Stream Pseudo-probabilistic Ciphers

Nikolay Andreevich Moldovyan[1], Dmitriy Nikolaevich Moldovyan[1], Quang Minh Le[2], Long Giang Nguyen[3], Sy Tan Ho[4], and Hieu Minh Nguyen[4(✉)]

[1] St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, St. Petersburg 199178, Russia
[2] The Information Technology Institute (ITI), Vietnam National University, Hanoi, Vietnam
[3] Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi, Vietnam
[4] Academy of Cryptography Techniques, Hanoi, Vietnam
hieuminhmta@gmail.com

Abstract. The paper considers methods and algorithms for stream pseudo-probabilistic encryption and introduces a novel design of such ciphers. In the known algorithms of such type two independent messages (fake and secret ones) are encrypted simultaneously (with using two different keys, fake and secret) and the produced ciphertext is computationally indistinguishable from the ciphertext produced by process of the probabilistic encryption of the fake message using the fake key. However in the known stream pseudo-probabilistic encryption schemes the algorithms for decrypting the fake and secret messages do not coincide completely. Therefore a potential attacker can use the last fact to distinguish the pseudo-probabilistic encryption from the probabilistic one. To provide resistance to such potential attacks in the paper there are proposed stream pseudo-probabilistic ciphers satisfying criterion of the sameness of the algorithms for decrypting the fake and secret messages. The introduced ciphers are sufficiently fast and represent interest for practical application to provide confidentiality of the communication protocols performed using public channels. The randomized pseudo-probabilistic stream ciphers have been also designed.

Keywords: Stream cipher · Pseudo-probabilistic encryption
Probabilistic cipher · Fake message · Secret message

1 Introduction

The notion of pseudo-probabilistic (PP) encryption was introduced in [1] as a particular implementation of the shared-key deniable encryption (DE) [2]. The notion of the DE is a special cryptographic primitive suitable for providing resistance to so called coercive attacks [2], i.e., attacks from the part of some potential coercive adversary has power to force a party of the communication session or

P. Cong Vinh and V. Alagar (Eds.): ICCASA 2018/ICTCC 2018, LNICST 266, pp. 36–47, 2019.
https://doi.org/10.1007/978-3-030-06152-4_4

the both parties simultaneously to open the encryption key and the ciphertext after the last has been sent. The paper [2] initiated a lot of investigations devoted to developing secure and efficient protocols for public-key DE [3,4] which have practical interest as a method for preventing vote buying in the internet-voting systems [3] and to provide secure multiparty computations [5].

The shared-key DS algorithms have practical significance as individual method for providing the information protection against unauthorized access in computer and communication systems in the case of coercive attacks. Such application of the shared-key DE algorithms are considered in detail in paper [1] where it had been also shown that for such application it is reasonable to implement the DE algorithms in the form of some PP cipher that generates the ciphertext indistinguishable from the ciphertext generated by some probabilistic encryption procedure applied to transform some fake message with some fake key. Actually the ciphertext contains two messages, the fake and the secret ones that have been encrypted simultaneously, however it is computationally infeasible to distinguish PP encryption process from the probabilistic one while performing cryptanalysis of the ciphertext. The paper [6] describes the block PP encryption algorithms possessing sufficiently high performance and representing practical interest. The papers [1,7] considers the stream DE ciphers, however the ciphers from [1] have sufficiently low performance and the cipher from [7] uses different algorithms for disclosing the fake and the secret message. Any differences between algorithms implementing decryption process while disclosing fake and secret messages can be potentially used by the coercive attacker to distinguish the ciphertext produced as the result of the PP encryption from the ciphertext produced as the result of the probabilistic encryption. Therefore, the criterion of the coincidence of the decryption algorithms for opening the fake and secret messages has practical significance.

Present paper introduces fast stream PP ciphers with the single decryption algorithm for disclosing the fake and secret messages. The paper is organized as follows. Section 2 describes the model of the coercive attack and the design criteria. Section 3 discusses the stream PP cipher proposed in [1]. Section 4 introduces a new stream PP cipher. In Sect. 5 there is considered randomized stream PP cipher. Section 6 concludes the paper.

