PUBLIC–KEY BLOCK CIPHER

Hong Dung Luu[1,*]

Abstract

The paper proposes a type of block cipher scheme based on cryptographic hash function and public key cryptography. The scheme proposed here is capable of verifying the origin and integrity of the encrypted message. On the other hand, establishing a shared secret key between the sender/encryptor and the receiver/decryptor can be done for each message separately.

Index terms

Block cipher, symmetric key cryptography, public key cryptography, encryption - authentication scheme, OTP cipher, discrete logarithm problem.

1. Introduction

In [1], [2] and [3], a solution for constructing a symmetric key cryptosystem has been proposed, which is developed from One-Time Pad (OTP) cipher. The advantage of the algorithms constructed in [1], [2] and [3] is that the security and efficiency are inherited from the OTP cipher, but the shared secret key between the sender/encryptor and the receiver/ decryptor can be used in the long run. Moreover, the establishment, management and distribution of keys are performed similarly to other symmetric key cryptosystems being applied in practice (DES, AES, ...). Based on the solution in [1], [2] and [3], a method of constructing public key block cipher schemes has been proposed in [4] and [5]. The schemes constructed according to this method, in addition to the security and authentication features of the encrypted message, also allow the establishment of a shared secret key between the two parties sender/encryptor - receiver/decryptor based on the mechanism of the public key cryptography for each message separately.

However, in the schemes proposed in [1]–[5], the Encryption and Decryption - Authentication algorithm can be performed only with messages whose size is predetermined before the time the algorithm is executed. Therefore, these schemes are not suitable for online applications where encryption and decryption - authentication algorithm needs to be performed with real-time data, that is, at the time the encryption and decryption algorithms of the scheme are executed, the size of the message may not yet be determined.

In this paper, we will propose a type of public key block cipher scheme for online applications as well as offline applications, meaning the schemes proposed here are not

[1]Institute of Information and Communication Technology, Le Quy Don Technical University
[*]Corresponding author, email: luuhongdung@gmail.com DOI: 10.56651/lqdtu.jst.v12.n02.747.ict

only applicable to messages whose size is known but is also applicable to messages whose size is not determined at the time these schemes are executed.

2. The proposed public-key block cipher schemes

2.1. The block cipher scheme developed based on OTP cipher

The scheme proposed here constructed based on the solution mentioned in [1], [2] and [3] with some improvements to be applicable (encrypt, decrypt) for online messages as well as offline messages. The scheme proposed here includes the Encryption algorithm (Algorithm 1.1) and the Decryption - Authentication algorithm (Algorithm 1.2), described as follows:

2.1.1. The Encryption algorithm: In this scheme, the Encryption algorithm takes as input a plaintext $\mathbf{P}$ and the shared secret key $\mathbf{K}$ of the sender (encryptor). The plaintext $\mathbf{P}$ is encrypted as n data blocks $\mathbf{P}_i$ of size m bits:

$$\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n.\}$$

It is important to note here that at the beginning of the execution of the Encryption algorithm, the value of parameter n may not yet be determined.

The one-time key $\mathbf{K_{OT}}$ used to encrypt P consists of n subkeys K_i whose size corresponds to the size of the plaintext block:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

The output of the algorithm consists some of components, where the $\mathbf{C}$ ciphertext also includes n data blocks $\mathbf{C}_i$ of size m bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}.$$

In addition to the $\mathbf{C}$ ciphertext, the output data of the Encryption algorithm has the following components:

- $\mathbf{C}_0$ is component responsible for generates subkey $\mathbf{K}_1$ of the $\mathbf{K_{OT}}$ key.
- $\mathbf{R}$ is component responsible for verifying the integrity of the post-decrypted message.

The Encryption algorithm is performed through the following steps:

Step 1: Generate a data block $\mathbf{P}_0$ of size m bits using the random/pseudo-random number generator $\mathbf{PRNG}()$:

$$\mathbf{P}_0 = \mathbf{PRNG}\left(\{1, 2, \ldots, 2^m - 1\}\right).$$

Step 2: Calculate the component $\mathbf{C}_0$ from data block $\mathbf{P}_0$ and key $\mathbf{K}$ by operator $\mathbf{XOR}$:

$$\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}.$$

Step 3: Generate key $\mathbf{K}_0$ from component $\mathbf{P}_0$ and shared secret key $\mathbf{K}$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_0 = \mathbf{H}\left(\mathbf{P}_0 \| \mathbf{K}\right).$$

Step 4: Encrypt the data blocks of the plaintext $\mathbf{P}$, the encryption process $\mathbf{P}$ will end if the value of the encrypted data block is $\mathbf{ESC}$

$i = 1;$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$i = i + 1$
end

Step 5: Generate component $\mathbf{R}$ from plaintext $\mathbf{P}$ and subkey $\mathbf{K}_i$ by the hash function $\mathbf{H}()$:

$$\mathbf{R} = \mathbf{H}\left(\mathbf{P} || \mathbf{K}_i\right).$$

The Encryption algorithm is described in pseudocode as follows:

Algorithm 1.1a :
input : $\mathbf{P}, \mathbf{K}$
output : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}$
[1]. $\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
[2]. $\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}$
[3]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K})$
[4]. $i = 1$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$i = i + 1$
end
[5]. $\mathbf{R} = \mathbf{H}(\mathbf{P} \| \mathbf{K}_i)$

Note:

- Operator " $\oplus$ " is a modulo 2 addition operation ($\mathbf{XOR}$).
- Operator "$\|$" is the operation to concatenate two bit strings.
- The hash function $\mathbf{H}()$ here is selectable as: MD5 [6], SHA-1/SHA-256 [7],...
- $\mathbf{ESC}$: End character of the message to be encrypted.

