (POPOVSKY, 2009).

In the program are specified the number of neurons in the input layer, the number of neurons in the hidden layer and the range of weights which ranges from a negative number to its opposite positive.

The program consists in initializing two TPM type ANNs with random weights using the function 'RandomWeight'. At each iteration is created a vector of size N x K (product of number of neurons of the input layer by the output) of random values by the 'FormRandomVector' function. The value of each entry (of each network) is calculated by the function 'CountResult', which performs the multiplication of weights of the neurons in the hidden layer by the input vector. If it's a value greater than zero, it returns as a result '1', otherwise it returns the value '-1'. This result is then compared with the other ANN. If the result is the same, the function 'UpdateWeight', responsible for making the adjustment of synaptic weights is triggered.

At each iteration a point is plotted in the graph, according to the result obtained in the comparison of the output of the ANN. If the result is equal, then a point is drawn toward baseline "Equal", otherwise a dot is drawn in the opposite direction, toward the line "Not equal", also moving horizontally at each iteration.

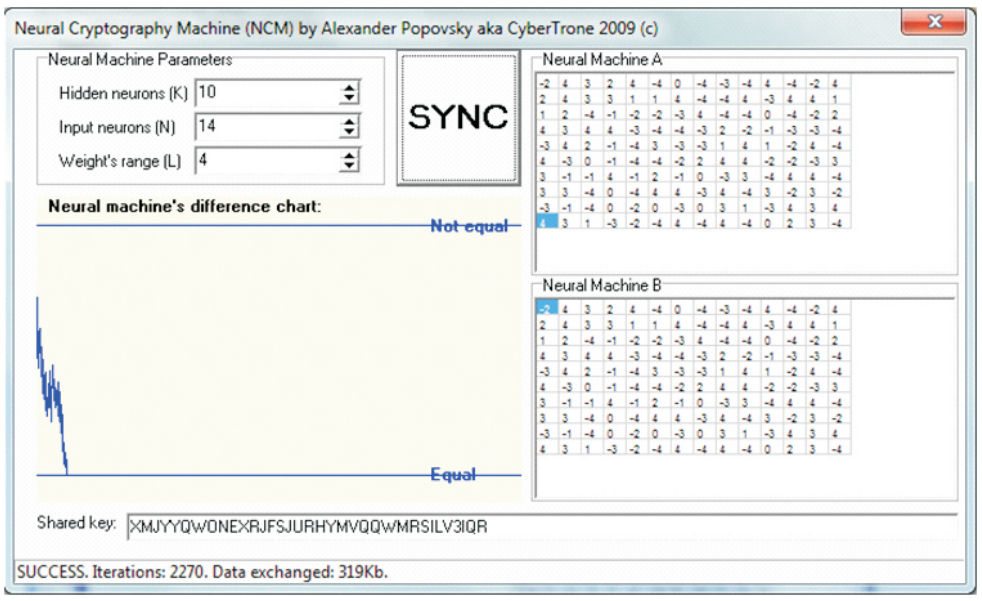


**Figure 7 - Simulation of ANNs synchronization in Neural Cryptography Machine software.**

When running the program, a table is generated from the two ANNs, forming a matrix of size N x K. At each iteration the data from both tables are updated and the graph corresponding to the result of the comparison of the outputs is plotted. The process ends when the two ANNs are able to produce exactly the same outputs.

With the synchronization completed, the encryption key is calculated by the weight vectors used during the iterations that resulted in the same output. This process of constructing the cryptographic key can be done in several ways. In this example, first the variable key_size is set, which is obtained by dividing the number of characters available in the variable ABC for the construction of the key (26 letters from A to Z, 01 underline and 10 numbers from 0 to 9, a total of 37 characters) by the weight value specified at the beginning of the program (in this case '4 '), plus 1, resulting in the value 4.

With the value obtained, the variable key_length is defined, performing the multiplication of the number of neurons in the input layer by the hidden layer (14 x 10 = 140), divided by the key_size (4), obtaining the value 35, which will be the length (number of characters) of the encryption key.

The system initializes a variable k with 1 as value and then enters in a loop (for), from 1 to 35. Within this loop is performed another repetition, by the number of times of key_size's variable value. At each iteration of this internal loop a calculation is done as follows:

Where 'A' is the first ANN, 'W[j]' is the vector of weights used which resulted in equal outputs at the two ANNs and 'L' is the weight set at the start of the operation. Performing calculations using the values of the matrix of Figure 7, we have: 1st iteration; 2nd iteration; 3rd iteration and 4th iteration.

Replacing 'k' in variable 'ABC' of characters, we have the letter 'X', located at position 24 of this vector. This process continues until the 35-character encryption key be set.

