Computer Science Journal of Moldova, vol.**24**, no.1(70), 2016

Stream Deniable-Encryption Algorithms

N.A. Moldovyan A.A. Moldovyan D.N. Moldovyan
V.A. Shcherbacov

Abstract

A method for stream deniable encryption of secret message is proposed, which is computationally indistinguishable from the probabilistic encryption of some fake message. The method uses generation of two key streams with some secure block cipher. One of the key streams is generated depending on the secret key and the other one is generated depending on the fake key. The key streams are mixed with the secret and fake data streams so that the output ciphertext looks like the ciphertext produced by some probabilistic encryption algorithm applied to the fake message, while using the fake key. When the receiver or/and sender of the ciphertext are coerced to open the encryption key and the source message, they open the fake key and the fake message. To disclose their lie the coercer should demonstrate possibility of the alternative decryption of the ciphertext, however this is a computationally hard problem.

Keywords: cryptology, algorithm, stream, deniable, encryption.

MSC 2000: 94A60, 11S05.

1 Introduction

This paper is an extended version of the article [1].

The notion of deniable encryption (DE) was introduced by Canetti et al. in 1997 [2] as property of cryptographic protocols and algorithms to resist the so called coercive attacks that are performed by some adversary (coercer) that intercepts the ciphertext and has power to force

sender or/and receiver to open both the sent message and the encryption key. If the sender encrypts the secrete message using public key of the receiver of the message, then we have the case of the public-key deniable encryption schemes. If the encryption of the secrete message is performed using a shared secret key, then we have the case of the shared-key deniable encryption schemes.

The public-key DE protocols are applicable for preventing vote buying in the internet-voting systems [3] and for providing security of multiparty computations [4]. The shared-key DE algorithms represent interest for information protection in computer and telecommunication systems. In literature the following cryptoschemes are considered: sender-deniable [2], [3] (coercer attacks the sender of the ciphertext), receiver-deniable [4] (coercer attacks the receiver of the ciphertext), and bi-deniable [5] (coercer attacks the both parties of the secure communication session) cryptoschemes. The encryption scheme is deniable, if it provides possibility to the sender or/and to the receiver to open a fake message and a fake key instead of the secret ones so that disclosing their lie is a computationally infeasible problem for the coercer. Practical methods for bi-deniable public-key encryption have been proposed in [6], [7].

Fast methods for block deniable encryption are described in [8]. Those methods implement deniable encryption as simultaneous transformation of two different messages, secret and fake ones, using two keys, secret and fake ones, into the single ciphertext. In the paper [8] it has been also introduced the notion of the computational indistinguishability of the DE from the probabilistic encryption. The DE algorithm is considered as possessing such property, if it produces the ciphertext that can be also produced by some probabilistic-encryption algorithm used for ciphering the fake message with the fake key and some random input. The stream DE algorithms proposed in [8] and [9] are indistinguishable from some probabilistic encryption algorithms, however those algorithms are very slow. At present no practical and fast algorithms for shared-key stream DE are described in the literature, such algorithms are very attractive for practical application to provide information protection in computer and telecommunication systems

though.

The present paper proposes a method and algorithm for sufficiently fast stream bi-deniable encryption. Computational indistinguishability from a probabilistic stream encryption is used as a design criterion. The paper is organized as follows. Section 2 presents the design criteria. Section 3 and 4 present method and algorithm for stream bi-deniable encryption, correspondingly. Section 5 discusses the proposed algorithm. Section 6 concludes the paper.

2 Design criteria

For designing a shared-key DE algorithm the following criteria have been used:

- the algorithm should implement the stream encryption;
- the used encryption method should provide possibility of the independent decryption of each symbol of the produced ciphertext; this criterion takes into account possible practical applications in the cloud-computing technologies for processing data contained in encrypted files having large size;
- the method should implement the DE procedure as simultaneous encryption of the secret and fake messages using the secret and fake keys;
- the output ciphertext generated by the algorithm should be computationally indistinguishable from the ciphertext produced by some probabilistic ciphering a fake message with a fake key;
- the algorithm should provide sufficiently high encryption speed;
- the algorithm should provide bi-deniability;
- one should provide possibility of the independent recovering of the secret and fake messages, using secret or fake key, correspondingly.

