Improving One-Time Pad Algorithm on Shamir's Three-Pass Protocol Scheme by Using RSA and ElGamal Algorithms

Agung Purnomo Sidik¹, Syahril Efendi², and Suherman Suherman³

Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan 20155, Indonesia

¹agungkomed@gmail.com
²syahril1@usu.ac.id
³suherman@usu.ac.id

Abstract. This study aims to cover the shortcomings of the one-time pad algorithm used in the Shamir's three-pass protocol scheme. In this study, the technique used to cover the weakness of one-time pad algorithm is by changing each ciphertext generated from the three paths in the three pass protocol scheme. Ciphertext modification is performed by encrypting the cipher-text by using the RSA and Elgamal algorithms to generate super ciphertext. The first line and the third line are encrypted by using the RSA algorithm and the second line is using ElGamal algorithm. Both algorithms are aimed at keeping no secret key exchange which is a major requirement of the three pass protocol scheme. The evaluation results show that the weakness of one-time pad algorithm can be overcome. Although the processing time of three pass protocol increase, but the longer the plaintext character the closer the processing time ratio to one. The research shows the only way for plaintext to be formed by cryptoanalysis is they must succeed to decrypt the three super ciphertexts back into the three initial ciphertexts first. But it is not easy if RSA and ElGamal algorithms use the big prime number.

1. Introduction
The three pass protocol scheme was developed by Adi Shamir, an Israeli professor around 1980 [1]. The protocol has a three-track framework for delivering messages or data. This protocol also allows two parties to exchange information safely without having to exchange keys, so the key distribution problem that existed in the symmetric algorithm can be resolved properly.

The XOR (exclusive-OR) technique can be implemented in this protocol scheme, such as one-time pad algorithm, Vigenere cipher, and Caesar cipher. In addition, matrix-based algorithms such as hill cipher can also be implemented in the scheme of this protocol [2][3]. When the XOR technique is implemented for the three pass protocol scheme, it has a fatal flaw, where one-time pad algorithm will be breakable very easily [4]. The fatal weakness is proven in the following analysis of the Equation 1.

\[(M \oplus K_A) \oplus (M \oplus K_A \oplus K_B) \oplus (M \oplus K_B) = M\]  

(1)

In Equation 1, if, for instance, Anton wants to send a message to Beby, Anton will encrypt the message M, with the key he has,i.eK_A with the formula \((M \oplus K_A)\), so it generates ciphertext A, afterward, the ciphertext A will be sent to Beby. Following that, Beby will encrypt ciphertext A with the key she has, i.e K_B with the formula \((M \oplus K_A \oplus K_B)\), so it generates ciphertext AB. Then, the
ciphertext AB will be sent back to Anton. Further, Anton will decrypt the ciphertext AB with the $K_A$ key, so it generates ciphertext B. Ciphertext B will be sent back to Beby, then she will decrypt the ciphertext B with the $K_B$ key so it generates plaintext M. This process proves that there is no key exchange occurred in the algorithm [5].

However, if, for example, Ucok succeeds in intercepting Anton and Beby's conversations and obtaining the three of ciphertexts, i.e. ciphertext A, ciphertext AB, and ciphertext B. Ucok will be very easily to reconstruct the plaintext M by performing XOR operations between ciphertext A, ciphertext AB, and ciphertext B. With this fatal weakness, the three pass protocol scheme designed for secure communication is failed. The one-time pad algorithm that is supposed to be an unbreakable algorithm would be a breakable algorithm due to the aforementioned weaknesses. In order to overcome this weakness, something should be done on ciphertext A, ciphertext AB, and ciphertext B so that if all of the ciphertext is operated using XOR, it will not produce M (plaintext).

Existing works have done by some researchers. For instance, using quantum techniques to apply to three-pass protocol [6]. Likewise, The weakness of the XOR technique on the Three-pass protocol can be overcome by adding some other algorithms like Hill-Cipher, DES or Pohlig-Hellman [7]. The other researcher also uses additional algorithms to improve security in the three-pass protocol [8]. This paper reports the use of the RSA and ElGamal algorithms [9] as additional algorithms to modify the ciphertext A, ciphertext AB, and ciphertext B to be super ciphertexts before being sent and passed through the communication paths. It is expected that message M cannot be reconstructed by XOR-ing the three ciphertexts.

2. Methodology
In this research, The RSA and Elgamal algorithms are used to modify ciphertext generated one-time pad algorithm. Figure 1 shows the existing process of the three pass protocol uses the one-time pad algorithm. Three-pass protocol consists of three stages of the delivery processes:

![Diagram of three-pass protocol]

Figure 1.Scheme of three-pass protocol

1. Anton awakens a secret key that is $K_A$ for encryption and decryption process and should not be published and distributed. The plaintext to be sent is first encrypted using the $K_A$ key. after that, the resulting ciphertext is sent to Beby.
2. Beby also generates a secret key that is KB for encryption and decryption process and should not be published and distributed. Ciphertext received from Anton is encrypted using the KB key. The resulting ciphertext is then sent back to Anton.

