Method of Training and Implementation on the Basis of Neural Networks of Cryptographic Data Protection

Ivan Tsmots [1], Vasyl Teslyuk [1], Yurii Lukashchuk [1] and Yurii Opotiak [1]

[1] *Lviv Polytechnic National University, 12 Bandery Str, Lviv, 79013, Ukraine*

Abstract

The analysis of publications is carried out, which indicates the urgency of the problem of cryptographic encryption and decryption of real-time data for mobile robotic systems. The method of singular matrix decomposition has been improved, which, in contrast to the principal components method, provides non-iterative learning and calculation of the matrix of weights. This choice was made in order to be able to work with an input matrix of arbitrary dimension, because the principal components method assumes that the input matrix is square. The program which provides performance of process of training and calculation of a matrix of weights for the set architecture of a neural network is developed. Visual Studio 2019 environment and C# programming language were used to develop the user interface. A method for determining the parameters of the neural network architecture for data encryption/decryption is proposed, which provides a choice of the number of neuroelements and the number of inputs by taking into account the bit size of the message and the bit size of the inputs. Implemented data encryption/decryption software for sixteen-bit message with two-bit inputs on the basis of the SoC Allwinner H2+. It is determined that encryption and decryption of data using the developed tools is performed in near real-time.

Keywords

cryptographic data protection, neural network architecture, neuroelements, matrix transformations, Jacobi rotation method, data encryption, software implementation.

1. Introduction

When transmitting important information and remotely controlling mobile robotic systems by unmanned aerial vehicles and various mobile transport systems, an important task is to ensure cryptographic protection of data transmission [1-3]. Solving this problem requires the development of new methods and algorithms for cryptographic protection, focused on effective real-time software and hardware implementation with restrictions on size, power consumption and cost [4-6]. One way to meet such requirements is to use an auto-associative neural network of direct propagation, which is trained on the basis of the principal components method. A feature of such neural networks is the ability to pre-calculate weights and use them to implement cryptographic protections. An auto-associative neural network with pre-calculated weights is a neural network. To implement the task of cryptographic data protection, it is proposed to use a neural network with encryption with symmetric keys. When implementing symmetric crypto means, the encryption key and the decryption key are the same or the decryption key is easily calculated from the encryption key. Neuro-like encryption occurs over plaintext using a key that is determined by neural network parameters (number of neurons, number of inputs and their bit size), a matrix of calculated weights, and masking operations. The main

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland
EMAIL: ivan.tsmots@gmail.com (I. Tsmots); vasyl.m.teslyuk@lpnu.ua (V. Teslyuk); urijlukas@gmail.com (Yu. Lukashchuk); yurii.v.opotiak@lpnu.ua (Yu. Opotiak)
ORCID: 0000-0002-4033-8618 (I. Tsmots); 0000-0002-5974-9310 (V. Teslyuk); 0000-0002-8933-8635 (Yu. Lukashchuk); 0000-0001-9889-4177 (Yu. Opotiak)


component of the encryption key is the architecture of the neural network and the matrix of weights calculated for it.

The choice of architecture, training, calculation of the matrix of weights of the neural network and the implementation of such means of cryptographic encryption and decryption of data is an urgent problem.

Achieving this goal in the work involves solving the following tasks:

- improvement of the method of singular matrix decomposition;
- development of software for calculating weights for a given architecture of a neural network;
- implementation of cryptographic neuro-like encryption and data decryption on the basis of a microcomputer.

2. Related Works

Analysis of the use of neural networks in mobile robotic systems for the implementation of cryptographic data protection [7, 8] shows that they have the following significant disadvantages:

- do not take into account the requirements of specific applications in terms of cost, size and power consumption;
- not focused on the use of modern computing tools and systems on the chip;
- their implementation often does not provide real-time mode.

The analysis [9, 10] shows that to ensure real-time mode and high technical and economic characteristics it is necessary to use neural networks with pre-calculated weights. [11, 12] considered the application of direct propagation neural networks for the construction of cryptographic protection systems, which study the back propagation algorithm, and [13, 14] considered the possibility of using recurrent neural networks, in particular the Hopfield network. Also in [15, 16] the use of neural networks of counter propagation and radial basis functions for the implementation of cryptographic data protection is considered. A common disadvantage of the considered neural networks is the use of iterative learning algorithms, which are not focused on the implementation of cryptographic protection in real-time systems.

In [17, 18] the adaptation of the auto-associative neural network with non-iterative learning for the problem of cryptographic encryption and data decryption was carried out. In such a neural network, weights are calculated as a result of its training based on the principal components method. A feature of this method is the use of a system of eigenvectors that correspond to the eigenvalues of the covariance matrix of input data [19, 20]. An auto-associative neural network with calculated weights is a neural network that focuses on cryptographic data protection. Analysis of neuro-like tools [21, 22], which are used for cryptographic encryption and decryption of data, showed that the basis of such tools are neuro-like elements. The peculiarity of such neuroelements is that their implementation is reduced to the calculation of the scalar product using pre-calculated weights. In [23] the element base and the main ways of realization of mobile means of cryptographic protection of data transmission in real time are analysed. The analysis shows that a promising way to create such tools is the use of problem-oriented approach and modern microcomputer tools.

Therefore, the development on the basis of modern element base using neural networks of real-time cryptographic protection with high technical and economic characteristics should be considered an urgent task.