In case the size of the message to be encrypted (value n) has been determined, the Encryption algorithm here is the same as the Encryption algorithm mentioned in [8] and is described in pseudocode as follows:

Algorithm 1.1b :
input : $\mathbf{P}, \mathbf{K}$
output : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}$
[1]. $\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
[2]. $\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}$
[3]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K})$
[4]. **for** $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
end
[5]. $\mathbf{R} = \mathbf{H}(\mathbf{P} \| \mathbf{K}_n)$

2.1.2. The Decryption - Authentication algorithm: The Decryption - Authentication algorithm whose input is ciphertext $\mathbf{C}$, components $(\mathbf{C_0}, \mathbf{R})$ and the shared secret key $\mathbf{K}$ of the receiver (decryptor). The $\mathbf{C}$ ciphertext consists of n data blocks $\mathbf{C}_i$ of size m bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}.$$

Similar to the Encryption algorithm, when starting to execute the Decryption - Authentication algorithm, the value of parameter n may not be determined yet.

The output of the algorithm is a post-decrypted message $\mathbf{M}$ consisting of n data blocks of size m bits:

$$\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}.$$

The one-time key $\mathbf{K_{OT}}$ used to decrypt the received message - similar to the sender/encryptor side, consists of n subkeys $\mathbf{K}_i$ of size m bits:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

The Decryption - Authentication algorithm is performed through the following steps:
Step 1: Decrypt the $\mathbf{C}_0$ component using the key $\mathbf{K}$ and the operator $\mathbf{XOR}$:

$$\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}.$$

Step 2: Generate key $\mathbf{K}_0$ from $\mathbf{M}_0$ and $\mathbf{K}$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_0 = \mathbf{H}\left(\mathbf{M}_0 \| \mathbf{K}\right).$$

Step 3: Decrypt the data blocks of ciphertext $\mathbf{C}$, the decryption process $\mathbf{C}$ will end if the value of the decrypted data block is $\mathbf{ESC}$:

$i = 1;$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1}||\mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$i = i + 1$
end

Step 4: Generate the value $\mathbf{V}$ from the post-decrypted message $\mathbf{M}$ and the subkey $\mathbf{K}_i$ by the hash function $\mathbf{H}()$:

$$\mathbf{V} = \mathbf{H}(\mathbf{M}||\mathbf{K}_i).$$

Step 5: Check if $\mathbf{V} = \mathbf{R}$ then the integrity of the post-decrypted message $\mathbf{M}$ is authenticated. Otherwise, the message has been modified.

The Decryption - Authentication algorithm is described in pseudocode as follows:

Algorithm 1.2a :
input : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}, \mathbf{K}$
output : $\mathbf{M}, \mathbf{TRUE}, \mathbf{FALSE}$
[1]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}$.
[2]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0||\mathbf{K})$
[3]. $i = 1$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1}||\mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$i = i + 1$
end
[4]. $\mathbf{V} = \mathbf{H}(\mathbf{M}||\mathbf{K}_i)$.
[5]. **if** $\mathbf{V} = \mathbf{R}$ **then return** $(\mathbf{M}, \mathbf{TRUE})$
else return $(\mathbf{M}, \mathbf{FALSE})$.

Note:

- $\mathbf{ESC}$: End character of the message to be decrypted.
- If the return is $(\mathbf{M}, \mathbf{TRUE})$, then the post-decrypted message $\mathbf{M}$ is exactly the plaintext $\mathbf{P}$, that is: $\mathbf{M} = \mathbf{P}$.
- If the return is $(\mathbf{M}, \mathbf{FALSE})$, then the post-decrypted message has been modified, that is: $\mathbf{M} \neq \mathbf{P}$.

In case the size of the message to be decrypted (value n) has been determined, the Decryption - Authentication algorithm here is the same as the Decryption - Authentication algorithm mentioned in [8] and described in pseudocode as follows:

Algorithm 1.2b :
input : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}, \mathbf{K}$
output : $\mathbf{M}, \mathbf{TRUE}, \mathbf{FALSE}$
[1]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}$.
[2]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K})$
[3]. **for** $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{M}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
end
[4]. $\mathbf{V} = \mathbf{H}(\mathbf{M} \| \mathbf{K}_n)$.
[5]. **if** $\mathbf{V} = \mathbf{R}$ **then return** $(\mathbf{M}, \mathbf{TRUE})$
else return $(\mathbf{M}, \mathbf{FALSE})$.

