THE POST-QUANTUM AND PUBLIC-KEY BLOCK CIPHER ALGORITHMS BASED ON C.E. SHANNON'S THEORY OF PERFECT SECRECY

Luu Hong Dung[1,*], *Nong Phuong Trang*[1]

Abstract

The article proposes a post-quantum and public-key block cipher algorithms constructed based on C.E. Shannon's theory of perfect secrecy. The post-quantum block cipher algorithm proposed here can resist various types of attacks assisted by quantum computers. In addition to high security, this algorithm also has the ability to authenticate the origin and integrity of the encrypted messages. The public key block cipher algorithms here are developed from the proposed post-quantum block cipher algorithm combined with the Diffie - Hellman protocol, so this algorithm can be used similarly to pre-quantum block cipher algorithms (DES, AES,...) but the establishment of the shared secret key is completely based on the public key infrastructure (PKI).

Index terms

Symmetric key cryptography; Encryption - Authentication algorithm; OTP cipher; block cipher; post-quantum cryptography; quantum-resistant cryptography.

1. Introduction

The Vernam cipher also known as OTP cipher is a type of stream cipher proposed by G.S. Vernam in 1917 [1]. This cryptographic algorithm uses the XOR operation to encrypt the plaintext P with a random secret key K. The C.E. Shannon's theory of perfect secrecy [2] proved that OTP cipher is absolutely secure if the key K used here can meet the requirements: a) random; b) used only once; c) has a size not smaller than the size of the plaintext P. In which the randomness of the key K ensures that the ciphertext C received after the encryption will not provide any information about the plaintext P, and using it only once is to maintain the perfect secrecy property of the encryption process of this algorithm. Since the encryption here is the XOR of each

[1] Institute of Information and Communication Technology, Le Quy Don Technical University
[*]Corresponding author, E-mail: luuhongdung@gmail.com DOI: 10.56651/lqdtu.jst.v13.n02.921.ict

bit of the plaintext P with each corresponding bit of the key K, it is obvious that the size of the key K must not be smaller than the size of the plaintext P. It can be seen that the randomness of the key K is the decisive factor for the perfect secrecy property of the OTP cipher. Based on C.E. Shannon's theory of perfect secrecy, it can generally be stated that if a plaintext P is encrypted by XOR operation with a secret key K, the ciphertext C obtained after encryption will not provide any information about the plaintext P and perfect secrecy will be achieved if the key K is random and is used only once.

Based on C.E Shannon's theory of perfect secrecy and the type of algorithm proposed in [3], section 2 of the article proposes a new post-quantum block cipher algorithms that is resistant to all types of attacks assisted by quantum computers (quantum attacks). Moreover, it can also be used as pre-quantum cryptographic algorithms (DES, AES,...) with high secure and performance, which is the difference between this algorithm and previously proposed post-quantum algorithms [4]–[20]. Therefore, section 3 of the article will continue to propose a solution to be able to conveniently use these algorithms as a pre-quantum block cipher algorithms (DES, AES,...). The solution here is simply to integrate the Diffie - Hellman key agreement protocol into the proposed post-quantum block cipher algorithms, so that the establishment of the shared secret key between the sender (encryptor) and the receiver (decryptor) is done entirely based on the public key infrastructure (PKI), and thus these algorithms are called public-key block ciphers.

2. The proposed post-quantum block cipher algorithms

2.1. The first algorithm

The first algorithm proposed here includes: the Encryption algorithm (Algorithm 1) and the Decryption - Authentication algorithm (Algorithm 2), the construction method of these algorithms is presented in the following sections 2.1.1 and 2.1.2. The proof of the correctness of the first algorithm is presented in section 2.1.3.

2.1.1. The Encryption algorithm

In the proposed type of algorithm, the Encryption algorithm takes as input a plaintext P and a shared secret key K of the sender, where the key K has size m bits. The plaintext P is encrypted as n data blocks P_i ($i = 1, 2, \ldots, n$) of size m bits:

$$P = \{P_1, P_2, \ldots, P_n\}$$

The One-time key K_{EOT} used to encrypt plaintext P consists of n subkeys K_{e_i} ($i = 1, 2, \ldots, n$) whose size corresponds to the size of the plaintext block:

$$K_{EOT} = \{K_{e_1}, K_{e_2}, \ldots, K_{e_n}\}$$

The output data of the Encryption algorithm includes the following components:

- C: The ciphertext includes n data blocks C_i ($i = 1, 2, \ldots, n$) of size m bits:

$$C = \{C_1, C_2, \ldots, C_n\}$$

- C_0: Component responsible for generating the One-time key K_{DOT} of the receiver, verifying the origin and the integrity of the post-decrypted message.

The Encryption algorithm is performed through the following steps:

- Step 1: Generate a number N_e from plaintext P and shared secret key K by the hash function $H()$ that has an output data size of m bits:
  $N_e = H(H(P \parallel K) \oplus H(K \parallel P))$

- Step 2: Generate the component C_0 from N_e and key K by the XOR operator:
  $C_0 = N_e \oplus K$

- Step 3: Generate key K_{e_0} from N_e and key K by the hash function $H()$:
  $K_{e_0} = H(H(N_e \parallel K) \oplus H(K \parallel N_e))$

- Step 4: Generate data block P_0 from N_e and K_{e_0} by the hash function $H()$:
  $P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$

- Step 5: Encrypt n data blocks of plaintext P:
  for $i = 1$ to n do
  begin
  $K_{e_i} = H((P_{i-1} \parallel K_{e_i-1}) \oplus (K_{e_i-1} \parallel P_{i-1}))$
  $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
  end

The Encryption algorithm (Algorithm 1) is described as follows:

Algorithm 1: Encryption algorithm

Input: P, K
Output: C_0, C

$N_e = H(H(P \parallel K) \oplus H(K \parallel P))$
$C_0 = N_e \oplus K$
$K_{e_0} = H(H(N_e \parallel K) \oplus H(K \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
  $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
  $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
return (C_0, C)

Note:

– Operator "$\oplus$" is a modulo 2 addition operation (XOR).

– Operator "$\parallel$" is the operation to concatenate bit strings.