3. Methods and Materials

3.1. Improving the method of singular matrix decomposition and its focus on calculating the weights of neural networks

The classic process of learning the neural network is to determine the matrix of weights. To implement this task, the method of Singular Value Decomposition (SVD) was chosen. Singular matrix decomposition is a certain type of decomposition of a rectangular matrix and is widely used due to its visual geometric interpretation in solving many applied problems. Reformulation of singular

decomposition – the so-called Schmidt decomposition is used in quantum information theory, for example, in confusion. SVD is similar to the principal component analysis (PCA), but is more general. PCA assumes that the input matrix is square, SVD has no such restriction.

The general formula of SVD is as follows:

$$A = UDV^T, \quad (1)$$

where A – input matrix $N \times n$; U – left singular matrix $N \times N$, columns contain eigenvectors of the matrix AA^T; D – diagonal matrix $N \times n$, containing singular (eigen) values; V – right singular matrix $n \times n$, columns contain eigenvectors of the matrix A^TA.

To calculate the eigenvalues and eigenvectors, the Jacobi rotation method was used, in which the eigenvalues and eigenvectors of the symmetric matrix are calculated by iteration. This process of calculating eigenvectors is known as diagonalization. The essence of the Jacobi method is to construct a sequence of orthogonal similar matrices $S^{(1)}$, $S^{(2)}$, …, $S^{(m)}$ for a given matrix $S = S^{(0)}$, which converge to a diagonal matrix on which the diagonals are proper the value of the matrix S. To construct this sequence, a specially selected matrix of rotation J_i is used, so that the norm of the super diagonal part:

$$\left|\left|A^{(i)}\right|\right|_{off} = \sqrt{\sum_{1 \le j < k \le n} \left(a_{jk}^{(i)}\right)^2}, \quad (2)$$

decreases with each two-sided rotation of the matrix:

$$A^{(i+1)} = J_i^T A^{(i)} J_i. \quad (3)$$

Therefore, to calculate the matrix U, the result of the product AA^T is transmitted to the input of the Jacobi method. And to find the matrix V – the result of the product A^TA. In order to find the matrix D, it is enough to take the eigenvalues that were found when calculating the matrix U or matrix V and place them on the main diagonal.

After finding the matrices U, V and D, the weights are calculated. Based on the formula from the source [8]:

$$Aw = UD, \quad (4)$$

where A – is the input matrix of dimension $N \times n$, w – is the matrix of weights of dimension $n \times n$, matrices U and D are taken from the result of SVD.

Now we express the matrix of weights w:

$$w = A^{-1}UD, \quad (5)$$

Matrix A^{-1} can be written according to the formula proposed in the source [10]:

$$A^{-1} = VD^{-1}U^T, \quad (6)$$

Substituting (6) into (5) we obtain the formula for calculating the weights:

$$w = VD^{-1}U^TUD, \quad (7)$$

It should be noted that in the case when the matrix D is rectangular, the pseudo-inverse matrix will be calculated.

Now in order to encrypt the input data it is enough to use the formula from the source [17]:

$$\bar{y} = w\bar{a}, \quad (8)$$

where y – is the vector of the resulting matrix with encrypted data, w – is the matrix of weights, a – is the vector of the input matrix.

3.2. Implementation of cryptographic neuro-like encryption and data decryption based on microcomputer

Cryptographic neuro-like encryption and decryption of data is performed using symmetric keys, in which the encryption key and the decryption key are the same or the decryption key is easily calculated from the encryption key. Encryption takes place over plaintext using a key that consists of a given number of neurons in the neural network N, a matrix of W_{ji} weights, and masking operations.

Consider the main stages of encryption and decryption of the message.

The first important step is to choose a neural network architecture for cryptographic encryption and data decryption. The architecture of the neural network is determined by the number of neuroelements *N*, the number of inputs *k* and the bit inputs *m*.

The type of neural network used for data encryption is shown in Figure 1, where M_j – is the mask for the *j-th* input, x_j is the *j-th* input data, XOR is the masking operation using the Exclusive OR elements.

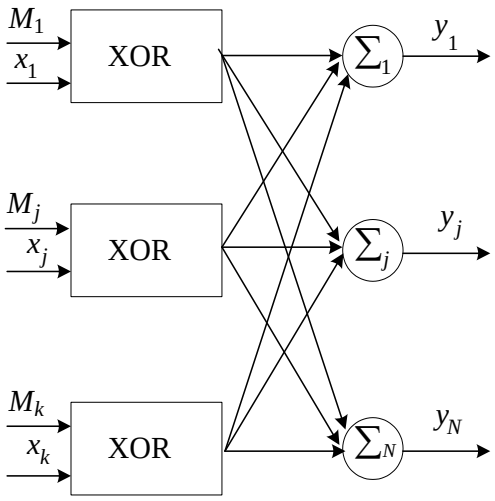

Figure 1: Neural network structure for data encryption

As known, the main disadvantage of classical neural networks is a fairly long process of learning them. The proposed neuro-like architecture allows the study of the neural network, which consists in determining the matrix of weights *W*, which is formed from the eigenvectors of the autocovariance matrix of input data *R*.