3. Ciphertext that has been received by Beby is decrypted by KA key. The result of the decryption is still a ciphertext which is then sent back to Beby. Beby who receives the ciphertext then redefines the ciphertext with the KB key. The results of the decryption will produce the original plaintext or message that is meant by Anton.

The working process of a one-time pad algorithm (also referred to as an unbreakable algorithm or also known as the holy grail algorithm [10]) can use classical or modern work processes. The classical work process can be done by creating an encoding table to convert characters into decimal places for encryption and decryption. Modern work processes can be done by XOR technique by XOR each work process can be done by creating an encoding table, while private key B 1st which is the key pair of public key B 1st is kept by Beby. This public key B 1st

The modification scheme also has three delivery stages, where one-time pad algorithm will be used at every stage of delivery. The ElGamal algorithm will be used in the first and third stages of delivery and the RSA algorithm is only used in the second stage. The ElGamal algorithm was designed by an Egyptian scientist named Taher ElGamal in 1984 based on the concept of the Diffie-Helman algorithm [12]. The key-forming process required a prime number, i.e p, two random numbers, i.e g and x, provided that g<p and 1≤x≤p-2. ElGamal's public key consists of pairs of 3 numbers (y, g, p) where:

\[ y = g^x \mod p \]  \hspace{1cm} (4)

The secret key is the number (x, p).

\[ a = g^k \mod p \]  \hspace{1cm} (5)
\[ b = y \mod p \]  \hspace{1cm} (6)

Where, k is a random number with terms 1≤k≤p-2. The calculation results will get the ciphertext block of character m in block (a, b).

\[ m = b \cdot c \mod p \]  \hspace{1cm} (7)

Where, m is plaintext. The value of the variable c can be searched using the equation as follows:

\[ c = a^{p-1} \mod p \]  \hspace{1cm} (8)

Where M is plaintext, p is prime number, g is random number, x is random number, a is ciphertext of 1st block, b is ciphertext of 2nd block and y is part of public key.

The RSA was created by Ron Rivest, Adi Shamir, and Leonard Adleman, according to the inventor's name, in the 1970s. This design relies on the complexity of integrating different integers from discrete algorithm settlement [13][14]. The RSA algorithm generation process generator as follows. Select two large random primes, i.e p and q, then calculate system modulus n = p * q. Afterward, select the encryption key e randomly using equation:

\[ \phi (n) = (p-1)(q-1) \]
\[ \gcd (e, \phi (n)) = 1, \text{where: } 1<e<\phi (n) \]  \hspace{1cm} (9)

Finally calculate decryption key d with the equation 11 with encryption and decryption use equations 12 and 13.

\[ e \cdot d = 1 \mod \phi (n) \text{ and } 0 \leq d \leq n \]  \hspace{1cm} (10)
\[ \text{public key = } \{e, n\} \text{ and } \text{private key = } \{d, n\} \]

\[ C_i = P_i^e \text{ (mod n)} \text{, Where: } 0 \leq p \leq n \]  \hspace{1cm} (11)
\[ P_i = C_i^d \text{ (mod n)} \]  \hspace{1cm} (12)

The modification process is shown in Figure 2. Before Anton sends a message to Beby, Anton must receive a public key first (public key B 1st) generated by Beby for ElGamal algorithm process, while private key B 1st which is the key pair of public key B 1st is kept by Beby. This public key B 1st
can only be used for a one-time process, for the next process to be resurrected different public keys. This is done to improve security and also to follow the flow of the one-time pad algorithm which has the condition that the key can only be used for a one-time process.

After receiving the public key B 1st from Beby, Anton will generate the random key and encrypt the plaintext using $K_A$ and produce ciphertext A. Ciphertext A will then be encrypted with the public key B 1st received from Beby using Elgamal algorithm which will produce Super Ciphertext A. Then Anton immediately generate public key A for RSA algorithm which will be done by Baby later. Super ciphertext A and public key A are sent to Beby. The private key A is stored by Anton.

After Beby receives the super ciphertext A and public key A from Anton, Beby will decrypt the super ciphertext A using the private key B 1st which generates ciphertext A. Afterward, Beby will raise the $K_B$ key at random and encrypt ciphertext A to generate ciphertext AB. The ciphertext AB will be re-encrypted using public key A received from Anton using the RSA algorithm that generate the Super Ciphertext AB. Beby re-generate the second public key (public key B 2nd) to be sent to Anton. Public key B 2nd is later used for encryption process using Elgamal algorithm by Anton, while private key B 2nd remains in Beby. Super Ciphertext AB and public key B 2nd will be sent back to Anton.

