Math-Net.Ru

Общероссийский математический портал

G. V. Marshalko, О безопасности схемы биометрической аутентификации, основанной на нейронной сети, *Матем. вопр. криптогр.*, 2014, том 5, выпуск 2, 87–98

DOI: <https://doi.org/10.4213/mvk120>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 54.209.52.79

26 апреля 2023 г., 17:23:14

On the security of a neural network-based biometric authentication scheme

G. B. Marshalko

Technical committee for standardization (TC 26), Moscow

Received 25.09.2013

We show that neuron weights used in neural network-based biometric authentication scheme defined in GOST R 52633 standard series contain all the information on biometric data and secret key of the legitimate user. So, the complexity of evaluating (with known tables of neuron weights) the legitimate user's secret key is equivalent to the complexity of evaluating one solution of a corresponding system of linear inequalities. Thus, first, neuron weights should be considered as a part of a secret key of the authentication system, and, second, several methods for neural networks protection proposed in the standard are inefficient.

Keywords: High-reliable biometric authentication, fuzzy extractors, neural networks, linear programming, GOST R 52633.

О безопасности схемы биометрической аутентификации, основанной на нейронной сети

Г. Б. Маршалко

Технический комитет по стандартизации (ТК 26), Москва

Резюме. Рассматривается определенная в стандарте ГОСТ Р 52633 схема биометрической аутентификации, основанная на нейронной сети. Показано, что используемое нейронной сетью преобразование содержит всю информацию о биометрических данных и секретном ключе законного пользователя. Поскольку нейронная сеть описывается системой линейных неравенств (над $\mathbb{R}$ или $\mathbb{Z}$, сложность нахождения секретного ключа пользователя по известным таблицам весов нейронов эквивалентна сложности нахождения одного решения соответствующей системы линейных неравенств. Таким образом, во-первых, веса нейронов следует считать частью секретного ключа в системе аутентификации, а во-вторых, предложенные в стандарте методы защиты нейронных сетей являются недостаточными.

Ключевые слова: биометрическая аутентификация, нечеткие преобразования, нейронные сети, линейное программирование, ГОСТ Р 52633.

1 Introduction

The widespread distribution of information technologies nowadays sets a difficult task to construct methods for simple, quick but, at the same time, secure techniques for accessing to information resources and services. Classical approaches such as paroles, tokens, smart-cards, RFID-tags usually contradict with ease of handling, because the users should remember symbolic sequences or control the safety of the mentioned devices.

One of the ways to make such access easier is to use biometric technologies (access by retina, fingerprints etc.). The main question here is the selection of algorithm for transforming biometric parameters into a pattern or a cryptographic key that will be used for further access. Such transformation should satisfy typical cryptographic requirement of independence and uniformity of output bits (key bits) and also should take into account on the instability of biometric parameters, which cause the requirement for mapping the set of "close" samples of biometric parameters into the same output vector.

A classical method for implementing such schemes consists in a comparison of an output vector with a predefined (secret) pattern, and, in case they match, granting access to the user. The obvious drawback of such scheme is the need to store secret pattern (a sort of a secret key), which may be difficult, for a example, in mobile devices.

Nevertheless, several approaches overriding this drawback have been developed recently. For example, Dodis et al. in [13] proposed rather general principles for constructing transformations of biometric data into secret keys using some additional non-secret parameters. They also proposed a general methodology for assessing the security of such systems.

The following two frameworks were proposed in [13]:

- secure sketches: a pair of transformations $SS(w) = s$ and $Rec(w', s) = w$, $w \approx w'$, where w is a vector of biometric parameters, s is a non-secret additional vector.
- fuzzy extractors: a pair of transformations $Gen(w) = (R, P)$ $Rep(w', P) = R$, $w \approx w'$, where w is a vector of biometric parameters, R is a secret key, P is a non-secret additional vector.

It has to be mentioned that a secure sketch could be a part of a fuzzy extractor. It is shown in [13] that there exist transformations with "negligible" leakage of a secret information through a non-secret additional vector. The core question is to assess the quantity of this "negligible" leakage for specific transformation. For example, Boyen in [14] shows the possibility to get the secret key when it's used multiple times for several transformations. So, the choice of a transformation does affect the security of biometric systems (as it is in classical cryptography).