In this case, the main operation of neural network data encryption is reduced to multiplying the matrix of weights *W*, in which each element on the input vector $\overline{x}$ in accordance with the following formula:

$$y_j = \begin{vmatrix} W_{11} & W_{12} & \cdots & W_{1k} \\ W_{21} & W_{22} & \cdots & W_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ W_{N1} & W_{N2} & \cdots & W_{Nk} \end{vmatrix} \times \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{matrix}. \quad (9)$$

Multiplying the matrix of weights W by the input vector $\overline{x}$ is reduced to performing *N* operations to calculate the scalar product:

$$y_j = \sum_{s=1}^{k} W_{js} x_s, \quad (10)$$

where *k* – number of products, *s*=1, 2, …, *k*, *j* = 1, 2, …, *N*.

For example, to encrypt a 16-bit control command into the following neural network architecture options:

- m=2, k=8, N=8;
- m=4, k=4, N=4;
- m=8, k=2, N=2.

The corresponding architectures of neural networks are shown in Figure 2.

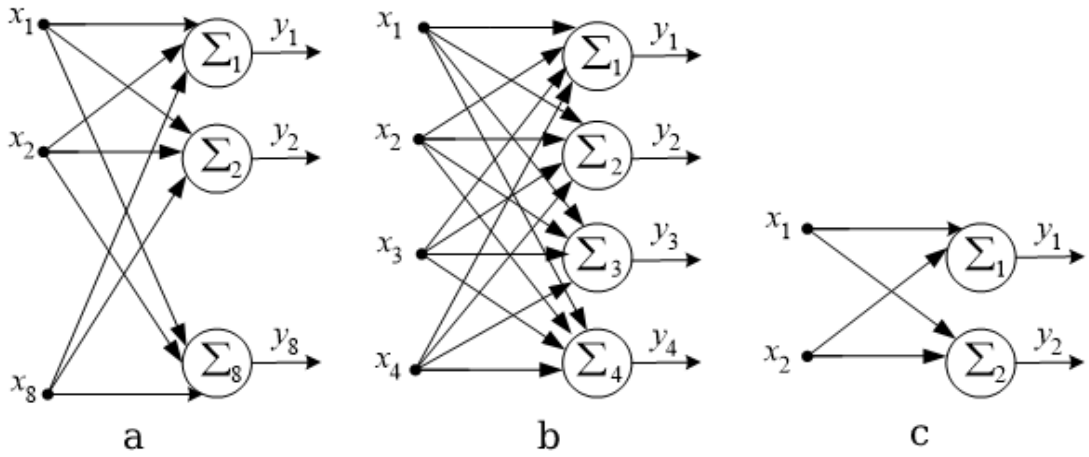

Figure 2: Architectures of neural networks: a) with 8 neural elements; b) with 4 neuro-like elements; c) with 2 neuro-like elements

The peculiarity of neural networks (Figure 4), which are used for cryptographic encryption, is that increasing the number of neural elements and inputs leads to a decrease in the bit size of the inputs. The bit inputs for a neural network with 2-a, 4-a and 8-a neuro-like elements are 8, 4 and 2 bits, respectively.

In this case, neural network data decryption requires the following steps. Encrypted data in the form of mantises y_{mj}^{h}, which are reduced to the largest common order m_m (block-floating point) are decrypted. Consider the main stages of decrypting encrypted data.

Configure the neural network architecture to decrypt encrypted data. The neural network architecture for decrypting encrypted data by the number of neural elements corresponds to the neural network used to encrypt data. In this neural network, the number of inputs and the number of neurons corresponds to the number of encrypted mantises y_{mj}^{h}.

The neural network architecture used to decrypt encrypted data is shown in Figure 3.

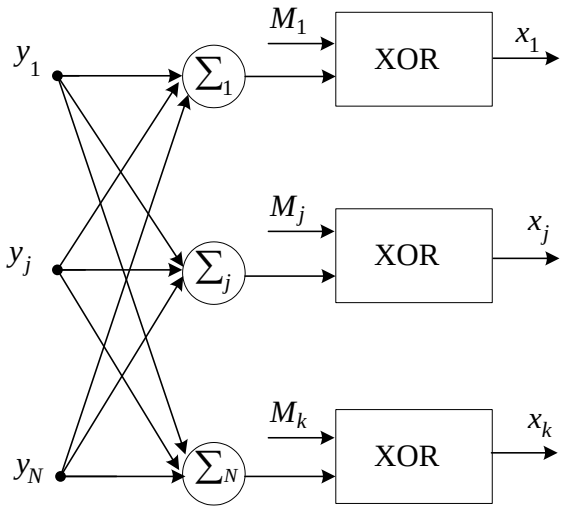

Figure 3: Neural network architecture for decrypting encrypted data

In a neural network for decrypting encrypted data, the bit rate of the inputs corresponds to the bit rate of the encrypted mantises y_{mj}^{h}, which determines the decryption time. To reduce the decryption time, you can discard the lower bits of the mantissa, which do not affect the recovery of the original message.

4. Experiment

To implement software for calculating the weights used to encrypt the incoming message, the C# programming language and Visual Studio 2019 development environment were chosen. The developed software uses an advanced method of singular matrix decomposition, and the Jacobi rotation method is used to find eigenvalues and eigenvectors. The practical value is that the developed tools provide fast calculation of coefficients for a given neural network architecture. A flexible user interface has also been developed, which allows you to clearly and in detail get acquainted with the operation of the algorithm. In the developed application, you can view each step of the calculations, which in turn allows you to check the correctness of the calculations at each stage.

