Higher Security for Login System Using RSA and One-time Pad Schemes

Kritsanapong Somsuk
Department of Computer and Communication Engineering, Faculty of Technology,
Udon Thani Rajabhat University, UDRU,
Udonthani, Thailand.
kritsanapong@udru.ac.th

***Abstract*—The aim of this paper is to propose a new methodology to increase the security for Login System using RSA and One-time pad (OTP). In the past, the application of RSA with Login System focused on keeping the private key in the server. However, this approach has limitation, in which the overall system can be broken whenever the private key is recovered. Therefore, this paper proposes a different method where the RSA's private key used for exchanging OTP's key is kept at the client's side, whilst the public key and modulus are kept in the database. Furthermore, as a preventive measure from the attackers, the OTP's key is generated in the server. If it is created at the client's side, the attackers can trap both the encrypted key and encrypted password from client. Accordingly, they can send both of them to the server without encrypting again and without knowing the password. In addition, the RSA's process in the server is an encryption process only. That means it takes only a small computation cost for computing modular exponentiation because the public key is always small when compared with the private key. Assuming that the client's private key is recovered, only the client system whose private key can be found will be broken, which implies that it does not affect to other clients. The experimental results show that although users must remember their private keys and consumes more time, the new proposed system is very strong and secure. Therefore, users who use this system can access the web application without worrying the attackers.**

***Index Terms*—Login System; RSA Scheme; One-time Pad (OPT); Security; Li Ming –Xin's Method; Time.**

I. Introduction

The accessibility of web applications, such as e-mail, e-book, e-Service, e-Commerce and Cloud Computing etc. via Internet is very popular nowadays. However, there are some limited rights to access the web applications. Login System is a process that allows users to access the web applications by identifying and authenticating themselves. In general, username and password are the important components for Login System since users without these two components cannot access the applications. Moreover, attackers who can capture other people's username and password sent through insecure channel are able to access the applications using the captured password and username.

In 2010, Li Ming – Xin and Kang Fend [1] proposed the application of RSA [2], which is a asymmetric cryptosystem [3] with Login System. Although the components of RSA are generated by the server, the public key and modulus are published to clients, while the private key is kept in the database. In this case, both the username and password will be encrypted with RSA scheme using JavaScript Language as the computation using this language can be done on the client's side. Therefore, the username and password will be encrypted before they are sent through the insecure channel. However, assuming that attackers can capture the ciphertexts of the username and password, they can send both of them to the server without encrypting again and without knowing the real username and password. When the Login System is broken, the server can accept the attackers' requests, considering that the username and password are in database of the server after the completion of the decryption process.

Other approaches to increase the security for Login System were also proposed. For example, in 2016, Ki Hwan Kim et al. [11] proposed the improved security of Login System for smart phone using tap and gesture. Furthermore, in the same year, Abdul Waheed et al. [12] reviewed the attacking methods for stealing Login System's username and password. In these studies, some authentication methods were described and analyzed.

This paper proposes the application of RSA and One-time pad (OTP) [4], which is one of symmetric key cryptosystems [5] and a perfect secrecy cryptosystem with Login System. In fact, the proposed method is different from Li Ming Xin's method because all of the RSA's components used for exchanging OTP's key are generated at the client's side. Therefore, each user will have the RSA's components by him/herself. In addition, only the public key and modulus sent through insecure channel to the server will be kept in the database. On the other hand, the OTP's key, which is used for encrypting and decrypting password will be created in the server and it must be encrypted using RSA scheme before sending it to user. In this case, the proposed system is very secure based on two reasons. First, even though attackers can trap the encrypted key, they cannot recover the key because they do not know the user's private key. Second, in cases where the attackers can trap the encrypted password sent by the user, the password cannot be recovered because the attackers do not have the OTP's key.

The remainders of this paper are organized as follows: In section II, the related works consisting of RSA, OTP and the application of RSA with Login System, focusing on the decryption process of asymmetric cryptosystem from the server are presented. Section III describes the proposed method of applying the combination of RSA and OTP with Login System. The results of the experiment are discussed in Section IV. Finally, the conclusion is presented in the last section, Section V.

