Neural network as a programmable block cipher

Piotr Kotlarz[1], Zbigniew Kotulski[2]

[1] Kazimierz Wielki University, Bydgoszcz,
piotrk@ukw.edu.pl
[2] Institute of Fundamental Technological Research, PAS
and Institute of Telecommunications, WUT
zkotulsk@ippt.gov.pl

Abstract. A model of Boolean neural network is proposed as a substitute of a bock cipher. Such a network has functionality of the block cipher and one additional advantage: it can change its cryptographic properties without reprogramming, by training the network with a new training set. The constriction of the network is presented with an analysis of the applied binary transformations. Also three methods of training the network (what corresponds to the re-keying of a block cipher) are presented. Their security and effectiveness are analyzed and compared.

1. Introduction

In popular cryptographic communication protocols, like SSL, IPSec, SCP, etc., the designers provide several encryption algorithms, with an option of extending their list. Among the reasons of such a solution one could give the permanent progress of cryptanalytic attacks that can compromise a cipher. The possibility of changing the cipher makes that even if some algorithm is broken, the whole protocol remains valid. For example, development of linear and differential cryptanalysis shaken DES security, protocols with additional cryptosystems (e.g. 3DES, AES) implemented preserved their functionality for secure communication.

The DES case shows that the problem of breaking protocols due to breaking component cryptographic algorithms is crucial in a case of the expected long life of communication devices with the security protocols built-in. The designers and producers of the information infrastructure must take this problem into account.

For several years one can find in the literature results concerning application of programmable logic arrays for implementation of cryptographic algorithms. This fact is especially important for hardware implementations of cryptosystems. Any change in hardware is very complicated, so application of programmable (and, therefore, reconfigurable) element in the hardware seems to be a remedy in a case when we must modernize an algorithm. Such solutions already exist, e.g., in the paper [1] a cryptographic accelerator realizing IDEA block cipher is presented. A proposal of application of the processor RipeRench for cryptographic algorithms is given in [2]. Further, the authors realize algorithms CRYPTON [3] and RC6 [4] using the reconfigurable RipeRench. Another implementation of CRYPTON in reconfigurable

logical chips is presented in [5]. More information about application of the programmable devices to ciphers implementation and, more general, signal processing, can be found in [6].

In spite of the application of the programmable logical chips could be a solution of the problem of updating cryptographic algorithms built-in the network security components, in this paper we propose an alternative solution. In our opinion (based on earlier studies) the proposed solution of application of neural networks as universal updatable encryption algorithms could be a good alternative. After an appropriate design of such a network the implementing a new block cipher in the network and, then, the training process with adequate training sets should do periodic update of the algorithm.

In papers we [7-9] we proposed an idea of realization of the elementary operations of block ciphers (permutations and non-linear substitutions, S-boxes) by neural networks. We considered problems of efficiency of training of such networks and possibility of composing complete block ciphers of the elementary component neural networks. Now it is time to study the security of functioning of such neural network-based cryptosystems at each phase of its life: training and regular work as a cipher. Let us concentrate on the training process. We assume that the encrypting neural network is installed at a remote server (servers). At the client side we have the owner of the system, who can modify (from his location) the properties of the network and chose the transformation performed by it. For the sake of clarity, we consider now the neural network that realizes some permutation. So, the changes of the cryptosystem are the changes of permutation performed by the neural network. In spite of such a solution is not a general encrypting neural network; we can build the cipher as a composition of traditional S-boxes and permutations realized by neural networks. The included neural networks play a role of the switching elements in the cipher that make possible to change the cryptographic properties.

2. Neural network realizing elementary permutations

The concept of application of the neural network realizing permutations for cryptographic algorithms was proposed and presented in details in the paper [7]. For the purposes of this presentation we give here the outline of this method. Generally, the idea of application of a neural network as a universal cryptographic algorithm is based on constructing any complicated structure transforming block of bits onto blocks of bits as a combination of small blocks (neural networks) realizing elementary permutations (and, if needed, substitutions). In Fig. 1 are presented two examples of the networks realizing permutations that will be used in this paper. In each case, the left picture presents symbolically the transformation of bits, which is the permutation while the right one is the scheme of neural network realizing this permutation.