A very unpleasant properties of biometric data is non homogeneity and dependence. For a classical code-based types of biometric transformations, this properties leads to the increasing of the code dimension, what is rather unsuitable for implementation. It also prevents the extracting enough randomness from biometric data [4, 9]. And, finally, it is obvious that biometric data of a user could be compromised.

These observations make specialists of JSC PNIEI and Penza State University to propose an approach for constructing biometric authentication system transformations, that potentially manage error correction more efficiently by means of the use of neural networks. Besides, in order to get "more" randomness from biometric data they proposed to use secret biometric data (see, [12, 6]). Later this proposal was adopted as a state standard [1]. By secret biometric data we mean biometric data by means of which user (through speech or handwriting) reproduces a secret password or pin-code. Actually, such system may be seen as a combination of a biometric and a password-based authentication systems. Such combination permits to extract enough randomness from relatively small passwords, see [1] for details.

Further we consider several "known-neural network" attacks on one of the biometric schemes defined by GOST R 52633 standard.

2 Description of the neural network-based biometric authentication scheme

GOST R 52633 standard defines various aspects of neural network-base biometric authentication [6]: general requirements, requirements for testing bases, requirements for testing biometric authentication schemes, requirements for neural network training etc. For our purposes it's enough to consider here general requirements, defined in GOST R 52633.0 standard [1], and a training algorithm, defined in GOST R 52633.5 standard [2].

Fig. 1 shows a general scheme of a neural network-based biometric authentication system. Given biometric data are transformed into a vector of biometric parameters and then processed by an a priori trained neural network. The parameters of the neural network (neuron weights and connections) are stored in, so called, neuronet container. The neural network transforms the vector into the cryptographic key, which at last goes to a sort of cryptographic authentication protocol. If the data are correct then an access is granted to the user.

Now we consider the training algorithm, defined in [2] in details.

The standard assumes two types of neural networks, which may be used in biometric systems:

- continuous biometric data-oriented systems processing data with bad quality;
- discrete biometric data-oriented systems processing data with good quality.

The standard describes training algorithms for one- and two-layer neural networks. Since it's also mentioned that for majority of applications a one-layer network is enough, we consider further only one-layer continuous data-oriented neural networks.

The training procedure is performed through processing several (no less than 21) samples of biometric data of a legitimate user, and no less than 64 samples of randomly selected natural samples.

We will use the following notations further:

Figure 1: The general scheme of a neural network-based biometric authentication system

- N_0 – the number of biometric parameters (external inputs of a network), this number depends on the biometric technology (e.g., for a handwritten biometric data $N_0 = 416$ [3]);
- N_1 – the number of neurons in the network, which is equal to the length of user's secret key (128, 256, 512 etc.);
- n – the number of inputs of each neuron, defined through the analysis of the biometric data quality (in [3] $n = 16$ is used);
- $\vec{\nu} = (\nu_1, \dots, \nu_{N_0})$ – the vector of biometric data;
- $\mathcal{M} = (\mu_i^j, i = \overline{0, n}, j = \overline{1, N_1})$ – the weights of j -th neuron inputs (weight table); the inputs $1, \dots, n$ are connected to the external inputs of the whole network, and the zero input is used for setting up the threshold output function in such a way, that output bits are equidistributed (see [10]);

- $\mathcal{D} = (d_i^j, i = \overline{1, n}, j = \overline{1, N_1})$ – the correspondence table between the external inputs of a network, and neuron inputs;
- $\vec{b} = (b_1, \dots, b_{N_1})$ – the cryptographic key (the output vector of a network);
- a_i – some additional auxiliary weight coefficients dependent on the biometric technology (and so these coefficients are known).

In order to set neuron weights, sample expectations $\overline{\nu}_t^c$ and dispersions $S^2(\nu_t^c)$ of the t -th biometric parameter $t = 1, \dots, N_0$ for samples of a legitimate user ($c = 0$) and randomly chosen samples ($c = 1$) have to be evaluated. The output value of the neuron for samples of a legitimate user is set by the desired value of a secret key bit. The schematic description of the training algorithm is presented on Fig. 2

Figure 2: The schematic description of the first neuron training procedure

The training algorithm [2]

- Input: vectors of sample expectations $\overline{\nu}_i^z$ and dispersions $S^2(\nu_i^z)$, $z = 0, 1$, $i = \overline{1, N_0}$, a cryptographic key $\vec{b}$;

