The cryptanalysis of the Rabin public key algorithm using the Fermat factorization method

M A Budiman, D Rachmawati, and R Utami

Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Alumni No. 9-A, Kampus USU, Medan 20155, Indonesia

Email: mandrib@usu.ac.id, dian.rachmawati@usu.ac.id, rayanti.utami@student.usu.ac.id

Abstract. As a public key cryptography algorithm, the Rabin algorithm has two keys, \( i.e., \) public key (\( n \)) and private key (\( p, q \)). The security of Rabin algorithm relies on the difficulty of factoring very large numbers, so the greater the private keys are used, the better the security becomes. In order to test how hard it is to cryptanalyze the Rabin public key \( n \), we use the Fermat factorization method to obtain the values of \( p \) and \( q \). After obtaining the factors, both of these factors are tested whether or not they are in accordance with the Rabin private key requirements. The first is to test whether or not the factors are prime numbers using Fermat Little Theorem. The second is to test whether or not the factors are congruent to 3 in modulo 4. If the results of the two tests turn out to be positive, then the factors are indeed \( p \) and \( q \), the private keys of the Rabin algorithm. The result of our experiment indicates that the value of public key \( n \) does not have a directly proportional correlation to the factoring time. A factor which may affect the factoring time is the difference between the private keys (\( p - q \)): the larger the difference, the longer the factoring time.

1 Introduction

The development of computer networks at this time allows us to send messages through computer networks. To keep the security and confidentiality of messages to be sent, a secure encoding is required. Cryptography is a field of science about encoding messages using encryption and decryption techniques. The essence of cryptography is protecting the secret of the message (plaintext) from unwanted parties who want to retrieve or read the secret message (tappers, attackers, hackers, opponents, or enemies). Cryptanalysis is the science and art to retrieve the original message (plaintext) without knowing the key. Message security can be done with a variety of cryptographic techniques. One of the algorithms used in cryptography is the Rabin algorithm.

The Rabin algorithm is an asymmetric cryptography algorithm similar to the RSA algorithm (Rivest-Shamir-Adleman algorithm), both of which have difficulty levels in factorization problems. In the encryption process, the Rabin algorithm uses only simple calculations, while the decryption process can use the Chinese Remainder Theorem to get the correct plaintext.
Rabin algorithm relies on the difficulty of factoring on large numbers. Thus, if the key being used are bigger, the more difficult the Rabin algorithm can be compromised by the unwanted parties. To understand how difficult it is to break the Rabin algorithm having certain key lengths, we need a factorization algorithm. The Fermat method is one of the famous factorization methods. This method is based on quadratic disputes, discovered by Pierre de Fermat in 1643, and still used 400 years later.

Therefore, to see how safely the Rabin algorithm survives the attack against its calculation of factorization, in this study the authors wanted to solve the public key cryptanalysis of the Rabin algorithm by using the Fermat factorization method.

2 Method

2.1 Rabin Algorithm

Rabin's cryptographic algorithm was first introduced in January 1979 by Michael O. Rabin in his journal entitled "Digitalized Signatures and Public-Key Functions as Intractable as Factorization". This algorithm applies the concept to get plaintext from ciphertext based on factorization problem. [1]

2.1.1. Key Generation of Rabin Algorithm

Rabin algorithm key generation steps:
1. Generate two very large prime numbers, p and q, satisfying the conditions below:
   \[ p \neq q \rightarrow p \equiv q \equiv 3 \pmod{4} \]
   
   For example:
   \[ p = 139 \quad q = 191 \]
   \[ 139 \quad \text{prime number} \]
   \[ 191 \quad \text{prime number} \]
   \[ 139 \equiv 3 \pmod{4} \]
   \[ 191 \equiv 3 \pmod{4} \]
2. Calculate the value of n.
   \[ n = p \cdot q \]
   \[ = 139 \cdot 191 \]
   \[ = 26549 \]
3. Publish n as public key and save p and q as private key.

2.1.2. Encryption of Rabin Algorithm

Rabin's algorithmic encryption steps:
1. Get the public key.
   \[ n = 26549 \]
2. Specify the message to be sent and convert to ASCII value. Then convert it to binary value, extend the binary value with itself, and change the binary value back to decimal.
   \[ m = \text{"R"} = 82 \]
   \[ m = 82_{10} = 1010010_2 \]
   \[ m = 1010010 \mid 1010010 \rightarrow \text{double extend} \]
   \[ m = 10578_{10} \]
3. Encrypt with the formula:
   \[ C = m^2 \pmod{n} \]
   \[ = 10578^2 \pmod{26549} \]
   \[ = 16598 \]
4. Send C to recipient.

