A type of public - key block cipher algorithm

Nong Phuong Trang, Luu Hong Dung*

Military Technical Academy.
*Corresponding author: luuhongdung@mta.edu.vn
Received 11 Oct. 2023; Revised 08 Dec. 2023; Accepted 12 Dec. 2023; Published 30 Dec. 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.CSCE7.2023.50-60

ABSTRACT

The paper proposes a type of block cipher algorithm based on cryptographic hash function and public key cryptography. The algorithm 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.

Keywords: Symmetric Key Cryptography; Public Key Cryptography; Encryption - Authentication Algorithm; OTP Cipher; Block Cipher; Public – Key Block Cipher.

1. INTRODUCTION

In [1-3], a solution for constructing a symmetric key cryptosystem has been proposed, which is developed from OTP cipher. The advantage of the algorithms constructed according to this method [1-3] is that the security and efficiency are inherited from the OTP cipher, but the shared secret key between the sender and the receiver 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, etc.).

Based on the method proposed in [1-3], the paper continues to propose a new type of block cipher algorithm, which is called a public key block cipher. Algorithms of this type, in addition to the security and authentication features of the encrypted message, also allow the establishment of a shared secret key between the sender and the receiver based on the mechanism of the public key cryptography for each message separately.

2. THE PROPOSED BLOCK CIPHER ALGORITHMS

2.1. The type of block cipher algorithm developed based on OTP Cipher

The type of block cipher algorithm proposed here is constructed based on the solution mentioned in [1-3] with some improvements to ensure exactly the same key usage, management and distribution as other block cipher algorithms. Furthermore, the shared secret key size need not be kept secret as in these algorithms in [1-3]. It includes: the Encryption algorithm (Algorithm 1.1) and the Decryption - Authentication algorithm (Algorithm 1.2). These algorithms are described as follows:

2.1.1. The Encryption algorithm

In the proposed type of block cipher algorithm, the Encryption algorithm takes as input a plaintext P and a shared secret key K of the sender, here, key K has size m bits. The plaintext P is encrypted as n data blocks P_i (i = 1,2,..,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_{ei} (i =

1,2,..,n) whose size corresponds to the size of the plaintext block:

$$K_{EOT} = \{K_{e1}, K_{e2},\ldots, K_{en}\}$$

The output of the algorithm consists some of the components, where the C ciphertext also includes n data blocks C_i (i = 1,2,..,n) of size m bits:

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

In addition to the C ciphertext, the output data of the encryption algorithm has the following components:

– C_0: Component responsible for generates subkey K_{e1} of the receiver's K_{EOT} key.

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

The Encryption algorithm is performed through the following steps:

– Step 1: Generate a data block P_0 of size m bits using the random/pseudo-random number generator PRNG():

$$P_0 = PRNG(\{1,2,\ldots, 2^m - 1\})$$

– Step 2: Calculate the component C_0 from data block P_0 and key K by operator XOR:

$$C_0 = P_0 \oplus K$$

– Step 3: Generate key K_{e0} from component P_0 and key K by the hash function H() has an output data size of m bits:

$$K_{e0} = H(P_0 \,||\, K)$$

– Step 4: Encrypt n data blocks of plaintext P:

for i = 1 to n do
begin
$K_{ei} = H(P_{i-1}||K_{ei-1})$
$C_i = P_i \oplus K_{ei}$
end

– Step 5: Generate component R from plaintext P and subkey K_{en} by the hash function H():

$$R = H(P||K_{en})$$

The Encryption algorithm (Algorithm 1.1) is described in pseudocode as follows:

Algorithm 1.1:

input: P, K.

output: C_0, C, R.

[1]. $P_0 = PRNG(\{1,2,\ldots,2^m-1\})$

[2]. $C_0 = P_0 \oplus K$

[3]. $K_{e0} = H(P_0 \,||\, K)$

[4]. for i = 1 to n do

begin

$K_{ei} = H(P_{i-1}||K_{ei-1})$

$C_i = P_i \oplus K_{ei}$

end

[5]. $R = H(P||K_{en})$

Note:

– Operator "⊕" is a modulo 2 addition operation (XOR).