2.1.3. The correctness of the proposed schema: What needs to be proved here is: if the received ciphertext is exactly the sent ciphertext, then the post-decrypted message is also the pre-encrypted message (plaintext): $\mathbf{M} = \mathbf{P}$ and the condition: $\mathbf{V} = \mathbf{R}$ will be satisfied. Therefore, after decryption, if the condition: $\mathbf{V} = \mathbf{R}$ 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 ($\mathbf{K}$) and the value $\mathbf{C}_0$ received is also the value $\mathbf{C}_0$ sent, so we have:

$$\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K} = \mathbf{P}_0 \oplus \mathbf{K} \oplus \mathbf{K} = \mathbf{P}_0.$$

It follows that the value $\mathbf{K}_0$ of the sender (encryptor) is also the value $\mathbf{K}_0$ of the receiver (decoder):

$$\mathbf{K}_0 = \mathbf{H}\left(\mathbf{P}_0 \| \mathbf{K}\right) = \mathbf{H}\left(\mathbf{M}_0 \| \mathbf{K}\right)$$

and how to generate subkey $\mathbf{K}_i$ of sender: $\mathbf{K}_i = \mathbf{H}(\mathbf{P}_{i-1} \| \mathbf{K}_{i-1})$ is also the way to generate the subkey $\mathbf{K}_i$ of the receiver: $\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1} \| \mathbf{K}_{i-1})$. It follows that the key $\mathbf{K_{OT}}$ of the receiver is also the key $\mathbf{K_{OT}}$ of the sender. From here we have the first thing to prove:

$$\mathbf{M} = \mathbf{C} \oplus \mathbf{K_{OT}} = \mathbf{P} \oplus \mathbf{K_{OT}} \oplus \mathbf{K_{OT}} = \mathbf{P}.$$

and there's the second thing to prove:

$$\mathbf{V} = \mathbf{H}\left(\mathbf{M} \| \mathbf{K}_i\right) = \mathbf{H}\left(\mathbf{P} \| \mathbf{K}_i\right) = \mathbf{R}.$$

2.1.4. Some evaluation of the security level of the proposed schema: Similar to the OTP cipher, the KOT key here is 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 typical block ciphers such as DES, AES,... are not effective with the proposed algorithm. The security level of the proposed scheme is evaluated by its ability to resist some typical attacks as follows:

- **Known-plaintext attack**: At the proposed scheme, if the plaintext is known, the attacker can calculate the subkey $\mathbf{K}_1$ according to:

  $$\mathbf{K}_1 = \mathbf{C}_1 \oplus \mathbf{M}_1.$$

  Based on:

  $$\mathbf{K}_1 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_0)$$

  and

  $$\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K})$$

  the attacker can establish the equation:

  $$\mathbf{K}_1 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{H}(\mathbf{M}_0 \| \mathbf{K})).$$

  However, to find the shared secret key $\mathbf{K}$ from the above equation with known value of $\mathbf{K}_1$, the attacker must perform a “brute force" attack with $\mathbf{M}_0$ also a secret/unknown value, that is impossible.
- **Spoofing attack**: The OTP cipher does not provide an authentication mechanism for an encrypted message, so an attacker could block the ciphertext which was sent and send the recipient a fake ciphertext of the same size as the true message. In the case of decrypt to a meaningless plaintext, the receiver may speculate that the tampering was made or caused by a communication error. However, if decrypt to a meaningful plaintext, then the receiver has no way to verify whether the plaintext is true or fake. With the proposed scheme, the origin and the integrity of the post-decrypted message is verified by the condition: $\mathbf{V} = \mathbf{R}$. Because, no one other than the sender (encryptor) and receiver (decryptor) can satisfy the above condition, it makes the proposed scheme to be resistant to spoofing attacks.

2.2. The public-key block cipher

The public-key block cipher scheme here is developed based on the block cipher scheme in section 2.1 which is capable of establishing a shared secret key (based on the mechanism of public key cryptography) for each encrypted message. The scheme includes the Parameter and Key Generation algorithm (Algorithm 2.1), the Encryption algorithm (Algorithm 2.2) and the Decryption-Authentication algorithm (Algorithm 2.3), described as follows:

2.2.1. The Parameter and Key Generation algorithm:

Algorithm 2.1 :

input : ℓ_p, ℓ_q.

output : p, q, g, x, y.

[1]. Choose a pair of prime numbers p, q with : $len(p) = \ell_p, len(q) = \ell_q$ and satisfy: $q \mid (p-1)$.

[2]. Choose a in the range $(1, p),$ calculate g according to the formula: $g = a^{\frac{p-1}{q}} \bmod p,$ satisfy: $g \neq 1$.

[3]. Select a secret key x in the range $(1, q)$.

[4]. Calculate the public key y calculate g according to the formula: $y = g^x \bmod p$.

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 end-user in system.
- p, q, g are system parameters.

Assuming, x_s is the secret key of the sender/encryptor and x_r is the secret key of the receiver/decryptor, then the corresponding public key of the sender/encryptor is:

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

and of the receiver/decryptor is:

$$y_r = g^{x_r} \bmod p.$$

2.2.2. The Encryption algorithm: In this scheme, the Encryption algorithm takes as input the plaintext $\mathbf{P}$, the sender's secret key x_s, the receiver's public key y_r and system parameters. The plaintext $\mathbf{P}$ is encrypted as n data blocks $\mathbf{P}_i$ of size m bits:

$$\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n.\}$$

It should also be noted that the value of n here may not be determined when the algorithm is executed.