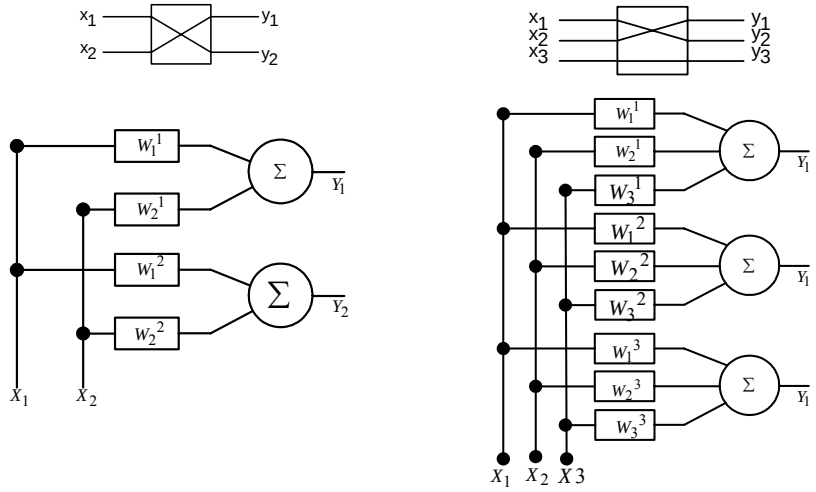


Fig. 1. Sample neural network blocks realizing elementary permutations

3. Modification of permutation realized by a neural network

Assume that we have the neural network that is trained to realize some transformation of the block of bits of a fixed length into the block of bits of the same length, playing a role of block cipher. The problem considered in this paper is: how to change the neural network to make it realizing some other block cipher (or the block cipher with different cryptographic properties). We assume that the structure of neural network remain constant during the lifetime of the system. The changing permutation is made by appropriate changes of the weight coefficients of neurons in the network. Thus, the replacement of one algorithm (old cryptosystem) by another (the new one) is by a certain training process, with the earlier designed training set. For this purpose we propose the supervised training method. For the user the network is considered as a black box located at a distant server. Thus, there is a need of preparing a method of remote modifying the algorithm realized by the neural network. Since we assumed that the security protocol applying our cryptographic algorithm works in the client-server architecture, we assign the role of client to system's administrator. Playing his role, he should be able to modify the algorithm at the remote server, performing some well-defined training process. This process can be made in two ways. One possibility is to train the network at the server using the adequate training set. In this case the training data must be transmitted from client to server. Another possibility is to train an identical copy of the neural network at the client side. Now, the resultant weights must be transmitted to the server and substituted at the network site.

The two proposed methods of training should be extensively analyzed with respect to efficiency and required bandwidth occupation at data transmission, quality of training process, necessary resources at the client and server side and the security and reliability of the obtained cryptographic algorithm. **Training at a server side.** As we

mentioned above, one of the possible ways of changing cryptographic algorithm realized by the neural network is performing the training process at the server (S). To do this we must transmit to the server the appropriate training data, which was earlier prepared at the client (C) side, to put into action the training program. In Fig. 2 is presented the architecture of the C-S system responsible for the training data exchange and the training procedure. The party S can work in two modes: the training mode (T1) and the operation mode (T2). In T1, operating just after the training data is supplied, the training process is performed and, as a result, the cryptographic algorithm realized by the neural network is changed. In T2 the neural network works as a black box realizing the cryptographic algorithm, in our case a block encryption. To complete the notation used in the picture, the open channel denotes some insecure communication channel, e.g., Internet. By cryptogram we understand the secured information being exchanged between the parties C and S of the communication protocol.

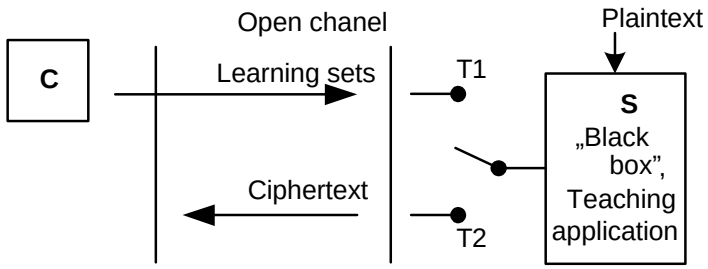
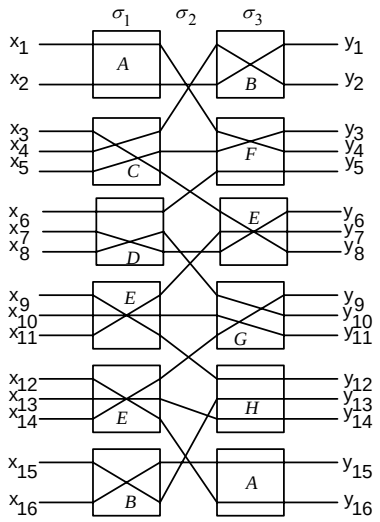


