Anna GROCHOLEWSKA-CZURYŁO[1], Marek RETINGER[1]

Poznan University of Technology (1)

doi:10.15199/48.2017.01.15

Biometrics as an authentication method in a public key infrastructure

Abstract. *This paper presents proposed authentication method for PKI based on fingerprint transformations discussed in "Secure fingerprint hashes using subsets of local structures". The authentication method uses Shamir secret sharing and blind signatures techniques. Biometrics data are stored as a one share. The latter approach helps to improve the security of biometric templates and allows to use them within PKI.*

Streszczenie. *Niniejszy dokument prezentuje koncepcyjną metodę uwierzytelniania biometrycznego w ramach infrastruktury klucza publicznego, bazującą na przekształceniach obrazu odcisku palca opisanych w "Secure fingerprint hashes using subsets of local structures". Mechanizm uwierzytelniania został zbudowany w oparciu o schemat dzielenia sekretu Shamira oraz technikę ślepych podpisów cyfrowych, co pozwala na transformację danych biometrycznych do postaci pojedynczego udziału. Takie podejście zapewnia akceptowalny poziom bezpieczeństwa oraz współdziałanie systemu z PKI.* ***(Biometria jako metoda uwierzytelniania w infrastrukturze klucza publicznego)***

Keywords: authentication, biometrics, PKI, fingerprints, security
Słowa kluczowe: uwierzytelnianie, biometria, PKI, odcisk palca, bezpieczeństwo

Introduction

Modern biometric-based authentication systems [11] were built in the way that provides the highest accuracy level and at the same time high performance. This approach is valid for systems working in a closed environment. The situation changes greatly when we decide to work within an open environment, like e.g. the Internet. User requirements are still very important, however the security aspect is even more significant. Currently no standardized, global biometric system exists, however some research teams have published several important documents describing the problem of using biometrics techniques and public key infrastructure (PKI) [5] [7] [16] together.

In 2007 a group of researchers from Mexico [8], designed a system using biometric authentication on a small scale. Their solution were based on modifications introduced to digital certificates (X.509) and setting up a unique mobile certificate authority (CA). The biometrics validation and recognition module was taken from open-source project [3]. The biggest disadvantage of this system was an unacceptable certificate size, depending on the chosen cryptographic method.

At the end of 2010, a valuable document [2] describing how to use biometrics with PKI was published. A group from United States of America, proposed revocable biotokens, which were a milestone in how biometric systems are perceived. The new look at that problem, contributed in building session biotokens which, when compromised, do not affect the possibility of obtaining an original biometric feature. They have also presented two attacks which bio-systems should be resistant to.

Finally, in 2013, South Korean researchers presented their own solution [10], based on advanced biometric template processing. They linked up biometric features with clients certificate public key, which allowed them to revocate digital certificate and processed values at the same time. What is more, proposed system neither required certificate modifications nor extensions.

System overview

Our system was built based on the experience and conclusions from related works [2] [8] [10]. We defined several assumptions, which our system has to meet. It allowed us to provide expected level of security. They are listed below.

1. Biometric authentication process has to be directly linked with client's digital certificate's public key.
2. Client should store only a minimal amount of data, which is required to respond the server requests.
3. Compromising processed biometric data stored on the server, should not allow to recover the original biometric template.
4. Only the hash value is revocable, not the raw biometric template.

Achieving criteria described above resulted in a system being able to validate biometrics data without an explicit template.

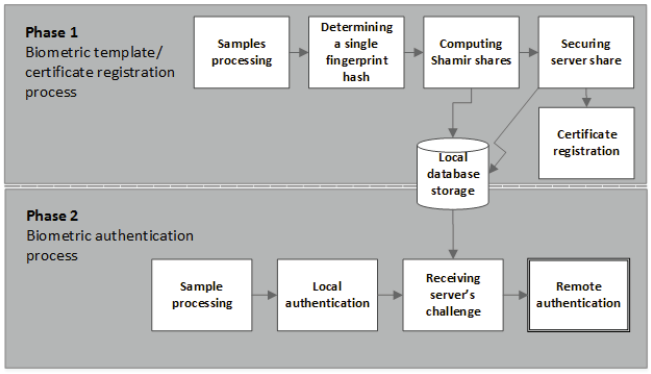

Fig. 1. Generalized system's model. Phase 1 contains blocks representing processing steps in our system — from reading the input data, through processing minutiae structures and generating Shamir shares, up to digital certificate registration. Phase 2 shows how the authentication process works. The system also contains a shared local database — *shared* means that database content is used in both phases.