II. Related Works

A. RSA Cryptosystem

RSA is the asymmetric key cryptosystem using a pair of keys for encryption and decryption processes. Assuming that the public key (e), private key (d) and modulus (n) are generated, the encryption process and decryption process are as follows:

$$\text{Encryption: } c = m^e \bmod n$$

where: m is the original message (plaintext)
c is the encrypted message which is unreadable (ciphertext)

$$\text{Decryption: } m = c^d \bmod n$$

In general, after finishing the decryption process, m is always recovered.

B. One-time pad

One-time pad (OTP) is one of symmetric key cryptosystems using the same key for encryption process and decryption process. In fact, after finishing one encryption process and one decryption process, the key, which is always generated randomly, must be left immediately, indicating that the OTP is the perfect secrecy cryptosystem. Moreover, OTP uses XOR operation, which is an easy and a fast technique for both of the encryption and decryption process.

$$\text{Encryption: } c = m \oplus k$$

where k is the random key

$$\text{Decryption: } m = c \oplus k$$

However, the bits- length of key for OTP should be equal or larger than m.

C. Applying for RSA with Login System

The application of the RSA with Login System to secure username and password was proposed by Li Ming – Xin and Kang Fend in 2010. In general, e, d and n are generated in the server before publishing e and n to clients. Therefore, users can use the server's e and n for encrypting their username (u) and password (p) before sending it to server. In fact, when the encrypted username (c_{i1}) and encrypted password (c_{i2}) arrive to the server, they can be recovered as u and p using the RSA's decryption process because d is kept in the database.

However, if c_{i1} and c_{i2} are trapped over the insecure channel, the attackers can create a fake login page to send c_{i1} and c_{i2} without encrypting them again to the server. Moreover, c_{i1} and c_{i2} can be recovered in the server as u and p, respectively. Therefore, the server can accept the attackers' requests to access the application.

III. The Proposed Method

In this paper, a new strong Login System is proposed by applying the combination of RSA and OTP. However, the objective of the proposed method is different from the method proposed by Li Ming – Xin because the RSA's components are generated at the client side in order to save the computation cost in the server and to avoid invasion from attackers. In general, the bits length of d is always larger than e in order to make the RSA stronger. This means that if the decryption process is implemented in the server, high computation costs are needed. In addition, the decryption process in the server must also be serviced by all of the users who request to access the application at the same time. Specifically, the OTP's key will be generated randomly in the server for encrypting the password at the client's side. In general, it must be encrypted with the RSA scheme before sending it to the client.

In fact, there are two proposed systems, which are the registration system and the login system. Assuming that only the username and the password are needed in the registration system, the algorithms of both systems will be as follows:

```
Algorithm 1: Registration System
Input: Username (u)
Client:
    1. Generate RSA's Components consists of public key (e), private key (d) and modulus (n).
    2. Send u, e, n to Server
  Note1: Client must remember d by him/herself because it is not transferred to Server

Server: Receive u, e, n from Client
    3. If u is already found in database
    4.   Reject this registration
    5. Else
    6.   Generate a client's password (p) randomly.
    7.   Generate a OTP's key (k) randomly.
    8.   Encrypt p using OTP scheme: $c_1 = p \oplus k$
    9.   Encrypt k using RSA scheme: $c_2 = k^e \bmod n$
    10.  Destroy value of k
    11.  Insert u, p, e, n to database
    12.  Send $c_1$ and $c_2$ back to Client
    13. EndIf

Client: Receive $c_1$ and $c_2$ from Server
    14. Decrypt $c_2$ using RSA scheme: $k = c_2^d \bmod n$
    15. Decrypt $c_1$ using OTP scheme: $p = c_1 \oplus k$
    16. Client knows Username is u and Password is p

Output: password (p)

  Note2: Registration is completed in Step 13
  Note3: Step 14 – 16 are the processes to send the secret password to client
```