2.1.2. The Decryption - Authentication algorithm

In the Decryption - Authentication algorithm, the input consists of ciphertext C, component C_0, and the receiver's shared secret key K. The ciphertext C consists of n data blocks C_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$C = \{C_1, C_2, \ldots, C_n\}$$

The output of the algorithm is a post-decrypted message M and a logical value $TRUE$ or $FALSE$, where M consists of n data blocks M_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$M = \{M_1, M_2, \ldots, M_n\}$$

The One-time key K_{DOT} used to decrypt the received message consists of n subkeys K_{d_i} $(i = 1, 2, \ldots, n)$ of size m bits:

$$K_{DOT} = \{K_{d_1}, K_{d_2}, \ldots, K_{d_n}\}$$

The Decryption - Authentication algorithm is performed through the following steps:

- Step 1: Decrypt the component C_0 using the key K and the operator XOR:
  $N_d = C_0 \oplus K$

- Step 2: Generate key K_{d_0} from the data block N_d and the key K by the hash function $H()$:
  $K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d))$

- Step 3: Generate data block M_0 from N_d and K_{d_0} by the hash function $H()$:
  $M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$

- Step 4: Decrypt the data blocks of ciphertext C:
  for $i = 1$ to n do
  begin
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
  end

- Step 5: Generate the value N_v from M and K by the hash function $H()$:
  $N_v = H(H(M \parallel K) \oplus H(K \parallel M))$

- Step 6: Check if $N_v = N_d$. If true, the integrity of the post-decrypted message M is authenticated; otherwise, the message has been modified.

The Decryption - Authentication algorithm (Algorithm 2) is described as follows:

Algorithm 2: Decryption - Authentication

Input: C_0, C, K
Output: $M, TRUE/FALSE$

$N_d = C_0 \oplus K$
$K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$N_v = H(H(M \parallel K) \oplus H(K \parallel M))$
if $(N_v = N_d)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

Note:

– If the return is $(M, TRUE)$, then the post-decrypted message M is exactly the plaintext P, that is: $M = P$.

– If the return is $(M, FALSE)$, the post-decrypted message has been modified, that is: $M \neq P$.

2.1.3. The correctness of the proposed algorithm

For the first algorithm, what needs to be proved here is: if the received ciphertext is exactly the sent ciphertext, then the post-decrypted message is also the plaintext: $M = P$ and the conditions: $N_v = N_d$ will be satisfied. Therefore, after decryption, if the condition: $N_v = N_d$ is satisfied, the receiver can confirm with certainty about the origin and integrity of the received message.

Indeed, since the sender's shared secret key and the receiver's shared secret key is only one and the value C_0 received is also the value C_0 sent, so we have:

$$N_d = C_0 \oplus K = N_e \oplus K \oplus K = N_e$$

From that, we have:

$$K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d)) = H(H(N_e \parallel K) \oplus H(K \parallel N_e)) = K_{e_0}$$

and:

$$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d)) = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e)) = P_0$$

Because: $M_0 = P_0$, $K_{d_0} = K_{e_0}$, and how to generate subkeys $K_{e_i} (i = 1, 2, .., n)$ of sender: $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$ is also the way to generate the subkeys $K_{d_i} (i = 1, 2, .., n)$ of the receiver: $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$. Infer that the key K_{DOT} of the receiver is also the key K_{EOT} of the sender. From here, we have the first thing to prove:

$$M = C \oplus K_{DOT} = C \oplus K_{EOT} = P \oplus K_{EOT} \oplus K_{EOT} = P$$

Therefore, we have:

$$N_v = H(H(M \parallel K) \oplus H(K \parallel M)) = H(H(P \parallel K) \oplus H(K \parallel P)) = N_e$$

From here, we have the second thing to prove: $N_v = N_d$

2.2. The second algorithm

In case M is a message consisting of n data blocks (n is a determined value), which are generated at different times and need to be encrypted at the time of generation, the first algorithm will not be applicable in this case. The second algorithm proposed here will be used instead of the first algorithm in such cases.

The second algorithm proposed here includes: the Encryption algorithm (Algorithm 3) and the Decryption - Authentication algorithm (Algorithm 4), the construction method of these algorithms is presented in the following sections 2.2.1 and 2.2.2. The proof of the correctness is presented in section 2.2.3.

2.2.1. The Encryption algorithm

The input data of the encryption algorithm here also includes the plaintext P and the sender's shared secret key K - similar to the first encryption algorithm mentioned in section 2.1.1. The output of the second encryption algorithm also includes the ciphertext C and the components C_0, R, where:

- C_0: Component responsible for generates the One-time key K_{DOT} of the receiver.

- R: Component responsible for verifying the origin and the integrity of the post-decrypted message.

The execution steps of the second encryption algorithm are basically the same as the first encryption algorithm, except that in the first step the value N_e is generated by a random/pseudorandom number generator with an output data size of m bits.

The Encryption algorithm (Algorithm 3) is described as follows:

Algorithm 3: Encryption

Input: P, K
Output: C_0, C, R
$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K$
$K_{e_0} = H(H(N_e \parallel K) \oplus H(K \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
  $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
  $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
$R = H(P_n \parallel (K_{e_n} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

Note:

– $PRNG(\{1, 2, \ldots, 2^m - 1\})$: the random/pseudorandom number generator with output data size m bits.

2.2.2. The Decryption - Authentication algorithm

The input data of the Decryption - Authentication algorithm is ciphertext C, components C_0, R, and the receiver's shared secret key K. The output data of the algorithm is a post-decrypted message M and a logical value $TRUE$ or $FALSE$.

The execution steps of the second decryption - authentication algorithm are basically the same as the first decryption - authentication algorithm, except that in Step [5].

The Decryption - Authentication algorithm (Algorithm 4) is described as follows:

Algorithm 4: Decryption - Authentication

Input: C_0, C, R, K
Output: $M, TRUE/FALSE$
$N_d = C_0 \oplus K$
$K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$V = H(M_n \parallel (K_{d_n} \oplus M_0 \oplus K_{d_0}))$
if $(V = R)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

Note:

– If the return is $(M, TRUE)$, then the post-decrypted message M is exactly the plaintext P, that is: $M = P$.

– If the return is $(M, FALSE)$, the post-decrypted message has been modified, that is: $M \neq P$.

2.2.3. The correctness of the proposed algorithm

For the second algorithm, what needs to be proven here is: if the received ciphertext is exactly the sent ciphertext, then the post-decrypted message is also the plaintext: $M = P$ and the conditions: $V = R$ will be satisfied.

