УДК 004.274

M.A. Aleksandrov, PhD student
Y.A. Bashkov, professor
Department of Applied Mathematics and Informatics
Donetsk National Technical University, Pokrovsk, Ukraine
eab23may@gmail.com, mykyta.aleksandrov@donntu.edu.ua

Use of interacting neural networks in cryptography

The computing power of computer technology, as well as the amount of data transmitted, is growing steadily every day. In this regard, existing encryption algorithms quickly become obsolete, due to insufficient cryptographic strength or bandwidth. There is a need to constantly create new encryption methods or modify old ones. As new algorithms can be algorithms using neural networks. This article presents the existing algorithms created on the basis of mutual synchronization of neural networks. The basic principles of their construction and application are described, as well as areas that require further research and modernization are identified.
Key words: neural networks, tree parity machines, synchronization, cryptography, encryption.
DOI: 10.31474/1996-1588-2020-1-30-19-24

Introduction

Cryptography is a science that ensures the secrecy of data transmission with the support of confidentiality, data integrity and user authentication identification. Cryptography is based on encryption mechanisms that allow you to convert an open message to a closed (encrypted) one. There are two basic approaches to encryption - symmetric methods with a secret key and asymmetric methods with an open prickly [1].

Public key cryptography consists of creating two keys (public and private), as well as a reliable encryption method that allows you to encrypt a message with a private key in such a way that you can subsequently decrypt it only using the public key, and vice versa. With this approach, the public and private keys must be generated in such a way that the calculation of the private key in the presence of the public key is impossible in a reasonable amount of time. The private key should not be transferred by its owner, only the public key is transferred. In this case, the key can be transmitted over an open channel, or the key can be made publicly available [2].

Systems built on this approach have a very high level of security, but at the same time, calculations in them require a large amount of computing resources, which greatly complicates application of this approach for large data flows.

Secret key cryptography is based on the existence of one single key for encrypting and decrypting messages, which is secretly transmitted between users over a secure communication channel. Compared with public key encryption, this approach is less resource-intensive, however, the main problem of the approach lies in the secure exchange of a shared key between users. To ensure the secure transfer of the secret key, special protocols are used, for example, the Diffie-Hellman protocol, which is based on the assumed complexity of the problem of solving the discrete logarithm. This protocol allows two users to obtain a common private key via a communication channel that is not protected from listening, but is protected from spoofing [3-5]. If it is possible to modify data, it becomes possible for an attacker to intervene in the key generation process. In addition to the classical algorithms created for key exchange, other approaches can be used, such as: neurocryptography, chaotic synchronization, and the exchange of quantum keys [5]. In this work, a key exchange option using neural networks is proposed, namely, their mutual synchronization.

Interacting neural networks

An artificial neural network is a model built on the principle of functioning of nerve cell networks of biological organisms. The first attempt to create an artificial neural network was in [6] when trying to simulate the processes occurring in the brain. Artificial neural networks are systems of integrated and communicating processors (neurons). A neural network usually consists of a multilevel connected structure of neurons, in which the first layer consists of input neurons, to which the initial data are supplied, and then data is transmitted through the synapses to the subsequent layers to the flesh of the last level of output neurons in which the result is finally formed. In synapses, parameters called "weights" are stored, which affect the data when calculating the resulting function [7]. Typically, neural networks have three types of parameters:

– scheme of communication between layers of neurons;

– the training process (updating weights);

– activation function.

Neural networks can be multi-level, and also have feedback between neurons [7]. However, an increase in the number of processors and levels in the neural network, as well as the addition of feedback, increases the amount of computation, which greatly complicates the work with a large data stream.

Neural networks need training to solve the necessary problem. However, if one neural network will train another with a similar structure, then at some point in time these two networks will become symmetrical and will give the same result with the same data set, even if the learning process continues. Suppose we have a learning network and the network that it teaches. Thus, the training network will provide its partner with sets of data, input and output, on which the second network will be trained. After a certain number of operations, both networks are synchronized and their weight vectors will be correlated, and upon receipt of input data not belonging to the training sample, both networks will continue to give identical results [8].