Figure 1 presents an overview of our system. The solution has two data processing paths — first path involves biometric template or certificate registration process, the second one involves biometric authentication process only. Some data is stored in a local database. It is important, because some values generated in the first step are required in the second one. This is described later in the paper.

Samples processing

Samples processing was presented in [9] in detail. The whole process is composed of minutiae detection, triangles determination (based on minutiae [4]), triangles translations, triplets of triangles combinations, synthetics fingerprints generation, quantization and finally hash values generation. In general, the steps above were well described, however quantization was in a certain sense skipped — important information about buckets ranges was not mentioned. Our team conducted some additional research and found the proper ranges, allowing us to balance the output values. We

have used 5 buckets[1].

First of all, we have gathered fingerprint images and built temporary database. Further, we have randomly chosen 100 elements. Afterwards, we have determined minutiae. Next we have built triangles and translated them in the coordinate system as described in the mentioned document. Our dedicated application counts the number of occurrences of each of x, y, θ and simply normalizes all the values into <0;1> range. Those normalized *data* is put together into one, averaged result which is shown in figures 2, 3 and 4 separately for each coordinate.

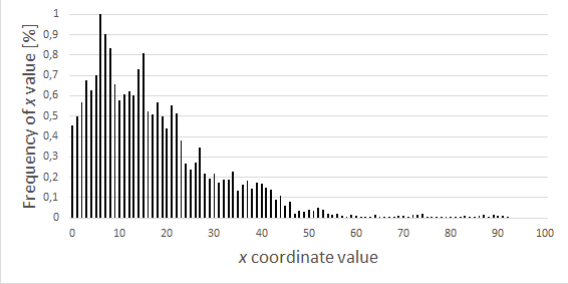

Fig. 2. Averaged distribution of values for x coordinate.

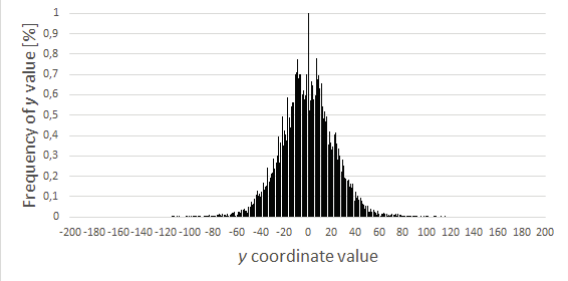

Fig. 3. Averaged distribution of values for y coordinate.

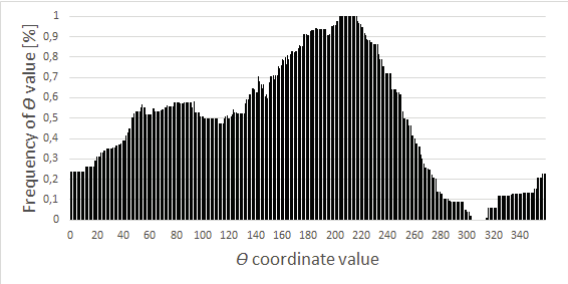

Fig. 4. Averaged distribution of values for θ coordinate.

As we can see, for coordinates of x and y, there is a distribution regularity. Buckets should be placed mainly in the first half of diagram (<0; 50>) for x coordinate and in range <-25;25> for y, because it contains the largest amount of corresponding values. In the case of θ, which represents the minutiae rotation direction, result is quite different. These values were irregular, so was impossible to divide range into proper slices where output would be fully balanced. We have chosen a small group[2] of fingerprints and determined the ranges as well as possible in an experimental way.

Determining a single fingerprint hash

Processing model from previous section, generates thousands of hash values. The choice of a single representative value from a huge group in a random way is in practice impossible. Our solution is based on sets intersections. We capture several fingerprint pictures and we look for intersections as long as we obtain a single value or a group of recurring values. Our tests have shown that this process is relatively fast. The algorithm is described below.

1. Load fingerprint hashes set and make it a base set B.
2. Load fingerprint hashes set F (different fingerprint image).
3. Build common elements set E, which contains hashes belonging to both sets ($\forall e \in E, \quad e \in B \wedge e \in F$).
4. If the E set is empty (probably fingerprint hashes set was malformed) go to step 2. Otherwise go to step 5.
5. Make set E a new base set B.
6. If cardinality of the base set is equal to 1, end intersection process and go to step 7. Otherwise go to step 2.
7. Determine random number S_H, which will be a salt.
8. Generate base hash value (formula 1).

$$h(H) = h(h(H)||S_H) \tag{1}$$

where H is the chosen hash value, h is SHA-256 (*Secure Hash Algorithm*) hash function.

