Implementation of RSA and RSA-CRT Algorithms for Comparison of Encryption and Decryption Time in Android-based Instant Message Applications

Achmad Wahyu Hidayat [1*], Riza Arifudin [1], Isa Akhlis [1]

[1]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang, Semarang, Indonesia
*Corresponding author: wahyoucf@gmail.com

ARTICLE INFO

Article history
Received 17 August 2020
Revised 15 September 2020
Accepted 12 October 2020

Keywords
Messaging
Cryptography
RSA-CRT Algorithm
Android

ABSTRACT

One of the advances in communication technology is producing instant messaging applications or instant messages. The confidentiality of instant messaging is still not maintained, so cryptography is needed. An example of a reliable cryptographic algorithm is Rivest-Shamir-Adleman (RSA), where RSA is a process of asymmetric key encryption (asymmetric key). Chinese Remainder Theorem (CRT) is an algorithm to reduce modular arithmetic calculation with a large modulus for the same calculation for each factor of the modulus. CRT can shorten the bit size of the decryption exponent d (which is the public key of RSA or RSA-CRT) by hiding *d* on a congruent system to speed up the decryption time, and it can be used with an RSA algorithm called RSA-CRT. This study uses three modulus n (key length), namely 1024 bits, 2048 bits, and 4096 bits. In the RSA-CRT 1024 bit decryption process, speed increases about 2.6 times faster than the 1024 bit RSA. In the RSA-CRT 2048 bit, the decryption process speed increases almost three times faster than the 2048 bit RSA. Whereas in the RSA-CRT 4096 bit, the decryption process's speed increases approximately 3.6 times faster than the RSA 4096 bit. From the results of this study, it can be concluded that the RSA-CRT algorithm can speed up the decryption process up to three times faster than the decryption process in the RSA algorithm. The longer or greater the modulus n used, the speed of the decryption process in the RSA-CRT algorithm will increase compared to the RSA algorithm.

1 Introduction

Technology in the communication sector nowadays is proliferating. The Technologies that is often used by the public is instant messaging. Instant message or Instant messaging is a communication facility chat for Internet users to communicate by sending a text message to another (Zuliarso & Februariyanti, 2013). The community uses instant messaging to mutually communicate through a message that is sent in the form of text. Instant messaging is easier for the user to send a message in a short time. With their instant messaging that raised the public's question widely associated with security information, if someone sends a confidential message through the facility of instant messaging, distribution of information is currently increasingly vulnerable to attack and disruption. As a result, the personal or confidential message is not guaranteed to all recipients without known information by parties who are not responsible. Instant messaging security on sent text messages is also not necessarily safe from cybercrime crimes such as tapping message transmissions and message manipulation. One of the handlings of that need is the manufacture of a system of security. The way to increase text message security in instant messaging is by using cryptography.

Cryptography is the study of mathematical techniques concerned with aspects of information security such as data confidentiality, data integrity, data integrity, and data authentication (Wulansari, Muslim, & Sugiharti, 2016). The cryptographic system or cryptosystem is a system for converting plaintext into ciphertext or *vice versa*. The main arithmetic operation in the RSA Cryptosystem is modular exponentiation defined as $C = M^e \bmod n$ for encryption and $M = C^d \bmod n$ for decryption, where C is the cipher, M is the message, e is the public key, d is the private key, and n is modulus (Sharma, Yadav, & Sharma, 2012). Cryptography consists of two processes, namely the encryption process and the decryption process. The encryption process is the process of encoding an open message into a secret message (ciphertext). After that, the ciphertext will be sent through an open communication channel. At the time of ciphertext received by the recipient of the message, then the secret message is converted back into an open message through the decryption process so that messages can be read by the recipient of the message (Muslim, Prasetiyo, & Alamsyah, 2016).