After Anton receives the super ciphertext AB and public key B 2nd from Beby, then Anton will decrypt super ciphertext AB using private key A to generate ciphertext AB, then ciphertext AB will be again decrypted using a rail key owned Anton to generate ciphertext B. Ciphertext B will be encrypted using public key B 2nd received from Beby using algorithm Elgamal which will produce Super Ciphertext B. Super ciphertext B is then sent to Beby. Super ciphertext B received by Beby is then decrypted by private key B 2nd to generate cipher B. This Ciphertext is then decrypted using $K_B$ owned by Beby to generate plaintext or original message.

![Figure 2. Proposed modification process](image-url)
3. Result and Discussion
Testing is performed by using ten text messages with different lengths produced various processing time as shown in Table 1. The processing time increases as text length increases.

| Lengths | Designed Algorithm (ms) | Previous Algorithm (ms) |
|---------|-------------------------|-------------------------|
| 10      | 0.0072388648986816      | 0.002739458465576       |
| 20      | 0.0093448162078857      | 0.003949165344238       |
| 50      | 0.015640020370483       | 0.0069119930267334       |
| 100     | 0.042346954345703       | 0.022449970245361       |
| 200     | 0.087727069854736       | 0.036191940307617       |
| 500     | 0.15630197525024        | 0.10586500167847       |
| 1000    | 0.32381987571716        | 0.20496797561646       |
| 2000    | 0.58216500282288        | 0.36745214462288       |
| 5000    | 1.6609079837799         | 1.2502000331879        |
| 10000   | 4.309779882431          | 3.801255941391         |

The difference in processing time from algorithms designed with previous algorithms is not very significant. Therefore, the increase of processing time on algorithms designed with improved security is still very feasible to use. This is due to the process time required algorithm designed with the previous algorithm is not much different. The processing time ratio between the previous algorithm and the designed algorithm is depicted in Figure 3. For low plain text length, the proposed algorithm produces much longer processing time. But for the longer plain text, the processing time is almost similar as the ratio approaches 1.

![Figure 3. Process time ratio](image)

In the designed algorithm, the XOR problem is completely resolved. As shown in Table 2, the XOR results are always different from the decrypted message.

| Block | Super Ciphertext A | Super Ciphertext AB | Super Ciphertext B | Result XOR | Plaintext |
|-------|-------------------|--------------------|-------------------|------------|-----------|
| 0     | 11001101111001001011 | 001001100001001000011 | 101001101000100100100100 | 01010110 | 01010110 |
| 1     | 0001000000110110011 | 11011010000001111111 | 000100010111011011101110 | 110110101001000100011011 | 01011110 |
| 2     | 0000001011101001010 | 1011101001000100010000 | 110100111010001010010101 | 011000001010000100010101 | 01100101 |
| 3     | 1000010101010100100001 | 010100011011001111111010 | 110110001111000110010101 | 001000001111011001001011 | 01101111 |
| 4     | 11111011101100011100011 | 0110110011011010100010 | 110100011001101100101010 | 011000001111011001001011 | 01101111 |
| 5     | 110101000011000000101 | 0001000111100100000000 | 0001011000110100010111 | 111000110010000100010101 | 01101111 |
| 6     | 00001000011110111001000 | 0100010101100000100010 | 111000011110100011001111 | 110110011011100100010110 | 01101111 |
| 7     | 101110100110011111010 | 0110000111101010100000 | 0001001010110100101010 | 111100110101001110111010 | 01101111 |
| 8     | 11011010010011110110 | 101100000100101101101110 | 010100111110110011001110 | 01101111 | 01101111 |
| 9     | 1101101111111000011000 | 0001101110101110101101 | 000010011111111011101010 | 1101010011000000101010 | 01101111 |
| 10    | 0000000110001110111111 | 110111110001011011111111 | 0110001110101110101111111 | 01100001 | 01100001 |
Each bit pattern from plaintext does not have any resemblance to the XOR bit pattern of operations, so the XOR operation against the three super ciphertext will not be able to form plaintext. The results of modulo $2^8$ XOR operations are also different from the plaintext as shown in Table 4. It shows that the proposed algorithm is very safe from XOR attack technique.