Computing Shamir shares

Single hash value can not be used in an authentication model. It could be captured easily and used by an adversary. Our system transforms data in such a way that only the person who has required hash and knows corresponding secret can successfully complete the server's challenge. We decided, that Shamir secret sharing [12] [14] [15] is an appropriate solution (*3 of 3*). First, it is required to generate three coefficients

$$\begin{aligned} a_0 &= secret \\ a_1 &= HMAC_s(kbio) \\ a_2 &= HMAC_s(a_1) \end{aligned} \tag{2}$$

where $secret(s)$ is a user key, $kbio$ is a single hash value.

which are necessary if the process should be repeatable. The secret is important part of the whole system — we have found potential vulnerability in [9] associated with quantization process.

Researches assumed that quantization produces 63-bit output. Based on this information, we simply obtained 2^{63} combinations of inputs data. However, system use only 5 quantization buckets (possibly one more extra) so combinatorial space is reduced to $2.2 * 10^{16} \approx 2^{54}$ only.

Usage of additional user key eliminates it perfectly. In this part, we adopted HMAC (*keyed-Hash Message Authentication Code*) algorithm with SHA-512 hash function.

$$S(s, x, a, p) = s + \sum_{j=1}^{2} a_j x^j \, mod \, p \tag{3}$$

where S is a Shamir's share, x is a share's identifier (0,1 or 2), a is an array of coefficients and p is a big prime number[3].

Formula 3 is applied to compute shares S_0, S_1 and S_2.

[1] Structure which transform defined range of values into the same sequences of bits.

[2] Set contains 15 fingerprint images.

[3] Length must be at least 1024 bits.

Securing server share

Processing raw server share (S_2) would be insecure, because some values would be used by several service providers. In consequence data which should be protected could be compromised. What is more, the raw value cannot be used in a group $\mathbb{Z}_N$ for unambiguous transformations — this causes authentication problems. We generate *secure server share*, which associates certificate public key with a server share. For S_2 we compute a random index (RI), such that

$$GCD(RI * S_2, n) = 1 \tag{4}$$

where GCD is a Greatest Common Divisor, n is a client's public key module.

Finally, we generate the secure server share (S_{SS}) as shown below.

$$S_{SS} = (RI * S_2)^{E_k} \tag{5}$$

where E_k is a client's public key.

Described method is based on a blind digital signatures approach [1]. Proposed dependency allows us to revocate server's share at the same time as keys pair. Finally, the result is encrypted with a client's digital certificate public key.

Certificate registration

Last step in the first part of preparation is registration of X.509 certificate for the client, with critical extension defined, containing a second Shamir share (S_1). It is important because S_1 will be required in a local authentication process described further in section *Local authentication*.

Local database storage

As was said before, the system should store as little data as possible. We only save a value of big prime number P, a random index value RI and a chosen hash's salt. Optionally, local database can contain client's certificate fingerprint or certificate public key.

Local authentication

Local authentication is a process preceding remote authentication to the server. It was introduced because biometric data processing is not as fast as classical challenge-response method. Furthermore, timestamps cannot be used (they will expire). At the beginning, we obtain a single hash value as shown in section *Determining a single fingerprint hash*. System also requires S_1 share, the random index RI, the big prime number P and the client secret s. Secret value must be entered by the client himself. After that, we compute coefficients for Shamir secret sharing and obtain shares S'_0, S'_2, S'_3. If determined S'_1 equals S_1, stored in the digital certificate extension, we can be sure that client is the person who he claims to be.

Receiving server's challenge

If we are sure that chosen single hash value is correct, two-way authentication process can be initiated. We have not described this part here, because SSL/TLS[4] protocol can be used instead. In a secure communication channel, we begin biometric authentication process. Whole authentication protocol was partially described below and in the next section.

[4] Protocol provides optional two-way certificate-based authentication mode.

1. Server randomly generates a challenge value R_V.
2. Server determines R_V^* value as described below

   $$R_V^* = R_V \cdot S_{SS} \, mod \, n \tag{6}$$

   where S_{SS} is the secure server share, n is the client's public key module.