An example that can be reliable to cryptographic algorithms is RSA, where RSA is a process of asymmetric key encryption (asymmetric key). RSA is an algorithm popularly used because of its simplicity and has a processing speed that has the same fast as other cryptographic algorithms (Busran & Putra, 2014). Chinese Remainder Theorem (CRT) is an algorithm to reduce the large modulus of modular arithmetic calculations for the same calculation for each factor of the modulus (Arief & Saputra, 2016).

Increasing security with regard to message delivery and modification of the RSA algorithm using the CRT theorem so it can be compared with the RSA algorithm, it is necessary to build an Android-based secret message exchange application with RSA-CRT cryptography, which is expected to be present as a solution.

2 Methods

Android-based secret message exchange application using RSA-CRT cryptography was built to improve security in sending messages. Several processes are carried out in the RSA and RSA-CRT cryptographic algorithms: the key generation process, the encryption, and the decryption process. In the key generation process, the RSA and RSA-CRT cryptographic algorithms generate two keys: a public key (public) and a private key (private), which will later be used for the encryption decryption process. Then for the encryption process, namely changing the original text to ciphertext using a public key (public) and for the decryption process, which is changing the ciphertext to original text / plaintext using a private key (private). The key used to encrypt messages is called the public key, a set of keys used to decrypt the password is called the private key (Sukarno, 2013). In this study, the RSA-CRT algorithm key generator flowchart can be seen in Figure 1 and Figure 2 as an instant message application flowchart.

2.1 Rivest-Shamir-Adleman (RSA)

RSA is based on the principle that some mathematical operations are easier to perform in one direction but otherwise very difficult without some additional information. In the case of RSA, the idea is that it is relatively easy to multiply but much more difficult to factor in. Multiplication can be calculated in polynomial time, where the factoring time can grow exponentially proportional to the size of the number (Dhakar, Gupta, & Sharma, 2012).

Many asymmetric key cryptographic algorithms have been made; the most popular algorithm is RSA. The safety of the RSA algorithm lies in the difficulty of factoring large numbers into prime factors. Factoring is done to obtain the private key. If no effective algorithm has been found to factor large numbers into prime factors, so long as the RSA algorithm's security is guaranteed.

The RSA algorithm has the following quantities:

1. p and q are prime numbers (secret)
2. $n = p \cdot q$ (no secret)
3. $\phi(n) = (p-1)(q-1)$ (secret)
4. e (encryption key) (no secret)
5. d (decryption key) (secret)

6. *m* (*plaintext*) (secret)
7. *c* (*chipertext*) (no secret)

RSA is a block cipher where the plaintext and ciphertext are integers between *0* and *n-1* for some *n*. Encryption and decryption are of the following forms, for secret text block *C* can be calculated by Formula 1 and some original text blocks *M* can be calculated by Formula 2.

$$C = M^e \ mod \ n \tag{1}$$

$$M = C^d \ mod \ n = (M^e)^d \ mod \ n = mod \ n = \ M^{ed} \ mod \ n \tag{2}$$

The sender or receiver block must know the values of *n* and *e*, and only the recipient knows the value of *d*. The public key encryption algorithm has a public key of $KU = \{e, n\}$ and a unique key of $KR = \{d, n\}$. The algorithm must meet the following conditions to qualify as good public-key encryption:

1. It is possible to find the values for *e*, *d*, *n* such that $M^{ed} = M \ mod \ n$ for all $M < n$.
2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is not easy to calculate *d*, given *e* and *n*.

The first two conditions can be fulfilled easily. Meanwhile, the third condition can only be fulfilled for large *e* and *n* values. RSA password security lies in the difficulty of factoring in large numbers. RSA is currently still widely trusted and used on the internet (Wibowo, Susanto, & Junius, 2009).