The one-time key $\mathbf{K_{OT}}$ used to encrypt $\mathbf{P}$ consists of n subkeys $\mathbf{K}_i$ whose size corresponds to the size of the plaintext block:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

The output of the algorithm consists of several components, where the ciphertext $\mathbf{C}$ also includes n data blocks $\mathbf{C}_i$ of size m bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}.$$

In addition to the $\mathbf{C}$ ciphertext, the output data of the encryption algorithm has the following components:

- $\mathbf{C}_0$ is the component responsible for generates subkey $\mathbf{K}_1$ of the $\mathbf{K_{OT}}$ key.
- $\mathbf{R}$ is the component responsible for verifying the integrity of the post-decrypted message.

The Encryption algorithm is performed through the following steps:

Step 1: Calculate the value of the sender's shared secret key $\mathbf{K}_s$ according to the formula:

$$\mathbf{K}_s = (y_r)^{x_s} \bmod p.$$

Step 2: Generate key $\mathbf{K}_e$ from key $\mathbf{K}_s$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_e = \mathbf{H}\left(\mathbf{K}_s\right).$$

Step 3: Generate a data block $\mathbf{P}_0$ of size m bits using the random/pseudo-random number generator $\mathbf{PRNG}()$:

$$\mathbf{P}_0 = \mathbf{PRNG}\left(\{1, 2, \ldots, 2^m - 1\}\right).$$

Step 4: Calculate the component $\mathbf{C}_0$ from data block $\mathbf{P}_0$ and key $\mathbf{K}_e$ by operator $\mathbf{XOR}$:

$$\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_e.$$

Step 5: Generate key $\mathbf{K}_0$ from data block $\mathbf{P}_0$ and key $\mathbf{K}_e$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_e).$$

Step 6: Encrypt the data blocks of the plaintext $\mathbf{P}$, the encryption process $\mathbf{P}$ will end if the value of the encrypted data block is $\mathbf{ESC}$

$i = 1;$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$i = i + 1$
end

Step 7: Generate component $\mathbf{R}$ from plaintext $\mathbf{P}$ and subkey $\mathbf{K}_i$ by the hash function $\mathbf{H}()$:

$$\mathbf{R} = \mathbf{H}\left(\mathbf{P} || \mathbf{K}_i\right).$$

The Encryption algorithm is described in pseudocode as follows:

Algorithm 2.2a :
input : $g, p, x_s, y_r, \mathbf{P}$
output : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}$
$[1].\mathbf{K}_s = (y_r)^{x_s} \bmod p$
$[2].\mathbf{K}_e = \mathbf{H}(\mathbf{K}_s)$
$[3].\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
$[4].\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_e$
$[5].\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_e)$
$[6].\ i = 1$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$i = i + 1$
end
$[7].\ \mathbf{R} = \mathbf{H}(\mathbf{P} \| \mathbf{K}_i)$.

In case the size of the message to be encrypted (value n) has been determined, the Encryption algorithm is described in pseudocode as follows:

Algorithm 2.2b :
input : $g, p, x_s, y_r, \mathbf{P}$
output : $\mathbf{C_0}, \mathbf{C}, \mathbf{R}$
$[1].\mathbf{K}_s = (y_r)^{x_s} \bmod p$
$[2].\mathbf{K}_e = \mathbf{H}(\mathbf{K}_s)$
$[3].\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
$[4].\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_e$
$[5].\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_e)$
$[6].$ **for** $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
end
$[7].\ \mathbf{R} = \mathbf{H}(\mathbf{P} \| \mathbf{K}_n)$.

2.2.3. The Decryption – Authentication algorithm: The Decryption - Authentication algorithm takes as input the ciphertext $\mathbf{C}$, components $(\mathbf{C_0}, \mathbf{R})$, the receiver's secret

key x_r, the sender's public key y_s and system parameters. The ciphertext $\mathbf{C}$ consists of n data blocks $\mathbf{C}_i$ of size m bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}.$$

It should also be noted that the value of n here may not be determined when the algorithm is executed.

The output data of this algorithm is a post-decrypted message $\mathbf{M}$ consisting of n data blocks of size m bits:

$$\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}.$$

The one-time key $\mathbf{K_{OT}}$ used to decrypt the received message - similar to the sender (encryptor) side, consists of n subkeys $\mathbf{K}_i$ of size m bits:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

The Decryption - Authentication algorithm is performed through the following steps:
Step 1: Calculate the value of the receiver's shared secret key $\mathbf{K}_r$ according to the formula:

$$\mathbf{K}_r = (y_s)^{x_r} \bmod p.$$

Step 2: Generate key $\mathbf{K}_d$ from key $\mathbf{K}_r$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_d = \mathbf{H}(\mathbf{K}_r).$$

Step 3: Decrypt component $\mathbf{C}_0$ using key $\mathbf{K}_d$ and operator $\mathbf{XOR}$:

$$\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_d.$$

Step 4: Generate key $\mathbf{K}_0$ from data block $\mathbf{M}_0$ and key $\mathbf{K}_d$ by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_d).$$

Step 5: Decrypt the data blocks of the ciphertext $\mathbf{C}$, the decryption process $\mathbf{C}$ will end if the value of the decrypted data block is $\mathbf{ESC}$:

$i = 1;$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1} || \mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$i = i + 1$
end

Step 6: Generate the value $\mathbf{V}$ from the post-decrypted plaintext $\mathbf{M}$ and the subkey $\mathbf{K}_i$ by the hash function $\mathbf{H}()$:

$$\mathbf{V} = \mathbf{H}(\mathbf{M} \| \mathbf{K}_i).$$

Step 7: Check if $\mathbf{V} = \mathbf{R}$ then the integrity of the post-decrypted plaintext $\mathbf{M}$ is authenticated. Otherwise, the message has been modified.