- Output: a weight table $\mathcal{M}$, a correspondence table $\mathcal{D}$.
1. set tables $\mathcal{M}$ and $\mathcal{D}$ to zero
  2. For $j = 1, \dots, \left\lceil \frac{N_0}{n} \right\rceil$ do 3
  3. For $i = 1, \dots, n$ do 4
  4. $\mu_i^j = \frac{(-1)^{b_j} (\nu_{jn+i}^1 - \nu_{jn+i}^0)}{a_2 \sqrt{S^2(\nu_{jn+i}^1) + \sqrt{S^2(\nu_{jn+i}^0)}}}$, $d_i^j = jn + i$
  5. For $j = \left\lceil \frac{N_0}{n} \right\rceil + 1, \dots, N_1$ do 6
  6. For $i = 1, \dots, n$ do 7 and 8
  7. $\tau = prnd(\mathcal{D})$
  8. $\mu_i^j = \frac{(-1)^{b_j} (\nu_\tau^1 - \nu_\tau^0)}{a_2 \sqrt{S^2(\nu_\tau^1) + \sqrt{S^2(\nu_\tau^0)}}}$
  9. For $j = 1, \dots, N_1$ do 10
  10. $\mu_0^j = norm(\mu_1^j, \dots, \mu_n^j)$

The algorithm uses two following procedures:

- $prnd(\mathcal{D})$ – a pseudo random selection of biometric parameters (coordinates of the external input vector) in such a way that the following conditions hold:
  - the minimum and maximum frequencies of the usage of an input parameter differ no more than by 2,
  - every 4 neighbor neurons do not have connections to common external inputs;
- $norm(\mu_1^j, \dots, \mu_n^j)$ – evaluation of the zero input weight.

3 Proposed attacks on the neural network-based biometric authentication scheme

The standard [1] among others defines the following security threats :

- guessing the input vector of biometric parameters $\vec{v}$;
- extraction of the secret information from the structure and parameters of the neural network (tables $\mathcal{M}$ and $\mathcal{D}$);
- guessing the secret key $\vec{b}$;

As main countermeasures for the described threats the standard proposes to use software integrity control and audit of computational processes that goes alongside of biometric authentication. In order to resist the first mentioned threat, the standard suggests increasing the complexity of a neural network (number of inputs, outputs, layers and connections of each neuron). In order to resist second – restricting an access to neuronet containers (tables $\mathcal{M}$ and $\mathcal{D}$), increasing dimensions of tables, creating unique neural network configurations for each user. It should be mentioned, that standard also defines other types of security threats, which is out of the scope of this paper since they do not deal with the internal structure of the neural network.

3.1 Neural network tables compromise

Consider the consequences of a compromise of a neuronet container (a weight table $\mathcal{M}$ and a correspondence table $\mathcal{D}$). We can see from the described training algorithm that the signs of weights of the neuron are set in order to provide the correct bit of a secret key as the output of the neuron for a legitimate user. If the desirable value of this bit is zero, then the sign of the weight is equal to the sign of the corresponding difference $(\overline{\nu_i^1} - \overline{\nu_i^0})$, in the other case it has the opposite sign. Considering the requirement for the equiprobable usage of the external inputs, the expected number of connections of an external input could be roughly assessed as $\left\lceil \frac{N_1 n}{N_0} \right\rceil$.

Then we conclude that when the tables $\mathcal{M}$ and $\mathcal{D}$ are compromised, an adversary learns two variants of a secret key: the correct key and its negation.

Indeed, let the first bit of a secret key is equal to $b_1 \in \{0, 1\}$, from the first row of $\mathcal{M}$ we get the signs of the weights μ_i^1 , $i = \overline{1, n}$. By looking further through the correspondence table $\mathcal{D}$ we get neuron numbers $(j_1, \dots, j_z)$, $z \leq \left\lceil \frac{N_1 n}{N_0} \right\rceil$, which are also connected to external inputs $i = \overline{1, n}$. According to the training algorithm, their weight's moduli are equal to the weight's moduli of the first neuron: $|\mu_i^{j_s}| = |\mu_i^1|$, $i = \overline{1, n}$, $s = \overline{1, z}$, and their signs are defined by the secret key bits b_{j_s} . Therefore, we get bits $b_{j_1}, \dots, b_{j_z}$ from the signs of weights of each neuron. Carrying on the same way further, we get other bits of a secret key, and, finally, in the best case we could have two variants of a secret key: $\vec{b}$ and $\vec{b} \oplus \vec{1}$ (one of them is the correct one). The actual number of variants of the secret key depends on the biometrical technology and specific biometric parameters of a user, but since the number of components of the biometric vector exceeds the number of bits in the secret key, we would always have a reduction of a possible secret keys set.