| Block | Result XOR with Modulo 2$^8$ | Plaintext |
|-------|-----------------------------|-----------|
| 0     | 01000110                    | 01001001  |
| 1     | 00011011                    | 01101110  |
| 2     | 00101011                    | 01101110  |
| 3     | 10100101                    | 01101110  |
| 4     | 00000000                    | 01110010  |
| 5     | 10000110                    | 01110101  |
| 6     | 00111111                    | 01100001  |
| 7     | 11111010                    | 01110100  |
| 8     | 00000000                    | 01101001  |
| 9     | 00100110                    | 01101011  |
| 10    | 01000011                    | 01100001  |

The easiest way to generate a keyless plaintext is to decrypt all three super ciphertexts to be the initial ciphertext so that with XOR operation a plaintext of the three ciphertexts will be generated. If only one or two super ciphertexts are successfully deciphered into the initial ciphertext, then the ciphertext is still protected by an unbreakable one-time pad algorithm and it is not possible to be decrypted into a keyless plaintext. Equations 14, 15 and 16 describe it. This requires cryptoanalysis to penetrate the RSA and ElGamal algorithms. It is not an easy task as the prime number used is a large prime number.

\[
(M \oplus K_A) \oplus (M \oplus K_A \oplus K_B) \neq M
\]  

(14)

\[
(M \oplus K_A) \oplus (M \oplus K_B) \neq M
\]  

(15)

\[
(M \oplus K_A \oplus K_B) \oplus (M \oplus K_B) \neq M
\]  

(16)

There has not been a sufficiently efficient way to solve the ciphertext of RSA and ElGamal algorithms with the big prime number; it takes a long time and a great resource if the ciphertext can be solved.

4. Conclusion

The paper has proposed three-pass protocol modification using the RSA and ElGamal algorithms. The evaluations show that the proposed method causes longer processing time, however, longer plaintext results insignificant processing time increment. The XOR process evaluation on super ciphertexts produce different decryption text from the plaintext, the same thing applies for 8 bits modulo test. These prove the security level of the proposed method. With this technique the whole ciphertexts are much secure and free of XOR attack. The proposed technique can be an alternative to apply for any communication that utilizes the three-pass protocol.

References

[1] Oktaviana, Boni., Andysah Putra Siahaan. 2016. Three-Pass Protocol Implementation in Caesar Cipher Classic Cryptography. IOSR Journal of Computer Engineering (IOSR-JCE)

[2] Siahaan, Andysah Putra. 2016. Three-Pass Protocol Concept in Hill Cipher Encryption Technique. Seminar Nasional Aplikasi Teknologi Informasi (SNATi)
[3] Abdullah, Alharith., Rifaat Khalaf, and Mustafa Riza. 2015. A Realizable Quantum Three-Pass Protocol Authentication Based on Hill-Cipher Algorithm. Hindawi Publishing Corporation Mathematical Problems in Engineering.

[4] Tasliyah, H. 2015. Algoritma One Time PAD Pada Skema Three Pass Protocol. Universitas Sumatera Utara, Medan.

[5] Lang, Jun. 2012. A no-key-exchange secure image sharing scheme based on Shamir’s three-pass cryptography protocol and the multiple-parameter fractional Fourier transform. OPTICS EXPRESS

[6] Kanamori, Yoshito. Yoo, Seong-Moo. 2009. Quantum Three-Pass Protocol: Key Distribution Using Quantum Superposition States. International Journal of Network Security & Its Applications (IJNSA)

[7] Rahim, Robbi., Ali Ikhwan. 2016. Study of Three Pass Protocol on data Security. International Journal of Science and Research (IJSR).

[8] André, G. D. U., 2007. A Three-Pass Protocol for Cryptography Based on Padding for Wireless Networks. IEEE, no. 1

[9] Okeyinka, A. 2015. Computational Speeds Analysis of RSA and ELGamal Algorithms on Text Data. Proceedings of the World Congress on Engineering and Computer Science

[10] Iqbal, Muhammad., Muhammad Akbar Syahbana Pane, and Andysah Putera Utama Siahaan. 2016. SMS Encryption Using One-Time Pad Cipher. IOSR Journal of Computer Engineering (IOSR-JCE).

[11] Srikantaswamy, S. Phaneendra, H. 2011. Enhanced OneTime Pad Cipher with More Arithmetic and Logical Operations with Flexible Key Generation Algorithm. International Journal of Network Security & Its Applications (IJNSA)

[12] Fujun, Zhan. 2013. The Application of ELGamal Encryption Technology to the Information Security of Digital Library. TELKOMNIKA

[13] Kallam, Ravindra Babu., S. Udaya Kumar, A. Vinaya Babu, and V.Shravan Kumar. 2011. An Enhanced RSA Public key Cryptographic Algorithm. International Journal of Advanced Research in Computer Science.

[14] Preetha, M., M. Nithya. 2013. A Study And Performance Analysis of RSA Algorithm. International Journal of Computer Science and Mobile Computing.