The Decryption - Authentication algorithm is described in pseudocode as in Algorithm 2.3a and in case the size of the message to be encrypted (value n) has been determined, the Decryption - Authentication algorithm is described in pseudocode as in Algorithm 2.3b:

Algorithm 2.3a :
input : $g, p, x_r, y_s, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
output : $\mathbf{M}$, **TRUE/FALSE**
[1]. $\mathbf{K}_r = (y_s)^{x_r} \bmod p$
[2]. $\mathbf{K}_d = \mathbf{H}(\mathbf{K}_r)$
[3]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_d$
[4]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_d)$
[5]. $i = 1$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1} || \mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$i = i + 1$
end
[6]. $\mathbf{V} = \mathbf{H}(\mathbf{M} \| \mathbf{K}_i)$.
[7]. **if** $\mathbf{V} = \mathbf{R}$ **then return** $(\mathbf{M}, \mathbf{TRUE})$
else return $(\mathbf{M}, \mathbf{FALSE})$.

Algorithm 2.3b :
input : $g, p, x_r, y_s, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
output : $\mathbf{M}$, **TRUE/FALSE**
[1]. $\mathbf{K}_r = (y_s)^{x_r} \bmod p$
[2]. $\mathbf{K}_d = \mathbf{H}(\mathbf{K}_r)$
[3]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_d$
[4]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_d)$
[5]. **for** $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1} || \mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
end
[6]. $\mathbf{V} = \mathbf{H}(\mathbf{M} \| \mathbf{K}_n)$.
[7]. **if** $\mathbf{V} = \mathbf{R}$ **then return** $(\mathbf{M}, \mathbf{TRUE})$
else return $(\mathbf{M}, \mathbf{FALSE})$.

Note:

- If the return is $(\mathbf{M}, \mathbf{TRUE})$, then the post-decrypted message $\mathbf{M}$ is exactly the plaintext $\mathbf{P}$, that is: $\mathbf{M} = \mathbf{P}$.
- If the return is $(\mathbf{M}, \mathbf{FALSE})$, then the post-decrypted message has been modified, that is: $\mathbf{M} \neq \mathbf{P}$.

2.2.4. The correctness of the proposed schema: It is easy to see that, the correctness of the proposed scheme here can be stated and proven completely similar to the scheme in section 2.1 if it can be confirmed that shared secret key ($\mathbf{K}_s$) of the sender (encryptor) is also shared secret key ($\mathbf{K}_r$) of the receiver (decryptor).

Indeed, we have:

$$\begin{aligned}\mathbf{K}_r &= (y_s)^{x_r} \bmod p = (g^{x_s} \bmod p)^{x_r} \bmod p \\ &= (g^{x_r} \bmod p)^{x_s} \bmod p = (y_r)^{x_s} \bmod p = \mathbf{K}_s.\end{aligned}$$

It follows that: $\mathbf{K}_d = \mathbf{K}_e$.

From that, it is possible to prove the correctness of the proposed scheme here similar

to the scheme in section 2.1.

2.2.5. The security level of the proposed schema: It is also easy to see that, the only difference of the proposed scheme here compared to the scheme in section 2.1 is that this scheme has 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 scheme proposed here, the cryptanalyst can attack the Parameter and Key Generation algorithm (Algorithm 2.1) 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 $\mathbb{F}_p$. Currently, no polynomial time algorithm has been published for this hard problem [9]–[20].

2.3. Public key block cipher scheme authenticates the integrity of each block of data

The scheme proposed here is construct to perform encryption and decryption - authentication with each data block of the message to be encrypted, so it will be suitable for encryption applications with real-time data. This scheme includes the Parameter and Key Generation algorithm, the Encryption algorithm and the Decryption - Authentication algorithm. In which, the Parameter and Key Generation algorithm is completely similar to the Parameter and Key Generation algorithm in the schema in section 2.1 (Algorithm 2.1), the Encryption algorithm (Algorithm 3.1) and the Decryption - Authentication algorithm (Algorithm 3.2) is described as follows:

2.3.1. The Encryption algorithm: In this scheme, the Encryption algorithm takes as input the plaintext $\mathbf{P}$, the sender's secret key x_s, the receiver's public key y_r, and system parameters.

The plaintext $\mathbf{P}$ is encrypted as n blocks of data $\mathbf{P}_i$ of size m bits:

$$\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n\}.$$

It should also be noted that the value of n here may not be determined when the algorithm is executed.