From Algorithm 1, although attackers can trap c_1 and c_2, they cannot recover p because they do not have d to recover k. Therefore, the registration is finished securely.

```
Algorithm 2: Login System
Input: Username (u), Password (p) and Private key (d)
Client:
    1. Send u to Server
Server: Receive u from Client
    2. If u is not in database
    3.   Reject this login request
    4. Else
    5.   Generate a OTP's key (k) randomly.
    6.   Encrypt k using RSA scheme: $c_2 = k^e \bmod n$
    7.   Insert k in database
    8.  Send $c_2$ to Client
    9. EndIf

Client: Receive $c_2$ from Server
    10. Decrypt $c_2$ using RSA scheme: $k = c_2^d \bmod n$
    11. Encrypt p using OTP scheme: $c_1 = p \oplus k$
    12. Send u and $c_1$ to Server

Server: Receive u and $c_1$ from Client
    13. Decrypt $c_1$ using OTP scheme: $p = c_1 \oplus k$
    14. Destroy k from database
    15. If both of u and p are found in database
    16. Login Complete
    17. Else
    18. Reject the Login using username: u and password:p
    19. EndIf
```

From Algorithm 2, although attackers can trap c_2 sent from server, they cannot recover k because they do not have d, which is kept secretly by the user. Moreover, if they know c_1 is sent from the user, they cannot know p because they cannot recover k. That means the proposed Login System is very strong and secure.

Nevertheless, although the new Login System gets the higher security, there are two disadvantages of the proposed method.

1. There are four requests – respond times between Client and Server that are more than the traditional Login System, which needs two times only (Client requests and Server responses). Therefore, the computation time of the proposed method is certainly higher than the traditional one.
2. Usually, users must only remember their username and password for the traditional Login System, but private key must be also be known for the proposed method.

In addition, Javascript Language is chosen for the computation process in the client side because the process can be completed in the user's device. Furthermore, Biginteger library [6] is chosen for the implementation because it can be used as the large size of integer. The library's initial variable can be assigned as follows:

```
var big_int = bigInt(num);
```

where: big_int is a Biginteger library's variable
num is an integer or the string

Furthermore, the important methods proposed in this paper are as follows:

1. multiply(num): The multiplication operation
2. mod(num): The modular operation
3. xor(num): The bitwise XOR operation

Method 1 and 2 are used to implement Square and Multiply Algorithm [7], [8], [9] which is the algorithm to speed up modular exponentiation. On the other hand, Method 3 is used for the OTP's encryption and decryption operation.

Moreover, Math_BigInteger class of PHP Language [10] is chosen to implement RSA algorithm, the encryption process in the server side. The object of this class can be assigned as follows:

```
$big_int = new Math_BigInteger($num);
```

where: $big_int is the Math_BigInteger class's object
$num is an integer or the string

However, modPow($num1, $num2), $num1 and $num2 are Math_BigInteger class's parameters, which is the method for computing modular exponentiation, is chosen to the encryption process.

IV. Results and Discussion

In this session, Intel® Core i3 Processor with 4 GB memory are the resources for the experiment. HTML, Javascript, and PHP are three languages chosen in this experiment. In fact, there are two main processes, which consist of the registration system and login system. Figure 1 until Figure 4 show the registration system. Figure 1 is the first page of this system. In this page, the user can use only "check" button for generating RSA's components.

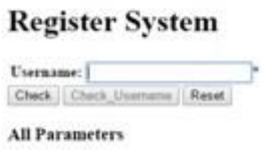
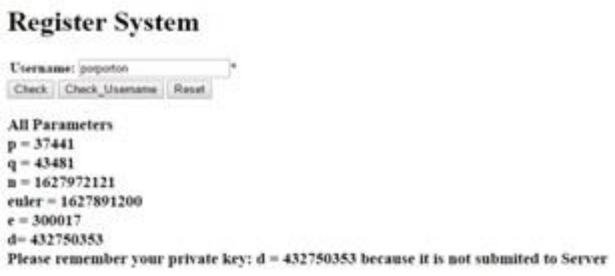