Indeed, since the sender's shared secret key and the receiver's shared secret key is only one and the value C_0 received is also the value C_0 sent, so we have:

$$N_d = C_0 \oplus K = N_e \oplus K \oplus K = N_e$$

From that, we have:

$$K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d)) = H(H(N_e \parallel K) \oplus H(K \parallel N_e)) = K_{e_0}$$

and:

$$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d)) = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e)) = P_0$$

Because: $M_0 = P_0$, $K_{d_0} = K_{e_0}$, and how to generate subkeys $K_{e_i} (i = 1, 2, .., n)$ of sender: $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$ is also the way to generate the

subkeys $K_{d_i}(i = 1, 2, .., n)$ of the receiver: $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$. Infer that the key K_{DOT} of the receiver is also the key K_{EOT} of the sender. From here, we have the first thing to prove:

$$M = C \oplus K_{DOT} = C \oplus K_{EOT} = P \oplus K_{EOT} \oplus K_{EOT} = P$$

Therefore, we have the second thing to prove:

$$V = H(M_n \parallel (K_{d_n} \oplus M_0 \oplus K_{d_0})) = H(P_n \parallel (K_{e_n} \oplus P_0 \oplus K_{e_0})) = R$$

2.3. The third algorithm

In case M is a message consisting of n blocks of data generated at different times and n is a non-predetermined value (e.g. message M is online voice data, ...), then the second algorithm needs to be modified accordingly. The third algorithm here will be used instead of the second algorithm in such cases.

The third algorithm proposed here includes: the Encryption algorithm (Algorithm 5) and the Decryption - Authentication algorithm (Algorithm 6), the construction method of these algorithms is presented in the following sections 2.3.1 and 2.3.2. The proof of the correctness is presented in section 2.3.3.

2.3.1. The Encryption algorithm

The input and output data of this third encryption algorithm are exactly the same as the second encryption algorithm mentioned in section 2.2.2.

The Encryption algorithm (Algorithm 5) is described as follows:

Algorithm 5: Encryption

Input: P, K
Output: C_0, C, R

$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K$
$K_{e_0} = H(H(N_e \parallel K) \oplus H(K \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
$i = 1$
while $P_i \neq ESC$ **do**
    $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
    $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
    $i = i + 1$
end
$R = H(P_{i-1} \parallel (K_{e_i-1} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

Note:

– ESC: character indicating end of message M.

2.3.2. The Decryption - Authentication algorithm

In case M is a message consisting of n blocks of data generated at different times and n is an undetermined value, the Decryption - Authentication algorithm (Algorithm 6) is described as follows:

Algorithm 6: Decryption - Authentication

Input: C_0, C, R, K
Output: $M, TRUE/FALSE$

$N_d = C_0 \oplus K$
$K_{d_0} = H(H(N_d \parallel K) \oplus H(K \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
$i = 1$
while $M_i \neq ESC$ **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
  $i = i + 1$
end
$V = H(M_{i-1} \parallel (K_{d_i-1} \oplus M_0 \oplus K_{d_0}))$
if $(V = R)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

Note:

– If the return is $(M, TRUE)$, then the post-decrypted message M is exactly the plaintext P, that is: $M = P$.

– If the return is $(M, FALSE)$, the post-decrypted message has been modified, that is: $M \neq P$.

2.3.3. The correctness of the proposed algorithm

For the third algorithm, what needs to be proven here is: if the received ciphertext is exactly the sent ciphertext, then the post-decrypted message is also the plaintext: $M = P$ and the conditions: $V = R$ will be satisfied.

The proof of the correctness of this algorithm is also performed similarly to the first algorithm mentioned in the section 2.2.3.

2.4. Some evaluation of the secure level of the proposed algorithms

Similar to the OTP cipher, the K_{EOT}/K_{DOT} keys here are only used once for each encrypted message, so types of attacks such as differential cryptanalysis, linear cryptanalysis, etc., and in general all known attack types for block ciphers (such as DES, AES, ...) are not effective with the proposed algorithms. The secure level of the proposed algorithm is assessed by its ability to resist some typical attacks as follows:

- **Ciphertext-only Attack**: According to C. E. Shannon's theory of perfect secrecy [2], if the keys K_{e_i}/K_{d_i} $(i = 1, 2, \ldots, n)$ are a random bit sequence, then there will not

be any relationship between the data blocks P_i of plaintext P and the data blocks C_i of ciphertext C, so: Probability$(P_i|C_i)$ = Probability(P_i) and the proposed algorithms will have "perfect secrecy". Because there is no relationship between the plaintext and the ciphertext, there will be no information from the ciphertext C that would allow an attacker (cryptanalyst) to select the plaintext P from the post-decrypted meaningful messages. Similarly, since N_e is a secret random value, there is no relationship between C_0 and the key K, so an attacker cannot find K if he only knows C_0. In the proposed algorithm, a hash function is used to generate the keys K_{e_i}/K_{d_i}, since the size of the keys K_{e_i}/K_{d_i} (160/256/.. bits) is very small compared to the repetition period of data at the output of the hash function, and furthermore, the attacker cannot calculate the repetition period of the hash function output data, so the keys K_{e_i}/K_{d_i} are possible to completely meet the requirement of randomness of the key. Therefore, the proposed algorithms can resist the "brute force" attack when the attacker only knows the ciphertext, even with the assistance of quantum computers.

- **Known-plaintext attack**: Suppose if for some reason the attacker knows the subkey P_0 and also suppose that with the assistance of quantum computers the one-way property of the hash function is broken, the attacker will find: $H(H(N_e||K_{e_0}) \oplus H(K_{e_0}||N_e))$. But C.E. Shannon's theory of perfect secrecy [2] has shown that the attacker cannot find exactly $H(N_e||K_{e_0})$ and/or $H(K_{e_0}||N_e)$, so it is impossible to find N_e and therefore it will be impossible to find the secret key K. Similarly, once K_{e_0} is known, the attacker will also find:$H(N_e||K) \oplus H(K||N_e)$ but cannot find exactly $H(N_e||K)$ and/or $H(K||N_e)$, so it is not possible to find the shared secret key K. Therefore, even if the one-way property of the hash function is broken, the attacker will not have a chance to attack the secret key K. Furthermore, with the algorithm proposed here, when the plaintext P is public, the one-time key K_{EOT}/K_{DOT} is still kept secret, and the attacker has no way to know/find this key. Therefore, if for some reason the attacker knows the first data block of the plaintext, the attacker cannot decrypt the remaining data blocks of the ciphertext.