The one-time key $\mathbf{K_{OT}}$ used to encrypt $\mathbf{P}$ consists of n subkeys $\mathbf{K}_i$ of size m bits corresponding to the size of the plaintext block:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

The output of the algorithm consists of several components, where the ciphertext $(\mathbf{C}, \mathbf{R})$ also includes n data blocks $(\mathbf{C}_i, \mathbf{R}_i)$ of size (m, m) bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$$

and:

$$\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n\}.$$

In addition to the ciphertext $\mathbf{C}$, the output of this algorithm has other components as follows:

- $(\mathbf{R}_0, \mathbf{C}_0)$: These components are responsible for verifying the origin of the encrypted message and generating subkey $\mathbf{K}_1$ of key $\mathbf{K_{OT}}$.
- $\mathbf{R}$: This component is responsible for verifying the integrity of data blocks of the encrypted message.

The Encryption algorithm is performed through the following steps:

Step 1: Calculate the value of the sender/encryptor's shared secret key $\mathbf{K}_s$ according to the formula: $\mathbf{K}_s = (y_r)^{x_s} \bmod p$.

Step 2: Generate a data block $\mathbf{P}_0$ of size m bits by the random/pseudo-random number generator $\mathbf{PRNG}()$: $\mathbf{P}_0 = \mathbf{PRNG}\left(\{1, 2, \ldots, 2^m - 1\}\right)$.

Step 3: Generate component $\mathbf{R}_0$ from data block $\mathbf{P}_0$ and sender's shared secret key $\mathbf{K}_s$ by hash function $\mathbf{H}()$ has an output data size of m bits: $\mathbf{R}_0 = \mathbf{H}\left(\mathbf{P}_0 \| \mathbf{K}_s\right)$.

Step 4: Generate the key $\mathbf{K}_0$ from the component $\mathbf{R}_0$ and the shared secret key $\mathbf{K}_s$ by hash function $\mathbf{H}()$: $\mathbf{K}_0 = \mathbf{H}\left(\mathbf{R}_0 \| \mathbf{K}_s\right)$.

Step 5: Calculate the component $\mathbf{C}_0$ from data block $\mathbf{P}_0$ and key $\mathbf{K}_0$ by operator $\mathbf{XOR}$:

$$\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_0.$$

Step 6: Send $(\mathbf{R}_0, \mathbf{C}_0)$ to the receiver (decryptor).

Step 7: Encrypt the data blocks of the plaintext $\mathbf{P}$, the encryption process $\mathbf{P}$ will end if the value of the encrypted data block is $\mathbf{ESC}$ and the Encryption algorithm is described in pseudocode as in Algorithm 3.1a:

$i = 1;$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
  begin
    $\mathbf{K}_i = \mathbf{H}\left(\mathbf{P}_{i-1} || \mathbf{K}_{i-1}\right)$
    $\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
    $\mathbf{R}_i = \mathbf{H}\left(\mathbf{K}_i || \mathbf{P}_i\right)$
    $i = i + 1$
    Send $(\mathbf{R}_i, \mathbf{C}_i)$ to receiver
  end

The Encryption algorithm is described in pseudocode as follows:

Algorithm 3.1a :
input : $g, p, x_s, y_r, \mathbf{P}$
output : $\mathbf{R_0}, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
[1]. $\mathbf{K}_s = (y_r)^{x_s} \bmod p$
[2]. $\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
[3]. $\mathbf{R}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_s)$
[4]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_s)$
[5]. $\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_0$
[6]. Send $(\mathbf{R}_0, \mathbf{C}_0)$ to receiver
[7]. $i = 1$
while $(\mathbf{P}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{P}_{i-1} || \mathbf{K}_{i-1})$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$\mathbf{R}_i = \mathbf{H}(\mathbf{K}_i \| \mathbf{P}_i)$
$i = i + 1$
Send $(\mathbf{C}_i, \mathbf{R}_i)$ to receiver
end

In case the size of the message to be encrypted (value n) has been determined, the Encryption algorithm is described in pseudocode as follows:

Algorithm 3.1b :
input : $g, p, x_s, y_r, \mathbf{P}$
output : $\mathbf{R_0}, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
[1]. $\mathbf{K}_s = (y_r)^{x_s} \bmod p$
[2]. $\mathbf{P}_0 = \mathbf{PRNG}(\{1, 2, \ldots, 2^m - 1\})$
[3]. $\mathbf{R}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_s)$
[4]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{P}_0 \| \mathbf{K}_s)$
[5]. $\mathbf{C}_0 = \mathbf{P}_0 \oplus \mathbf{K}_0$
[6]. Send $(\mathbf{R}_0, \mathbf{C}_0)$ to receiver
[7]. **for** $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{P}_{i-1} || \mathbf{K}_{i-1})$
$\mathbf{C}_i = \mathbf{P}_i \oplus \mathbf{K}_i$
$\mathbf{R}_i = \mathbf{H}(\mathbf{K}_i \| \mathbf{P}_i)$
Send $(\mathbf{C}_i, \mathbf{R}_i)$ to receiver
end

2.3.2. The Decryption – Authentication algorithm: The input data of the Decrypt - Authentication algorithm includes ciphertext $(\mathbf{C}, \mathbf{R})$, components $(\mathbf{R_0}, \mathbf{C_0})$, receiver's secret key x_r, sender's public key y_s and system parameters.

The ciphertext $(\mathbf{C}, \mathbf{R})$ consists of n data blocks $(\mathbf{R_i}, \mathbf{C_i})$ of size (m, m) bits:

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$$

and

$$\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n\}.$$

Similar to the Encryption algorithm, it is important to note here that at the beginning of the implementation of the Decrypt - Authentication algorithm, the value of the parameter n is undefined.