3. Server sends a message m, such that

   $$m = (R_V^* || T_B)^{D_s} \tag{7}$$

   where T_B is a server timestamp, D_s is a server's private key.

4. Client receives message m and checks the timestamp validity.
5. Client starts releasing procedure.

Client is obligated to release R_V value and send it back to the server. This part is called *Remote authentication*.

Remote authentication

After message m is received and R_V^* value is obtained, client begins releasing procedure. The original value is computed based on the recovered share S_2. The whole process requires determination of a value which is an inverse of the secure server share in group $\mathbb{Z}_N$ and several RSA encryptions.

1. Client begins releasing procedure:
   - determine R'_V

     $$R'_V = (R_V^*)^{D_k} \, mod \, n \tag{8}$$

   - obtain single hash value as described in sections *Samples processing* and *Determining a single fingerprint hash*;
   - compute Shamir shares;
   - determine inverse of the secure share in group $\mathbb{Z}_N$, such that

     $$S_S = (S_2 \cdot RI)^{-1} \tag{9}$$

     where S_2 is an original Shamir share value (last one), RI is a random index value stored in local database.

   - remove the blindness of R'_V

     $$R_V^{**} = R'_V \cdot S_S \, mod \, n \tag{10}$$

   - encrypt R_V^{**} using your own public key and obtain R_{VK} value. R_{VK} is the released challenge value.
2. Client sends message m, such that

   $$m = (R_{VK} || T_A)^{D_k} \tag{11}$$

3. Server receives message m, verifies the timestamp value and compares R_{VK} with R_V. If R_{VK} is equal to R_V, the authentication process was completed successfully — client has some data, which allows him to release R_V.

The above process can be expressed by formula 12.

$$R_{VK} = \Big[[(RI * S_2)^{E_k} * R_V \, mod \, n]^{D_k} * (RI * S_2)^{-1} \, mod \, n \Big]^{E_k} \, mod \, n \tag{12}$$

Simplified authentication process was shown in figure 5.

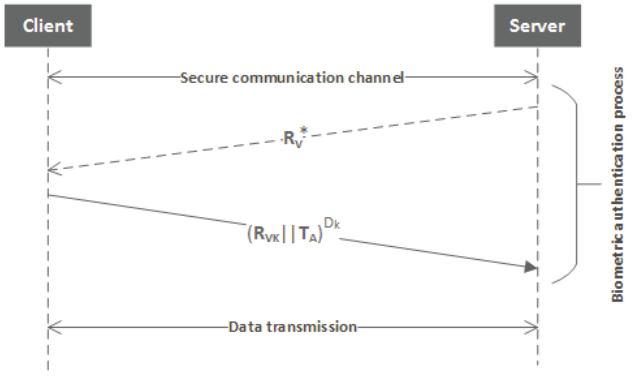

Fig. 5. Simplified biometrics authentication sequence diagram. Classical two-way authentication was skipped in the figure. T_A is a client's timestamp, D_k is a client's private key.

Main results

We have tested our system using *SecuGen Hamster IV* device and our own fingerprint set, which contains over 600 images. We did not use FVC set (*Fingerprint Verification Competition*) [6], due to the fact that it was built for systems matching templates with input data — in our solution we cannot obtain a raw template. Performed tests are real tests, which includes Shamir shares comparison. Results contain 4 chosen, representative tests, where FRR (*False Rejection Rate*) and FAR (*False Acceptance Rate*) were tested. Both are presented in table 1.

Table 1. Chosen tests results. Grayed rows represent values obtained in the document mentioned earlier. Tests results are not corresponding.

Type	Test 1	Test 2	Test 3	Test 4
FAR	0,0%	0,0%	0,3%	0,0%
FRR	58,3%	30,8%	15,4%	76,9%
Type	**DB1**	**DB2**	**DB3**	**DB4**
FAR	3,8%	3,2%	0,3%	2,4%
FRR	22,1%	19,9%	57,0%	39,7%

Obtained results are similar to those described in [9], however different test sets were used — researchers were able to compare template with a read from an input and use the FVC set (using raw data). Their comparison method was based on set intersection, which is partially comparable with the single hash determining, described in *Determining a single fingerprint hash* (first common set determining). The biggest difference is in the biometric subsystem's output — our subsystem produces a *single* hash, which is common to all analysed fingerprints, where [9] produces a *set* containing all hashes for the obtained fingerprint. Summarizing the above, the presented system is more sensitive to distortions and so it is more reliable.