- **Spoofing attack**: With the proposed algorithms, the origin and the integrity of the post-decrypted message are verified by the condition: $N_v = N_d$ (for the first algorithm) or: $V = R$ (for the second and third algorithms). The input data of the authentication function are the components C_0, C (for the first algorithm) or C_0, C, R (for the second and third algorithms) and the shared secret key K. Therefore, if the hash function used in this algorithm has such weak collision resistance that an attacker can create a meaningful message whose hash value matches the hash value of the plaintext P, the spoofing attack cannot be performed. Thus, it can be seen that the weak collision resistance of the hash function along with the high speed of quantum computers does not have any significance for the resistance to spoofing attacks ability of the proposed algorithm.

3. The proposed public-key block cipher algorithms

As mentioned in section 1, the public-key block cipher algorithms proposed here are simply the integration of the Diffie - Hellman key agreement protocol into the post-quantum block cipher algorithms proposed in section 2, so these algorithms can be used as pre-quantum block cipher algorithms (DES, AES,...), but the establishment of the shared secret key between the sender (encryptor) and the receiver (decryptor) is done entirely based on the PKI.

3.1. The public-key block cipher algorithms based on discrete logarithm problem on the finite field

The public-key block cipher algorithms proposed here are constructed based on the solution proposed in [21], [22] which is capable of establishing a shared secret key based on the public/private key pair of the sender (encryptor) and the receiver (decryptor). The generation of these public/private key pairs is based on the hard of the discrete logarithm problem on the finite fields, similar to that in public key cryptosystems that are being applied in practice (RSA, ElGamal). The public-key block cipher algorithms proposed here include: the Key Generation algorithm (Algorithm 7), the Encryption algorithms (Algorithm 8, 9, 10) and the Decryption - Authentication algorithms (Algorithm 11, 12, 13). The construction method of these algorithms is presented in the following sections 3.1.1, 3.1.2 and 3.1.3. The proof of the correctness and evaluation of the security of the algorithm are presented in sections 3.1.4 and 3.1.5.

3.1.1. The Key Generation algorithm

In the block cipher algorithms proposed here, the shared secret key between the sender and receiver is established based on the public/private key pair of the sender and receiver.

The End-User's public/private key pair is generated by the Key Generation algorithm based on the set of domain parameters, which includes:

- p, q is a pair of prime numbers and satisfy: $q|(p-1)$.
- g is an element of the finite field F_p of order q calculated according to:

$$g = \alpha^{\frac{p-1}{q}} \mod p, \quad \text{where} \quad \alpha \in (1, p).$$

Note:

The domain parameters here can be generated as specified in ISO/IEC 14888-3 [23], FIPS 186-4 [24], or GOST R34.10-94 [25].

The private key of the End-User is a random or pseudo-random integer d that is secretly created so that: $0 < x < q$.

The corresponding public key y is calculated according to the formula:

$$y = g^x \mod p$$

The Key Generation algorithm (Algorithm 7) is described as follows:

Algorithm 7: Key Generation

Input: l_p, l_q

Output: p, q, g, x, y

1 Generate a pair of prime numbers p, q with: $len(p) = l_p$, $len(q) = l_q$ and satisfy: $q|(p-1)$

2 Choose α in the range $(1, p)$, calculate g according to the formula:

3 $g = \alpha^{\frac{p-1}{q}} \mod p$, satisfy: $g \neq 1$

4 Select a private key x in the range $(1, q)$

5 Calculate the public key y according to the formula:

6 $y = g^x \mod p$

Note:

- $len()$ is the function that calculates the length (in bits) of an integer.

- x, y are the public key and the private key of the end-user in the system.

- p, q, g are the system parameters.

Assuming x_s is the private key of the sender and x_r is the private key of the receiver, then the corresponding public key of the sender y_s is:

$$y_s = g^{x_s} \mod p$$

and the corresponding public key of the receiver y_r is:

$$y_r = g^{x_r} \mod p$$

3.1.2. The Encryption algorithms

In the public-key block cipher algorithms proposed here, the Encryption algorithm takes as input the plaintext P, the sender's private key x_s, the receiver's public key y_r, and system parameters.

The plaintext P is encrypted as n data blocks P_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$P = \{P_1, P_2, \ldots, P_n\}$$

The One-time key K_{EOT} used to encrypt plaintext P consists of n subkeys K_{e_i} $(i = 1, 2, \ldots, n)$ whose size corresponds to the size of the plaintext block:

$$K_{EOT} = \{K_{e_1}, K_{e_2}, \ldots, K_{e_n}\}$$

The output data of the Encryption algorithm includes the following components:

- C: The ciphertext includes n data blocks C_i $(i = 1, 2, \ldots, n)$ of size m bits:
$C = \{C_1, C_2, \ldots, C_n\}$

- C_0: Component responsible for generating the One-time key K_{DOT} of the receiver, verifying the origin and the integrity of the post-decrypted message.

The Encryption algorithm (Algorithm 8) is described as follows:

Algorithm 8: Encryption Algorithm

Input: g, p, x_s, y_r, P
Output: C_0, C

$K_s = (y_r)^{x_s} \mod p$
$K_e = H(K_s)$
$N_e = H(H(P \parallel K_e) \oplus H(K_e \parallel P))$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
  $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
  $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
return (C_0, C)

In case M is a message consisting of n blocks of data generated at different times and n is a predetermined value, then the Encryption algorithm (Algorithm 9) is modified as follows:

Algorithm 9: Encryption Algorithm

Input: g, p, x_s, y_r, P
Output: C_0, C, R

$K_s = (y_r)^{x_s} \mod p$
$K_e = H(K_s)$;
$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
  $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
  $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
$R = H(P_n \parallel (K_{e_n} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

In case M is a message consisting of n blocks of data generated at different times and n is an undetermined value (e.g. message M is online voice data,...), then the Encryption algorithm (Algorithm 10) is described as follows:

Algorithm 10: Encryption Algorithm

Input: g, p, x_s, y_r, P
Output: C_0, C, R

$K_s = (y_r)^{x_s} \mod p$
$K_e = H(K_s)$
$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
$i = 1$;
while $P_i \neq ESC$ **do**
    $K_{e_i} = H((P_{i-1} \oplus K_{e_i-1}) \parallel (K_{e_i-1} \oplus P_{i-1}))$
    $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
    $i = i + 1$
end
$R = H(P_{i-1} \parallel (K_{e_{i-1}} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

It should be noted that Algorithms 9 and 10 can be applied to encrypt pre-determined messages just like Algorithm 8.