3 Encryption method

One can consider text files as sequence of small data blocks having fixed size, i.e. as sequence of bit strings with which symbols are coded. Thus,

for encrypting a file or a message it is possible to apply formally the fast bi-deniable block-encryption method proposed in [7]. To encrypt a secret message $T = (T_1, T_2, \ldots, T_i, \ldots, T_n)$ represented as sequence of the b-bit data blocks T_i ($b = 32, 64, 128$, or 256), in that method it is supposed to generate a fake message $M = (M_1, M_2, \ldots, M_i, \ldots, M_n)$, where M_i are the b-bit data blocks, having the same size as the secret one and then to encrypt simultaneously all pairs of the data blocks T_i and M_i ($i = 1, 2, \ldots, n$) as follows:

1. Using some known secure block cipher with b-bit input data block, encrypt the data block M_i into the b-bit block C_{M_i} of intermediate ciphertext in accordance with the formula

$$C_{M_i} = E_K(M_i), \tag{1}$$

where E is the used block cipher; K is the fake key.

2. Encrypt the data block T_i into the b-bit block C_{T_i} of intermediate ciphertext in accordance with the formula

$$C_{T_i} = E_Q(T_i), \tag{2}$$

where Q is the secret key.

3. Compute the ith $(2b)$-bit block of the output ciphertext C_i as $(2b)$-bit binary polynomial satisfying the system of congruences

$$\begin{cases} C_i \equiv C_{M_i} \bmod \mu(x) \\ C_i \equiv C_{T_i} \bmod \lambda(x), \end{cases} \tag{3}$$

where binary polynomial $\mu(x) = 1||\mu'(x)$, $||$ denotes the concatenation operation; $\mu'(x)$ is the binary polynomial, which is given by the right b bits of the fake key K (i.e. the right b bits of the secret key K are interpreted as binary polynomial); binary polynomial $\lambda(x) = 1||\lambda'(x)$; $\lambda'(x)$ is the binary polynomial, which is given by the right b bits of the secret key Q.

In the method described in [8] the keys K and Q are generated as a pair of random bit strings such that polynomials $\mu'(x)$ and $\lambda'(x)$ are mutually irreducible, therefore the last system of congruences has

unique solution $C_i < \lambda(x)\mu(x)$ and can be computed as follows:

$$C_i = [C_{M_i}\lambda(x)(\lambda^{-1}(x) \bmod \mu(x)) + \\ + C_{T_i}\mu(x)(\mu^{-1}(x) \bmod \lambda(x))] \bmod \mu(x)\lambda(x).$$

In the case of small values of the data blocks the described method is insecure, for example, in the case of simultaneous encryption of the files $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$ and $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$, where t_i and m_i are symbols having size $b \leq 16$ bits. To overcome this problem we propose to modify the key for each value $i = 1, 2, \ldots, n$. Due to such modification it becomes possible to simplify computation of the blocks C_{M_i} and C_{T_i}, if the sequences of the modified values of the fake and secret key are generated in the form of some pseudorandom sequence that is computationally indistinguishable from the uniform random sequence. Besides, we propose to use unique fake and secret key sequences for encryption of each secret message T. Thus, we have come to idea to generate fake (Γ) and secret (Γ') key sequences using the block cipher E in accordance with the following formulas

$$E_K(i||V) \bmod 2^{2b} = (\alpha_i||\beta_i) \quad \text{and} \quad E_Q(i||V) \bmod 2^{2b} = (\alpha'_i||\beta'_i),$$

where α_i, β_i, α'_i, and β'_i are b-bit strings such that binary polynomials $\mu_i(x) = 1||\beta_i$ and $\lambda_i(x) = 1||\beta'_i$ are mutually irreducible; V is the 64-bit initialization vector generated at random for each encrypted message or file (the value V is not secret, therefore V can be transmitted via insecure channel).

The sequences Γ and Γ' can be written as follows:

$$\Gamma = \{(\alpha_1||\beta_1), (\alpha_2||\beta_2), \ldots, (\alpha_i||\beta_i), \ldots, (\alpha_n||\beta_n)\} \text{ and}$$
$$\Gamma' = \{(\alpha'_1||\beta'_1), (\alpha'_2||\beta'_2), \ldots, (\alpha'_i||\beta'_i), \ldots, (\alpha'_n||\beta'_n)\}.$$

The elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of these sequences are to be used to encrypt simultaneously the couple of symbols t_i and m_i. Instead of formulas (1) and (2) one can use the following transformation of the ith symbol of the fake and secret messages, respectively:

$$c_{m_i} = m_i \oplus \alpha_i \quad (4)$$
$$c_{t_i} = t_i \oplus \alpha'_i, \quad (5)$$

where $\oplus$ is the XOR operation. The b-bit symbols c_{m_i} and c_{t_i} of the intermediate ciphertext are to be mixed into the single $(2b)$-bit symbol c_i of the output ciphertext in accordance with the following formula

$$\begin{aligned} c_i = [c_{m_i}\lambda_i(x)(\lambda_i^{-1}(x) \bmod \mu_i(x)) + \\ + c_{t_i}\mu_i(x)(\mu_i^{-1}(x) \bmod \lambda_i(x))] \bmod \mu_i(x)\lambda_i(x), \end{aligned} \tag{6}$$

where $\mu_i(x) = 1||\beta_i$ and $\lambda_i(x) = 1||\beta'$ are mutually irreducible binary polynomials. Formula (6) defines solution of the following system of congruences

$$\begin{cases} c_i \equiv c_{m_i} \bmod \mu_i(x) \\ c_i \equiv c_{t_i} \bmod \lambda_i(x). \end{cases} \tag{7}$$

System (7) defines the following formulas for computing the symbols c_{m_i} and c_{t_i} from c_i:

$$c_{m_i} = c_i \bmod \mu_i(x), \tag{8}$$

$$c_{t_i} = c_i \bmod \lambda_i(x). \tag{9}$$

Then the ith symbols t_i and m_i of the source texts T and M are computed using the values α_i and α'_i with the following formulas ($i = 1, 2, \ldots, n$):

$$m_i = c_{m_i} \oplus \alpha_i, \tag{10}$$

$$t_i = c_{t_i} \oplus \alpha'_i. \tag{11}$$

4 The stream deniable encryption algorithm

Suppose we have a secure block cipher E with 128-bit input data block and 128-bit key K. Using the method described in Section 3 (in which it is supposed that two parties of the communication session share the secret 128-bit key Q and the fake 128-bit key K) we have constructed the following algorithm for performing the stream DE of the secret message T:

INPUT: the secret message $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$ and encryption keys K and Q.

1. Generate a fake message M having the same length as the message T.

2. Generate a random value of the 64-bit initialization vector V.

3. For $i = 1$ to n do the following steps.

3.1. Using the procedure **Form_**$\alpha\beta$ generate the ith elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of the key sequences Γ and Γ'.

3.2. Compute the b-bit symbols c_{m_i} and c_{t_i} of the intermediate ciphertext using formulas (4) and (5).

3.3. Compute the $(2b)$-bit symbol c_i of the output ciphertext as solution of the system of two linear congruences (7), which is defined by formula (6).

4. Compose the output ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$.

OUTPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$ and the initialization vector V.

The procedure **Form_**$\alpha\beta$ used at step 3 is described as follows:

INPUT: two 128-bit keys K and Q and two 64-bit values i and V.

1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$, where E is some specified 128-bit block cipher; α_i and β_i are b-bit strings; the value $E_K(i||V)$ is considered as binary number.

2. Compose the bit string $\mu_i = (1||\beta_i)$.

3. Compute the value $(\alpha'_i||\beta'_i) = E_Q(i||V) \bmod 2^{2b}$.

4. Compose the bit string $\lambda_i = (1||\beta'_i)$.

5. Considering the bit strings μ_i and λ_i as binary polynomials $\mu_i(x)$ and $\lambda_i(x)$ of the degree b, respectively, compute the greatest common divisor $D = gcd(\mu_i(x), \lambda_i(x))$.

6. If $D \neq 1$, then increment $\beta'_i \leftarrow \beta'_i + 1 \bmod 2^b$ (here the bit string β'_i is considered as binary number) and go to step 4, otherwise STOP.

OUTPUT: two $(2b)$-bit elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of the key sequences Γ and Γ'.

Decryption of the ciphertext C produced by the proposed DE algorithm requires using the value V assigned to C (i.e. sent together with the ciphertext C) and both the secret and fake keys. The following algorithm describes the decryption procedure.

Algorithm for decrypting the secret message.

INPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$, the encryption key Q, the fake key K, and the initialization vector V.

1 For $i = 1$ to n do the following steps.

1.1. Using the procedure **Form_**$\alpha\beta$ generate the ith element $(\alpha'_i||\beta'_i)$ of the key sequence Γ'.

1.2. Compute the b-bit symbol c_{t_i} of the intermediate ciphertext using the formula (9).

1.3. Compute the b-bit symbol t_i of the secret message using formula (11).

2. Compose the message $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$.

OUTPUT: the opened message T.