– Operator "||" is the operation to concatenate two bit strings.

– The hash function H() here is m bits in size and selectable as: MD5 [4], SHA-1/ SHA-256 [5],...

2.1.2. The Decryption - Authentication algorithm

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

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

The output of the algorithm is a post-decrypted message M consisting of n data blocks M_i (i = 1,2,..,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 side, consists of n subkeys K_{di} (i = 1,2,..,n) of size m bits:

$$K_{DOT} = \{K_{d1}, K_{d2},\ldots, K_{dn}\}$$

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

– Step 1: Decrypt the C_0 component using the key K and the operator XOR:

$$M_0 = C_0 \oplus K$$

– Step 2: Generate the key K_{r0} from the component M_0 and the key K by the hash function H() has an output data size of m bits:

$$K_{d0} = H(M_0 \,||\, K)$$

– Step 3: Decrypt the data blocks of ciphertext C to get post-decrypted message M:

for i = 1 to n do

begin

$K_{di} = H(M_{i-1}||K_{di-1})$

$M_i = C_i \oplus K_{di}$

end

– Step 4: Generate the value V from the post-decrypted message M and the subkey K_{dn} by the hash function H():

$$V = H(M||K_{dn})$$

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

The Decryption - Authentication algorithm (Algorithm 1.2) is described in

pseudocode as follows:

Algorithm 1.2:

input: C_0, C, R, K.

output: M,TRUE/FALSE.

[1]. $M_0 = C_0 \oplus K$

[2]. $K_{d0} = H(M_0 \,||\, K)$

[3]. for i = 1 to n do

begin

$K_{di} = H(M_{i-1}||K_{di-1})$

$M_i = C_i \oplus K_{di}$

end

[4]. $V = H(M||K_{dn})$

[5]. 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 plantext 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

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): M = P and the conditions: V = R will be satisfied. Therefore, after decryption, if the condition: V = 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 and the value C_0 received is also the value C_0 sent, so we have:

$$M_0 = C_0 \oplus K = P_0 \oplus K \oplus K = P_0$$

and:

$$K_{d0} = H(M_0 \,||\, K) = H(P_0 \,||\, K) = K_{e0}$$

Because: $M_0 = P_0$, $K_{d0} = K_{e0}$ and how to generate subkey K_{ei} (i = 1,2,..,n) of sender: $K_{ei} = H(P_{i-1}||K_{ei-1})$ is also the way to generate the subkey K_{di} (i = 1,2,..,n) of the receiver: $K_{di} = H(M_{i-1}||K_{di-1})$. 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:

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

And there's the second thing to prove:

$$V = H(M||K_{dn}) = H(P||K_{en}) = R$$

2.1.4. Some evaluation of the security level of the proposed algorithm

Similar to the OTP cipher, the K_{OT} 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 algorithm is assessed by its ability to resist some typical attacks as follows:

– *Ciphertext-only Attack*: The analysis in [1-3] has shown that a direct attack on the proposed algorithm is not viable when only ciphertext is available. If the K_{EOT} key were a truly random string of bits, there wouldn't be any relationship between the plaintext and the ciphertext. Therefore, a "brute force" attack can decrypt a ciphertext into any meaningful message of the same length (as the ciphertext). So, for an attacker, all plaintexts after decryption are likely to be encrypted messages (true plaintext). That is, there will not be any information in the ciphertext that would allow an attacker to select the correct plaintext from the meaningful messages after decryption using a "brute-force" attack. Additionally, if K_{EOT} is truly random, then from a known key, an attacker cannot find whether other keys were generated before or after.

