Analysis of RSA variants in securing message

Nadia Widari Nasution[1], Syahril Efendi[1], Sawaluddin[2]

[1]Faculty of Computer Science and Technology Information, University of Sumatera Utara, Medan, Indonesia
[1]Faculty of Computer Science and Technology Information, University of Sumatera Utara, Medan, Indonesia
[2]Faculty of Mathematics and Natural Science, University of Sumatera Utara

nadianwidari@gmail.com, syahnyata1@gmail.com, sawal@usu.ac.id

Abstract. Message security is an important thing to prevent the interference from third parties. RSA cryptographic algorithm is believed to be a powerful algorithm in securing message. However, RSA computing process takes a long time so it takes several variants of RSA, namely R prime RSA and Multi-factor RSA which can reduce time and computing cost on the encryption and decryption side. In this paper, the author use three prime numbers to improve better security than only use two prime numbers and minimize the value of private key "*d*" is by using Chinese Reminder Theorm (CRT).

1. Introduction

There are many settlement algorithms to solve a problem especially problem of message security. Cryptography is a science that learn how to keep data or messages safe when were sent, from sender to recipient without getting interruption from third parties.
By analyzing some algorithms, can be identified an algorithm that was more efficient. Efficiency of the algorithm can be measured from execution time of algorithm and space memory needed.

Rprime RSA is a combination of Rebalanced RSA and Multi-prime RSA [1], but Multi-factor RSA is a combination of Multi-prime RSA and Multi-power RSA.

Analysis of algorithm was needed for the comparison of the algorithm without depending the specification of machine. It help us to solve the problem based on the condition and data. It is not an absolute tool to choose the best algorithm, but help to understand the behavior of the algorithm when applied. Therefore, in this research will be compared the performance of two variants algorithms (Rprime RSA and Multi-factor RSA) by using analysis of time complexity and computer simulation.

2. Material and Methods

2.1 Cryptography

Cryptography is a science or art in securing message, and it was done by a cryptographer. But cryptanalysis is a science and art for breaking ciphertext and the person who do it that's called cryptanalyst.

According to Bruce Scheiner in his book "Applied Cryptography", cryptography is a science and art to keep data or messages safe.
The principles are underlying cryptography as follows:

- Confidelity
- Data Integrity

- Authentication
- Non-Repudiation

Cryptographic system or cryptosystem is a facility to convert plaintext to ciphertext and vice versa. In this system, set of parameters that determine a specific cipher transformation namely a set of keys. The encryption and decryption processes are governed by one or more cryptographic keys. Generally, the keys that are used for them are necessarily identical, depend on the system used.

The field of cryptography includes algorithms, methods and protocols for the encryption for a message and its safe traversal over a network (i.e without delay safe from the hands of an intruder) [2].

2.2 Rprime RSA

Rprime improves the computational cost at the decryption side by combining rebalanced RSA and Mprime RSA. The security of Rprime RSA, as well as that of Rebalanced RSA, depends on the security offered by the private exponent d and on the size of the used primes [3].

Rprime RSA consists of three processes, i.e. key generation, encryption and decryption.

a. Key Generation
   1. Generate k distinct prime numbers $p_1, p_2, ..., p_k$ with
      $\gcd(p_1 - 1, p_2 - 1, ..., p_k - 1) = 2$
   2. Compute $n = p_1 \cdot p_2 \cdot .... p_k$,
      $\varphi(n) = (p_1 - 1)(p_2 - 1)(p_k - 1)$
   3. Generate k random numbers $d_{p_1}, d_{p_2}, ..., d_{p_k}$
      $\gcd(d_{p_1}, p_1 - 1) = 1,$
      $\gcd(d_{p_2}, p_2 - 1) = 1, ...,$ and $d_{p_1} = d_{p_2} = ... = d_{p_k} \bmod 2$
      $\gcd(d_{p_k}, p_k - 1) = 1$
   4. Compute d with Chinese Reminder Theorem (CRT)
   5. Compute $e = d^{-1} \bmod \varphi(n)$
      The public key is (n, e) and the private key is $(p_1, p_2, ..., p_k, d_{p_1}, d_{p_2}, ..., d_{p_k})$

b. Encryption
   1. Represent the plaintext message (M)
   2. Encrypt message $C = M^e \bmod n$

c. Decryption
   1. Accept ciphertext from sender
   2. Decrypt message $M = C^d \bmod n$

2.3 Multi-factor RSA

Multi-factor RSA is based on modifying the structure of the RSA Modulus, i.e. $n = pqr$ or $n = p^2 r$. In the key generation process, given an additional parameter namely b. The security of Multi-factor RSA depends on the difficulty of factoring integers of the form n [4].