3.1.3. The Decryption - Authentication algorithms

The Decryption - Authentication algorithm takes as input the ciphertext C, the component C_0, the receiver's private key x_r, the sender's public key y_s, and system parameters.

The ciphertext C consists of n data blocks C_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$C = \{C_1, C_2, \ldots, C_n\}$$

The output data of this algorithm is a post-decrypted message M, consisting of n data blocks M_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$M = \{M_1, M_2, \ldots, M_n\}$$

The One-time key K_{DOT} used to decrypt the received message - similar to the sender/encryptor side, also consists of n subkeys K_{d_i} $(i = 1, 2, \ldots, n)$ of size m bits:

$$K_{DOT} = \{K_{d_1}, K_{d_2}, \ldots, K_{d_n}\}$$

The Decryption - Authentication algorithm (Algorithm 11) is described as follows:

Algorithm 11: Decryption - Authentication Algorithm

Input: g, p, x_r, y_s, C_0, C
Output: M, $TRUE/FALSE$

$K_r = (y_s)^{x_r} \mod p$
$K_d = H(K_r)$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d \parallel K_d) \oplus H(K_d \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$N_v = H(H(M \parallel K_d) \oplus H(K_d \parallel M))$
if $N_v = N_d$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

Note:

– If the return is $(M, TRUE)$, then the post-decrypted message M is exactly the plaintext P, that is: $M = P$.

– If the return is $(M, FALSE)$, the post-decrypted message has been modified, that is: $M \neq P$.

In case M is a message consisting of n blocks of data generated at different times and n is a predetermined value, the Decryption - Authentication algorithm (Algorithm 12) is described as follows:

Algorithm 12: Decryption - Authentication Algorithm

Input: $g, p, x_r, y_s, C_0, C, R$
Output: M, $TRUE/FALSE$

$K_r = (y_s)^{x_r} \mod p$
$K_d = H(K_r)$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d \parallel K_d) \oplus H(K_d \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$V = H(M_n \parallel (K_{d_n} \oplus M_0 \oplus K_{d_0}))$
if $V = R$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

In case M is a message consisting of n blocks of data generated at different times and n is an undetermined value (e.g. online voice data), the Decryption - Authentication algorithm (Algorithm 13) is described as follows:

Algorithm 13: Decryption - Authentication Algorithm

Input: $g, p, x_r, y_s, C_0, C, R$
Output: M, $TRUE/FALSE$

$K_r = (y_s)^{x_r} \mod p$
$K_d = H(K_r)$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d \parallel K_d) \oplus H(K_d \parallel N_d))$
$M_0 = H(H(N_d \parallel K_{d_0}) \oplus H(K_{d_0} \parallel N_d))$
$i = 1$;
while $M_i \neq ESC$ **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1}) \parallel (K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
  $i = i + 1$
end
$V = H(M_{i-1} \parallel (K_{d_i-1} \oplus M_0 \oplus K_{d_0}))$
if $(V = R)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

It should be noted that Algorithms 12 and 13 can be applied to decrypt pre-determined messages just like Algorithm 11.

3.1.4. The correctness of the proposed algorithms

It is easy to see that the correctness of the proposed algorithms here can be stated and proven completely similarly to the algorithms in section 2 if it can be confirmed that the shared secret key (K_s) of the sender (encryptor) is also the shared secret key (K_r) of the receiver (decryptor).

Indeed, we have:

$$\begin{aligned} K_r = (y_s)^{x_r} \mod p &= (g^{x_s} \mod p)^{x_r} \mod p \\ &= (g^{x_r} \mod p)^{x_s} \mod p = (y_r)^{x_s} \mod p = K_s \end{aligned}$$

It follows that: $K_d = K_e$

From that, it is possible to prove the correctness of the proposed algorithms here, similar to the algorithms in section 2.

3.1.5. Some evaluation of the secure level of the proposed algorithms

It is also easy to see that, the only difference of the proposed algorithms here compared to the algorithms in section 2 is that these algorithm have the additional feature of establishment a shared secret key between the sender (encryptor) and the receiver (decryptor) based on the mechanism of public key cryptography. It also means that in the algorithms proposed here, the cryptanalyst can attack the Key Generation algorithm (Algorithm 7) to find out the secret key (private key) of the sender (encryptor) or receiver (decryptor), from which will calculate the secret key shared between the sender (encryptor) or receiver (decryptor). However, to find the secret key of the user in the system, the cryptanalyst needs to solve the discrete

logarithm problem on the finite field F_p. Currently, no polynomial time algorithm has been published for this hard problem [26]–[34].

3.2. The public-key block cipher algorithms based on discrete logarithm problem on the elliptic curve

The public-key block cipher algorithms proposed here are constructed in a similar way to the block cipher algorithms proposed in section 3.1, the difference between these algorithms and the proposed algorithms in section 3.1 is that the End-user's public/private key pairs are established based on the hard of the discrete logarithm problem on an elliptic curve instead of on a finite field. The public-key block cipher algorithms proposed here include: the Key Generation algorithm (Algorithm 14), the Encryption algorithm (Algorithm 15,16,17) and the Decryption - Authentication algorithm (Algorithm 18,19,20). The construction method of these algorithms is presented in the following sections 3.2.1, 3.2.2 and 3.2.3. The proof of the correctness and evaluation of the security of the algorithm are presented in section 3.2.4 and section 3.2.5.

3.2.1. Key Generation algorithm

In the block cipher algorithms proposed here, the shared secret key between the sender and receiver is established based on the public/private key pair of the sender and receiver.

The End-User's public/private key pair is generated by the Key Generation algorithm based on the set of domain parameters, which includes:

- p is a prime number specifying the underlying finite field F_p.
- $E(F_p)$ is elliptic curve $E(a, b)$ defined on the finite field F_p by the equation: $y^2 = x^3 + ax + b$ with: $a, b \in F_p$ and satisfies: $4a^3 + 27b^2 \neq 0 \mod q$.
- G is a point of order q on the elliptic curve $E(F_p)$, called the base point in $E(F_p)$.
- q is the order of G in $E(F_p)$.

Note:

The domain parameters here can be generated as specified in ISO/IEC 14888-3 [23], FIPS 186-4 [24], or GOST R34.10-2012 [35].