Fig. 2. Training at the server S side

Since this paper we concentrate on the security of the process of dynamical changes of the cryptographic algorithm inserted in the neural network, for the sake of clarity of presentation we restrict our consideration to a binary transformation of the block of 16 bits onto the block of 16 bits. More precisely, the transformation realizes a permutation of the block of 16 bits. The scheme of the permutation is presented in Fig. 3. For the construction of the complete 16-bit permutation we used blocks of elementary neural networks that realize 2-bit and 3-bit permutations. The details of such blocks and their properties were discussed in details in the paper [7]. Let us remark that using analogous blocks it is possible to build also non-linear substitutions, which can be used for building S-boxes, the supplementary component elements of many present-day block ciphers. The examples of such structures are also presented in the papers [7], [8], and [9]. However, now we concentrate on permutations. The detailed description of the 16-bit permutation build of some number of elementary neural network blocks is presented in the diagram below. In Fig. 3 a network built of two layers of blocks of smaller neural networks is presented. Each of the layers realizes different permutation, what is symbolically denoted as σ_1, σ_3. The additional permutation σ_2 reflects the connections between the outputs of one layer of the blocks and the inputs of the blocks of the other layer. In the right hand side of the Fig. 3 the permutation realized by the network is presented.



$$\sigma = (\sigma_1 \circ \sigma_2 \circ \sigma_3) =$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 2 & 5 & 3 & 4 & 6 & 8 & 7 & 11 & 10 & 9 & 14 & 13 & 12 & 16 & 15 \end{pmatrix} \circ$$

$$\circ \begin{pmatrix} 1 & 2 & 5 & 3 & 4 & 6 & 8 & 7 & 11 & 10 & 9 & 14 & 13 & 12 & 16 & 15 \\ 3 & 2 & 6 & 1 & 4 & 5 & 8 & 9 & 12 & 10 & 7 & 16 & 14 & 11 & 13 & 15 \end{pmatrix} \circ$$

$$\circ \begin{pmatrix} 3 & 2 & 6 & 1 & 4 & 5 & 8 & 9 & 12 & 10 & 7 & 16 & 14 & 11 & 13 & 15 \\ 4 & 1 & 8 & 2 & 3 & 5 & 6 & 10 & 12 & 11 & 7 & 16 & 14 & 9 & 13 & 15 \end{pmatrix} =$$

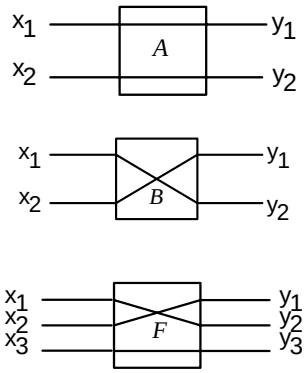
$$= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 4 & 1 & 8 & 2 & 3 & 5 & 6 & 10 & 12 & 11 & 7 & 16 & 14 & 9 & 13 & 15 \end{pmatrix}$$

Fig. 3. Neural network realizing a permutation

To summarize the diagram, the permutation realized by the resultant neural network is a composition of three 16-bit permutations, realized by individual layers, what is expressed by Eq. 1.

$$\sigma = (\sigma_1 \circ \sigma_2 \circ \sigma_3) \tag{1}$$

In the above scheme, the permutations σ_1, σ_2 are these, which are being changed during the training procedure, while the permutation σ_2 remains unchangeable. In the methodology, which is implemented and verified at the moment we propose the training procedure independent for each of the permutations σ_1 and σ_3. Thus, we need two training sets for the two permutations. In further studies we plan to propose an effective procedure of training the three-layer network with a single training set. Such a procedure would be more convenient and more confidential, because it would not need knowledge of the internal structure of the neural network by the training procedure. The preparation of a security protocol with the neural network-supported encryption algorithm runs in several steps. First, the network is implemented at the server; its structure is presented in Fig.3. Next, random values of weights are inserted to it. Finally, using some training set (earlier prepared at the client side, individually for each requires permutation, and transmitted to the server) the training program at the server side performs the training of the network. In this paper we present some selected sample fragments of the training sets for the permutations σ_1 , σ_3. In Fig. 4 are presented fragments of the training sets for several chosen sample elementary modules, but not the complete training set. One, having the full knowledge about the complete neural network and the content of the training sets for individual blocks, can reconstruct the training set for the complete network. How to construct general training sets, can be found in our earlier papers [7, 8]. However, making the methods effective needs additional studies.