Multi-factor RSA consists of three processes, i.e. key generation, encryption and decryption.

a. Key Generation
   1. Generate *b* distinct prime numbers $p_1, ..., p_b$
   2. Compute $n = p_1 \cdot ... p_b$
   3. Compute $\varphi(n) = (p_1 - 1)(p_2 - 1)(p_b - 1)$
   4. Pick the same *e* used in standard RSA public key, namely $e = 65537$. Then compute $d = e^{-1} \bmod \varphi(n)$

      The public key is (n, e) and the private key is d.

b. Encryption
   1. Represent the plaintext message (M)
   2. Encrypt message $C = M^e \bmod n$

c. Decryption
   1. Accept ciphertext from sender
   2. Decrypt message using the Chinese Reminder Theorem (CRT)

   Let $d_i = d \bmod p_i - 1$. Compute $M_i = C^{d_i} \bmod p_i$ for each *i*, $1 \le i \le b$ then combines the M_i's using CRT to obtain $M = C^d \bmod n$.

3. Result and Discussion

3.1 Analysis of Rprime RSA Algorithm

Here are the processes:

a. Key Generation
   1. Generate k distinct prime numbers

      $k = 3$

      e.g $p_1 = 23, p_2 = 19, p_3 = 29$
   2. Compute

      $n = p_1 \cdot p_2 \cdot p_3 = 23 \cdot 19 \cdot 29$

      $= 12.673$

      Then,

      $\varphi(n) = (p_1 - 1)(p_2 - 1)(p_3 - 1)$

      $= 22 \cdot 18 \cdot 28$

      $= 11.088$
   3. Generate k random numbers $d_{p_1}, d_{p_2}, ..., d_{p_k}$

      e.g $d_{p_1} = 17, d_{p_2} = 5, d_{p_3} = 9$
   4. Compute *d* with Chinese Reminder Theorem (CRT)

      $d \equiv d_{p_1} (\bmod p_1 - 1)$

      $d \equiv d_{p_2} (\bmod p_2 - 1)$

      $d \equiv d_{p_3} (\bmod p_3 - 1)$

Then,

$d \bmod 22 = 17 \Leftrightarrow d \equiv 17 \pmod{22}$

$d \bmod 18 = 5 \Leftrightarrow d \equiv 5 \pmod{18}$

$d \bmod 28 = 9 \Leftrightarrow d \equiv 9 \pmod{28}$

So, d =149

5. Compute $e = d^{-1} \bmod \varphi(n)$

$e = d^{-1} \pmod{\varphi(n)}$

$e\ d \bmod \varphi(n) = 1$

$e \cdot 149 \bmod 11.088 = 1$

So, $e = 893$

Public Key

$(n, e) = (12.673, 893)$

Private Key

$(p_1, p_2, ..., p_k, d_{p_1}, d_{p_2}, ..., d_{p_k}) = (23,19,29,17,5,9)$

b. Encryption

1. Represent the plaintext message (M)

$M =$ " A "

Then, convert to ASCII Table

A = 65

2. Encrypt message

$C = M^e \bmod n$

$= 65^{893} \bmod 12.673$

$= 11.507$

c. Decryption

1. Accept ciphertext from sender

$C = 11.507$

2. Decrypt message

$M = C^d \bmod n$

$= 11.507^{149} \bmod 12.673$

$= 65$

is a character "A"

3.2 Analysis of Multi-factor RSA Algorithm

Here are the processes:

a. Key Generation

1. Generate b distinct prime numbers $p_1, ..., p_b$

e.g $p_1 = 23, p_2 = 19, p_3 = 29$

2. Compute $n = p_1 \cdot ... p_b$

$n = p_1 \cdot p_2 \cdot p_3$

$= 23 \cdot 19 \cdot 29$

$= 12.673$

3. Compute

$\varphi(n) = (p_1 - 1)(p_2 - 1)(p_3 - 1)$

$= 22 \cdot 18 \cdot 28$

$= 11.088$

4. Pick the same *e* used in standard RSA public key, namely $e = 65537$. But it can also use another number $(e = 893)$ Then, compute $d = e^{-1} \bmod \varphi(n)$