Note that it is also possible for the adversary to get secret key when only weight table $\mathcal{M}$ is compromised. In this case we can distinguish external inputs connected to neurons by the moduli of their weights (if we have weights with the same modulus for two neurons, they are likely to be connected to the same external input). However the possibility of such attack depends on the biometric technology that is used.

3.2 Estimation of the legitimate biometric parameters through the compromised neural network tables

Suppose that an adversary is unable to use directly the secret key obtained from compromised tables. In such case, he has to get the correct vector of biometric parameters, in order to use it for standard access procedure. For example, when biometric parameters is obtained from handwritten pass phrase, by a digital pen, an adversary could make his own pen, which would send to the biometric authentication system the predefined vector of biometric parameters.

A neural network consisting of neurons defined earlier (see Fig. 2) may be described in terms of a system of linear inequalities:

$$\mu_0^j + \mu_{d_1}^j \nu_{d_1}^j + \mu_{d_2}^j \nu_{d_2}^j + \dots + \mu_{d_n}^j \nu_{d_n}^j \begin{matrix} \leq \\ \geq \end{matrix} 0, j = 1, \dots, N_1,$$

where the specific symbols of inequalities are defined by the values of secret key bits (recall that two variants of the secret key are known from the compromised tables) and by the values of the coefficients – by the training algorithm.

So, we can reduce the problem of estimation of biometric parameter vector to the problem of finding one solution satisfying one of two systems of linear inequalities associated with the neural network. Note that the systems always has a solution.

It was mentioned in [5] that the complexity of solving of such system by the classical Chernikova algorithm [11] is $O\left(N_0^{\frac{N_1}{2}+1}\right)$ operations. Indeed, such estimate is known (see [7]), but it is true (a common situation for optimization theory) for the specific case of "extremely bad" systems, that produce the maximum number of steps of the algorithm. Moreover, this estimate is true for the algorithm describing the whole set of solutions of the system.

More precisely, such algorithms consist of series of transformations of the system of inequalities. On every step new conditions (inequalities) may emerge, which, in their turn, should also be solved. This leads to the increasing the number of steps of the algorithm. There exist examples of systems of inequalities with the maximum number of steps of Chernikova algorithm, which leads to the mentioned estimate.

Note that along with the described algorithm there exists a class of interior points algorithms which may effectively solve systems of linear inequalities when initial approximation is known (some restriction on starting vector). Specifically, in [8] several algorithms of such kind are proposed and results of experiments are presented. These results show that the algorithms are quite effective even for size of the system larger than the size of the system in our case.

We have conducted experiments that showed that finding a solution of a randomly chosen system (128 inequalities of 416 variables) may be effectively done even on a netbook (Asus EeePC 1003 with Atom N208 and 1 GB RAM, we used Mathematica 7 software).

Finally, we mention again that it is not necessary to describe the whole set of solutions: to find only one (which is the correct vector of biometric parameters) is enough.

3.3 A "black box" attack

Suppose that an attacker has no direct access to the neural network tables $\mathcal{M}$ and $\mathcal{D}$, but he is able to observe the neural network output $\vec{b}$ evaluated on arbitrary chosen vectors $\vec{v}$ of biometric parameters – a classical "black box" attack. It is hard to imagine whether such situation could be realized in practice, since it's more likely that the neuronet container is compromised, so, the following results mostly characterize properties of the discussed neural transformation.

In a "black box" attack, we have to evaluate the weights and connections of a neural network, and then apply earlier proposed algorithms.

Consider a simple dichotomic algorithm for evaluating neuron weights. Note that weights of each neuron could be treated separately (this follows directly from the training algorithm). Let's observe i -th neuron. Choose at random N_0 input biometric vectors $\vec{v}_i^0$, $i = 1, \dots, N_0$, in such a way, that the corresponding output bit is equal to zero and N_0 input vectors $\vec{v}_i^1$, $i = 1, \dots, N_0$, with corresponding bit equal to one. Denote neuron function as f , $f: \mathbb{R}^{N_0} \rightarrow V_1$. Consider intervals $[\vec{v}_i^0, \vec{v}_i^1]$, $i = 1, \dots, N_0$, and fix parameter $0 < \epsilon < 1$ which describes the algorithm accuracy.