x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	1

Fig. 4. Selected sample blocks of the network with the fragmentary training sets

As we remarked, the training set must be delivered to the server to the program training the network. The transmission of this set is through an insecure communication channel. Asking about the security of the algorithm update procedure, we must assume that an adversary can have the content of the training set. If the complete training set is transmitted, the adversary has the complete information about the permutation (cryptographic algorithm). He simply obtains this information by some analysis, because the training set corresponds to a unique permutation. The essential information for the adversary is the topological structure of the network. Certainly, we could assume that this structure is secret, but such a solution seems to be inefficient. First, it contradicts the presently valid rule of knowledge the algorithm (the secret algorithm cannot be analysed and its security is never sufficiently sure). Moreover, the adversary can always train a similar network (e.g., with the same number of inputs and outputs) and obtain very similar algorithm, what together with other cryptanalytic techniques would completely break the cipher. So, the remedy could be transmission not the complete training set, but only some partial information, which together with some trapdoor information would make possible the reconstruction of the complete training set at the server side and performing the training procedure. In Fig. 3 the elementary blocks of neural network are denoted as A, B, F. Using these notations (reflecting the network internal structure) we can transmit the fragmentary training sets for each blocks, as it is presented in Fig. 4. Using these components one can construct, at the server side, the complete training set for the whole neural network realizing the cryptographic algorithm. Such a solution gives some security for the algorithm update process provided the internal structure of the complete network (its topology) remains secret.

The problem of security of the algorithm update process at the server side is strictly connected the efficiency of such a training process. First, the server must be equipped with the training software, which must be secure during learning the network and during the storage in inter-learning period. Then, the process of training is server resources- and time-consuming, what is especially important in a case of large number of servers located over the network and serviced by the training systems. Finally, the servers are often located in an un-trusted environment, so such attacks, as

sniffing/spoofing and analysis of electronic discharge are possible. For this reasons, one should look for alternative solutions of dynamic changing encryption algorithms built-in the remote network security devices equipped with neural networks. **Training at a client side**. A natural alternative for training neural network at the server side is training the identical copy of the network at the client side and transmission of the obtained result to the server. As we know, the neural network is completely defined (in the model we apply for encryption) by the architecture of the functional blocks (being certain neural networks), the structure of neurons inside the blocks and the weights of these neurons. Thus, information sufficient to change the algorithm is contained in the weights (we assume that the structure of neural network remains unchangeable in the server). The result of the training is vector of weights and this vector must be transmitted form client to server to update the cryptographic algorithm.

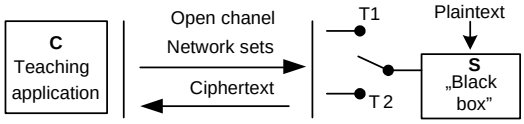


Fig. 5. Training at the client C side

The communication scheme for training the neural network at the client side is presented in Fig. 5. Now the server S again works in two modes. Mode T1 is the update of the weights in the neural network blocks, what corresponds to inserting a new cryptographic algorithm into the server's "black box". Mode T2 is the operation mode, where a plaintext inputs the "black box" and the corresponding ciphertext outputs it. Certainly, the "black box" performs also the inverse operation.

The presented solution is especially effective (less server time consuming, smaller volume of information transmitted) in the case of updating several servers. Also the neural network update program at the server side is less complicated. It works without the observer's supervision, what is required in the case of the neural network training at the server side. It is possible the automatic update of the weights with some self-check algorithm afterwards.