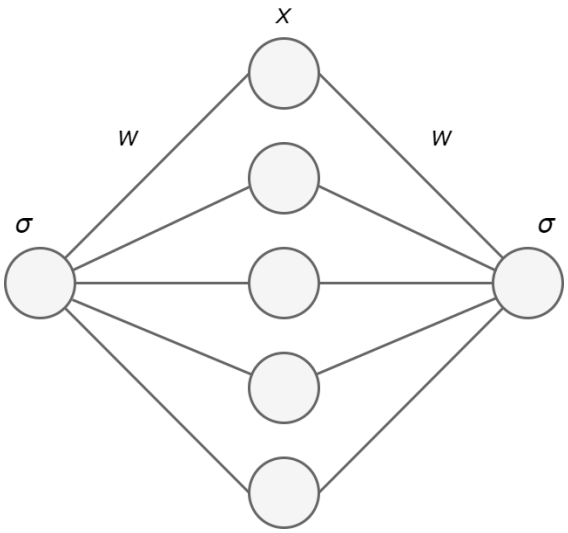

Figure 1 – Two perceptrons receive an identical input x and learn their mutual output bits σ

As an example, fig. 1 shows the mutual learning of neural networks for a simple system: two perceptrons receive a common random input vector x and change their weights w according to their mutual bit σ [9].

Mutual learning of networks can be done in two ways: batch and interactive. In batch training, all training sets are saved and subsequently used to minimize learning errors. In interactive training, only one new example is saved for use at each step of the training, so such training can be considered as a dynamic process. In interactive learning, examples are generated by a static learning network, however, the learning network can also generate examples and participate in the process of mutual learning [10].

In mutual learning, in which both the learning network and the network that is learning take part, it is almost impossible to learn the same third network, which has the ability to intercept the data sent for training, since it will not be able to participate in the process of creating new vectors for learning. This makes the peer-to-peer learning process safe and allows learning through the open channel.

Use of synchronized neural networks in cryptography

Synchronization of two neural networks can be used to generate keys in cryptography. Based on the identical weights of two synchronized neural networks, you can create a key or a one-time notepad, which can later be used as an initial number for generating pseudorandom sequences, which can also act as a key [8, 11]. When creating two neural networks on the basis of which keys will be created, it should be borne in mind that in order to prevent hacking, it is necessary to hide as much information as possible so that an attacker could not synchronize his network with the system. But at the same time, enough information should be transmitted for synchronization [11]. In [12], to solve such a problem, it was proposed to use multilayer networks with hidden elements. Based on them, the algorithm of tree parity machines (TPM) was created, the synchronization of which is similar to the synchronization of two chaotic oscillators in the theory of chaotic connections.

The tree parity machine (TPM) is a special (fig. 2) type of multilevel direct distribution neural network. This neural network has one output neuron, K hidden neurons and $K \times N$ input neurons.

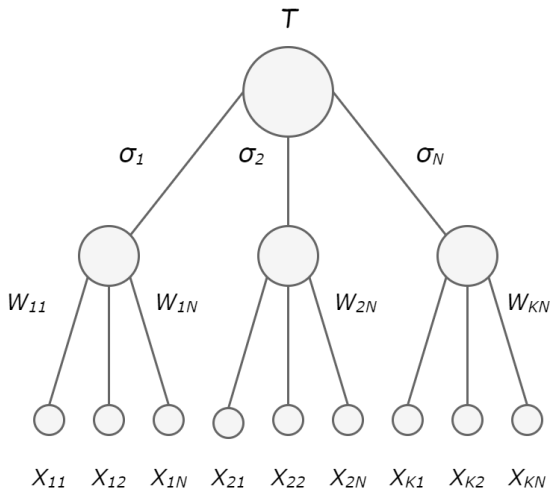

Figure 2 – Structure of tree parity machines

Input neurons x_{ij} take binary values:

$$x_{ij} \in \{-1,+1\}. \quad (1)$$

The weights between the input and hidden neurons w_{ij} are:

$$w_{ij} \in \{-L,...,0,...+L\}. \quad (2)$$

The value for each hidden neuron σ_i is the sum of the products of the input value and the weight coefficient:

$$\sigma_i = \mathrm{sgn}(\sum_{j=1}^{N} w_{ij} x_{ij}), \quad (3)$$

$$\mathrm{sgn}(x) = \begin{cases} -1 & if \quad x \le 0 \\ 1 & if \quad x > 0 \end{cases}. \quad (4)$$

The value of the output neuron is the product of all hidden neurons *K*:

$$\tau = \prod_{i=1}^{K} \sigma_i. \quad (5)$$

The output value under these conditions is also binary [10].