For cryptographic encryption and decryption of data, neural networks are used, the architecture of which is determined by the number of neuroelements *N*, the number of inputs *k* and their bit size *m*. The number of neural elements in a neural network is determined by the following formula:

$$N = \frac{n}{m}, \tag{11}$$

where *n* is the bit size of the message, *m* is the bit size of the inputs.

Encrypted incoming messages can have different bits *n*, and to encrypt them using a neural network with a different architecture. The architecture of the neural network depends on the value of the message bit *n* and the number of inputs *k*. The following variants of the neural network architecture are possible for the n = 16 bit message: m = 2, k = 8, N = 8; m = 4, k = 4, N = 4; m = 8, k = 2, N = 2, and for n = 24 are: m = 2, k = 12, N = 12; m = 3, k = 8, N = 8; m = 4, k = 6, N = 6, m = 6, k = 4, N = 4; m = 8, k = 3, N = 3; m = 12, k = 2, N = 2.

The work of the developed simulation model is demonstrated below. For example, an input message with a bit size of 16 and an input bit size of 2 was chosen. Based on these data, the input matrix will have a dimension of 8×2. The incoming message is set by the user. As a result, a matrix of 2×2 weights is calculated. In the future, this matrix will be used to encrypt and decrypt the incoming message. However, there are requirements for the message, namely: the bit size of the message and the bit size of the inputs must be the same as when finding the matrix of weights.

Sixteen-bit message with bit input – 2 was used for the test example. Based on these data, the number of neuroelements and the number of inputs – 8 were calculated.

The input data (Figure 4) for the program are *n* – bit message, *m* – bit inputs and training matrix.

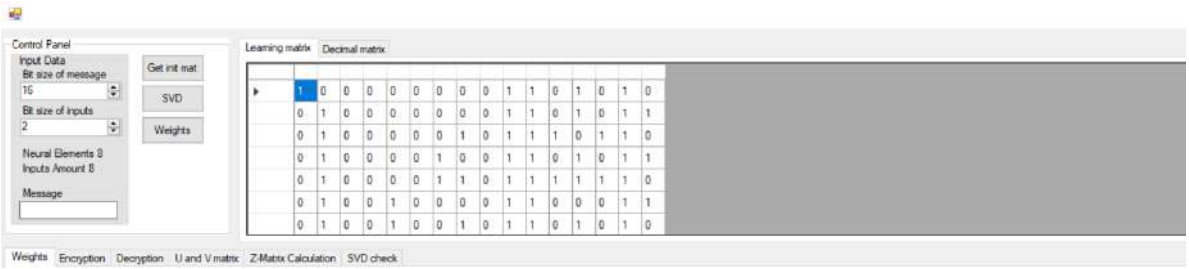

Figure 4: Input data for the program

The dimension of the training matrix is 14x16, where each line is a sixteen-bit command (Figure 5).

1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0
0	1	0	0	0	0	0	0	0	1	1	0	1	0	1	1
0	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0
0	1	0	0	0	0	1	0	0	1	1	0	1	0	1	1
0	1	0	0	0	0	1	1	0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0	0	1	1	0	0	0	1	1
0	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0
0	1	0	0	1	0	1	0	0	1	1	1	1	1	1	1
0	1	0	0	1	0	1	1	0	1	1	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	1	1	1	0	1	1
0	1	0	1	1	0	0	0	0	1	1	0	1	1	1	0
0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1
0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	1
0	1	1	0	0	0	1	0	0	1	1	1	0	0	1	1

Figure 5: Learning matrix

After entering the data, the program calculates the number of neuroelements and inputs. The *Get init mat* button is responsible for this operation.

Next, find the matrices U, V and D using the SVD algorithm, which is triggered by pressing the SVD button. As described above, the Jacobi rotation method is used to calculate the eigenvalues and eigenvectors. To find U, the SVD input is the result of the product AA^T, where A is the training matrix, and to find V is the result of the product A^TA. Matrix D consists of eigenvalues placed on the main diagonal.

To calculate the weights, the user must press the *Weights* button. The coefficients are calculated by formula (7). Formula (4) is used to check the correctness of the calculations. The result is shown in Figure 6.

Weights | Encryption | Decryption | U and V matrix | Z-Matrix Calculation | SVD check

Weights matrix

0.2365	-0.0436	-0.0413	0.0702	0.2453	-0.097	-0.8856	0.2861
0.1322	-0.4923	-0.0464	-0.1332	-0.8273	-0.0889	-0.1668	0.0035
0.1567	0.3824	0.2335	0.8059	-0.3371	-0.0931	0.0096	0.0525
0.2542	0.4588	0.6344	-0.5124	-0.1834	0.1373	-0.0836	0.0219
0.2234	-0.0371	0.0224	0.0668	0.0859	-0.0083	-0.2139	-0.9438
0.5489	-0.2009	0.1376	-0.0496	0.2193	-0.6944	0.3128	0.0929
0.3888	0.5098	-0.7202	-0.1861	-0.1812	-0.0012	0.0523	0.0134
0.5787	-0.3118	0.0313	0.1544	0.1409	0.688	0.1848	0.1253

Figure 6: Matrix of weights

This matrix can now be used to encrypt an incoming message, which the user can enter in a special text box – *Message*. Then, going to the *Encryption* tab and clicking on the *Encrypt* button, the user will display an encrypted message according to formula (8). The result is shown in Figure 7.