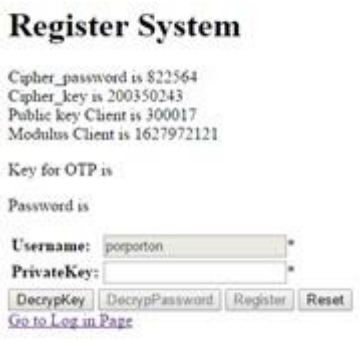
Figure 1: Registration Page



```
All Parameters
p = 37441
q = 43481
n = 1627972121
euler = 1627891200
e = 300017
d= 432750353
Please remember your private key: d = 432750353 because it is not submited to Server
```

Figure 2: Key generation for RSA in Client Side

After, finishing the key generation process, "check_ Username" button will be enabled. If the user presses this button, u, e and n will be sent to the server.



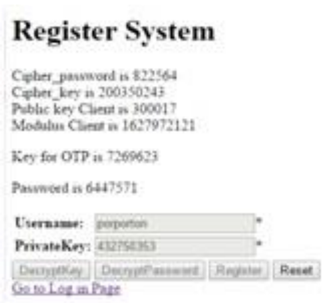
```
Cipher_password is 822564
Cipher_key is 200350243
Public key Client is 300017
Modulus Client is 1627972121

Key for OTP is

Password is
```

Figure 3: Encrypted password and Encrypted key from Server

If u is not found in the database of the server, p and k will be generated randomly and the registration is successful. However, p is not still sent to user. Therefore, it has to be encrypted by using OTP scheme, c_1, and k must be also encrypted using RSA scheme, c_2. Finally, the server sends both c_1 and c_2 instead of p to the client. Figure 3 shows the response page receiving c_1 and c_2 from the server.



Register System

Cipher_password is 822564
Cipher_key is 200350243
Public key Client is 300017
Modulus Client is 1627972121

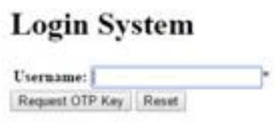
Key for OTP is 7269623

Password is 6447571

Username: porporton *
PrivateKey: 432750353 *

Figure 4: Recovering key and password by Client

In Figure 4, user can get his/her p by using the private key to disclose k, "DecryptKey" button, and using k to recover p, "DecryptPassword" button.

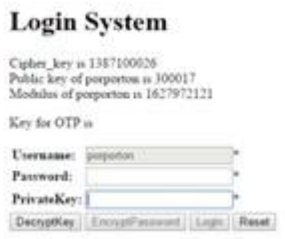


Login System

Username: *

Figure 5: Login Page

Figure 5 is the first page for login process requesting only u from user. If u is assigned in the textbox and "Request OTP Key" button is pressed respectively, u will be sent to the server to check in the database. The new value of k is generated randomly and is also encrypted using RSA scheme with e and n, c_2 = k^e mod n, whenever u is found in the database. Next, c_2 is sent back to the user.



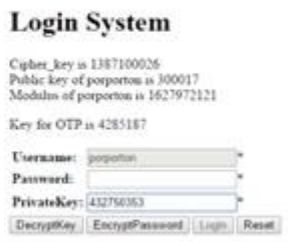
Login System

Cipher_key is 1387100026
Public key of porporton is 300017
Modulus of porporton is 1627972121

Key for OTP is

Username: porporton *
Password: *
PrivateKey: *

Figure 6: Encrypted key (OTP) from Server



Login System

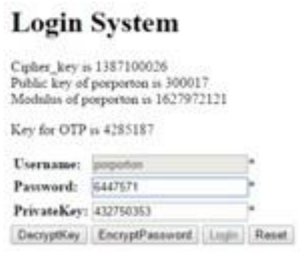
Cipher_key is 1387100026
Public key of porporton is 300017
Modulus of porporton is 1627972121

Key for OTP is 4285187

Username: porporton *
Password: *
PrivateKey: 432750353 *

Figure 7: Recovering key by Client

The user can recover k by pressing "DecryptKey" button after filling d in the Textbox in Row 3, as shown in Figure 7.