The neural network weights update can be performed in two ways. In the following two sections of the papers we show the outline how the two methods can be performed. **Transmission of updated weights**. In Fig. 5 we present the transmission of parameters of a neural network. In this case the parameters are new values of weights. The basic assumption to make such a procedure effective is that both parties of the protocol, client C and server S are equipped with the neural networks with identical internal architecture. After the training stage, the weights of neurons obtained at the client side are transmitted to the server. The server is switched to the mode T1 and the weights are introduced into the neural network, overwriting the previous values. Then, the server is switched to the mode T2 and starts realizing a new encryption algorithm. For the neural network presented in Fig. 3 we have 64 weights and such a number of coefficients must be sent through the communication channel, see Eq. 2.

$$w = [w_1 ... w_{64}] \tag{2}$$

The information sent must contain the assignment of the weights to neurons. Since some functionally identical blocks in the network architecture repeat (as it is seen in Fig. 3), the weights for identical blocks can be sent only once. Next, they must be substituted as many times as needed. This property reduces the amount of information transmitted during the update process. Eq. 3 presents the sample weight vectors for the defined blocks A, B,..., H of the neural network presented in Fig. 3.

$$W_A = [w_{A1} ... w_{A4}], W_B = [w_{B1} ... w_{B4}], W_C = [w_{C1} ... w_{C6}] W_D = [w_{D1} ... w_{B6}] \tag{3}$$

$$W_E = [w_{E1} ... w_{E6}], W_F = [w_{F1} ... w_{F6}], W_G = [w_{G1} ... w_{G6}], W_H = [w_{H1} ... w_{H6}]$$

Consider now the security of the weights updating process. Assume at the beginning that the transmission of the weights is by an open communication channel (that is, an adversary knows the weights). Trying to answer the question, what information about the cryptographic algorithm is contained in the weights we must say, that even if the weights together with the architecture of the neural network completely describe the cryptographic algorithm, then the weights alone give no information about it. For a successful attack an adversary should know the structure of blocks, their internal connections and assignment of weights to particular neurons. Thus, the algorithm remains secure provided the adversary does not know the neural network (our "black box") architecture, even if the weights are transmitted by an open communication channel. The last sentence, in fact, contradicts the widely used rule of the public knowledge of applied cryptographic algorithm (see Chapter 3.1). In the following sections we propose two methods how to solve this problem.

Before we go further, let us remark, that the proposed solution of application of the neural network for implementation of cryptographic algorithms slightly differs form analogous application of the programmable logical circuits. To update the algorithm, in the first case we can send only a sequence of numbers (the weights), while in the second one we need the complete code of the cipher. **Transmission of differences of weights.** Assume now that we wish to protect the transmitted weights against interception by an adversary. The method proposed in this section does not apply cryptography for increasing security of the update procedure. The adversary, to attack successfully the weights values should permanently trace traffic between client and server to collect all transmitted updates of weights. This effect can be obtained, if instead of the complete set of new weights W' we will transmit to the server their differences R to the previous values of weights W. The updating program at the server, knowing the old weights W and differences R can easily restore the new weights W', what symbolically is presented in Eq. 4. Here we present only the updating procedure for two weights; the others are completely analogous.

$$R = W \pm W', \qquad W'_A = [W_{A1} \pm W_{R1}, ..., W_{A4} \pm W_{R4}], W'_B = [W_{B1} \pm W_{R1}, ..., W_{B4} \pm W_{R4}], ... \tag{4}$$

In the proposed method, the effective updating the neural network is possible under two conditions. First, both networks: at the client side and at the server side must be initiated with the same values of weights. Then, the updating must be synchronous, what means that no weight update can be neglected at the server side. This forces the server to permanent following signals about the weights updates (remembering

present state of weights and the prompt receiving the differences of weights by the network updating program). If we additionally assume that the initial state of the neural network is a secret (the adversary cannot restore the initial values of weights), then both the internal structure of the neural network and the updates of weights (the transmitted differences of weights) need not be confidential to keep confidentiality of the communication protected by the neural network.

4. A unified solution for a permutation update

The methods presented in Chapter 3 assumed that the updating data (the training set, the weights, or the differences of weights) was transmitted by an open communication channel. However, we can assume a possibility of using a secure (encrypted) channel for transmission of the critical updating data. Fig. 6 presents this situation.

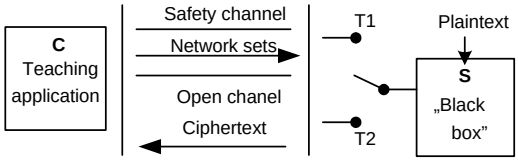


Fig. 6. Data transmission through the secure communication channel

The amount of the required data is relatively small (especially in the case of weights), so we can use asymmetric cryptography to deliver the data in a secure way. This method should not introduce a significant delay in operating the system. An alternative could be application of some popular secure tunnelling method, e.g., SSL.