2 Model of the Coercive Attack and Design Criteria

It is assumed that some potential adversary attacks the sender of the encrypted message or/and the receiver after the ciphertext has been sent via a public communication channel. Besides. the adversary has possibility to intercept the ciphertext and the both parties of the communication session to open the following:

- the source text corresponding to the ciphertext;
- encryption and decryption algorithms;
- the source software code used for performing decryption (not encryption) of the ciphertext;
- the encryption key.

Accordingly to [1,6] the resistance to the described potential attack can be provided using the stream PP cipher applied to performing simultaneous encryption of the fake and secret message which produces the ciphertext exactly the same as the ciphertext produced by some stream probabilistic cipher applied to encrypt the fake message (criterion of the computational indistinguishability from probabilistic encryption [1]). Like in the PP ciphers introduced in [1,7] the simultaneous stream encryption of the fake and secret messages is to be performed using two different keys, the fake and secret ones correspondingly. At time of the coercive attack the sender and the receiver of the ciphertext will open the fake message and the fake key. Besides, they will open the stream probabilistic encryption algorithm (called associated probabilistic encryption algorithm [1]) and related stream decryption algorithm. The last two algorithms are attributed to some stream probabilistic cipher called associated probabilistic cipher. The parties of the secure communication session will lie plausible they used the associated stream probabilistic cipher. While using the opened fake key, the decryption procedure of the last cipher will transform the ciphertext into the fake message. Therefore the coercer will have no arguments to expose their lie, until he shows possibility of the alternative decryption of the ciphertext.

Thus, we propose the following design criteria for creating the stream PP ciphers:

- the stream encryption should be performed as simultaneous transformation of two messages, secret one and fake one, using secret and fake keys shared by sender and receiver;
- a stream probabilistic cipher should be associated with the constructed stream PP cipher;
- the associated stream probabilistic cipher should potentially transform the fake message with the fake key into the same ciphertext that is produced by the stream PP cipher;
- disclosing the secret message, while performing cryptanalysis on the base of the known fake massage and the known fake key, should be computationally infeasible;
- the cipher should include encryption and decryption algorithms possessing sufficiently high performance;
- the decryption algorithm should provided independent recovering of the secret and fake messages;
- the algorithms for recovering the fake and secret messages should completely coincide;
- using the fixed-size shared keys should provide performing secure PP encryption of messages having arbitrary length.

3 Pseudo-probabilistic of Two Different Messages with the Single Decryption Algorithm for Independent Disclosing Each of the Messages

In the paper [1] the introduced notion of pseudo-probabilistic encryption was illustrated by the proposed stream encryption algorithm based on using some one-way transformation function, for example on the base of some secure hash-function F_H. Using the hash-function as the base primitive of the encryption procedure is as follows. Suppose T is a secret message represented as sequence of u-bit symbols t_i: $T = \{t_1, t_2, \ldots, t_i, \ldots, t_z\}$, for example $u = 4$ to 16.

The following algorithm proposed in [1] performs probabilistic encryption of the message T with using the hash-function F_H and the key Q.

Algorithm for Probabilistic Encryption

1. Set counter $i = 1$ and random 128-bit initialization vector V. (The value V is not secret and is to be sent by sender to receiver of the secret message T.)
2. Set counter $j = 0$.
3. If $j < 2^{k+1}$, then generate a random k-bit ($k > u$) value r. Otherwise output the message "The ith data block is not encrypted", increment the counter $i = i + 1$ and go to step 2.
4. Compute the value $t = F_H(Q||V||i||r) \bmod 2^u$, where $||$ is the concatenation operation. If $t \neq t_i$, then increment $j = j + 1$ and go to step 3.
5. Set $r_i = r$. If $i \neq z$, then increment $i = i + 1$ and go to step 2. Otherwise STOP.

The described algorithm is a probabilistic procedure. Indeed, it uses random selection of the current ciphertext symbol r_i (see step 3). The size of the output ciphertext $R = \{r_1, r_2, \ldots, r_i, \ldots, r_z\}$ is larger than the size of the source message T, since the size of symbols of the input text is larger than the size of the symbols of the output ciphertext ($k > u$). As it was shown in [1] mechanism of random selection some part of the has-function argument, until the hash value takes on the required value of the symbol of the source message, can be put into the base of procedure of simultaneous encryption of two different input message, fake and secret, using two different keys, fake and secret respectively. In this case the generated ciphertext contains both messages and each of the lasts can be decrypted simultaneously with the same decryption algorithm, i. e., the procedure for decrypting the fake message coincide exactly with the procedure for decrypting the secret message. Thus, the encryption procedure of two messages produces the ciphertext that potentially could be produced by process of encrypting only one fake message with using the fake key. Therefore, the distinguishing the ciphertext produced by the process of simultaneous encryption of the fake and secret messages from the ciphertext produced by the process of probabilistic encryption of the fake message will require disclosing the secret message. The process of simultaneous encryption of two messages, called pseudo-probabilistic encryption, is described as follows.