The output data of this algorithm is a post-decrypted message $\mathbf{M}$ consisting of n data blocks of size m bits:

$$\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n\}.$$

The one-time key $\mathbf{K_{OT}}$ used to decrypt the received message - similar to the sender (encryptor) side, consists of n subkeys $\mathbf{K}_i$ of size m bits:

$$\mathbf{K_{OT}} = \{\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n\}.$$

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

Step 1: Calculate the value of the receiver's shared secret key $\mathbf{K}_r$ according to the formula:

$$\mathbf{K}_r = (y_s)^{x_r} \bmod p.$$

Step 2: Generate the key $\mathbf{K}_0$ from the component $\mathbf{R}_0$ and the shared secret key $\mathbf{K}_r$ of the receiver by the hash function $\mathbf{H}()$ has an output data size of m bits:

$$\mathbf{K}_0 = \mathbf{H}(\mathbf{R}_0 \| \mathbf{K}_r).$$

Step 3: Decrypt component $\mathbf{C}_0$ by key $\mathbf{K}_0$ and operator $\mathbf{XOR}$:

$$\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_0.$$

Step 4: Generate the value of $\mathbf{V}_0$ from $\mathbf{M}_0$ and $\mathbf{K}_r$ by the hash function $\mathbf{H}()$:

$$\mathbf{V}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_r).$$

Step 5: Check if: $\mathbf{V}_0 \neq \mathbf{R}_0$ then stop. Otherwise, the identity of the sender (encryptor) or the origin of the post-decrypted message is authenticated, executes the next steps.

Step 6: Decrypt the data blocks of ciphertext $\mathbf{C}$, the decryption process $\mathbf{C}$ will end if the value of the decrypted data block is $\mathbf{ESC}$ as follows:

$i = 1; j = 1$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1}||\mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$\mathbf{V}_i = \mathbf{H}(\mathbf{K}_i\|\mathbf{M}_i)$.
if $(\mathbf{V}_i \neq \mathbf{R}_i)$ then
begin
ErrorList[j]=i
$j = j + 1$
end
$i = i + 1$
end

The Decryption - Authentication algorithm is described in pseudocode as follows:

Algorithm 3.2a :
input : $g, p, x_r, y_s, \mathbf{R_0}, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
output : $\mathbf{M}, \mathbf{ErrorList}$
[1]. $\mathbf{K}_r = (y_s)^{x_r} \bmod p$
[2]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{R}_0\|\mathbf{K}_r)$
[3]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_0$
[4]. $\mathbf{V}_0 = \mathbf{H}(\mathbf{M}_0\|\mathbf{K}_r)$
[5]. **If** $(\mathbf{V}_0 \neq \mathbf{R}_0)$ **then break;**
[6]. $i = 1; j = 1;$
while $(\mathbf{M}_i \neq \mathbf{ESC})$
begin
$\mathbf{K}_i = \mathbf{H}(\mathbf{M}_{i-1}||\mathbf{K}_{i-1})$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$\mathbf{V}_i = \mathbf{H}(\mathbf{K}_i||\mathbf{M}_i)$
If $(\mathbf{V}_0 \neq \mathbf{R}_0)$ **then**
begin
ErrorList[j]=i
$j = j + 1$
end
$i = i + 1$
end

Note:

- $\mathbf{ErrorList}$: The list of data blocks of the received message is not verified for integrity.

In case the size of the message to be encrypted (value n) has been determined, the Decryption - Authentication algorithm is described in pseudocode as follows:

Algorithm 3.2b :
input : $g, p, x_r, y_s, \mathbf{C_0}, \mathbf{C}, \mathbf{R}$
output : $\mathbf{M}, \mathbf{ErrorList}$
[1]. $\mathbf{K}_r = (y_s)^{x_r} \bmod p$
[2]. $\mathbf{K}_0 = \mathbf{H}(\mathbf{R}_0 \| \mathbf{K}_r)$
[3]. $\mathbf{M}_0 = \mathbf{C}_0 \oplus \mathbf{K}_0$
[4]. $\mathbf{V}_0 = \mathbf{H}(\mathbf{M}_0 \| \mathbf{K}_r)$
[5]. **If** $(\mathbf{V}_0 \neq \mathbf{R}_0)$ **then break;**
[6]. $j = 1$;
for $i = 1$ **to** n **do**
begin
$\mathbf{K}_i = \mathbf{H}\left(\mathbf{M}_{i-1} || \mathbf{K}_{i-1}\right)$
$\mathbf{M}_i = \mathbf{C}_i \oplus \mathbf{K}_i$
$\mathbf{V}_i = \mathbf{H}\left(\mathbf{K}_i || \mathbf{M}_i\right)$
If $(\mathbf{V}_i \neq \mathbf{R}_i)$ **then**
begin
ErrorList[j]=i
$j = j + 1$
end
end

2.3.3. The correctness of the proposed schema: The basic difference between the scheme proposed here and the schemes mentioned in section 2.1 and section 2.2 is that data integrity verification is performed for each data block of the encrypted message, but not for the whole message as in the schemes in section 2.1 and section 2.2. However, proving the correctness of the scheme proposed here can be done in exactly the same way as the schemes in section 2.1 and section 2.2.