Login System

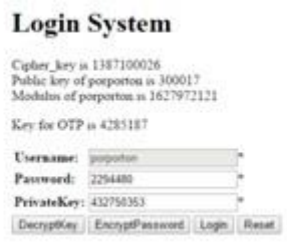
Cipher_key is 1387100026
Public key of porporton is 300017
Modulus of porporton is 1627972121

Key for OTP is 4285187

Username: porporton *
Password: 6447571 *
PrivateKey: 432750353 *

Figure 8: Client's password

After receiving k, user must fill his/her p in the Textbox in Row 2, Figure 8, to be encrypted with the OTP scheme, $c_1 = p \oplus k$, when "EncryptPassword" button is pressed, as shown in Figure 9. Finally, u and c_1 are sent back to the server.



Login System

Cipher_key is 1387100026
Public key of porporton is 300017
Modulus of porporton is 1627972121

Key for OTP is 4285187

Username: porporton *
Password: 2294480 *
PrivateKey: 432750353 *

Figure 9: Encrypting client's password

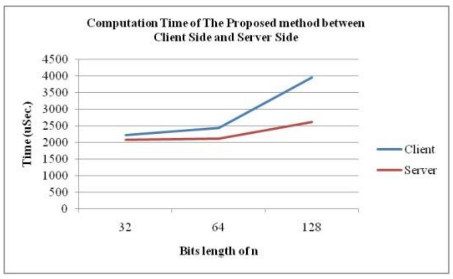


Figure 10: Computation time in Client and Server side

As shown Figure 10, the blue curve is represented as the computation time in the client side. In this side, there are two processes, which consist of decryption with RSA scheme and encryption with OTP scheme. However, the red curve is represented as the computation in the server. Three processes in the server are the encryption with RSA scheme, decryption with OTP scheme and key generation for OTP. The experimental results show that the computation time in client side is higher than the others in the server although the processes in the server are larger than client's side. The reason is that there is only decryption process with RSA in the client's side. Usually, the decryption process usually consumes time more than encryption process for the RSA because d is always larger than e.

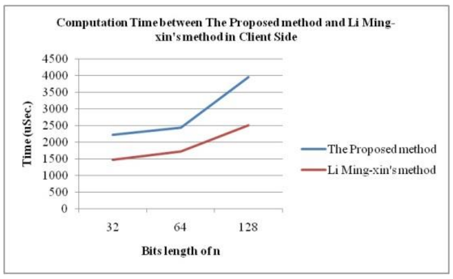


Figure 11: Computation time between The Proposed method and Li Ming-Xin's method in Client Side

The information in Figure 11 shows that the proposed method consumes time more than Li Ming-Xin's method. This is because the decryption process for RSA and encryption process for OTP of the proposed method is at the client's side. Similar to Li Ming-Xin's method, the process requires only encryption process of RSA.

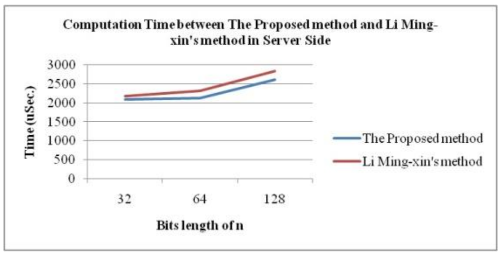


Figure 12: Computation time between The Proposed method and Li Ming-Xin's method in Server Side

In Figure 12, the encryption process of RSA for the proposed method is in the server, while the decryption process of Li Ming - Xin's method is required in this side. Although the OTP scheme is also implemented with the RSA for the proposed method, it consumes only a little time when compared with the RSA process. Therefore, the proposed method is faster in the server than Li Ming - Xin's method. Furthermore, if many users need to request the Login System at the same time, then the space time between the proposed method and Li Ming-Xin's method is increased.

In general, for overall system per one log in, the proposed method consumes more time than Li Ming - Xin's method because two cryptosystems are needed. However, the proposed method is stronger than Li Ming - Xin's method based on the reasons showed in Table 1.