Our system has obtained highest FRR value for *Test 3*, which is an acceptable upper limit for biometric purpose. We also gathered information from an enrolment process, which shows how many intersections are required to obtain a single hash (table 2).

Chosen set of tests presents dependency between FRR value and enrolment process. Greater number of samples in enrolment process causes lower FRR values, however FAR is still at the same, promising level.

System security

System security is based on considerations contained in [9] and details described in this section. We have analysed 3 possible attacks — attack on triangles coordinates, attack on quantization process and attack on Shamir schema.

Table 2. Intersections of chosen tests. Greater number of samples in the enrolment process caused lower FRR value.

Sample's number	Test 1	Test 2	Test 3	Test 4
1	265529	598526	304818	209272
2	172	322	953	74
3	8	84	498	4
4	6	63	159	1
5	4	6	18	
6	1	4	8	
7		1	7	
8			5	
9			1	
FRR	58,3%	30,8%	15,4%	76,9%

Description of triangles coordinates attack requires some assumptions:

- each minutiae is described by x, y and θ coordinates,
- maximal euclidean distance between grouped minutiae is less than or equal to 100 units (pixels),
- at least 7 minutiae in triplet have to be unique,
- fingerprint image size is 300x300 pixels.

Attack complexity on a single triangle can be described as

$$101^3 * 401^3 * 360^3 = 3.1 * 10^{21} \tag{13}$$

where first value is x (<0;100>), second is y (<-200; 200) and the last one is θ (<0;359>).

Our system is based on triples of triangles, what significantly improves security at this level. The minimal combinatorial space is

$$101^7 * 401^7 * 360^7 = 1.4 * 10^{50} \tag{14}$$

and the maximal reaches

$$101^9 * 401^9 * 360^9 = 2.9 * 10^{64} \tag{15}$$

Attack on quantization process was presented in [9], however it is possible to reduce combinatorial space which was mentioned in section *Computing Shamir shares*. In the original attack, adversary checks all possible sequences of bits

$$2^{6*3+6*3+9*3} = 2^{63} \tag{16}$$

where the numbers mean:

- 3 — quantization output length in bits,
- 6 — total number of x and y coefficients used in quantization process,
- 9 — total number of θ coefficient used in quantization process.

If we generate all possible sequences of three bits and we are using only 6 buckets, it is possible to reduce complexity to $\approx 2^{54}$

$$6^6 * 6^6 * 6^9 = 2.2 * 10^{16} \approx 2^{54} \tag{17}$$

Note that we have used only 6 x and y coefficients due to repeatable (0,0) coordinate in each triangle. This is caused by translation process.

Attack on Shamir schema can be divided into two sub-attacks — attack on *Shamir's* input and server share generation. First of them could be based on input space reduction where the space is reduced to $\approx 2^{54}$. In our system,

we used HMAC algorithm with SHA-512 hash function. The key is defined by user, but it cannot be shorter than 64 bits (8 ASCII characters), which implicates that adversary has to generate 2^{512} values or $(2^{54})^{kl}$ values, where kl is a user key length. Server share generation attack is much more dangerous. Public key module is publicly known. Adversary has to find two numbers (RI and S_2), such that

$$NWD(RI * S_2, n) = 1 \tag{18}$$

where S_2 is the original Shamir share value (last one), RI is the random index value.

The length of RI is (by default) 64 bits, which means that attacker has to generate and store 2^{64} values and further, to compare them with values satisfying the above condition. Note that Adversary has to know all cryptographical keys used in two-way authentication phase. Moreover, the attacker is able to gain access only to data associated with one selected server and only for time limited by digital certificate validity (default: 1 year).

Conclusions

This paper presents a biometric-based authentication system for public key infrastructure, which works on secured templates. Our solution, as one of just a few, provides security mechanism at biometric feature level. As opposed to other approaches mentioned in the paper, we applied sample processing producing a single secure hash value in a cryptographic sense and Shamir secret sharing with blind digital signatures. Also, the system structure was presented with some protocol details.

Our system obtains average FRR results and low FAR values as expected. Those two indexes are related. In general it is not possible to obtain both at the same low level. The lowest value of FRR obtained in our tests was 15,4%. In our opinion, low FAR is more desired than low FRR, mainly due to security reasons. We also observed correlation between enrolment process and FRR value. Shorter enrolment phase causes higher and unacceptable FRR level. Furthermore, a good quality algorithm for processing images is required for better minutiae detection. Some of our tests have failed because of that.