To modify the weights of networks *A* and *B*, you can use the following training rules [11]:

Hebb's Anti Rule:

$$w_i^+ \in w_i + \sigma_i x_i \theta(\sigma_i \tau) \theta(\tau^a \tau^b), \quad (6)$$

Hebb's Rule:

$$w_i^+ \in w_i - \sigma_i x_i \theta(\sigma_i \tau) \theta(\tau^a \tau^b), \quad (7)$$

Random straying:

$$w_i^+ \in w_i + x_i \theta(\sigma_i \tau) \theta(\tau^a \tau^b). \quad (8)$$

Studies have shown that for large values of *L* (2), the synchronization time of a third-party network is so long that an attack of a system with such an architecture is impossible. This means that this approach is safe and with an increase in the number *L*, the cryptographic stability of the system increases, but in this case, the computational complexity and time required for synchronization increases. Therefore, the use of such systems to transmit large data streams is difficult [10].

Group synchronization using mutual training

The algorithm described in the previous section assumes the synchronization of neural networks with the subsequent creation of an encryption key for only two end users, however, in modern realities, it is often necessary to create a group encryption key. The way out of this situation is a group synchronization mechanism using tree parity machines. Group synchronization mechanisms are presented in the work [13].

The essence of the algorithm is that *N* TPM must be synchronized together, and they are represented by M number of leaves of the complete parity tree. There are two approaches to collective synchronization of TPM:

– Binary Tree with Election (BTWE);

– Binary Tree with Swap (BTWS).

In the end, both give the desired result, but come to it in different ways.

Binary Tree with Election

With the BTWE approach, *N* TPMs are represented by *M* leaves, at each iteration of the J algorithm, the parity tree is divided by 2^j and mutual learning is applied for each pair of leaves sharing the same parent. In the next step, J increases and in each subtree TPM nodes are selected, which also use the mutual learning algorithm, and then send bits to their subgroups.

Such iterations continue until the root is reached by the algorithm, after which it ends. If the root is synchronized, then all TPMs are also synchronized and have the same weight vectors [12].

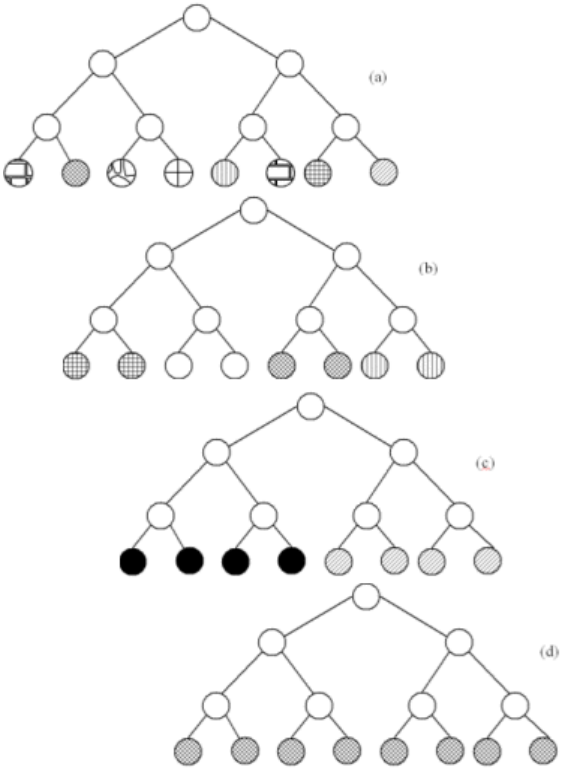

Figure 3 – Sequence of Neural synchronization between four TPMs using BTWS

Binary Tree with Swap

In the BTWS approach (Fig. 3), the mutual learning algorithm is applied between every two sides having the same parent in the even tree structure. If $mdepth$ is the maximum possible depth of the binary tree, and $cdepth$ is the depth at

which the algorithm is at a given time. Starting from $depth = mdepth - 1$, the mutual learning algorithm is applied between each pair of leaves (TPM), which belong to the same parent. After synchronization is achieved, a transition to a higher level is performed ($cdepth = cdepth - 1$), and a permutation mechanism is applied between all the subtrees of this $cdepth$ between the right leaves of the right and left branches. As soon as the value $cdepth$ becomes equal to zero, all leaves (TPM) will be synchronized [13].