Table 1 shows the comparison about time and risk during three algorithms

1. Time: Traditional Login System is the fastest algorithm because it does not include time for encryption and decryption process, while the proposed method is the slowest algorithms because two cryptosystems are implemented in the algorithm.
2. Risk: The proposed method has the lowest risk because although, attackers can trap any information from the user (Client) or the Server, they cannot find the user's password. On the other hand, for Li Ming –Xin's method, if attackers can trap c_{i1} and c_{i2} from user, they can send both of them to the server directly without encrypting and without knowing the real username and password. After decrypting c_{i1}, c_{i2}, the server finds that the user's username and password are in the database. Therefore, the server accepts the attackers' requests.

Table 1
Time and Risk during three algorithms

	Traditional Login System	Li Ming –Xin's method	The proposed method
Time	<u>Low</u>	High	High
Risk	Medium	Medium	<u>Low</u>

V. Conclusion

This paper proposes a new strong Login System with increased security by applying two cryptosystems along with the RSA, which is used to exchange the OTP's key, and OTP to transform the password before sending through the insecure channel. Specifically, the RSA's key generation and decryption process are done at the client's side in order to reduce time for computing modular exponentiation in the server. Moreover, attackers can break only some parts of system, because one private key is assigned for only one user. In addition, if they want to break the overall system, they must find all private keys of all users. However, for Li Ming –Xin's method the private key is kept in the server and the overall system will be broken if it is recovered. In general, for the proposed system, users must keep their private keys by themselves. On the other hand, the public key and modulus are submitted so that they can be kept in the database. Nevertheless, although user must remember the private key and the computation speed will be slower, this Login system is very strong and very difficult to attack. Therefore, users using this system to access web application do not worry about the danger from attackers. In the future work, researchers will focus on using this system with smart phone. Elliptic Curve cryptosystem (ECC) [13], which is another public key cryptosystem will be chosen instead of RSA. This is because the bits length of ECC is always less than RSA with the same security level. Therefore, ECC is more suitable tool than RSA for smart phone that is a low computation device.

References

[1] L.M. Xin, K. Feng, "An improved sign-in scheme based on RSA cryptosystem", International Conference on Computer Application and System Modeling, Taiyuan, 2010, pp.35 – 37.
[2] R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", Communications of ACM, vol. 21, 1978, pp. 120 – 126.
[3] W.Diffie, M.E. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, vol. 22, 1976, pp. 644–654.
[4] M. Sarvabhatla, C.M. Reddy , C.S. Vorugunti, "A Secure and Light Weight Authentication Service in Hadoop using One Time Pad", International Symposium on Big Data and Cloud Computin, Chennai, 2015, pp.81-86.
[5] J.A. Buchmann. "Introduction to Cryptography", United States of America: Springer-Verlag, 2000.
[6] Big-integer: https://www.npmjs.com/package/big-integer.
[7] O. Nibouche, M. Nibouche, A. Bouridane, "Highspeed FPGA implementation of RSA encryption algorithm",Electronics, circuits and Systems, vol. 1, 2003, pp. 204-207.
[8] A.Mazzero, L.Romano, G.P. Saggese, N. Mazzocca, "FPGA-based Implementation of a serial RSA processor", Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 582 – 587.
[9] J. Elbirt. "Understanding and Applying Cryptography and Data Security", United Kingdom: Auerbach Publications, 2009.
[10] Class: Math_BigInteger: https://pear.php.net/package/Math_BigInteger /docs/latest/ Math_BigInteger/Math_BigInteger.html.
[11] K. Kim, B. Ndibanje, S. Park, H. Lee, "New Security Login System Using Tap and Gesture on Smartphone Touchscreen", International Conference on Advanced Communication Technology, Pyeongchang, 2016, pp. 628 – 633.
[12] A. Waheed, M.A. Shah, A. Khan, "Secure login Protocols: An Analysis on Modern Attacks and Solutions", International Conference on Automation and Computing, Colchester, 2016, pp. 535 – 541.
[13] W. Trappe, L. Washington. "Introduction to Cryptography with Coding Theory", United States of America: Pearson, 2005.