Future work will mainly include FRR ratio improvements without decreasing FAR values and performance boost while processing samples. We will also add several minutiae detection algorithms, which can increase the number of minutiae found.

REFERENCES

[1] Bilski T.,Pankowski T., Stoklosa J.: Bezpieczenstwo danych w systemach informatycznych, Wydawnictwo Naukowe PWN, Poznań, 2001.

[2] Bishop W., Boult T.E., Scheirer W.J.: Beyond PKI: The Biocryptographic Key Infrastructure, 2010. [web page] http://www.wjscheirer.com/papers/wjs_spb2011_bki.pdf. [Accessed on 15 Mar. 2016.].

[3] [web page] http://www.bioapi.org. [Accessed on 25 APR. 2016.].

[4] Bolle R. M., Senior A. W., Ratha N. K., Pankanti S.: Fingerprint Minutiae: A Constructive Definition, 2002. [web page] https://www.research.ibm.com/people/a/aws/documents/papers/BolleMinutiae.pdf. [Accessed on 17 Jan. 2016.].

[5] Buchmann J. A., Karatsiolis E., Wiesmaier A.: Introduction to Public Key Infrastructures, Springer, Berlin, 2013.

[6] Cappelli R., Jain A.K., Maio D., Maltoni D., Wayman J.L.: FVC2004: Third Fingerprint Verification Competition, 2004, [web page] http://www.cse.msu.edu/biometrics/Publications/Fingerprint/FVC/Maioetal_FVC2004_ICBA04.pdf. [Accessed on 17 Jan. 2016.].

[7] Chmiel K., Grocholewska-Czuryło A., Molenda B, Socha P., Stokłosa J., Szymański W.: Ochrona danych i zabezpieczenia w systemach teleinformatycznych, Wydawnictwo Politechniki Poznańskiej, Poznań, 2005.

[8] Cruz-Cortes N., Ertaul L., Martinez-Silva G., Rodriguez-Henriquez F.: On the Generation of X.509v3 Certificates with Biometric Information, 2007. [web page] http://delta.cs.cinvestav.mx/~francisco/certFRH_SAM07.pdf. [Accessed on 19 Jan. 2016.].

[9] Effland T., Schneggenburger M., Schuler J.: Secure fingerprint hashes using subsets of local structures, 2014. [web page] http://www.cse.buffalo.edu/~hartloff/papers/SPIE2014.pdf. [Accessed on 14 Feb. 2016.].

[10] Han-Ul J., Heung-Kyu L.: Biometric-PKI Authentication System Using Fingerprint Minutiae, 2013. [web page] http://dx.doi.org/10.4236/jcc.2014.24004. [Accessed on 22 Feb. 2016.].

[11] Hołyst B., Pomykała J.: Biometria w systemach uwierzytelniania, 2011. [web page] http://wwwold.wat.edu.pl/M000000/_Biuletyn/mfhandler.php?file=24_4-2011_m.pdf&table=3_baza_artykulow&field=dodaj_pobierz&pageType=print&key1=132. [Accessed on 28 Mar. 2016.].

[12] Kulesza K., Nowosielski P.: Kiedy doskonały nie jest idealny, czyli matematyczne metody dzielenia sekretu, 2006. [web page] http://www.matstos.pjwstk.edu.pl/no7/no7_kulesza_nowosielski.pdf. [Accessed on 05 Jan. 2016.].

[13] Network Working Group, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008. [web page] https://www.ietf.org/rfc/rfc5280.txt. [Accessed on 05 Jan. 2016.].

[14] Shamir A.: How to share a secret, Communications of the ACM 22 (11), 612—613, 1979. [web page] http://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf. [Accessed on 15 Mar. 2016.].

[15] Shieh S.-P., Sun H.-M.: Construction of dynamic threshold schemes, Electronics Letters 30(24), 2023—2025, 1994.

[16] Tannenbaum A. S.: Sieci komputerowe. Helion, Gliwice, 682–685, 2004.

***Authors**: Ph.D. Anna Grocholewska-Czuryło, Institute of Control and Information Engineering, Laboratory of Information Technology Security, Poznan University of Technology, ul. Piotrowo 3a, 60-965 Poznan, Poland, email: anna.grocholewska-czurylo@put.poznan.pl*