2.1.3. Decryption of Rabin Algorithm

In the decryption process Rabin algorithm, recipient (recipient) certainly already know the private key p and q. In the decryption process the solution can use the Chinese Remainder Theorem (as difficult as RSA decryption) to get the correct plaintext. Rabin's decryption algorithm produces a choice of 4 possible answers in which only one original plaintext in the selection. [2]
Rabin algorithm decryption steps:

1. Accept C from sender.
   \( C = 16598 \)

2. Specify \( y_p \) and \( y_q \) with Extended Euclidean GCD.
   \( y_p \cdot p + y_q \cdot q = \text{GCD}(p, q) \)
   \( y_p \cdot p + y_q \cdot q = 1 \)
   \( 139 \cdot y_p + 191 \cdot y_q = 1 \)
   Find \( y_p \) and \( y_q \) with Extended Euclidean like table 1 below:

   \[ y_p = 11 \quad \text{and} \quad y_q = -8. \]

3. Calculate \( m_p \) and \( m_q \).
   \[ m_p = 16598^{139+1} \mod 139 \]
   \[ m_p = 125 \]
   \[ m_q = 16598^{191+1} \mod 191 \]
   \[ m_q = 118 \]

4. Calculate the value of \( r, s, t \) and \( u \) using Chinese Remainder Theorem (CRT).
   - Calculate the variable value \( v \) and \( w \) first to facilitate the calculation.
     \( v = y_p \cdot p \cdot m_q = 11 \cdot 139 \cdot 118 = 180422 \)
     \( w = y_q \cdot q \cdot m_p = -8 \cdot 191 \cdot 125 = -191000 \)
   - Calculate the values of \( r, s, t \) and \( u \).
     \( r = (v + w) \mod n \quad = 15971 \)
     \( s = (v - w) \mod n \quad = 26285 \)
     \( t = (-v + w) \mod n \quad = 264 \)
     \( u = (-v - w) \mod n \quad = 10578 \)

5. Convert \( r, s, t \) and \( u \) into binary.
   \( r = 15971_{10} = 1111100110001_2 \)
   \( s = 26285_{10} = 110011010101101_2 \)
   \( t = 264_{10} = 100001000_2 \)
   \( u = 10578_{10} = 10100101010010_2 \)

6. Determine the plaintext (original message) from the values of \( r, s, t \) and \( u \).

   \[ \begin{array}{c|c|c|c}
   r & 15971_{10} & 1111100110001_2 & \text{digits number is odd} \\
   s & 26285_{10} & 110011010101101_2 & \text{digits number is odd} \\
   t & 264_{10} & 100001000_2 & \text{digits number is odd} \\
   u & 10578_{10} & 10100101010010_2 & \text{left = right} \\
   \end{array} \]

   The message is in \( u \):
   \( m = 10100101010010_2 = 82_{10} = R \)

2.2 Fermat Factorization Method
Fermat factorization method known as Fermat's Difference of Squares Methods. This method is based on quadratic disputes, discovered in 1643 and still used almost 400 years later. Suppose that \( n \) is a composite and is written \( n = pq \), where \( 1 < q < \sqrt{n} \) so \( p > q \). Then it can be written:

\[
n = \left(\frac{p + q}{2}\right)^2 - \left(\frac{p - q}{2}\right)^2
\]

Where:

\[
s = \frac{p + q}{2}, \quad t = \frac{p - q}{2}
\]

so it can be written:

\[
n = s^2 - t^2 = (s + t)(s - t) = p \cdot q
\]

In the calculation, the Fermat factorization method looks for the value of \( y^2 - n \), until it finds a perfect root value, where the list starts from \( y = \lfloor \sqrt{n} \rfloor + 1, \lfloor \sqrt{n} \rfloor + 2, \) and so on. The above argument shows that this algorithm will ultimately work. This calculation will then find the factor value of the found root value. [3]

Example:
Use the Fermat method for \( n = 26549 \) factor.

Resolution:
The root of \( n = 26549 \rightarrow \sqrt{n} = \sqrt{26549} = 162,9386 ... \rightarrow \lfloor \sqrt{n} \rfloor = 162 \rightarrow y_0 \).
Starting from \( y_1 = y_0 + 1 \) \( y_1 = 162 + 1 = 163 \). The process is described in table 3 below.

| \( y_i \) | \( y_i^2 \) | \( y_i^2 - n \) | \( \sqrt{y_i^2 - n} \) |
|---|---|---|---|
| 163 | 26569 | 20 | 4,472135955 |
| 164 | 26896 | 347 | 18,62793601 |
| 165 | 27225 | 676 | 26 |

It can be written:

\[
26549 = (165 + 26)(165 - 26) = (191)(139)
\]

Since 191 and 139 are prime, then the prime factor is obtained from 26549.