5 Discussion

5.1 Security against the two-side coercive attack

Suppose a coercive adversary intercepts the ciphertext and initialization vector sent by sender to receiver of secret message and then forces both the parties to open the message, the encryption and decryption algorithms, and the encryption key. The encryption algorithm proposed in Section 4 resists this attack, since the sender and the receiver are able to fulfill coercers demands without opening the secret message. For this purpose they open the following:

- the fake key K declared as the secret one;
- the fake message M declared as the secret one;
- probabilistic encryption algorithm that allegedly produced the ciphertext intercepted by the coercer;
- decryption algorithm that discloses the fake message from the cryptogram, while using the fake key.

To catch them in a lie, the coercer should show conclusively that the ciphertext contains another message. The last can be performed by guessing the secret key Q, however this method is impractical due to sufficiently large size of the value Q (128 bits).

Let us also consider the known-plaintext attack, i.e. suppose the coercer knows the secret message. If he is able to compute the secret key

Q, then he is able to prove that the sender and the receiver are cheating (the proving consists in opening the message T from the ciphertext C, while using the key Q). Suppose additionally that, using the known message T and the value V, the coercer is able to compute the key sequence Γ' and then all values $E_K(i||V)$, where $i = 1, 2, \ldots, n$ (see step 3 in description of the procedure **Form_**$\alpha\beta$).

In this case the assumption about possibility to compute the key Q from the known 128-bit input $i||V$ and output values $E_K(i||V)$ leads to conclusion about insecurity of the used block cipher E against the known-plaintext attack. However in the proposed DE algorithm it is used a secure block cipher, for example, AES that surely resists such attacks and is recommended by the standard ISO/IET 18033-3:2010 [10].

Thus, one can conclude the proposed DE algorithm provides bi-deniability. The probabilistic encryption algorithm to be opened to the coercer is described as follows.

Associated probabilistic stream encryption algorithm

INPUT: the message $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$ and the encryption key K.

1. Generate a random value of the 64-bit initialization vector V.

2. For $i = 1$ to n do the following steps.

2.1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$, where E is the specified 128-bit block cipher; α_i and β_i are b-bit strings; the value $E_K(i||V)$ is considered as binary number.

2.2. Compose the bit string $\mu_i = (1||\beta_i)$.

2.3. Generate randomly two b-bit strings ρ and η'.

2.4. Compose the bit string $\eta = (1||\eta')$.

2.5. Considering the bit strings μ_i and η as binary polynomials $\mu_i(x)$ and $\eta(x)$ of the degree b, respectively, compute the greatest common divisor $D = gcd(\mu_i(x), \eta(x))$.

2.6. If $D \neq 1$, then increment $\eta' \leftarrow \eta' + 1 \bmod 2^b$ (here the bit string η' is considered as binary number) and go to step 2.4, otherwise go to step 2.7.

2.7. Compute the b-bit symbol c_{m_i} of the intermediate ciphertext using formula (4).

2.8. Compute the $(2b)$-bit symbol c_i as solution of the following system of two linear congruences:

$$\begin{cases} c_i \equiv c_{m_i} \bmod \mu_i(x) \\ c_i \equiv \rho(x) \bmod \eta(x), \end{cases} \tag{12}$$

where the bit string c_{m_i} is considered as binary polynomial and $\rho(x)$ is the binary polynomial represented by the bit string ρ.

3. Compose the output ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$.
OUTPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$ and the initialization vector V.

The value c_i at step 2.8 can be computed using the following formula:

$$\begin{aligned} c_i = [c_{m_i}\eta(x)(\eta^{-1}(x) \bmod \mu_i(x)) + \\ + \rho(x)\mu_i(x)(\mu_i^{-1}(x) \bmod \eta(x))] \bmod \mu_i(x)\eta(x). \end{aligned}$$

It is easy to see that for each symbol c_i of the ciphertext C there exist different bit strings η' and ρ satisfying system (12). Indeed, for given c_i and arbitrary η' such that $gcd(\mu_i(x), \eta(x)) = 1$ the value ρ satisfying (12) can be computed as binary polynomial $\rho(x) = c_i \bmod \eta(x)$, where the bit string c_i is considered as binary polynomial.

Thus, while using the encryption key K the associated probabilistic encryption algorithm can potentially encrypt the message M into the cryptogram C produced by the DE algorithm. Since it is computationally difficult to prove that the ciphertext C was produced by the DE process, but not by the probabilistic encryption, one can say the proposed DE algorithm is computationally indistinguishable from the associated probabilistic encryption algorithm.