The private key of the End-User is a random or pseudo-random integer d that is secretly created so that $0 < d < q$. The corresponding public key K is:

$$K = (x_k, y_k) = d.G$$

The Key Generation algorithm (Algorithm 14) is described as follows:

Algorithm 14: Key Generation

Input: $E(F_p) = (p, a, b, G, q)$
Output: d, K
$d = PRNG(\{1, 2, \ldots, 2^q - 1\})$
$K = (x_k, y_k) = d.G$
return (d, K)

Note:

- p, a, b, G, q are the system parameters.

- d, K are the private key and the public key of the End-User in the system.

- $PRNG()$ is the random/pseudo-random number generator.

Assuming d_s is the private key of the sender and d_r is the private key of the receiver, then the corresponding public key of the sender K_s is:

$$K_s = (x_{k_s}, y_{k_s}) = d_s.G$$

and the corresponding public key of the receiver K_r is:

$$K_r = (x_{k_r}, y_{k_r}) = d_r.G$$

3.2.2. The Encryption algorithms

In the public-key block cipher algorithms proposed here, the Encryption algorithm takes as input the plaintext P, the sender's private key d_s, the receiver's public key K_r, and system parameters.

The plaintext P is encrypted as n data blocks P_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$P = \{P_1, P_2, \ldots, P_n\}$$

The One-time key K_{EOT} used to encrypt plaintext P consists of n subkeys K_{e_i} $(i = 1, 2, \ldots, n)$ whose size corresponds to the size of the plaintext block:

$$K_{EOT} = \{K_{e_1}, K_{e_2}, \ldots, K_{e_n}\}$$

The output data of the Encryption algorithm includes the following components:

- C is the ciphertext includes n data blocks C_i $(i = 1, 2, \ldots, n)$ of size m bits:

$$C = \{C_1, C_2, \ldots, C_n\}$$

- C_0 is component responsible for generating the One-time key K_{DOT} of the receiver, verifying the origin and the integrity of the post-decrypted message.

The Encryption algorithm (Algorithm 15) is described as follows:

Algorithm 15: Encryption Algorithm

Input: $E(F_p) = (p, a, b, G, q), d_s, K_r, P$
Output: C_0, C

$K_{se} = (x_{k_{se}}, y_{k_{se}}) = d_s.K_r$
$K_e = H(x_{k_{se}})$
$N_e = H(H(P \parallel K_e) \oplus H(K_e \parallel P))$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
    $K_{e_i} = H((P_{i-1} \parallel K_{e_i-1}) \oplus (K_{e_i-1} \parallel P_{i-1}))$
    $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
return (C_0, C)

In case M is a message consisting of n blocks (n is a determined value), which are generated at different times and need to be encrypted at the time of generation. The following Algorithm 16 will be used instead of Algorithm 15 in such cases:

Algorithm 16: Encryption Algorithm

Input: $E(F_p) = (p, a, b, G, q), d_s, K_r, P$
Output: C_0, C, R

$K_{se} = (x_{k_{se}}, y_{k_{se}}) = d_s.K_r$
$K_e = H(x_{k_{se}})$
$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
for $i = 1$ ***to*** n **do**
    $K_{e_i} = H((P_{i-1} \parallel K_{e_i-1}) \oplus (K_{e_i-1} \parallel P_{i-1}))$
    $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
end
$R = H(P_n \parallel (K_{e_n} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

In case M is a message consisting of n blocks of data generated at different times and n is an undetermined value (e.g. message M is online voice data, ...), then the Encryption algorithm (Algorithm 17) is modified as follows:

Algorithm 17: Encryption Algorithm

Input: $E(F_p) = (p, a, b, G, q), d_s, K_r, P$
Output: C_0, C, R

$K_{se} = (x_{k_{se}}, y_{k_{se}}) = d_s.K_r$
$K_e = H(x_{k_{se}})$
$N_e = PRNG(\{1, 2, \ldots, 2^m - 1\})$
$C_0 = N_e \oplus K_e$
$K_{e_0} = H(H(N_e \parallel K_e) \oplus H(K_e \parallel N_e))$
$P_0 = H(H(N_e \parallel K_{e_0}) \oplus H(K_{e_0} \parallel N_e))$
$i = 1$
while $P_i \neq ESC$ **do**
    $K_{e_i} = H((P_{i-1} \parallel K_{e_i-1}) \oplus (K_{e_i-1} \parallel P_{i-1}))$
    $C_i = P_i \oplus K_{e_i} \oplus P_0 \oplus K_{e_0}$
    $i = i + 1$
end
$R = H(P_{i-1} \parallel (K_{e_i-1} \oplus P_0 \oplus K_{e_0}))$
return (C_0, C, R)

It should be noted that Algorithms 16 and 17 can be applied to encrypt pre-determined messages just like Algorithm 15.

3.2.3. The Decryption - Authentication algorithms

The Decryption - Authentication algorithm takes as input the ciphertext C, component C_0, the receiver's private key d_r, the sender's public key K_s and system parameters.

The ciphertext C consists of n data blocks C_i $(i = 1, 2, \ldots, n)$ of size m bits:
$$C = \{C_1, C_2, \ldots, C_n\}$$

The output data of this algorithm is a post-decrypted message M consisting of n data blocks M_i $(i = 1, 2, \ldots, n)$ of size m bits:
$$M = \{M_1, M_2, \ldots, M_n\}$$

The One-time key K_{DOT} used to decrypt the received message — similar to the sender/encryptor side, consists of n subkeys K_{d_i} $(i = 1, 2, \ldots, n)$ of size m bits:
$$K_{DOT} = \{K_{d_1}, K_{d_2}, \ldots, K_{d_n}\}$$

The Decryption - Authentication algorithm (Algorithm 18) is described as follows:

Algorithm 18: Decryption - Authentication

Input: $E(F_p) = (p, a, b, G, q), d_r, K_s, C_0, C$
Output: $M, TRUE/FALSE$

$K_{rd} = (x_{k_{rd}}, y_{k_{rd}}) = d_r.K_s$
$K_d = H(x_{k_{rd}})$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d||K_d) \oplus H(K_d||N_d))$
$M_0 = H(H(N_d||K_{d_0}) \oplus H(K_{d_0}||N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1})||(K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$N_v = H(H(M||K_d) \oplus H(K_d||M))$
if $(N_v = N_d)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

Note:

- If the return is $(M, TRUE)$, then the post-decrypted message M is exactly the plaintext P, that is: $M = P$.

- If the return is $(M, FALSE)$, the post-decrypted message has been modified, that is: $M \neq P$.