2.3 Fermat Little Theorem
This theorem was put forward by a French mathematician named Pierre de Fermat in the 17th century. He proposed his discovery of the relationship between primes and modular arithmetic. He says that \( p \) is a prime number if it qualifies:

\[
a^{p-1} \equiv 1 \pmod{p}
\]

For \( p \) is prime and \( a \) is an integer not divisible by \( p \), where a number satisfies the requirement \( 1 < a < p - 1 \). Example:

\[
p = 7, \text{ then:}
2^6 \equiv 64 \equiv 1 \pmod{7}
3^6 \equiv 729 \equiv 1 \pmod{7}
4^6 \equiv (2^3)^6 \equiv 2^{18} \equiv 2^3 \cdot 64 \equiv 4096 \equiv 1 \pmod{7}
5^6 \equiv 64^2 \equiv 4096 \equiv 1 \pmod{7}
6^6 \equiv (2^6). (3^6) \equiv 1.1 \equiv 1 \pmod{7}
\]
so, \( p = 7 \) is prime.

2.4 Previous Research
1. The research entitled "Cryptanalysis of Public Key Cryptosystems Based on Non-Abelian Factorization Problems" by Jinhui Liu, Aiwan Fan, Jianwei Jia, Huanguo Zhang, Houzhen Wang, and Shaowu Mao (2016) concluded that two public key cryptosystems (BKT-B cryptosystem and BKT-FO cryptosystems) based on non-Abelian factorization problems is not safe in the sense that attackers capable of solving homogeneous linear equations with high efficiency in the general linear group given are also able to solve two schemes made in the study. [4]

2. The research entitled "Improving Fermat Factorization Algorithm by Dividing Modulus into Three Forms" by Kritsanapong Somsuk and Kitt Tientanopajai (2016) examined the development of Fermat factorization called "Multi Forms of Modulus for Fermat Factorization Algorithm (Mn-FFA)". Experimental results show that Mn-FFA can reduce computational iterations for all modulus values when compared to Fermat Factorization Algorithm (FFA) and other developed algorithms. [5]

3. In a study entitled "New Factoring Algorithm: Prime Factoring Algorithm" by Muhammad Usman, Zaman Bajwa, and Mudassar Afzal (2015) examines the development of a new factorization algorithm that is prime factorization algorithm and compares with Fermat factorisation algorithm which combines Fermat V2 Factorizing Algorithm and Algorithm Fermat V3 factorization And resulted in that the prime factorization algorithm is better than MFFV2 and MFFV3 because this method takes less time to compute. [6]

3 Result and Discussion

3.1 Testing Multiple Values of Public Key n Against Process Time

Based on testing for some randomly assigned public key n values by sorting them from the smallest to the largest, the results are found in Table 4 below.

| digit | Public Key n (bit) | Privat Key p (digit) | Privat Key q (digit) | Processing Time (ms) | Time (ms) |
|-------|--------------------|----------------------|----------------------|----------------------|------------|
| 7     | 4214533            | 23                   | 3863                | 1091                 | 10         |
| 8     | 12909361           | 24                   | 9923                | 1307                 | 4          |
| 10    | 19956401           | 25                   | 8803                | 2267                 | 17         |
| 10    | 1820592013         | 31                   | 91639               | 19867                | 270        |
| 10    | 2151915853         | 32                   | 70079               | 30707                | 5          |
| 11    | 6121315357         | 33                   | 78823               | 77659                | 0          |
| 12    | 88802400961        | 37                   | 820247              | 108263               | 7445       |
| 12    | 188364274433        | 38                  | 968419              | 194507               | 2748       |
| 13    | 489117187261        | 39                   | 87367               | 557483               | 730        |
| 14    | 17169389502961        | 44                  | 8045671             | 2133991              | 55633      |
| 14    | 21409668401441       | 45                  | 7296179             | 2969779              | 26346      |
| 15    | 46036937434801       | 46                  | 7294687            | 6311023              | 879        |
| 16    | 1206406819781309     | 51                   | 36123007             | 33397187            | 1458       |
| 17    | 2078189353947557     | 51                   | 61313311           | 33894587            | 137920     |
| 18    | 5369319140861209     | 53                   | 82504394            | 65079191            | 33539      |
| 19    | 13508498732809049     | 57                   | 400043443           | 337687043          | 101449     |
| 18    | 735695118563808778  | 60                   | 977753731           | 752433967          | 601035     |
| 18    | 763874117116360513    | 60                   | 956138167           | 798916039           | 286324     |

From Table 4, one may conclude as follows:
• While it is true that there is a trend that the larger the value of n, the larger the factoring time needed, the value of n does not always directly correlate with the factoring (processing) time. It can be seen at the value of n 16 digits, where the value $n_{13} < n_{14} < n_{15}$, and the process time obtained with $n_{13} = 1458$ ms, $n_{14} = 137920$ ms, and $n_{15} = 33539$ ms.