$e\ d \bmod \Phi(n) = 1$

$893 \cdot d \bmod 11.088 = 1$

So, $d = 149$

Public Key

$(n, e) = (12.673, 893)$

Private Key

$d = 149$

b. Encryption

1. Represent the plaintext message (M)

$M =$ " L "

Then, convert to ASCII Table

L = 76

2. Encrypt message

$C = M^e \bmod n$

$= 76^{893} \bmod 12.673$

$= 3.838$

c. Decryption

1. Accept ciphertext from sender

$C = 3.838$

2. Decrypt message

$M = C^d \bmod n$

$= 3.838^{149} \bmod 12.673$

$= 76$

is a character "L"

For example, a message contains "CRYPTOGRAPHY" in ASCII TABLE:

C	R	Y	P	T	O	G	R	A	P	H	Y
67	82	89	80	84	79	71	82	65	80	72	89

Then, split the message into block which contains 3 digits.

$m_1 = 678$ $m_3 = 808$ $m_5 = 718$ $m_7 = 807$

$m_2 = 289$ $m_4 = 479$ $m_6 = 265$ $m_8 = 289$

Encrypt messsage

$c_i = m_i^e \bmod n$

$c_1 = m_1^e \bmod n$
$= 678^{893} \bmod 12.673$
$= 1788$

$c_2 = m_2^e \bmod n$
$= 289^{893} \bmod 12.673$
$= 7046$

$c_3 = m_3^e \bmod n$
$= 808^{893} \bmod 12.673$
$= 4954$

$c_4 = m_4^e \bmod n$
$= 479^{893} \bmod 12.673$
$= 3227$

$c_5 = m_5^e \bmod n$
$= 718^{893} \bmod 12.673$
$= 9750$

$c_6 = m_6^e \bmod n$
$= 265^{893} \bmod 12.673$
$= 10126$

$c_7 = m_7^e \bmod n$
$= 807^{893} \bmod 12.673$
$= 9296$

$c_8 = m_8^e \bmod n$
$= 289^{893} \bmod 12.673$
$= 7046$

So, ciphertext is C = 1788 7046 4954 3227 9750 10126 9296 7046

Then, Decrypt message $m_i = c_i^d \bmod n$

$m_1 = c_1^d \bmod n$
$= 1788^{149} \bmod 12.673$
$= 678$

$m_2 = c_2^d \bmod n$
$= 7046^{149} \bmod 12.673$
$= 289$

$m_3 = c_3^d \bmod 12.673$
$= 4954^{149} \bmod 12.673$
$= 808$

$m_4 = c_4^d \bmod 12.673$
$= 3227^{149} \bmod 12.673$
$= 479$

$m_5 = c_5^d \bmod n$
$= 9750^{149} \bmod 12.673$
$= 718$

$m_6 = c_6^d \bmod n$
$= 10126^{149} \bmod 12.673$
$= 265$

$m_7 = c_7^d \bmod n$
$= 9296^{149} \bmod 12.673$
$= 807$

$m_8 = c_8^d \bmod n$
$= 7046^{149} \bmod 12.673$
$= 289$

So, the result are the same as the initial plaintext.

4. Conclusion

The use of three prime numbers can improve better security than only use two prime numbers. And one way to minimize the value of private key “d” is by using Chinese Reminder Theorm (CRT). So, the message become fast and more secure.

5. References

[1] Verma, S. & Garg, D. 2015. Improvement in Rebalanced CRT RSA. The International Arab Journal of Information Technology, vol.12, no.6, pp.524-531

[2] Kannan, V et al. 2014. Review and Recent Trends in Cryptography. International Journal of Scientific Engineering and Technology Research. ISSN 2319-8885 vol.03, issue.22,pp.4450-4455.

[3] Paixao, C. A. 2005. An Efficient Variant of The RSA Cryptosystem. pp.1-9

[4] Boneh, D. & Shacham, H. 2002. Fast Variants of RSA. CryptoBytes, vol.5, no.1, pp.1-9.