In case M is a message consisting of n blocks of data generated at different times and n is a predetermined value, then the Decryption - Authentication algorithm (Algorithm 19) is modified as follows:

Algorithm 19: Decryption - Authentication Algorithm

Input: $E(F_p) = (p, a, b, G, q), d_r, K_s, C_0, C, R$
Output: $M, TRUE/FALSE$

$K_{rd} = (x_{k_{rd}}, y_{k_{rd}}) = d_r.K_s$
$K_d = H(x_{k_{rd}})$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d||K_d) \oplus H(K_d||N_d))$
$M_0 = H(H(N_d||K_{d_0}) \oplus H(K_{d_0}||N_d))$
for $i = 1$ ***to*** n **do**
  $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1})||(K_{d_i-1} \oplus M_{i-1}))$
  $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
end
$V = H(M_n||(K_{d_n}||M_0 \oplus K_{d_0}))$
if $(V = R)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

In case M is a message consisting of n blocks of data generated at different times and n is an undetermined value (e.g., message M is online voice data, etc.), then the Decryption - Authentication algorithm (Algorithm 20) is described as follows:

Algorithm 20: Decryption - Authentication

Input: $E(F_p) = (p, a, b, G, q), d_r, K_s, C_0, C, R$
Output: $M, TRUE/FALSE$

$K_{rd} = (x_{k_{rd}}, y_{k_{rd}}) = d_r.K_s$
$K_d = H(x_{k_{rd}})$
$N_d = C_0 \oplus K_d$
$K_{d_0} = H(H(N_d||K_d) \oplus H(K_d||N_d))$
$M_0 = H(H(N_d||K_{d_0}) \oplus H(K_{d_0}||N_d))$
$i = 1$
while $P_i \neq ESC$ **do**
    $K_{d_i} = H((M_{i-1} \oplus K_{d_i-1})||(K_{d_i-1} \oplus M_{i-1}))$
    $M_i = C_i \oplus K_{d_i} \oplus M_0 \oplus K_{d_0}$
    $i = i + 1$
end
$V = H(M_{i-1}||(K_{d_i-1}||M_0 \oplus K_{d_0}))$
if $(V = R)$ **then return** $(M, TRUE)$ **else return** $(M, FALSE)$

It should be noted that Algorithms 19 and 20 can be applied to decrypt pre-determined messages just like Algorithm 18.

3.2.4. The correctness of the proposed algorithms

It is easy to see that, the correctness of the proposed algorithms here can be stated and proven completely similar to the algorithms in section 3.1 if it can be confirmed that the sender's shared secret key (K_e) is also the receiver's shared secret key (K_d).

Indeed, we have:

$$K_{rd} = d_r.K_s = d_r.(d_s.G) = d_s.(d_r.G) = d_s.K_r = K_{se}$$

Inferred: $x_{k_{rd}} = x_{k_{se}}$

So, we have:

$$K_d = H(x_{k_{rd}}) = H(x_{k_{se}}) = K_e$$

From that, it is possible to prove the correctness of the proposed algorithms here similar to the algorithms in section 3.1.

3.2.5. The security level of the proposed algorithms

It is also easy to see that the only difference between the proposed algorithms here and the algorithms in section 3.1 is that its Key Generation algorithm is constructed on the hard of the discrete logarithm problem on elliptic curves instead of on finite fields. It also means that in the algorithms proposed here, the cryptanalyst can attack the Key Generation algorithm (Algorithm 7) to find out the private key of the sender or receiver, from which will calculate the secret key shared between the sender or receiver. However, to find the private key of the user in the system, the cryptanalyst needs to solve the discrete logarithm problem on the elliptic curve $E(F_p)$. Currently, no polynomial time algorithm has been published for this hard problem [26]–[37].

4. Conclusions

The post-quantum and public-key block cipher algorithms proposed here are developed based on C.E. Shannon's theory of perfect secrecy and hash functions. The advantage of these post-quantum block cipher algorithms is that it is not only resistant to quantum attacks but also resistant to spoofing attacks, which is one of the basic requirements for practical applications. One advantage of the newly proposed algorithms compared to previous post-quantum cryptographic algorithms [4]–[20] is that it can also be used in a similar way to pre-quantum algorithms (such as DES, AES,...) with high security and performance in practical applications. The proposed public key block cipher algorithms are then solutions that allow the proposed post-quantum block cipher algorithms to be used similarly to the block ciphers currently used in practice (such as DES, AES, etc.), however the secret key shared between the sender (encryptor) and the receiver (decryptor) is established based on PKI similar to public-key cryptosystems (RSA, ElGamal), which allows these algorithms to be used more conveniently in practical applications.

References

[1] G. Vernam, "Secret signaling system patent," U.S. Patent No. 1,310,719, July 22, 1919.

[2] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949. DOI: 10.1002/j.1538-7305.1949.tb00928.x

[3] L. H. Dung, "A type of post - quantum cryptographic algorithm," *Journal of Science and Technique - Section on Information and Communication Technology (ICT)*, vol. 13, no. 1, pp. 36–43, June 2024. DOI: 10.56651/lqdtu.jst.v13.n01.818.ict

[4] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *Journal of Cryptology*, vol. 5, no. 1, pp. 3–28, 1992. DOI: 10.1007/BF00191318

[5] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. DOI: 10.1137/S0097539795293172

[6] L. K. Grover, "From Schrödinger's equation to the quantum search algorithm," *American Journal of Physics*, vol. 69, no. 7, pp. 769–777, 2001. DOI: 10.1119/1.1359518

[7] C. Elliot, "Quantum cryptography," *IEEE Security & Privacy Journal*, vol. 2, no. 4, pp. 57–61, 2004. DOI: 10.1109/MSP.2004.54

[8] D. J. Bernstein, J. B., and E. D., *Post quantum cryptography*. Springer Science & Business Media, 2009.

[9] A. Broadbent and C. S., "Quantum cryptography beyond quantum key distribution," *Designs, Codes and Cryptography*, vol. 78, no. 1, pp. 351–382, 2016. DOI: 10.1007/s10623-015-0157-4

[10] L. Chen, S. Jordan, Y.-K. Liu ..., and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, vol. 12, 2016.