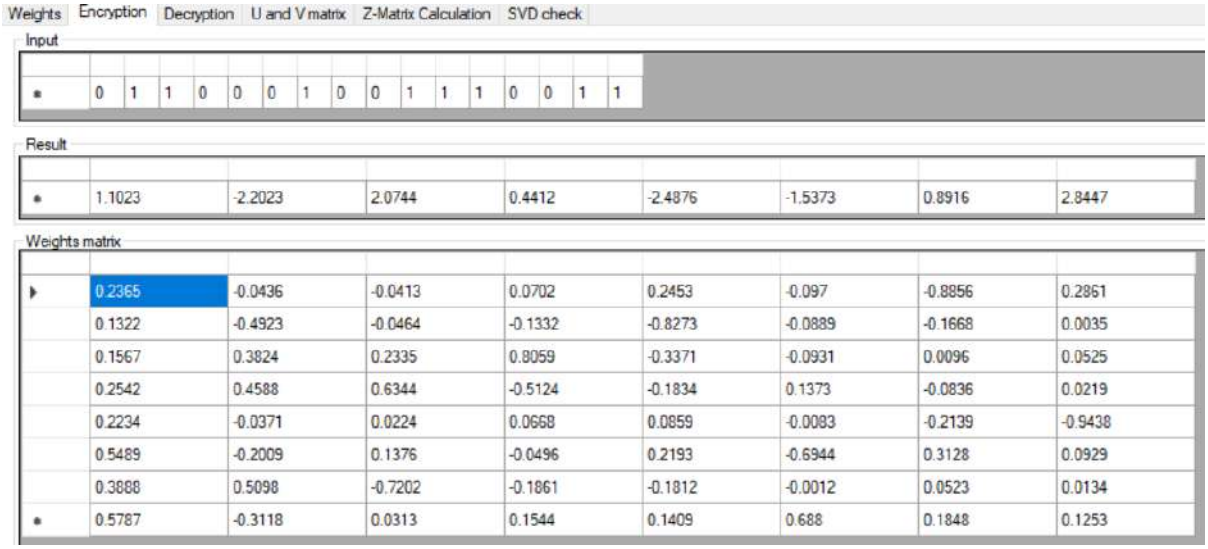

Figure 7: Encryption of incoming message

Now to decrypt the encrypted data you need to go to the *Decryption* tab and click on the *Decrypt* button. The decryption result is shown in Figure 8.

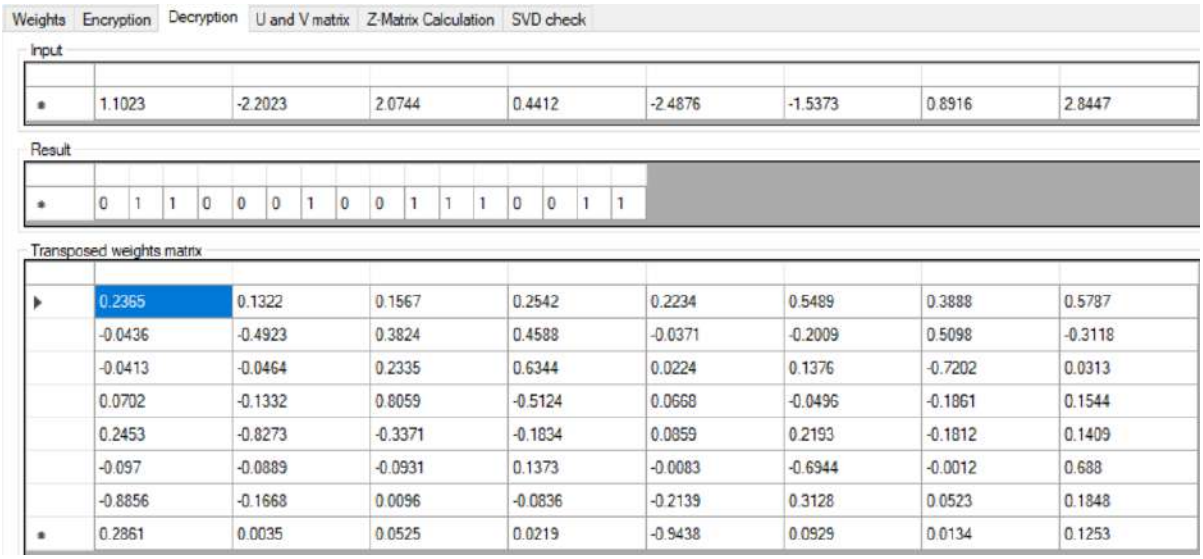

Figure 8: Decryption of incoming message

5. Results and Discussions

To implement the tasks of encryption/decryption, a software package has been developed, which includes software modules: neural network encryptor, neural network decoder and their configuration module. Software implementation of these modules is carried out in high-level C language, which is due to the need to ensure compatibility of implementations on different hardware platforms. Because a microcomputer is used to implement on-board modules, a standard GCC compiler is used to compile and debug software modules.