Suppose the sequence $M = \{m_1, m_2, \ldots, m_i, \ldots, m_z\}$, where symbols m_i have size u bits, represents the fake message and K is the fake key. The next algorithm proposed in [1] encrypts the secret T and fake message M with using the secret Q and fake K keys.

Algorithms for Simultaneous Encryption of Two Messages

1. Set counter $i = 1$ and random 128-bit initialization vector V.
2. Set counter $j = 0$.
3. If $j < 2^{k+1}$, then generate a random k-bit ($k > 2u$) value r. Otherwise output the message "The ith data block is not encrypted", then increment $i = i + 1$ and go to step 2.
4. Compute the values $t = F_H(Q||V||i||r) \bmod 2^u$ and $m = F_H(K||V||i||r) \bmod 2^u$.
5. If $t \neq t_i$ or $m \neq m_i$, then increment $j = j + 1$ and go to step 3.
6. Set $r_i = r$. If $i \neq z$, then increment $i = i + 1$ and go to step 2. Otherwise STOP.

The decryption algorithm corresponding to the both described encryption algorithm is as follows:

Decryption Procedure

1. Set counter $i = 1$, decryption key $X (X \leftarrow Q$ or $X \leftarrow K)$ and random 128-bit initialization vector V.
2. Compute the value $w_i = F_H(X||V||i||r_i) \bmod 2^u$.
3. If $i \neq z$, then increment $i = i + 1$ and go to step 2. Otherwise STOP.

The decryption algorithm outputs the sequence of u-bit data blocks w_i: $W = \{w_1, w_2, \ldots, w_i, \ldots, w_z\}$. The correctness of the decryption procedure is evident and $W = M$, if $X = K$, or $W = T$, if $X = Q$. A major drawback of the two encryption algorithms discussed above is applying the mechanism of the exhaustive search for finding the value of each symbol of the ciphertext. On the average, there are required about 2^u and 2^{2u} trials for selecting appropriate random value r at step 3 of the first and second algorithm respectively. Because of this drawback, their productivity is very low. In order to increase the speed of encryption, the next section proposes an encryption method that is free from using the mentioned mechanism of the exhaustive search.

4 Implementation Using Block Encryption Function

4.1 General Method for Stream Pseudo-probabilistic Encryption

Suppose that the fake message $M = (m_1, m_2, \ldots, m_i, \ldots, m_z)$ and the secret message $T = (t_1, t_2, \ldots, t_i, \ldots, t_z)$ are represented if the form of the sequences of u-bit symbols m_i and $t_i (i = 1, 2, \ldots, z)$ correspondingly and some secure block encryption function E_S is used to generate the following three key streams Γ,

Γ', and Γ'' depending on the values of the used block-encryption key $S = K$, $S = Q$, and $S = U$ correspondingly:

$$\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_z\},$$
$$\Gamma' = \{\beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_z\}, \quad \text{and}$$
$$\Gamma'' = \{(\lambda_1, \mu_1), (\lambda_2, \mu_2), \ldots, (\lambda_i, \mu_i), \ldots, (\lambda_z, \mu_z)\},$$

where elements α_i, β_i, λ_i, and μ_i represent the u-bit symbols. The key stream Γ'' is generated so that the bit strings $1||\lambda_i$ and $1||\mu_i$, where the sign "||" denotes the concatenation operation, represent two mutually irreducible polynomials. The fake key represents the pair (K, U) and the secret key represents the pair (Q, U), where the random subkeys K and Q are generated so that they have different oddness.