[11] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, 2017, pp. 188–194. DOI: 10.1007/978-3-540-88702-7

[12] S. S. Mehrdad, "Quantum cryptography: An emerging technology in network security," in *Technologies for Homeland Security (HST), IEEE International Conference*, 2011, pp. 13–19. DOI: 10.1109/THS.2011.6107841

[13] A. C. H. Chen, "Post-quantum cryptography neural network," in *2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES)*. India, 2023, pp. 1–6. DOI: 10.1109/ICSSES58299.2023.10201083

[14] T. Takagi, "Recent developments in post-quantum cryptography," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E101, no. A, p. 3, 2018. DOI: 10.1587/transfun.E101.A.3

[15] A. Cohen, R. G. L. D'Oliveira, S. Salamatian, and M. Médard, "Network coding-based post-quantum cryptography," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, pp. 49–64, 2021. DOI: 10.1109/JSAIT.2021.3054598

[16] O. S. Althobaiti and M. Dohler, "Quantum-resistant cryptography for the internet of things based on location-based lattices," *IEEE Access*, vol. 9, pp. 133 185–133 203, 2021. DOI: 10.1109/ACCESS.2021.3115087
[17] I. P. A. E. Pratama and I. G. N. A. K. Adhitya, "Post quantum cryptography: Comparison between RSA and McEliece," in *2022 International Conference on ICT for Smart Society (ICISS)*. IEEE, 2022, pp. 1–5. DOI: 10.1109/ICISS55894.2022.9915232
[18] J. Pinto, "Post-quantum cryptography," *Advanced Research on Information Systems Security*, vol. 2, no. 2, pp. 4–16, 2022. DOI: 10.56394/aris2.v2i2.17
[19] B. S. Solanki and A. Saini, "Review paper on quantum computing and quantum cryptography," *International Journal of Advanced Research in Science Communication and Technology*, May 2023. DOI: 10.48175/IJARSCT-10712
[20] J. Jency Rubia, R. Babitha Lincy, N. Ezhil E, C. Sherin Shibi, and A. Rosi, "A survey about post quantum cryptography methods," *EAI Endorsed Transactions on Internet of Things*, vol. 10, Feb. 2024. DOI: 10.4108/eetiot.5099
[21] N. P. Trang and L. H. Dung, "A type of public-key block cipher algorithm," *Journal of Military Science and Technology*, December 2023. DOI: 10.54939/1859-1043.j.mst.CSCE7.2023.49-59
[22] L. H. Dung, "Public-key block cipher," *Journal of Science and Technique - Section on Information and Communication Technology (ICT)*, vol. 12, no. 2, pp. 7–26, December 2023. DOI: 10.56651/lqdtu.jst.v12.n02.747.ict
[23] ISO/IEC, "Information technology – security techniques – digital signatures with appendix," November 2006, second edition, ISO/IEC 14888-3.
[24] National Institute of Standards and Technology, "NIST FIPS PUB 186-4. Digital Signature Standard," 2013.
[25] GOST R34.10-94, "Russian federation standard information technology. Cryptographic data security produce and check procedures of electronic digital signature based on asymmetric cryptographic algorithm," 1994.
[26] I. Shparlinski, *Cryptographic applications of analytic number theory. Complexity lower bounds and pseurandomness*. Birkhäuser, 2003.
[27] S. S. Wagstaff, *Cryptanalysis of number theoretic ciphers.* Chapman & Hall /CRC, 2003.
[28] J. Katz and Y. Lindell, *Introduction to modern cryptography.* Chapman & Hall/CRC, 2008.
[29] J. Hoffstein, J. Pipher, and J. H. Silverman, *An introduction to mathematical cryptography*. Springer-Verlag, 2008. ISBN 978-0-387-77993-5
[30] D. R. Stinson, *Cryptography: Theory and practice.* Chapman & Hall/CRC, 2006.
[31] R. A. Mollin, *An introduction to cryptography*. Chapman & Hall/CRC, 2006.
[32] J. Talbot and D. Welsh, *Complexity and cryptography: An introduction*. Cambridge University Press, 2006.
[33] J. Buchmann, *Introduction to cryptography*. Springer-Verlag, 2004.
[34] W. Mao, *Modern cryptography: Theory and practice*. Pearson Education, 2004.
[35] GOST R34.10–2012, "Russian federation standard information technology. Cryptographic data security produce and check procedures of electronic digital signature based on asymmetric cryptographic algorithm," 2012.
[36] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*. Cambridge University Press, 2000.
[37] L. C. Washington, *Elliptic Curves. Number theory and cryptography.* Chapman & Hall/CRC, 2008.

Manuscript received 12-03-2024; Accepted 27-12-2024.

■

Luu Hong Dung graduated in Electronics and Communications in 1989 and PhD in 2013 at Le Quy Don Technical University, Viet Nam. Currently working in the Institute of Information and Communication Technology, Le Quy Don Technical University. Research field: Cryptography and information security. E-mail: luuhongdung@gmail.com

Nong Phuong Trang graduated in Information System Security in 2024 at Le Quy Don Technical University. Currently working in the Department of Information Technology, Le Quy Don Technical University. Research field: Cryptography and information security. E-mail: ptrang6201@gmail.com

MỘT SỐ THUẬT TOÁN MÃ KHỐI HẬU LƯỢNG TỬ VÀ MÃ KHỐI KHÓA CÔNG KHAI DỰA TRÊN LÝ THUYẾT C.E. SHANNON VỀ ĐỘ MẬT HOÀN THIỆN

Lưu Hồng Dũng, Nông Phương Trang

Tóm tắt

Bài báo đề xuất một dạng thuật toán mã khối hậu lượng tử và mã khối khóa công khai dựa trên lý thuyết C.E. Shannon về độ mật hoàn thiện. Các thuật toán mã khối hậu lượng tử được đề xuất ở đây có khả năng chống lại các dạng tấn công khác nhau với sự trợ giúp của máy tính lượng tử. Ngoài tính bảo mật cao, thuật toán này còn có khả năng xác thực nguồn gốc và tính toàn vẹn của các thông điệp được mã hóa. Các thuật toán mã khối khóa công khai ở đây được phát triển từ các thuật toán mã khối hậu lượng tử đã được đề xuất trước đó bằng việc kết hợp với giao thức trao đổi khóa Diffie - Hellman, do đó thuật toán này có thể được sử dụng tương tự như các thuật toán mã khối tiền lượng tử đang được ứng dụng trong thực tiễn (DES, AES,...) nhưng việc thiết lập khóa bí mật chia sẻ hoàn toàn dựa trên cơ sở hạ tầng khóa công khai (PKI).

Từ khóa

Mật mã khóa đối xứng; thuật toán mã hóa – xác thực; mật mã OTP; mật mã khối; mật mã hậu lượng tử; mật mã kháng lượng tử.