Since the neural network technology of cryptographic data protection is focused on hardware and software implementation, an important step is to assess the performance of individual components in the complex and determine the components whose time delays in data processing are maximum. By building a system based on a component approach, it is possible to independently improve individual components in the process of debugging and testing. The set of implemented software modules of

neural network cryptographic encryption/decryption of data includes the following components: Training_ANN, EnCrypt_ANN, DeCrypt_ANN.

The software module for training the neural network Training_ANN is used once to provide training for a specific implementation of the neural network for data encryption/decryption. This module prepares the input data in accordance with the specified values of the input parameters, namely: bit input neurons of the neural network; the number of neurons in the input layer of the neural network: the number and bit rate of training vectors. Based on the input data, the network weights for a given architecture are calculated. The result of the program is information for further configuration of the neural network for data encryption. The results of the calculations are recorded in files that are used to configure neural networks when running encryption/decryption programs. It should be noted that the configuration of the neural network is performed once and does not change during the further operation of the configured network in the process of encrypting/decrypting data.

The EnCrypt_ANN neural network data encryption software program uses configuration files created with Training_ANN. The network architecture (bit size of the input neurons of the neural network, the number of neurons of the input layer of the neural network) and the weights for a given architecture are configured. The input of the thus configured neural network is fed input data in the form of a vector with bit size n, and at the output of the neural network we obtain encrypted. The DeCrypt_ANN software decryption module also uses configuration files created with Training_ANN similar to the encryption process. The matrix of weights of the neuronal decryption network is easily obtained from the matrix used in encrypting data by transposing it.

Neuro-like cryptographic encryption/decryption of data for the onboard part is implemented on the basis of a modern microcomputer based on SoC H2+ from Allwinner. The use of microcomputers provides high flexibility in modifying data processing algorithms and sufficient speed to accomplish the task. The use of the Linux operating system involves the use of standard software development tools, standard compilers, including GCC, which provides portability of software code from one microcomputer platform to another with minimal need for modification. A FriendlyElec NanoPi Duo microcomputer was used to develop neural network cryptographic encryption and decryption.

An estimate of the time spent on the implementation of neural network cryptographic encryption and data decryption. To do this, use the regular command of the OS *time* in the following format:

time ./EnCrypt_ANN.

As noted above, three possible configurations of neural network architecture in configurations were debugged and tested to encrypt sixteen-bit messages on a microcomputer:

- m=2, k=8, N=8;
- m=4, k=4, N=4;
- m=8, k=2, N=2.

For this purpose, three separate directories were created to test previously developed and debugged files of programs Training_ANN, EnCrypt_ANN, DeCrypt_ANN. These programs were configured to implement these three neural network configurations. Next, it was performed sequentially using a neural network with a defined architecture using configuration files, encrypting data based on a neural network using the software module EnCrypt_ANN and decrypting them using the software module DeCrypt_ANN. Similar tests were performed for two other configurations of neural network architecture - m = 4, k = 4, N = 4 and m = 8, k = 2, N = 2. At the same time, identical results were obtained, ie neural networks with these architectures successfully provided encryption and decryption of files with identical input vectors and responded to corrupted input data.

According to the test results for the three neural network architectures, the following data were obtained (Table 1).

Table 1

Learning, encryption and decryption execution time for the three neural network architectures

Module/Architecture	m=2, k=8, N=2	m=4, k=4, N=4	m=8, k=2, N=2
Training_ANN, ms	189,7	197,4	203,2
EnCrypt_ANN, ms	37,2	29,3	37,8
DeCrypt_ANN, ms	38,4	22,9	24,9

To obtain more accurate results, the software modules were run 10 times and the results were averaged. The obtained results are visualized in Figure 9.

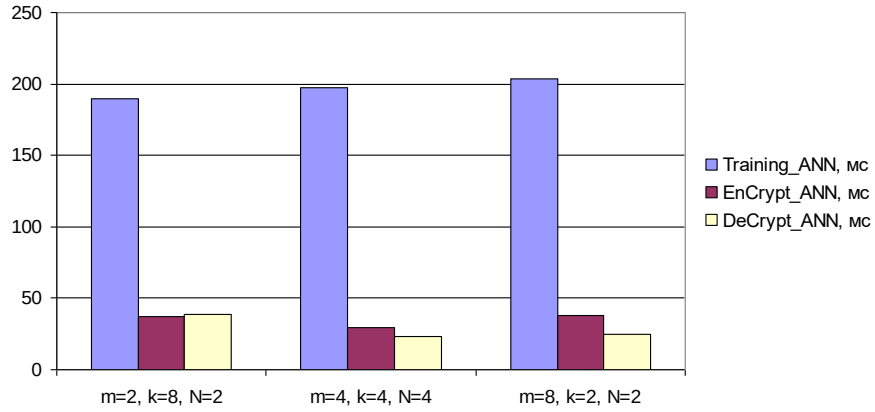

Figure 9: Time diagram of learning, encryption and decryption operations for three neural network architectures

The results of testing the implementation of neural network encryption/decryption of data show that the longest operation is the formation and training of the neural network, and its execution time on a microcomputer is about 200 ms and does not depend on the architecture of the selected neural network. On the other hand, the execution time of neural network cryptographic encryption and decryption of data blocks when implemented on a microcomputer is 30-38 ms and 23-35 ms, respectively.