Transformation of the pair of the symbols t_i and m_i. is performed simultaneously as solving the following system of two linear congruencies:

$$\begin{cases} c_i \equiv m_i \oplus \alpha_i \bmod \eta_i \\ c_i \equiv t_i \oplus \beta_i \bmod \psi_i \end{cases} \tag{1}$$

where c_i is the ith symbol of the output ciphertext; $\oplus$ is the bit wise modulo 2 addition operation; $(\eta_i;\ \psi_i) = (1||\lambda_i;\ 1||\mu_i)$, if K is odd, and $(\eta_i;\ \psi_i) = (1||\mu_i;\ 1||\lambda_i)$, if K is even.

Elements of the key sequences Γ, Γ', and Γ'' depend on the subkeys K, Q, and U correspondingly, on sequential number of the message symbol i, and on the initialization vector V that is not secret and can be sent by the sender to the receiver via an insecure channel. While using random initialization vector V different pairs of input messages will be encrypted with different triples of the key sequences Γ, Γ', and Γ''.

Decryption of the ith symbol of the fake and secret message is performed as generating the respective elements of the key sequences and using the following two formulas:

$$m_i = c_i \oplus \alpha_i \bmod \eta_i \quad \text{and} \quad t_i = c_i \oplus \beta_i \bmod \psi_i$$

4.2 Algorithms for Generating the Key Sequences

Suppose E_S be a secure block encryption function with 128-bit input (for example, AES). The following algorithm performs generation of the elements α_i of the key sequence Γ depending on the value $S = K$.

Algorithm 1.

1. Set the 64-bit counter $i = 1$ and the value of the random 64-bit initialization vector V. (The value V is not secret and is to be sent by sender to receiver of the secret message T).
2. Compute the value $\alpha_i = E_S(V||i) \bmod 2^u$.
3. If $i < z$, then increment the value i: $i \leftarrow i + 1$ and go to step 2. Otherwise STOP.

The next algorithm performs generation of the elements β_i of the key sequence Γ' depending on the value $S = Q$.

Algorithm 2.

1. Set the 64-bit counter $i = 1$ and the value of the random 64-bit initialization vector V.
2. Compute the value $\beta_i = E_S(V||i) \bmod 2^u$.
3. If $i < z$, then increment the value i: $i \leftarrow i + 1$ and go to step 2. Otherwise STOP.

The next algorithm performs generation of the pair of key elements (λ_i, μ_i) of the key sequence Γ'' depending on the value $S = U$. Note that in frame of the Algorithms 1 and 2 there is executed the same sequence of operations. They differ only by the value of the key used to compute the output value of the block encryption function $E_S(V||i)$.

Algorithm 3.

1. Set the 64-bit counter $i = 1$ and the value of the random 64-bit initialization vector V.
2. Compute the $2u$-bit value $(\mu_i||\lambda_i) = E_S(V||i) \bmod 2^{2u}$.
3. Compute the greatest common divisor (gcd) of the binary polynomials $1||\mu_i$ and $1||\lambda_i$.
4. If $gcd(1||\mu_i; 1||\lambda_i) \neq 1$, then, considering the bit string λ_i as binary number, modify the value λ_i as follows: $\lambda_i \leftarrow \lambda_i + 1 \bmod 2^u$, where $\leftarrow$ denotes the assignment operation, and go to step 3.
5. If $i < z$, then increment the value i: $i \leftarrow i + 1$ and go to step 2. Otherwise STOP.

4.3 Algorithm for Stream Pseudo-probabilistic Encryption

The stream pseudo-probabilistic encryption of two input messages M and T is performed as consecutive transformation of the pairs of the symbols m_i and t_i (for $i = 1, 2, \ldots, z$) into the $2u$-bit symbols c_i of the output ciphertext. The pair (m_i, t_i) is transformed as follows:

Pseudo-probabilistic Encryption Algorithm
Input: the values K, Q, U, i, t_i, and m_i.

1. Using Algorithm 1 generate the key element α_i.
2. Using Algorithm 2 generate the key element β_i
3. Using Algorithm 3 generate the pair of key elements (λ_i, μ_i).
4. If the value K is even (and the value Q is odd), then define the values $\eta_i = 1||\mu_i$ and $\psi_i = 1||\lambda_i$ and go to step 6.
5. Define the values $\eta_i = 1||\lambda_i$ and $\psi_i = 1||\mu_i$.
6. Compute the $2u$-bit symbol ci of the output ciphertext, using the following formula that describes solution of the system of congruencies (1):

$$c_i = \left[(m_i \oplus \alpha_i)\psi_i(\psi_i^{-1} \bmod \eta_i) \oplus (t_i \oplus \beta_i)\eta_i(\eta_i^{-1} \bmod \psi_i)\right] \bmod \eta_i\psi_i. \quad (2)$$

Output: the ith ciphertext symbol c_i.