This approach of ANNs synchronization enables a robust and secure communication, since only the input vectors and the total output of the network are made public. The internal parameters of each network, for example, the weight of the neurons of the hidden layers remains secret, difficulting an attempt to attack, since the calculation for the update of synaptic weights depends on these parameters. The attacker would need to guess these values so that the weights would be changed correctly (RUTTOR, 2006).

As the ANNs interrupts the synchronization process when this is reached, the probability of an attacker can synchronize his network C in the middle of a communication between two networks A and B decreases as the iterations are being held between them, mainly because requires a much greater processing, as he will need to intercept the data, analyze the outputs of the two networks and thus try to identify the weights used internally by them.

## 3. CONCLUSION

The use of ANNs for the development of secure cryptographic algorithms is a recent approach.

However, the results and analysis show that the methodology is a promising technology capable of providing robust security compared to traditional encryption methods.

The absence of a cryptographic key to the realization of the communication and the use of random values, without a pre-established pattern, is one of the biggest attractions of ANNs, making it difficult to carry out an attack, even with a possible interception of data.

To make this approach more robust, it is possible to apply, for example, a symmetric-key algorithm encryption such as AES (Advanced Encryption Standard) at a level of 128-bit encryption, in the key generated in the synchronization process, increasing the difficulty in carrying out an attack attempt.

Based on studies, the use of Artificial Intelligence in data encryption through the ANN proved to be very efficient. These that were heavily criticized at the time of its conception, today, more mature, have shown great potential for solving many types of problems, which is extremely important to conduct research for the continuous improvement of this promising technology.

## 4. REFERENCES

BRAGA, Antônio de Pádua; CARVALHO, André Ponce de Leon F. de; LUDERMIR, Teresa Bernarda. Redes Neurais Artificiais: teoria e aplicações. 2nd. ed. Rio de Janeiro: LTC, 2011.

FILHO, José Macêdo Firmino et al. SISMA – Sistema de Monitoramento e Auditoria HOSPITALAR utilizando criptografia neural. Available at: <http://www.sigaa.ufrn.br/sigaa/verProducao?idProducao=829182&key=1b797f53cac719a4ea109c2beafc3d37>. Accessed at: 01 july 2013.

HAYKIN, Simon. Redes Neurais: princípios e prática. 2nd. ed. Porto Alegre, RS:Bookman, 2001.

HEATON RESEARCH. Encog Machine Learning Framework. Version 3.2.0-beta2. [S.l]:Reaton Research, 2013.

NOAMAN, KHALED M. G.; JALAB, HAMID ABDULLAH. Data Security Based on Neural Networks. Task Quarterly, Yemen, v. 9, n. 44, p. 409-414, may 2005. Available at: <http://www.task.gda.pl/files/quart/TQ2005/04/TQ409J-E.PDF>. Accessed at: 05 jun. 2013.

OLIVEIRA, Ronielton Rezende. Criptografia simétrica e assimétrica: os principais algoritmos de cifragem. Available at: <www.ronielton.eti.br/ publicacoes/artigorevistasegurancadigital2012.pdf?>. Acessed at: 13 june 2013.

PIAZENTIN, Denis R. M.; DUARTE, Maurício. Troca de chaves criptográficas com redes neurais artificiais. Available at: <www.peotta.com/sbseg2011/resources/ downloads/wticg/92045.pdf?>. Accessed at: 10 june 2013.

POPOVSKY, Alexander. Neural Cryptography. Code Project, 2009. Available at: <http://www.codeproject.com/Articles/39067/Neural-Cryptography>. Accessed at: 01 july 2013.

RUSSEL, Stuart; NORVIG, PETER. Artificial Intelligence: a modern approach. 2rd ed. New Jersey: Pearson Education, 2010.

RUTTOR, Andreas. Neural synchronization and cryptography. 2006. Dissertation (Doctorate) – University of Würzburgo, Germany. Available at: <http://opus.bibliothek.uni -wuerzburg.de/volltexte/2007/2361/pdf/dissertation.pdf>. Accessed at: 10 june 2013.

SHIHAB, Khalil. A cryptographic scheme based on neural networks. 10th WSEAS International Conference on Communications, Greece, p. 7-12, july 2006. Available at: <http://www.wseas.us/e-library/conferences/2006cscc/papers/534-959.pdf>. Accessed at: 05 june 2013.

VOLNA, Eva et al. Cryptography based on neural network. Available at: <http://www.scs-europe.net/conf/ecms2012/ecms2012%20accepted%20papers/ is_ECMS_0113.pdf>. Accessed at: 05 june 2013.