Another important parameter when creating a data processing system based on a microcomputer is the temperature of the device, which is the result of CPU usage and indirectly indicates the intensity of these calculations. To assess the dynamics of temperature change, a script was used, which performs neural network cryptographic encryption and decryption operations in the loop. The script in the loop alternately runs the developed encryption and decryption software modules with a delay of 0.1s and thus loads the microcomputer processor. Next, the processor temperature parameter is read and output to the control console. The script was run for a long time (30 minutes) for testing. The results of the RPI Monitor utility are shown in Figure 10.

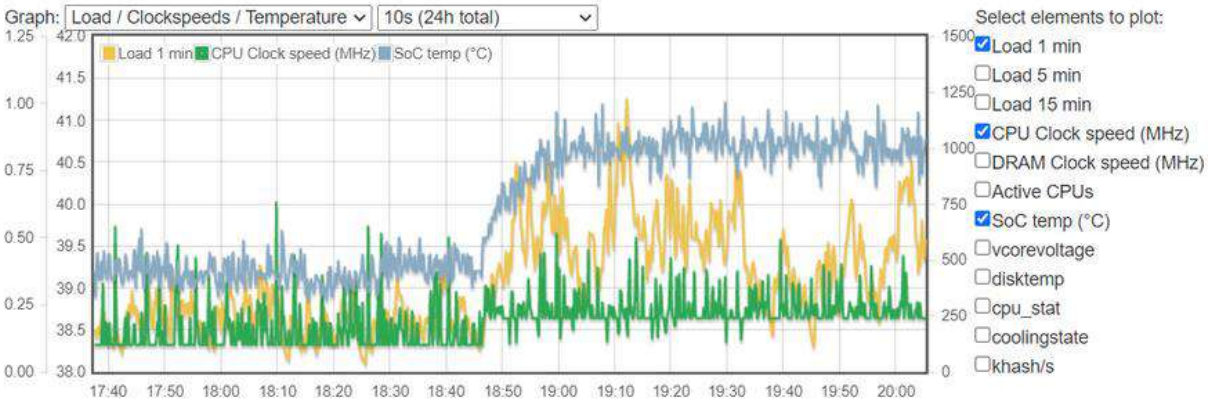

Figure 10: Graphs of temperature and load of the microcomputer core based on SoC H2+ during cyclic execution of software modules *EnCrypt_ANN* and *DeCrypt_ANN*

Therefore, in the normal operation of software for neural network cryptographic encryption and decryption of data blocks based on microcomputers, problems related to the temperature of the embedded system should not occur.

6. Conclusions

The paper presents an approach to build neural like networks of cryptographic data protection. The application of the SVD method allowed to calculate the weights to adjust the neural network. In the process of calculating the coefficients there is a certain error. This error is since the training matrix has a large number of zero elements, which affects the calculation of matrices. However, the results of modelling data encryption/decryption processes using a simulation model demonstrated the validity of the obtained results and the uniqueness of the encryption processes of the incoming message and its decryption.

The method of singular matrix decomposition has been improved, this was done by deriving formulas (5) - (7), after analysing the literature, in particular [8]. Also, it is focused on the calculation of the matrix of weights for a given architecture of a neural network. Software tools for calculating the matrix of weights that are adapted to the architecture of a neural network have been developed.

The encryption/decryption software was implemented on the basis of the SoC H2+ microcomputer and it was determined that the encryption/decryption time is approximately 30 ms, i.e., operate in near real-time. The obtained encryption/decryption time does not depend much on the configuration of the neural network architecture and is acceptable for the implementation of these tasks on embedded systems. The time of the operation of neural network formation and its training (calculation of weights) is an order of magnitude longer. However, this operation is performed once when changing the configuration of the neural network (i.e., encryption keys), and therefore does not affect the duration of the procedures themselves encryption/decryption.

Analysis of the obtained graphs (Fig. 10) of the microcomputer CPU load shows that the software implementation of encryption/decryption procedures does not significantly increase its temperature (apparently, this is facilitated by an efficient regular radiator) and problems with embedded system temperature should not occur.

Further research is aimed at reducing the execution time of encryption/decryption operations by further optimizing the relevant software modules and the use of hardware accelerators based on FPGA [24, 25].

7. Acknowledgements

This work was performed within the R&D "Experimental system of neural network cryptographic protection and real-time data transmission using barque-like codes" carried out by Lviv Polytechnic National University and funded from the state budget of the Ministry of Education and Science of Ukraine for 2021-2022.

8. References

[1] D. Hutabarat, M. Rivai, D. Purwanto and H. Hutomo, "LIDAR-based obstacle avoidance for the autonomous mobile robot", Proc. 2019 Int. Conf. Inf. Commun. Technol. Syst. ICTS 2019, pp. 197-202, 2019.

[2] M. Rohfadli, M. Rivai, M. Attamimi and D. Aulia, "Gas Leak Inspection System Using Mobile Robot Equipped With LIDAR," 2021 5th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM), 2021, pp. 50-54, doi: 10.1109/ELTICOM53303.2021.9590099.

[3] P. Denysyuk, V. Teslyuk and I. Chorna, "Development of mobile robot using LIDAR technology based on Arduino controller", 2018 14th Int. Conf. Perspect. Technol. Methods MEMS Des. MEMSTECH 2018 - Proc., pp. 240-244, 2018.