The decryption algorithm to be opened to the coercer is described as follows.

Dishonest decryption algorithm

INPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$, the encryption key K, and the initialization vector V.

1. For $i = 1$ to n do the following steps.

1.1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$.

1.2. Compose the bit string $\mu_i = (1||\beta_i)$.

1.3. Compute the b-bit symbol c_{m_i} of the intermediate ciphertext using the formula (8).

1.4. Compute the b-bit symbol m_i using the formula (10).

2. Compose the message $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$.

OUTPUT: the opened message M.

5.2 Estimation of the encryption speed

For comparing the performance of the proposed algorithm with the stream DE algorithm described in [7] one can roughly assume that time complexity of computation of the value c_i in accordance with the formula (6) is equal to the time complexity of one block encryption operation. Besides, the time complexity of generation of the values $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$ and $(\alpha'_i||\beta'_i) = E_Q(i||V) \bmod 2^{2b}$ is approximately equal to 1 and 2 block-encryption operations, correspondingly.

Thus, the time complexity of the encryption of one symbol of the secret message is equal to ≈ 4 block-encryption operations. Taking the last into account one can get the following formula for the encryption speed of the proposed algorithm:

$$S = \frac{1}{4} \cdot \frac{b}{128} S_E, \tag{13}$$

where b is the bit length of the symbols with which the secret message is written; S_E is the encryption speed of the block cipher E.

While implementing the DE method from [8] using the block cipher E, encryption of one symbol of the secret message takes on the average 2^{2b+1} operations of the block encryption and defines the following formula for estimating the speed:

$$S_{[7]} = \frac{b}{128} \cdot \frac{S_E}{2^{2b+1}}. \tag{14}$$

Comparing (13) with (14) one can state that the proposed stream DE algorithm is significantly faster (by 2^{2b-1} times) than algorithm by method in [8]. For example, in the case $b = 8$ the ratio S/S_E is equal to 2^{15}.

5.3 Probabilistic deniable encryption

The feasibility of practical use of the probabilistic encryption is related to its providing better statistical properties of ciphertext. This thesis provides credence to both the associated probabilistic stream encryption algorithm and the dishonest decryption algorithm that are to be presented to the coercive attacker together with the fake key. In the stream deniable encryption algorithm described in Section 4 no random values are used, i.e. it is deterministic after the fake message was generated. From practical point of view it is interesting that one can modify the proposed deniable encryption algorithm into the probabilistic deniable encryption one. Indeed, the probabilistic deniable encryption of the messages T and M can be performed as follows:

1. Generate a random value of the 64-bit initialization vector V.

2. For $i = 1$ to n do the following steps.

2.1. Using the procedure **Form_**$\alpha\beta$ generate the ith elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of the key sequences Γ and Γ'.

2.2. Compute the b-bit symbols c_{m_i} and c_{t_i} of the intermediate ciphertext using formulas (4) and (5).

2.3. Generate at random two b-bit strings ρ and η'.

2.4. Compose the bit string $\eta = (1||\eta')$.

2.5. Considering the bit strings μ_i, λ_i, and η as binary polynomials $\mu_i(x)$, $\lambda_i(x)$, and $\eta(x)$ of the degree b, respectively, compute the greatest common divisors $D_1 = gcd(\mu_i(x), \eta(x))$ and $D_2 = gcd(\lambda_i(x), \eta(x))$.

2.6. If $D_1 \neq 1$ or $D_2 \neq 1$, then increment $\eta' \leftarrow \eta' + 1 \bmod 2^b$ (here the bit string η' is considered as binary number) and go to step 2.4, otherwise go to step 2.7.

2.7. Compute the $(3b)$-bit symbol c_i of the output ciphertext as

solution of the system of the following three linear congruences

$$\begin{cases} c_i \equiv c_{m_i} \bmod \mu_i(x) \\ c_i \equiv c_{t_i} \bmod \lambda_i(x) \\ c_i \equiv \rho(x) \bmod \eta(x), \end{cases}$$

which is defined by the formula

$$\begin{aligned} c_i = \lfloor & c_{m_i}\lambda_i(x)\eta(x)\left(\lambda_i^{-1}(x)\eta^{-1}(x) \bmod \mu_i(x)\right) + \\ & + c_{t_i}\mu_i(x)\eta(x)\left(\mu_i^{-1}(x)\eta^{-1}(x) \bmod \lambda_i(x)\right) + \\ & + \rho(x)\lambda_i(x)\mu_i(x)\left(\lambda_i^{-1}(x)\mu_i^{-1}(x) \bmod \eta(x)\right)\rfloor \bmod \lambda_i(x)\mu_i(x)\eta(x) \end{aligned}$$

3. Compose the output ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$.

The associated probabilistic encryption algorithm connected with the last one is the same as that described in Subsection 5.1, except at step 2.3 there are generated $(2b)$-bit random values ρ and η'. The corresponding dishonest decryption algorithm is exactly the same as that described in Subsection 5.1.