• The greater the value of n, the factoring time is not always bigger (slower). Furthermore, the smaller the value of n, the factoring time is not always smaller (faster). It seems that there are some other factors rather than only the value of n that determine the factoring time.

The comparison graph between the length of the public key and the processing time is shown in Figure 1 below.

![Figure 1. Comparison Chart of Bits n Public Key Score and Processing Time](image)

3.2 Personal Key Difference Test (p-q) Against Process Time

Based on the test for several values n randomly obtained difference between p and q. The difference value of each value of n is sorted by ascending by its digits and compared with the processing time, thus obtaining the conclusion result. In this test, the private key digits (p, q) used are 6 and 7. The results for the 6-digit-private-keys difference are shown in Table 5.

| n (digit) | Public Key | Privat Key | Key Difference (p-q) | Processing Time (ms) |
|-----------|------------|------------|----------------------|---------------------|
| 11        | 36901683293 | 197683     | 186671              | 11012               | 1          |
| 11        | 82945786373 | 415379     | 199687              | 215692              | 685        |
| 12        | 489117187261 | 877367     | 557483              | 6                   | 319884     | 730        |
| 12        | 188364274433 | 968419     | 194507              | 773912              | 7244       |
| 12        | 113726862113 | 986707     | 115259              | 871448              | 10472      |

The results for the 7-digit-private-keys difference are shown in Table 6.

| n (digit) | Public Key | Privat Key | Key Difference (p-q) | Processing Time (ms) |
|-----------|------------|------------|----------------------|---------------------|
| 11        | 36901683293 | 197683     | 186671              | 11012               | 1          |
| 11        | 82945786373 | 415379     | 199687              | 215692              | 685        |
| 12        | 489117187261 | 877367     | 557483              | 319884              | 730        |

Table 5. Comparison of 6-Digit-Private-Keys Difference (p - q) to Processing Time

Table 6. Comparison of 7-Digit-Private-Keys Difference (p - q) to Processing Time
The data in Tables 5 and 6 can yield some conclusions, including:

1. For each private key digit:
   - The smallest key difference value \((p - q)\) yields the smallest (fastest) factoring (processing) time and the largest key \((p - q)\) value results in the largest (longest) factoring (processing) time.
   - The larger the key difference value \((p - q)\) the greater (the longer) the time it takes to process, and vice versa. Thus, the key difference value is directly proportional to factoring time.

2. This does not apply if the key difference value \((p - q)\) between the digits 6 and 7 is combined, since the key difference of the second row (digit 6) = 215692 and on the first row key (digits 7) = 255912, where 255912 > 215692, but the processing time of the first row key (digit 7) is smaller than the second row key (digit 6), which is 90 <685. This also occurs if the key difference of the fifth row (digit 6) is compared with the second row key (digits 7).

Based on Tables 5 and 6, the comparison graph between the private key difference \((p - q)\) 6 and 7 digits and the processing time is shown in Figure 2 below.

**Figure 2. Comparison Chart of Private Key Differences 6 and 7 Digits to the Processing Time**

4 Conclusion

The conclusions of this experiment are as follows.

1. While it is true that there is a trend that the larger the value of \(n\), the larger the factoring time needed, the value of \(n\) does not always directly correlate with the factoring (processing) time.

2. The factor that affects the speed of factoring Rabin’s public key \(n\) with the Fermat factorization method is the difference between the private keys \((p - q)\). The greater the difference of the private keys \((p - q)\), the the longer the time needed to factor the Rabin’s public key \(n\).

References

[1] Rabin, M O 1979 *Digitalized Signatures and Public-Key Function As Intractable As Factorization*

[2] Srivastava A K, Mathur A 2013 *The Rabin Cryptosystem & Analysis in Measure of Chinese Reminder Theorem 3* (6)

[3] Batten L M 2013 *Cryptography : Applications and Attacks* Melbourne: IEEE Press

[4] Liu J, Fan A, Jia J, Zhang H, Wan H, Moa S 2016 *Cryptanalysis of Public Key Cryptosystems Based on Non-Abelian Factorization Problems* 21 (3) pp 344-351

[5] Somsuk K, Tientanopajai K 2016 *Improving Fermat Factorization Algorithm by Dividing Modulus into Three Forms* 43 (52) pp 350-353

[6] Usman M, Bajwa Z, Afzal M 2015 *New Factoring Algorithm: Prime Factoring Algorithm* 5 (1) pp 75-77