[4] Chen, Y.; Xie, S.; Zhang, J. A Hybrid Domain Image Encryption Algorithm Based on Improved Henon Map. Entropy 2022, 24, 287. https://doi.org/10.3390/e24020287

[5] Khan, S.; Han, L.; Lu, H.; Butt, K.; Bachira, G.; Khan, N. A New Hybrid Image Encryption Algorithm Based on 2D-CA, FSM-DNA Rule Generator, and FSBI. IEEE Access 2019, 7, 81333–81350.

[6] Corona-Bermúdez, E.; Chimal-Eguía, J.C.; Téllez-Castillo, G. Cryptographic Services Based on Elementary and Chaotic Cellular Automata. Electronics 2022, 11, 613. https://doi.org/10.3390/electronics11040613
[7] Tsmots I., Tsymbal Y., Skorokhoda O., Tkachenko R. Neural-like methods and hardware structures for real-time data encryption and decryption. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2019, 3, pp. 248–253, 8929809
[8] Diamantaras K.I., Kung S.Y. (1996). Principal Component Neural Networks. Theory and Applications. Wiley. ISBN 9780471054368. (T)(C) 270 s.
[9] Tkachenko R., Tkachenko P., Izonin I., Vitynskyi P., Kryvinska N., Tsymbal Y. (2019) Committee of the Combined RBF-SGTM Neural-Like Structures for Prediction Tasks. Lecture Notes in Computer Science, 2019, vol 11673. pp. 267-277, Springer, Cham, https://doi.org/10.1007/978-3-030-27192-3_21
[10] Śledź, S.; Ewertowski, M.W.; Piekarczyk, J. (2021). Applications of unmanned aerial vehicle (UAV) surveys and Structure from Motion photogrammetry in glacial and periglacial geomorphology. Geomorphology. Vol. 378. 107620.
[11] Verma, A.; Ranga, V. (2020). Security of RPL based 6LoWPAN Networks in the Internet of Things: A Review. IEEE Sens. Vol. 20. P. 5666–5690.
[12] Volna E., Kotyrba M., Kocian V., Janosek M. (2012). Cryptography Based On Neural Network. Proceedings of the 26th European Conference on Modeling and Simulation. P. 386–391.
[13] Shihab K. (2006). A backpropagation neural network for computer network security. Journal of Computer Science. Vol. 2. No. 9. P. 710–715.
[14] Logoyda, M., Nazarkevych, M., Voznyi, Y., Dmytruk, S., & Smotr, O. (2019). Identification of Biometric Images using Latent Elements. CEUR Workshop Proceedings. EID: 2-s2.0-85074659529.
[15] Arvandi M., Wu S., Sadeghian A., Melek W.W., Woungang I. (2006). Symmetric cipher design using recurrent neural networks. Proceedings of the IEEE International Joint Conference on Neural Networks. P. 2039–2046.
[16] Tsmots, I., Tsymbal, Y., Khavalko, V., Skorokhoda, O., Tesluyk, T. (2018). Neural-Like Means for Data Streams Encryption and Decryption in Real Time. Processing of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing, DSMP 2018. P.438-443. 84788513.
[17] Rabyk, V., Tsmots, I., Lyubun, Z., Skorokhoda, O. (2020). Method and Means of Symmetric Real-time Neural Network Data Encryption. 2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020 – Proceedings. Vol. 1, P. 47–50. 9322006.
[18] Chang A.X.M., Martini B., Culurciello E. (2015). Recurrent neural networks hardware implementation on FPGA: arXiv preprint arXiv:1511.05552.
[19] Sooyong Jeong, Cheolhee Park, Dowon Hong, Changho Seo and Namsu Jho. (2021). Neural Cryptography Based on Generalized Tree Parity Machine for Real-Life Systems. Security and Communication Networks. Vol. 2021. Article ID 6680782. Retrieved from: https://doi.org/10.1155/2021/6680782
[20] Kevin Gurney. 2003. An Introduction to Neural Networks. 317 p. Retrieved from: https://www.inf.ed.ac.uk/teaching/courses/nlu/assets/reading/Gurney_et_al.pdf
[21] Cryptowiki. Retrieved from: http://cryptowiki.net/index.php?title= Neural_networks_in_cryptography
[22] Eva Volna, Martin Kotyrba, Vaclav Kocian, Michal Janosek. (2012). Cryptography Based On Neural Network. ECMS 2012 Proceedings edited by: K. G. Troitzsch, M. Moehring, U. Lotzmann. European Council for Modeling and Simulation. Retrieved from: http://dx.doi.org/10.7148/2012-0386-0391
[23] Wolfgang Kinzel, Ido Kanter. (2002). Neural cryptography. Proceedings of the 9th International Conference on Neural Information Processing, ICONIP '02. Retrieved from: http://dx.doi.org/10.1109/ICONIP.2002.1202841.

[24] Bao Z, Guo J, Zhang W, Dang H. DSCU: Accelerating CNN Inference in FPGAs with Dual Sizes of Compute Unit. Journal of Low Power Electronics and Applications. 2022; 12(1):11. https://doi.org/10.3390/jlpea12010011

[25] Hao, C.; Zhang, X.; Li, Y.; Huang, S.; Xiong, J.; Rupnow, K.; Hwu, W.; Chen, D. FPGA/DNN Co-Design: An Efficient Design Methodology for 1oT Intelligence on the Edge. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6