Disclosing each of two source message from the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_z)$ is performed independently as consecutive transformation of the ciphertext symbols $c_i (i = 1, 2, \ldots, z)$ using the following decryption algorithm.

Common Decryption Algorithm

Input: the value i, the ciphertext symbol ci, the initialization vector V, and the decryption key (W, U), where $W = K$ for disclosing the fake message or $W = Q$ for disclosing the secret message T.

1. Compute the u-bit value $\gamma_i = E_W(V||i) \bmod 2^u$.
2. Compute the $2u$-bit value $(\mu_i||\lambda_i) = E_U(V||i) \bmod 2^{2u}$.
3. Compute the greatest common divisor (gcd) of the binary polynomials $1||\mu_i$ and $1||\lambda_i$.
4. If $gcd(1||\mu_i; 1||\lambda_i) \neq 1$, then, considering the bit string λ_i as binary number, modify the value i as follows: $\lambda_i \leftarrow \lambda_i + 1 \bmod 2^u$, where $\leftarrow$ denotes the assigning operation, and go to step 3.
5. If $i < z$, then increment the value i: $i \leftarrow i + 1$ and go to step 2.
6. If the value W is even, then assign $\sigma \leftarrow 1||\mu_i$ and go to step 8. Otherwise go to step 7.
7. If the value W is odd, then assign $\sigma \leftarrow 1||\lambda_i$.
8. Compute the ith symbol τ_i of the source message: $\tau_i \equiv c_i \oplus \gamma_i \bmod \sigma$.

Execution of the last algorithm for $i = 1, 2, \ldots, z$ provides disclosing the fake message M, if the fake key $(W, U) = (K, U)$ is used, or disclosing the secret message T, if the secret key $(W, U) = (Q, U)$ is used.

The ciphertext produced by the pseudo-probabilistic encryption algorithm potentially can be produced by the following probabilistic encryption algorithm applied to encryption of the fake message with the fake key:

Associated Probabilistic Encryption Algorithm

Input: the values K, U, i, and m_i.

1. Using Algorithm 1 generate the key element α_i.
2. Using Algorithm 3 generate the pair of key elements (λ_i, μ_i).
3. If the value K is even, then define the values $\eta_i = 1||\mu_i$ and $\psi_i = 1||\lambda_i$ and go to step 5. Otherwise go to step 4.
4. If the value K is odd, then define the value $\eta_i = 1||\lambda_i$ and $\psi_i = 1||\mu_i$.
5. Generate random u-bit value ρ.
6. Compute the $2u$-bit symbol c_i of the output ciphertext as solution of the following system of congruencies:

$$\begin{cases} c_i \equiv m_i \oplus \alpha_i \bmod \eta_i \\ c_i \equiv \rho \bmod \psi_i \end{cases} \quad (3)$$

Output: the ith ciphertext symbol c_i.

System (3) has solution, since $gcd(\eta_i, \psi_i) = 1$ (see step 4 of Algorithm 3). The solution is described as follows:

$$c_i = \left[(m_i \oplus \alpha_i)\psi_i(\psi_i^{-1} \bmod \eta_i) \oplus \rho\eta_i(\eta_i^{-1} \bmod \psi_i)\right] \bmod \eta_i\psi_i. \tag{4}$$

The decryption procedure corresponding to the associated probabilistic encryption algorithm is precisely described by the Common decryption algorithm. This fact demonstrates that ciphertext generated by the pseudo-probabilistic encryption algorithm applied to encrypt simultaneously the fake M and secret T messages can be potentially generated by the associated probabilistic-encryption algorithm applied to encrypt the fake M message.

It is easy to see that the initially proposed design criteria are satisfied by the constructed stream pseudo-probabilistic cipher, including the criterion of the computationally indistinguishability from probabilistic encryption. Since the ciphertext contains two different messages, it is potentially possible to show that the ciphertext had not been generated by probabilistic encryption algorithm, however this would require to compute the secret message in the case when the subkey Q is unknown.