6 Conclusion

It is proposed a method and algorithm for fast stream deniable encryption satisfying criterion of the computational indistinguishability from the stream probabilistic encryption. It has been shown that the DE algorithm resists two-side coercive attack. As compared with the stream DE algorithm presented in [8] the proposed one is significantly faster, the algorithm from [8] has one interesting advantage though. The advantage consists in using the same decryption algorithm for opening both the secret and the fake messages from the ciphertext. Such property is significant for providing security against coercive attacks combined with measuring duration of the decryption process. In our future research we plan to develop a fast stream DE method with the same algorithm that decrypts the secret and fake message.

References

[1] A. A. Moldovyan, D. N. Moldovyan, and V. A. Shcherbacov, "Stream deniable-encryption algorithm satisfying criterion of the computational indistinguishability from probabilistic ciphering," in *Workshop on Foundations of Informatics, August 24-29, 2015, Chisinau, Proceedings*, Chisinau, 2015, pp. 318–330.

[2] R. Canetti, C. Dwork, M. Naor, and R.Ostrovsky, "Deniable Encryption," *Proceedings Advances in Cryptology CRYPTO 1997* (Lecture Notes in Computer Science, vol. 1294), pp. 90–104, 1997.

[3] J. Howlader and S. Basu, "Sender-side public key deniable encryption scheme," in *Advances in Recent Technologies in Communication and Computing. Proceedings of the ARTCom '09 International Conference*, 2009, pp. 9–13, dOI: 10.1109/ARTCom.2009.107.

[4] B.Meng and J. Wang, "A receiver deniable encryption scheme," in *Proceedings of the 2009 International Symposium on Information Processing (ISIP09)*, Huangshan, P. R. China, Aug. 2009, pp. 254–257.

[5] A. O'Neil, C. Peikert, and B. Waters, "Bi-deniable public-key encryption," in *Advances in Cryptology CRYPTO 2011* (Lecture Notes in Computer Science, vol. 6841), Berlin: Springer Verlag, 2011, pp. 525–542.

[6] A. Moldovyan and N. Moldovyan, "Practical method for bi-deniable public-key encryption," *Quasigroups and related systems*, vol. 22, pp. 277–282, 2014.

[7] A. Moldovyan, N. Moldovyan, and V. A. Shcherbacov, "Bi-deniable public-key encryption protocol secure against active coercive adversary," *Buletinul Academiei de Stiinte a Republicii Moldova. Matematica*, no. 3, pp. 23–29, 2014.

[8] E. Morozova, Y. Mondikova, and N.A.Moldovyan, "Methods of Deniable Encryption with a Shared Key," *Informatsionno-upravliaiushchie sistemy (Information and Control Systems)*, no. 6(67), pp. 73–78, 2013, (in Russian).

[9] N. Moldovyan, A. R. Birichevskiy, and Y. Mondikova, "Deniable encryption based on block ciphers," *Informatsionno-upravliaiushchie sistemy(Information and Control Systems)*, no. 5(72), pp. 80–86, 2014, (in Russian).

[10] I. standard ISO/IEC 18033-3:2010, "Information technology – security techniques – encryption algorithms – part 3: Block ciphers," 2010.

N. A. Moldovyan[1] A. A. Moldovyan[2],
D. N. Moldovyan[3], V. A. Shcherbacov[4],

Received July 10, 2015

[1] Professor, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
14 Liniya, 39, St.Petersburg, 199178
Russia
E–mail: nmold@mail.ru

[2] Professor, ITMO University
Kronverksky pr., 10, St.Petersburg, 197101
Russia
E–mail: maa1305@yandex.ru

[3] Dr., St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
14 Liniya, 39, St.Petersburg, 199178
Russia E–mail: mdn.spectr@mail.ru

[4] Dr., Institute of Mathematics and Computer Science
Academy of Sciences of Moldova Academiei str. 5, MD−2028 Chişinău
Moldova
E–mail: scerb@math.md