The presented methods use simple calculations and allow you to quickly create a group secret key. However, these algorithms are very vulnerable in cases where an attacker can participate in the protocol and obtain a secret key. Based on this, there is a need for an authentication scheme to protect the group from such attacks.

To build an authentication scheme, it is assumed that the group receives a secret password that can be used to authenticate the exchange protocol. This password can be mapped to some public parameter of neural cryptography, which becomes secret and, therefore, provides suitable privacy.

In [14], an authentication scheme that keeps input patterns secret was proposed. Therefore, only parties that know a secret can synchronize together. Also in [13], a scheme was proposed in which the password was used as a seed for a random number generator that encrypts output bits in the same way as [15, 16, 17].

The BTWE and BTWS algorithms are a logical continuation of the peer learning algorithm. It was also shown [13, 18] that the complexity of the algorithms is logarithmically proportional to the number of parties that need to be synchronized together. From which we can conclude that with the growth of the size of the group, synchronization time of the subscribers of the group will also increase.

Conclusion

The paper discusses existing approaches to using the mutual learning properties of neural networks.

Neural networks have such properties as: mutual learning, self-learning and stochastic behavior, low sensitivity to noise and inaccuracies. These properties allow solving various problems of cryptography with a public key, a private key, hashing, generating pseudorandom numbers, etc. And, importantly, they allow generating a secret key without the need for its subsequent transfer to the recipient, based on synchronization the neural networks with the same structure. This approach makes it possible to significantly increase the security level of cryptosystems with a private key.

In addition to the possibility of creating paired systems, based on this approach, group neural systems with the generation of a secret key can be created by synchronizing all neural networks in the group. However, when using this approach in significantly large groups of users, the problem of minimizing the time spent on group networks synchronizing arises. A lot of factors influence the time of network synchronization in a group:

– frequency of feedbacks when synchronizing every two neural networks or two groups of neural networks (less is faster, more is safer).

– the total quality of group members (more is longer).

– the total quality of elements in the structure of the neural network (less is faster, more is safer).

– the quality of hidden network elements compared to open ones (less is faster, more is safer).

Further research is aimed at modeling and testing systems with various variations of the above parameters. This will determine the degree of influence of each of them on the total time of group networks synchronization.

In the future, modeling and testing will allow us to consider possible ways to modernize existing approaches in order to create a distributed system for group generation of a secret key with acceptable time characteristics and preservation of the cryptographic strength of the system.

References

1. Forouzan B. Cryptography and Network Security. First Edition. McGraw-Hill Publishing, 2007. 721p.

2. Diffie W., Hellman M.E. New Directions in Cryptography. IEEE Trans. Inf. Theory. F. Kschischang IEEE. 1976. Vol. 22, Iss. 6. P. 644-654.

3. Pikovsky A., Rosenblum M., Kurths J. Synchronization: A Universal Concept in Nonlinear Sciences. Cambridge, U.K.: Cambridge Univ. Press, 2003. 411p.

4. Kim C.-M., Rim S., and Kye W.-H. Sequential synchronization of chaotic systems with an application to communication. Phys. Rev. Lett., 2001. Vol. 88, no. 1, P. 014103-1-014103-4.

5. Allam A.M., Abbas H.M. On the Improvement of Neural Cryptography Using Erroneous Transmitted Information with Error Prediction. IEEE transactions on neural networks. 2010. Vol. 21, no. 12. P. 1915-1924.

6. Mcculloch W.S., Pitts W. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics. 1943. Vol. 5. P. 115-133.

7. Hadke P.P., Kale S.G. Use of Neural Networks in Cryptography: A Review. WCFTR'16. 2016. P. 1-4.

8. Kanter I., I, Kessler D. A., Priel A., Eisenstein E. Analytical Study of Time Series Generation by Feed-Forward Networks. Phys. Rev. Lett. 1995. Vol. 75 (13). P. 2614–2617.

9. Klein E., Mislovaty R., Kanter I., Ruttor A., Kinzel W., Saul L.K., Weiss Y., Bottou L. Synchronization of neural networks by mutual learning and its application to cryptography. Advances in Neural Information Processing Systems. 2005. Vol 17. P. 689–696.