In the proposed scheme, the method of generating a K_{EOT} key for each encryption of a message can completely meet the requirements for randomness of the key in the following aspects: a) from a known key, attackers cannot find keys that have been previously generated or will be generated later; b) each key is a string of bits with no repeating cycle, so there will be no relationship between the plaintext and the ciphertext. Therefore, the proposed scheme can completely resist the "brute force" attack when the attacker only knows the ciphertext.

– *Known-plaintext attack:* Knowing the plaintext, the attacker will calculate the K_{d1} subkey according to: $K_{d1} = C_1 \oplus M_1$. From: $K_{d1} = H(M_0 \| K_{d0})$, $K_{d0} = H(M_0 \| K)$ the attacker will be able to establish the equation:

$$K_{d1} = H(M_0 \| H(M_0 \| K))$$

However, to find the shared secret key K from the above equation with known values of K_{d1}, the attacker must perform a "brute force" attack.

– *Spoofing attack*: The OTP cipher does not provide verification for an encrypted message, so an attacker could block the ciphertext was sent and send the recipient a fake ciphertext of the same size as the true message. In the case of decrypting to a meaningless plaintext, the receiver may speculate that the tampering was made or caused by a communication error. However, if decrypted to a meaningful plaintext, then the receiver has no way of confirming 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: V = R. Furthermore, no one other than the sender and receiver can satisfy the above conditions. That allows the scheme to be resistant to spoofing attacks.

2.2. The type of public-key block cipher algorithm

The type of public-key block cipher algorithm proposed here is developed based on the type of block cipher algorithm proposed 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 type of public-key block cipher algorithm proposed here includes the Parameter and Key Generation algorithm (Algorithm 2.1), the Encryption algorithm (Algorithm 2.2) and the Decryption - Authentication algorithm (Algorithm 2.3). These algorithms are described as follows:

2.2.1. The Parameter and Key Generation algorithm

In the block cipher scheme 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. These key pairs are generated by Algorithm 2.1 as follows:

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)* = 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:

$$g = \alpha^{\frac{p-1}{q}} \bmod p, \text{ satisfy: } g \neq 1.$$

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

[4]. Calculate the public key y according to the formula:

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

Note:

– *len()*: The function that calculates the length (in bits) of an integer.

– x, y: The public key and the private key of end-user in system.

– p, q, g: 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 is:

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

and the corresponding public key of the receiver is:

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

2.2.2. The Encryption algorithm

In the type of public-key block cipher algorithm 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,..,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_{ei} (i = 1,2,..,n) whose size corresponds to the size of the plaintext block:

$$K_{EOT} = \{K_{e1}, K_{e2}, \ldots, K_{en}\}$$

The output of the algorithm consists of several components, where the ciphertext C also includes n data blocks C_i (i = 1,2,..,n) of size m bits:

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

In addition to the ciphertext C, the output data of the algorithm has the following components:

– C_0: Component used to generate subkey K_{e1} of the receiver's key K_{EOT}.

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

The Encryption algorithm is performed through the following steps:

– Step 1: Calculate the value K_s according to the formula:

$$K_s = \left(y_r\right)^{x_s} \bmod p$$

– Step 2: Calculate the value of the sender's shared secret key K_e by the hash function H() has an output data size of m bits:

$$K_e = H(K_s)$$

– Step 3: Generate a data block P_0 of size m bits using the random/pseudo-random number generator PRNG():

$$P_0 = PRNG(\{1,2,\ldots,\ 2^m - 1\})$$

– Step 4: Calculate the component C_0 from data block P_0 and key K_e by operator XOR:

$$C_0 = P_0 \oplus K_e$$

– Step 5: Generate key K_{e0} from component P_0 and key K_e by the hash function H():

$$K_{e0} = H(P_0 \,||\, K_e)$$

– Step 6: Encrypt n data blocks of plaintext P:

```
for i = 1 to n do
    begin
      Kei = H(Pi-1||Kei-1)
      Ci = Pi ⊕ Kei
    end
```

– Step 7: Generate component R from plaintext P and subkey K_{en} by the hash function H():

$$R = H(P||K_{en})$$

The Encryption algorithm (Algorithm 2.2) is described in pseudocode as follows:

Algorithm 2.2:

input: g, p, x_s, y_r, P.

output: C_0, C, R.