2.3.4. The security level of the proposed schema: Evaluating the security level of the scheme proposed here can be done similarly to the schemes mentioned in section 2.1 and section 2.2. It can be seen that the security level of this scheme is completely equivalent to the security level of the scheme in section 2.2.

3. Conclusions

The block cipher schemes proposed here are developed based on the mechanism of OTP cipher, hash function and public key cryptography. The advantage of these schemes is that their security and performance are inherited from the OTP cipher, but the shared secret key can be established for the encryption of different messages based on the mechanism of the public-key cryptography. These are very important properties for these block cipher schemes to be applicable in practice. In addition, because of the mechanism to authenticate the origin and integrity of the encrypted message, these block cipher schemes are also resistant to spoofing attacks, which is one of the basic requirements that practical applications posed. However, to apply the above results in practice, there needs to be a deeper assessment in terms of cryptanalysis, preventing spoofing attacks and improving implementation speed,...

References

[1] L. H. Dung and N. A. Viet, "A solution to build a symmetric key crypto system," *Journal of Science and Technology on Information Security, ISSN 1859 - 1256*, vol. 057, no. 5, pp. 21–25, 2020.

[2] L. H. Dung, N. A. Viet, and D. T. B. Ngoc, "An encryption and authentication algorithm developed based on the one – time pad cipher," *Journal of Military Science and Technology, ISSN: 1859 - 1403*, pp. 87–93, 12 2020. doi: 10.54939/1859-1043.j.mst

[3] L. H. Dung, T. M. Duc, and B. T. Truyen, "A variant of otp cipher with symmetric key solution," *Journal of Science and Technique - Section on Information and Communication Technology (ICT), ISSN: 1859 - 0209*, vol. 9, no. 16, pp. 38–47, 12 2020. doi: 10.56651/lqdtu.jst.v9.n02.210.ict

[4] N. K. Thanh, T. M. Thanh, and L. H. Dung, "A high performance encryption-authentication scheme," *Journal of Science and Technique - Section on Information and Communication Technology (ICT), ISSN: 1859 - 0209*, vol. 11, no. 01, pp. 114–121, 2022. doi: 10.56651/lqdtu.jst.v11.n01.360.ict

[5] L. H. Dung, "A method for constructing public-key block cipher schemes," *Journal of Military Science and Technology*, pp. 114–121, Dec 2022. doi: https://doi.org/10.54939/1859-1043.j.mst.CSCE6.2022.114-121

[6] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*. Boston, Kluwer Academic Publishers, 1993.

[7] "Nist fips pub 180-1," Tech. Rep., Apr 1995.

[8] N. P. Trang and L. H. Dung, "A type of public-key block cipher algorithm," *Journal of Military Science and Technology*, dec 2023. doi: https://doi.org/10.54939/1859-1043.j.mst.CSCE7.2023.50-60

[9] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC, Aug 2007.

[10] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*. New York Springer, 2014.

[11] L. C. Washington, "Elliptic curves - number theory and cryptography," May 2003. doi: https://doi.org/10.4324/9780203484029

[12] D. R. Stinson, *Cryptography*. Chapman and Hall/CRC, Nov 2005.

[13] R. A. Mollin, *An Introduction to Cryptography*. Chapman and Hall/CRC, Sep 2006.

[14] J. Talbot and D. Welsh, *Complexity and Cryptography*. Cambridge University Press, Jan 2006.

[15] J. H. Silverman, "Elliptic curves and cryptography," pp. 91–112, 2005.

[16] J. A. Buchmann, *Introduction to Cryptography*. Springer US, 2001.

[17] W. Mao, *Modern Cryptography. Theory and Practice*. Pearson Education, 2004.

[18] I. Shparlinski, Ed., *Cryptographic Applications of Analytic Number Theory*. Birkhäuser Basel, 2003.

[19] S. S. Wagstaff, *Cryptanalysis of Number Theoretic Ciphers*. Chapman and Hall/CRC, Aug 2019.

[20] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*. Cambridge University Press, Jul 1999.

Manuscript received 31-07-2023; Accepted 21-12-2023.

■

Hong Dung Luu graduated from University in Radio Electronics in 1989, Master in Electronics and Communication Engineering in 2000, Doctorate in Electronic Engineering in 2013 from Le Quy Don Technical University. Currently, he is a lecturer at the Institute of Information and Communication Technology - Le Quy Don Technical University. Research field: cryptography and information security. E-mail: luuhongdung@gmail.com

MẬT MÃ KHỐI KHÓA CÔNG KHAI

Lưu Hồng Dũng

Tóm tắt

Bài báo đề xuất một dạng lược đồ mật mã khối dựa trên hàm băm mật mã và mật mã khóa công khai. Lược đồ được đề xuất ở đây có khả năng xác minh nguồn gốc và tính toàn vẹn của bản tin được mã hóa. Mặt khác, việc thiết lập khóa bí mật chung giữa người gửi/người mã hóa và người nhận/người giải mã có thể được thực hiện riêng biệt cho từng bản tin.

Từ khóa

Mã khối, mật mã đối xứng, mật mã khoá công khai, lược đồ mã hoá có xác thực, mã hoá một lần OTP, bài toán logarithm rời rạc.