Suppose the coercive attacker gets the fake key (K, U). Then he has possibility to compute the sequence of u-bit symbols $c'_i = c_i \bmod \psi_i = t_i \oplus \beta_i$. The sequence $C_t = (c'_1, c'_2, \ldots, c'_i, \ldots, c'_z)$ represents the intermediate ciphertext obtained as result of encryption the secret message M with using the key stream Γ'. The last is generated using secure block encryption function E (for example, AES), therefore the generated key stream Γ' and the sequence of the ciphertext symbols is computationally indistinguishable from uniform random sequence and computing the message T from C_t is computationally infeasible.

5 Randomized Stream Pseudo-probabilistic Cipher

To provide a higher resistance of the stream PP encryption to the coercive attacks at which the attacker has possibility to block the communication channel and to cause repeated encryption of the same source messages in [8] it has been proposed to imbed randomization into the block PP ciphers. Let us consider construction of the randomized stream PP cipher as imbedding randomization mechanism in the PP cipher described in the previous section. The idea of the proposed randomization consists in using two random bit strings, u-bit string π and $(u+1)$-bit string R, at step of computing the ciphertext symbol c_i. The last is performed as process of finding solution of the system of the following three congruencies:

$$\begin{cases} c_i \equiv m_i \oplus \alpha_i \bmod \eta_i \\ c_i \equiv t_i \oplus \beta_i \bmod \psi_i \\ c_i \equiv \pi \bmod R \end{cases} \tag{5}$$

where R is random binary polynomial of the degree u such that the following conditions hold: $gcd(\eta_i, R) = 1$ and $gcd(\psi_i, R) = 1$.

The randomized stream PP encryption of two input messages M and T is performed as consecutive randomized transformation of the pairs of the symbols m_i and t_i (for $i = 1, 2, \ldots, z$) into the $3u$-bit symbols c_i of the output ciphertext. The randomized transformation of the pair of symbols (m_i, t_i) is executed as follows:

Randomized Pseudo-probabilistic Encryption Algorithm

Input: the values K, U, i, t_i, and m_i.

1. Using Algorithm 1 generate the key element α_i.
2. Using Algorithm 2 generate the key element β_i.
3. Using Algorithm 3 generate the pair of key elements (λ_i, μ_i).
4. If the value K is even (and the value Q is odd), then define the values $\eta_i = 1||\mu_i$ and $\psi_i = 1||\lambda_i$ and go to step 8.
5. Define the values $\eta_i = 1||\lambda_i$ and $\psi_i = 1||\mu_i$.
6. Generate uniformly random u-bit string π.
7. Generate random $(u + 1)$-bit string R such that $gcd(\eta_i, R) = 1$ and $gcd(\psi_i, R) = 1$, where R is interpreted as binary polynomial of the degree u.
8. Compute the $3u$-bit symbol c_i of the output ciphertext, using the following formula that describes solution of the system of congruencies (5):

$$\begin{aligned} c_i = &[(m_i \oplus \alpha_i)\psi_i R(\psi_i^{-1}R^{-1} \bmod \eta_i) \oplus (t_i \oplus \beta_i)\eta_i R(\eta_i^{-1}R^{-1} \bmod \psi_i) \\ &\oplus R\eta_i\psi_i(\psi_i^{-1}\eta_i^{-1} \bmod R)] \bmod \eta_i\psi_i R. \end{aligned} \tag{6}$$

Output: the ith ciphertext symbol c_i having size equal to $3u$ bits.

The probabilistic stream cipher associated with the described randomized stream PP cipher is described as follows:

Associated Probabilistic Stream Cipher

Input: the values K, U, i, and m_i.