[1]. $K_s = \left(y_r\right)^{x_s} \bmod p$

[2]. $K_e = H(K_s)$

[3]. $P_0 = PRNG(\{1,2,\ldots,2^m\text{-}1\})$

[4]. $C_0 = P_0 \oplus K_e$

[5]. $K_{e0} = H(P_0 \,||\, K_e)$

[6]. for i = 1 to n do

begin

$K_{ei} = H(P_{i-1}||K_{ei-1})$

$C_i = P_i \oplus K_{ei}$

end

[7]. $R = H(P||K_{en})$

2.2.3. The Decryption – Authentication algorithm

The Decryption - Authentication algorithm takes as input the ciphertext C, components C_0, R, 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,..,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,..,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_{di} (i = 1,2,..,n) of size m bits:

$$K_{DOT} = \{K_{d1}, K_{d2},\ldots, K_{dn}\}$$

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

– Step 1: Calculate the value K_r according to the formula:

$$K_r = \left(y_s\right)^{x_r} \bmod p$$

– Step 2: Generate the value of the receiver's shared secret key K_d by the hash function H() has an output data size of m bits:

$$K_d = H(K_r)$$

– Step 3: Decrypt component C_0 using key K_d and operator XOR:

$$M_0 = C_0 \oplus K_d$$

– Step 4: Generate the key K_{d0} from M_0 and the key K_d by the hash function H():

$$K_{d0} = H(M_0 \,||\, K_d)$$

– Step 5: Decrypt the data blocks of ciphertext C to get post-decrypted plaintext M:

for i = 1 to n do

begin

$K_{di} = H(M_{i-1}||K_{di-1})$

$M_i = C_i \oplus K_{di}$

end

– Step 6: Generate the value of V from the post-decrypted plaintext M and the subkey K_{dn} by the hash function H():

$$V = H(M||K_{dn})$$

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

The Decryption - Authentication algorithm (Algorithm 2.3) is described in

pseudocode as follows:

Algorithm 2.3:

input: g, p, x_r, y_s, C_0, C, R.

output: M, TRUE/FALSE.

[1]. $K_r = (y_s)^{x_r} \bmod p$

[2]. $K_d = H(K_r)$

[3]. $M_0 = C_0 \oplus K_d$

[4]. $K_{d0} = H(M_0 \,||\, K_d)$

[5]. for i = 1 to n do

begin

$K_{di} = H(M_{i-1}||K_{di-1})$

$M_i = C_i \oplus K_{di}$

end

[6]. $V = H(M||K_{dn})$

[7]. 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 plantext P, that is: M = P.

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

2.2.4. The correctness of the proposed algorithm

It is easy to see that, the correctness of the proposed algorithm here can be stated and proven completely similar to the algorithm in ***section 2.1*** if it can be confirmed that the sender's shared secret key (K_s) is also the receiver's shared secret key (K_r).

Indeed, we have:

$$K_r = (y_s)^{x_r} \bmod p = \left(g^{x_s} \bmod p\right)^{x_r} \bmod p = \left(g^{x_r} \bmod p\right)^{x_s} \bmod p = (y_r)^{x_s} \bmod p = K_s$$

Inferred: $K_d = K_e$

From that, it is possible to prove the correctness of the proposed algorithm here which is similar to the algorithm in section 2.1.

2.2.5. The security level of the proposed algorithm

It is also easy to see that, the only difference of the proposed algorithm here compared to the algorithm in section 2.1 is that this algorithm has the additional feature of establishing a shared secret key between the sender and the receiver based on the mechanism of public key cryptography. It also means that in the algorithm proposed here, the cryptanalyst can attack the Parameter and Key Generation algorithm (Algorithm 2.1) 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 finite field F_p. Currently, no polynomial time algorithm has been published for this hard problem [6-18].