10. Червяков Н., Евдокимов А. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. М.: ФИЗМАЛИТ. 2012. 280с.

11. Kinzel W., Kanter I. Neural cryptography. 9th International Conference on Neural Information Processing, 2002. Vol. 3. P. 1351-1354.

12. Малік О.Р. Емпіричне дослідження функції розподілу часу синхронізації нейронних мереж в протоколі обміну ключа. Технологический аудит и резервы производства. 2014. № 4/1(18). P. 26-31.

13. Allamand A., Abbas H. Group key exchange using neural cryptography with binary trees. IEEE CCECE. 2011. P. 783-786.

14. Volkmer M., Wallner S. Entity Authentication and Authenticated Key Exchange with Tree Parity Machines, 2012. Vol. 112. P. 1–6.

15. Allamand A., Abbas H. Improved Security of Neural Cryptography Using Don't-Trust-My-Partner and Error Prediction. International Joint Conference on Neural Networks (IJCNN09), 2009. P. 121-127.

16. Allamand A., Abbas H. On theim provement of the neural cryptography using erroneoustransmitted information with error prediction. IEEE Transaction on Neural Networks. 2010. Vol. 21. P. 1915-1924.

17. Hertz J., Krogh A., Palmer R.G. Introduction to the Theory of Neural Computation. Redwood City: Addison Wesley. 1991. 352 p.

18. Eisenstein E., Kanter I., Kessler D.A., Kinzel W. Generation and Prediction of Time Series by a Neural Network. Phys. Rev. Lett. 1995. Vol. 74. P. 4-6.

References

1. Forouzan, B. (2007), *Cryptography and Network Security*, First Edition. McGraw-Hill, USA, 721 p.

2. Diffie, W., Hellman, M. E. (1976), "New Directions in Cryptography", *IEEE Trans. Inf. Theory*, F. Kschischang, IEEE, Vol. 22, Iss. 6., pp. 644–654., ISSN 0018-9448 – doi:10.1109/TIT.1976.1055638.

3. Pikovsky, A., Rosenblum, M., Kurths J. (2003), *Synchronization: A Universal Concept in Nonlinear Sciences*, Cambridge, U.K.: Cambridge Univ. Press., 411p.

4. Kim, C.-M., Rim, S., Kye, W.-H. (2001), "Sequential synchronization of chaotic systems with an application to communication", *Phys. Rev. Lett.*, vol. 88, no. 1, pp. 014103-1–014103-4.

5. Allam, A. M., Abbas H. M. (2010), "On the Improvement of Neural Cryptography Using Erroneous Transmitted Information with Error Prediction", *IEEE transactions on neural networks*, Vol. 21, no. 12, pp. 1915–1924.

6. Mcculloch W. S., Pitts W. (1943), "A logical calculus of the ideas immanent in nervous activity", *Bulletin of mathematical biophysics*, volume 5, pp. 115-133.

7. Hadke P.P., Kale S.G. (2016), "Use of Neural Networks in Cryptography: A Review", *WCFTR'16*, pp. 1-4.

8. Kanter I., Kessler D.A., Priel A., Eisenstein E. (1995), "Analytical Study of Time Series Generation by Feed-Forward Networks", *Phys. Rev. Lett.*, 75(13), pp. 2614–2617.

9. Klein E., Mislovaty R., Kanter I., Ruttor A., Kinzel W. (2005), "Synchronization of neural networks by mutual learning and its application to cryptography", *Advances in Neural Information Processing Systems*, volume 17, pp. 689-696.

10. Chervyakiv, N., Yevdokimov, A. (2012), *Application of artificial neural networks and residual classes in cryptography [Primenenie iskusstvenny`kh nejronny`kh setej i sistemy` ostatochny`kh klassov v kriptografii*], M., Fizmatlit, 280 p.

11. Kinzel, W., Kanter, I (2002), "Neural cryptography", *9th International Conference on Neural Information Processing*, Vol. 3, pp. 1351-1354.

12. Malik, O.R. (2014), "Empirical study of the time distribution function of neural network synchronization in the key exchange protocol" ["Empi`richne dosli`dzhennya funkczi`yi rozpodi`lu chasu sinkhroni`zaczi`yi nejronnikh merezh v protokoli` obmi`nu klyucha"], *Technological audit and production reserves*, № 4/1(18), pp. 26-31.