1. Using Algorithm 1 generate the key element α_i.
2. Using Algorithm 3 generate the pair of key elements (λ_i, μ_i).
3. If the value K is even, then define the values $\eta_i = 1||\mu_i$ and $\psi_i = 1||\lambda_i$ and go to step 5.
4. If the value K is odd, then define the values $\eta_i = 1||\lambda_i$ and $i = 1||\mu_i$.
5. Generate uniformly random $2u$-bit string π.
6. Generate random $(2u + 1)$-bit string R such that $gcd(\eta_i, R) = 1$, where R is interpreted as binary polynomial of the degree $2u$.
7. Compute the $3u$-bit symbol c_i of the output ciphertext as solution of the following system of two congruencies:

$$\begin{cases} c_i \equiv m_i \oplus \alpha_i \bmod \eta_i \\ c_i \equiv \rho \bmod R \end{cases} \tag{7}$$

Output: the ith ciphertext symbol ci having size equal to $3u$ bits.

The system (7) has solution, since $gcd(\eta_i, R) = 1$ (see step 6 of the algorithm). The solution is described as follows:

$$c_i = [(m_i \oplus \alpha_i)R(R^{-1} \bmod \eta_i) \oplus \rho\eta_i(\eta_i^{-1} \bmod R)] \bmod \eta_i R. \quad (8)$$

6 Conclusion

The design of stream PP cipher with the single decryption algorithm for disclosing the fake message and for disclosing the secret message, depending on the used key (fake or secret respectively) has been proposed. The randomized stream PP cipher constructed by means of embedding a randomization mechanism in the first PP cipher has been also proposed. Each of the introduced stream PP ciphers satisfies criterion of computational indistinguishability from probabilistic encryption. The last fact has been confirmed with presenting the associated probabilistic stream cipher that potentially generates the same ciphertext as the corresponding stream PP cipher. The decryption algorithm relating to the probabilistic cipher exactly coincide with the decryption algorithm relating to the corresponding PP cipher.

In the paper it has been considered the case of equal size of the symbols of the fake and secret messages, however the proposed PP encryption algorithms can be easily extended for the case of different size of the symbols m_i and t_i. Analogous remark can be attributed to the size of random bit string mixed with the transformed symbols m_i and t_i in the proposed randomized stream PP cipher.

The proposed designs of stream PP ciphers use the block encryption function E for generating the key streams Γ, Γ', and Γ'', therefore performance of the introduced two PP ciphers depends on the performance of the used function E. It is mentioned case of using the block encryption standard AES as function E. Since AES is sufficiently fast for hardware and for software implementations the introduced PP-encryption algorithm have performance sufficient for many potential practical implementations. They have significantly higher performance than the stream PP ciphers described in [1].

To increase the performance it is possible to use block encryption functions having comparatively small size of input data block, for example, 32-, 48-, and 64-bit block ciphers. However detailed consideration of such cases represent individual research topic. It is also interesting to use the hash-functions for generating the key streams Γ, Γ', and Γ''.

Acknowledgements. The reported study was funded by Russian Foundation for Basic Research (project $\#18 - 57 - 54002 - Viet\_a$) and by Vietnam Academy of Science and Technology (project $\#QTRU01.08/18 - 19$).

References

1. Moldovyan, N.A., Nashwan, A.A.-M., Nguyen, D.T., Nguyen, N.H., Nguyen, H.M.: Deniability of symmetric encryption based on computational indistinguishability from probabilistic ciphering. In: Bhateja, V., Nguyen, B.L., Nguyen, N.G., Satapathy, S.C., Le, D.-N. (eds.) Information Systems Design and Intelligent Applications. AISC, vol. 672, pp. 209–218. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7512-4_21
2. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052229
3. Barakat, M.T.: A new sender-side public-key deniable encryption scheme with fast decryption. KSII Trans. Internet Inf. Syst. **8**(9), 3231–3249 (2014)
4. Dachman-Soled, D.: On minimal assumptions for sender-deniable public key encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 574–591. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_33
5. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_23
6. Moldovyan, N.A., Moldovyan, A.A., Tam, N.D., Hai, N.N., Minh, N.H.: Pseudo-probabilistic block ciphers and their randomization. J. Ambient Intell. Hum. Comput. (2018). https://doi.org/10.1007/1265201807916
7. Moldovyan, N.A., Moldovyan, A.A., Moldovyan, D.N., Shcherbacov, V.A.: Stream deniable-encryption algorithms. Comput. Sci. J. Moldova **24**(1), 68–82 (2017)
8. Moldovyan, A.A., Moldovyan, N.A., Berezin, A.N., Shapovalov, P.I.: Randomized pseudo-probabilistic encryption algorithms. In: Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017, pp. 14–17 (2017)