3. CONCLUSIONS

The block cipher algorithms proposed here are developed based on the mechanism of OTP cipher, hash function and public – key cryptography. The advantage of these algorithms is that their security and performance are inherited from the OTP, 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 algorithms to be applicable in practice. In addition, because of the mechanism to authenticate the origin and integrity of the encrypted message, these block cipher algorithms are also resistant to spoofing attacks, which is one of the basic requirements that practical applications posed.

REFERENCES

[1]. Lưu Hồng Dũng, Nguyễn Ánh Việt. *"Một giải pháp xây dựng hệ mật khóa đối xứng"*. Tạp chí An toàn Thông tin, ISSN 1859 - 1256, Số 5 (057), (2020) (in Vietnamese).

[2]. Lưu Hồng Dũng, Nguyễn Ánh Việt, Đoàn Thị Bích Ngọc. *"Thuật toán mã hóa - xác thực thông tin phát triển từ mật mã otp"*. Journal of Military Science and Technology, CSCE special issue, 87-93, (2020) (in Vietnamese).

[3]. Luu Hong Dung, Tong Minh Duc, Bui The Truyen. *"A variant of OTP cipher with symmetric key solution"*. Journal of Science and Technique - Section on Information and Communication Technology (ICT) - No. 16, (2020). DOI: 10.56651/lqdtu.jst.v9.n02.210.ict.

[4]. A. Menezes. *"Elliptic Curve Public Key Cryptosystems"*. The Kluwer International Series in Engineering and Computer Science, 234. Kluwer Academic Publishers, Boston, (1993)

[5]. National Institute of Standards and Technology, NIST FIPS PUB 180-1. (1995)

[6]. J. KATZ, Y. LINDELL. *"Introduction to Modern Cryptography"*. Chapman & Hall/CRC (2008).

[7]. Jeffrey Hoffstein, Jill Pipher and Joseph H. Silverman. *"An Introduction to Mathematical Cryptography"*. ISBN 978-0-387-77993-5. Springer - Verlag, (2008).

[8]. L.C. WASHINGTON. *"Elliptic Curves. Number Theory and Cryptography"*. Chapman & Hall/CRC, (2008).

[9]. D.R. STINSON. *"Cryptography. Theory and Practice"*. Chapman & Hall/CRC, (2006).

[10]. R.A. MOLLIN. *"An Introduction to Cryptography"*. Chapman & Hall/CRC, (2006).

[11]. J. Talbot and D. Welsh. *"Complexity and Cryptography: An Introduction"*. Cambridge University Press, (2006).

[12]. J. H. Silverman. *"Elliptic curves and cryptography"*. In Public-Key Cryptography, volume 62 of Proc. Sympos. Appl. Math, pages 91–112. Amer. Math. Soc., Providence, RI, (2005).

[13]. J. BUCHMANN. *"Introduction to Cryptography"*. Springer–Verlag, (2004).

[14]. W. MAO. *"Modern Cryptography. Theory and Practice"*. Pearson Education, (2004).

[15]. I. SHPARLINSKI. *"Cryptographic Applications of Analytic Number Theory"*. Complexity Lower Bounds and Pseurandomness. Birkhäuser, (2003).

[16]. S.S. WAGSTAFF. *"Cryptanalysis of Number Theoretic Ciphers"*. Chapman & Hall/CRC, (2003).

[17]. I. F. Blake, G. Seroussi, and N. P. Smart. *"Elliptic Curves in Cryptography"*, volume 265 of London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, (2000).

[18]. I. BLAKE, G.SEROUSSI & N. SMART. *"Elliptic Curves in Cryptography"*. Cambridge University Press, (2000).

TÓM TẮT

Một dạng thuật toán mã khối khóa công khai

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

Từ khoá: Mật mã khóa đối xứng; Mật mã khóa công khai; Thuật toán mã hóa - xác thực; Mật mã OTP; Mật mã khối; Mật mã khối khóa công khai.