Algorithm for neuron weights evaluation

- Input: $[\vec{v}_i^0, \vec{v}_i^1]$, $i = 1, \dots, N_0$, ϵ – parameter.
- Output: Weight table $\mathcal{M}$ and correspondence table $\mathcal{D}$.

1. $s = 0$
2. $\vec{x}^{(i)}(s) = \vec{v}_i^0$, $\vec{y}_i(s) = \vec{v}_i^1$, $i = 1, \dots, N_0$
3. If $\max_i \left(\sqrt{\sum_{j=1}^{N_0} (x_j^{(i)}(s) - y_j^{(i)}(s))^2} \right) > \epsilon$ do 4 – 7, else – 8
4. $s = s + 1$
5. For $i = 1, \dots, N_0$,
6. For $j = 1, \dots, N_0$, $x_j = \frac{x_j^{(i)}(s-1) + y_j^{(i)}(s-1)}{2}$
7. If $f(\vec{z}) = 0$, then $\vec{x}^{(i)}(s) = \vec{z}$, else $\vec{y}^{(i)}(s) = \vec{z}$
8. Evaluate the hyperplane equality $a_1 z_1 + a_2 z_2 + \dots + a_{N_0} z_{N_0} + a_0 = 0$:

$$\begin{vmatrix} z_1 - x_1^{(1)}(s) & z_2 - x_2^{(1)}(s) & \dots & z_{N_0} - x_{N_0}^{(1)}(s) \\ x_1^{(2)}(s) - x_1^{(1)}(s) & x_2^{(2)}(s) - x_2^{(1)}(s) & \dots & x_{N_0}^{(2)}(s) - x_{N_0}^{(1)}(s) \\ & & \ddots & \\ x_1^{(N_0-1)}(s) - x_1^{(1)}(s) & x_2^{(N_0-1)}(s) - x_2^{(1)}(s) & \dots & x_{N_0}^{(N_0-1)}(s) - x_{N_0}^{(1)}(s) \end{vmatrix} = 0$$

The moduli of non zero coefficients in hyperplane equation correspond (with given accuracy ϵ) to the moduli of neuron weights. The positions of these coefficients determine neuron connections in the correspondence table $\mathcal{D}$. We have to make several remarks. First, in some cases the algorithm may fail. This would happen, when selected vectors belong to a hyperspace with smaller dimension, but the probability of such event is negligible. Second, the algorithm may be improved: we can run additional step in order to determine neuron connections (essential variables of f) by consequently changing coordinates of biometric vectors. After this step we would need only $2N_0$ instead of $2n$ vectors for evaluating weights. Nevertheless, since the practical dimensions are moderate such improvement is not critical.

Suppose that moduli of the weights are evaluated. Now we have to place the correct sign for each of them. This could be done, first, by evaluating the output from the zero vector – the sign of the output value is equal to the sign of μ_0 , then, we can determine other signs by evaluating input vectors with one non zero essential variable. Finally, we get the full description of the neural network, and can evaluate, further, the secret key or the correct vector of biometric parameters.

Let's assess the complexity of the method. Suppose that the coordinates of the input vector are bounded: $|\nu_i| \leq B$, $i = 1, \dots, N_0$, then the expected length of initial interval is $B\sqrt{N_0}$, and after k steps of the algorithm the length would be $\frac{B}{2^k}\sqrt{N_0}$. From that, for a fixed ϵ we can find the required number of steps: $k \geq \frac{1}{2} \log_2 \left(\frac{B^2}{\epsilon^2} N_0 \right)$. On each step we have to perform N_0 floating point operations. Then we have to evaluate the determinant of $(N_0 \times N_0)$ -matrix, this requires $\frac{N_0(N_0-1)}{2}$ operations. Finally, we get the signs of the neurons for $n+1$ operations. The overall complexity of non-optimized algorithm is $N_1 \left[\frac{N_0^2}{2} \log_2 \left(\frac{B^2}{\epsilon^2} N_0 \right) + \frac{N_0(N_0-1)}{2} \right]$ floating point operations. If, for example, $\epsilon = 0.001$, $B = 128$, $N_0 = 416$, $N_1 = 128$ then the complexity is $4.8 \cdot 10^8$.