13. Allam, A. M., Abbas, H. M. (2011), "Group key exchange using neural cryptography with binary trees", *IEEE CCECE*, pp. 783-786.

14. Volkmer, M., Wallner, S. (2006), "Entity Authentication and Authenticated Key Exchange with Tree Parity Machines", Vol. 112, pp. 1-6.

15. Allam, A.. Abbas, H. (2009), "Improved Security of Neural Cryptography Using Don't-Trust-My-Partner and Error Prediction", *International Joint Conference on Neural Networks (IJCNN09*), pp. 121-127.

16. Allamand, A., Abbas, H. (2010), "On theim provement of the neural cryptography using erroneoustransmitted information with error prediction", *IEEE Transaction on Neural Networks*, vol. 21, pp. 1915 – 1924.

17. Hertz, J., Krogh, A., Palmer, R.G. (1991), *Introduction to the Theory of Neural Computation.* Redwood City: Addison Wesley, 352 p.

18. Eisenstein, E., Kanter, I., Kessler, D.A., Kinzel, W. (1995), "Generation and Prediction of Time Series by a Neural Network", *Phys. Rev. Lett.*, V. 74, pp. 4-6.

Надійшла до редакції 12.06.2020

М.О. АЛЕКСАНДРОВ, Є.О. БАШКОВ
ДВНЗ «Донецький національний технічний університет», м. Покровськ, Україна
eab23may@gmail.com, mykyta.aleksandrov@donntu.edu.ua

ВИКОРИСТАННЯ ВЗАЄМОДІЮЧИХ НЕЙРОННИХ МЕРЕЖ В КРИПТОГРАФІЇ

Обчислювальні потужності комп'ютерної техніки та обсяг переданих даних неухильно зростають з кожним днем. У зв'язку з цим існуючі алгоритми шифрування швидко застарівають через недостатню криптостійкість або пропускну здатність. З'являється потреба у постійному створенні нових методів шифрування або ж модифікації старих. В якості нових алгоритмів можуть виступити алгоритми, що використовують нейронні мережі. В даній статті наведені існуючі алгоритми, створені на основі взаємної синхронізації нейронних мереж. Описано основні принципи їх побудови і застосування. Велике значення надається проблематиці використання нейронних мереж, які беруть участь в груповій синхронізації. Виділено фактори, що впливають як на загальний час синхронізації мереж, так і на потенційну криптостійкість, такі як: кількість вхідних і прихованих нейронів, кількість мереж, що беруть участь в синхронізації, тощо. У висновку були виділені основні напрямки подальшого дослідження, засновані на існуючій проблематиці досліджених методів і підходів.

Ключові слова: нейронні мережі, деревовидні машини парності, синхронізація, криптографія, шифрування.

Н.А. АЛЕКСАНДРОВ, Е.А. БАШКОВ
ГВУЗ «Донецкий национальный технический университет», г. Покровск, Украина
eab23may@gmail.com, mykyta.aleksandrov@donntu.edu.ua

ИСПОЛЬЗОВАНИЕ ВЗАИМОДЕЙСТВУЮЩИХ НЕЙРОННЫХ СЕТЕЙ В КРИПТОГРАФИИ

Вычислительные мощности компьютерной техники и объем передаваемых данных неуклонно растут с каждым днем. В связи с этим существующие алгоритмы шифрования быстро устаревают по причине недостаточной криптостойкости или пропускной способности. Появляется необходимость в постоянном создании новых методов шифрования или же модификации старых. В качестве новых алгоритмов могут выступить алгоритмы, использующие нейронные сети. В данной статье приведены существующие алгоритмы, созданные на основе взаимной синхронизации нейронных сетей. Описаны основные принципы их построения и применения. Большое значение придается проблематике использования нейронных сетей, которые участвуют в групповой синхронизации. Выделены факторы, влияющие как на общее время синхронизации сетей, так и на потенциальную криптостойкость, такие как: количество входных и скрытых нейронов, количество сетей, принимающих участие в синхронизации, и т.д. В заключении были выделены основные направления дальнейшего исследования, основанные на существующей проблематике исследованных методов и подходов.

Ключевые слова: нейронные сети, древовидные машины четности, синхронизация, криптография, шифрование.