The data exchange scheme presented in Fig. 6 shows that the secure channel is used only in the updating T1 mode. After the update the “black box” is switched to the operating T2 mode and the server starts regular work, which is the cryptographic algorithm implemented in the neural network starts encryption communication.

To conclude, if the pair client-server has no secure communication channel (the asymmetric secure communication is to absorbing for the parties and the parties do not have a symmetric secure channel), then can be used the simplest solution. After calculating the new weights the operator of the communication protocol can update the weights manually, moving the required data in some mobile device.

5. Summary

In this paper we considered possible difficulties and threats during the process of updating cryptographic algorithm working at a remote server. We postulated the secure process of modifying the cipher. For the sake of clarity, we restricted our considerations to the neural networks realizing permutations. However, using methods of representing S-boxes in neural networks, proposed in [7], we could first

represent any substitution-permutation block cipher as a neural network and then extend the update method to such a cipher.

We showed that the transmission of the training set by an open channel is not a good solution. An adversary who obtains the training set can reconstruct the cryptographic algorithm realized by the neural network. Updating the weights of neurons of the neural network is much more secure solution. The supplementary application of asymmetric cryptography solves the problem security of the operation on the level of security equivalent to security of the asymmetric algorithm applied in the protocol. However, not always such a method can be applied, because the asymmetric cryptography requires relatively strong computational abilities of the server and significantly complicate the cryptographic algorithm updating procedure.

The results presented in this paper are mostly of demonstrative character. They show that it is possible a secure mechanism of dynamic updating the cryptographic algorithms realized by appropriately designed neural networks. However, to study practical effectiveness and exploitation efficiency of such algorithms, one must define the procedure for the complete cipher, containing both linear (permutations) and nonlinear (S-box) blocks and perform its practical Internet tests.

To summarise, lest us remark that our proposal is in some sense more general than so-called neural cryptography, which utilizes neural networks for a secret key agreement through a public communication channel (see [10]). The proposed solution , in spite it is still not perfect, agrees a complete cryptographic algorithm.

References

[1] E. Mosanya, Ch. Teuscher, H.F. Restrepo, P. Galley, E. Sanchez, CryptoBooster: A Reconfigurable and Modular Cryptographic Coprocessor, in: Cryptographic Hardware and Embedded Systems: Proc. CHES'99, LNCS 1717, Springer Berlin 1999.

[2] R. Taylor, S. Goldstein, A High-Performance Flexible Architecture for Cryptography, Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems, Worcester August 1999.

[3] C.H. Lim, CRYPTON: A New 128-bit Block Cipher, Proceedings of the First Advanced Encryption Standard Candidate Conference, Ventura, California, NIST, 1998.

[4] L.R. Knudsen, Correlations in RC6, Department of Informatics, University of Bergen, N-5020 Bergen:, July 29, 1999.

[5] W. Laskowski, Programmable logical circuits as tools supporting cryptographic data protection, Przegląd Telekomunikacyjny, Vol. LXXIV, no. 3/2001. (In Polish)

[6] T. Łuba, K. Jasiński, B. Zwierzchowski, Programmable logical circuits processing signals and information – digital circuit engineering in multimedia and cryptography, Przegląd Telekomunikacyjny Vol. LXXVI no. 8–9/2003. (In Polish)

[7] P. Kotlarz, Z. Kotulski, Application of neural networks for implementation of cryptographic functions, in: Multimedia in Business and Education, ISBN83-9182218-7-0

[8] P.Kotlarz, Z.Kotulski, On application of neural networks for S-boxes design, in: P. S. Szczepaniak, J.Kacprzyk, A. Niewiadomski, ed. Advances in Web Intelligence, AWIC 2005, LNCS 3528, pp. 243-248, Springer, Berlin 2005.

[9] P. Kotlarz, Z. Kotulski, Artificial intelligence methods in the present-day cryptography, Proceedings of Ploug'05, Zakopane 2005. (In Polish)

[10] I. Kanter, W. Kinzel, E. Kanter, Secure exchange of information by synchronization of neural networks, Europhys., Lett. 57, 141 2002.