4 Conclusion

The proposed attacks show that the neural network-based authentication scheme defined in GOST R 52633 standard does not withstand the compromise of the neuronet container (the tables that describe the neural network). Hence, the conclusion of [12] that "...neural technologies allow to securely hide from unauthorized users personal biometric information of a legitimate user" is incorrect. It should be mentioned that in the later article [6] the authors admit the risk of a neuronet container compromise, and propose a "selfcipherring", when the output vector $\vec{b}$ of length N_1 is split into m subvectors $\vec{b}_i$, $i = 1, \dots, m$, of length $\frac{N_1}{m}$. The first m neurons trained as it was described earlier, and subsequent $N_1 - m$ are encrypted with key $\vec{b}_1$, subsequent $N_1 - 2m$ with key $\vec{b}_1 || \vec{b}_2$ etc. In case of an access of a legitimate user we will get the correct key by sequential decryption.

But we can perform the same decryption with the described algorithms since all neurons could be treated separately. The only problem here is that we, perhaps, have to

run exhaustive search through all possible values of $\vec{b}_1$ (since for several m ($m \leq \lceil \frac{N_0}{n} \rceil$) biometric vector coordinates may be used only once – see training algorithm for details). Nevertheless, taking into consideration the proposed in [6] value for m ($m=32$), this would not increase the number of operations significantly.

Finally, returning to the properties of fuzzy extractors, we see that the described neural network-based authentication system is not of this kind, and it may be secure if the neuronet container (which is actually a part of a secret key) is securely protected. We also conclude that the requirements of the standard are incomplete.

References

- [1] GOST R 52633.0-2006. Information protection. Information protection technology. Requirements to the means of high-reliability biometric authentication.
- [2] GOST R 52633.5-2011. Information protection. Information protection technology. The neural net biometry-code convertor automatic training.
- [3] *Efimov O. V., Funtikov V. A., Jazov U. K.* The neuronet converter "biometry-code" structure and interconnections choice strategy // Neurocomputers: development, application. — 2009. — V. 6. — P. 14–16.
- [4] *Kuznetsov V. V., Ushmaev O. V.* Algorithms for protected biometric verification based on the binary representation of fingerprint topology // Informatics and its applications. — 2012. — V. 6. No 1. — P. 132–140 (in Russian).
- [5] *Majorov A. V.* Biometric hash-function // In proc. TPCYR-2009, JSC PNIEL. — 2009 (in Russian).
- [6] *Nazarov I. G., Efimov O. V., Yazov Y. K.* The package of national standards, ensuring biometric and neural network protection of mass circulated personal data privacy // Neurocomputers: development, application. — 2012. — V. 3. — P. 9–16.
- [7] *Zolotykh N. Yu.* Deciphering of threshold and close to threshold q-ary logic functions. Ph. D. Thesis. NNGU, 1998 (in Russian).
- [8] *Zorkal'tsev V. I., Filatov A. Yu.* New variants of dual interior point algorithms for systems of linear inequalities // Zh. Vychisl. Mat. Mat. Fiz. — 2004. — V. 44. No 7. — P. 1234–1243.
- [9] *Ushmaev O. S.* Protected biometric authentication, CSE Days. Theory 2011, Ekaterinburg, 2011 (in Russian).
- [10] *Haykin S.* Neural Networks: A Comprehensive Foundation, 2nd Edition. — Prentice-Hall, 1999.

- [11] *Chernikova V.* Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities // Zh. Vychisl. Mat. Mat. Fiz. — 1965. — V. 5. No 2. — P. 334–337.
- [12] *Jazov U. K., Nazarov I. G., Ivanov A. I., Yefimov O. V.* Neuronet technology for protection of personal data // Neurocomputers: development, application. — 2009. — V. 6. — P. 5–10.
- [13] *Dodis Y., Reyzin L., Smith A.* Fuzzy extractors: How to generate strong keys from biometrics and other noisy data // In proc. EUROCRYPT'04, LNCS 3072. — 2004. — P. 523–540.
- [14] *Boyen X.* Reusable cryptographic fuzzy extractors // in proc. 11th ACM Conference on Computer and Communication Security. — 2004.