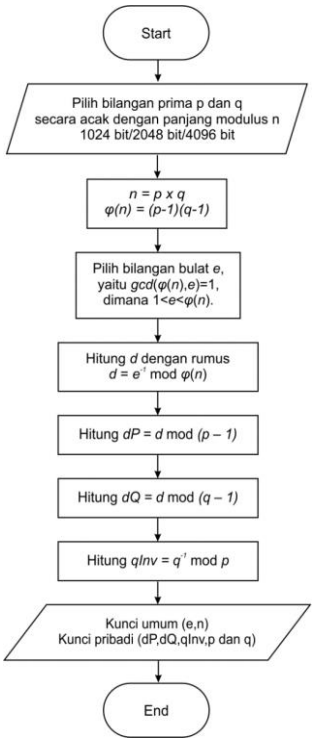

Figure 1. Flowchart of the RSA-CRT key generator

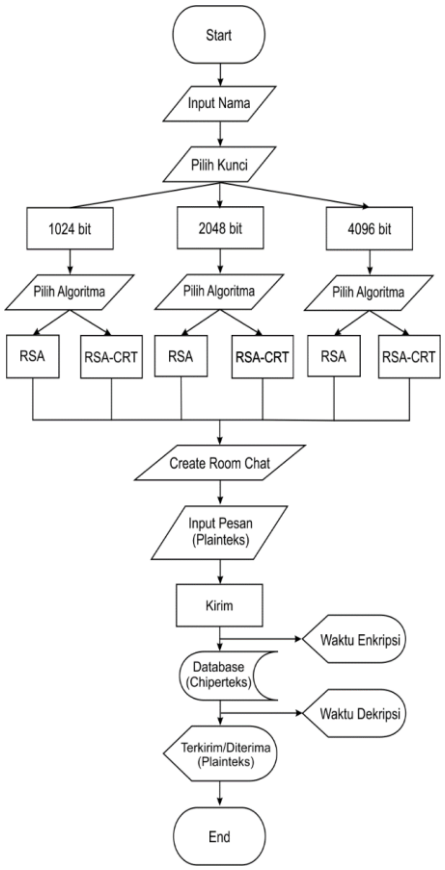

Figure 2. Flowchart of the instant message application

2.2 Chinese Remainder Theorem (CRT)

CRT is one of the main theorems in mathematics. As the perfect combination of beauty and usability continues to present itself in new contexts and a landscape open to new types of applications. Code theory and cryptography are two of the more recent application areas (Ekodeck & Ndoundam, 2016).

There are numbers $n_1, n_2, \ldots, n_k$ are positive integers where the pairs are relatively prime. For example: FPB(n_i,n_j) = 1 where $i \neq j$. Furthermore, $n = n_1,n_2, \ldots, n_k$ and $x_1, x_2, \ldots, x_k$ are integers. Then the system is congruent:

$$x \equiv x_1 \bmod n_1,$$

$$x \equiv x_2 \bmod n_2,$$

$$\ldots$$

$$x \equiv x_k \bmod n_k$$

The solutions are simultaneously congruent on all, and any two solutions mutually congruent modulo. Furthermore, there is exactly one solution between 0 and n-1. Unique solutions of simultaneous congruence satisfying $0 \leq x \leq n$ can be calculated by Formula 3.

$$x = \left(\sum_{i=1}^{k} x_i r_i s_i \right) mod$$
$$= (x_1 r_1 s_1 + x_2 r_2 s_2 + x_k r_k s_k) \, mod \, n \quad (3)$$

Where $r_i = \frac{n}{ni}$ and $s_i = r_i^{-1} \bmod ni$ for i = 1, 2, ... ,k.

If integers $n_1, n_2, \ldots, n_k$ are relatively prime pairs and $n = n_1, n_2, \ldots, n_k$, then for all integers a, b must be valid where $a \equiv b \bmod n$ if and only if $a \equiv b \bmod n_i$ for every i = 1,2, ... , k.

As a consequence of CRT, each positive integer $a < n$ can be represented uniquely as a *k-tuple* [$a_1, a_2, \ldots, a_k$] and vice versa. Where a_i denotes the residual $a \bmod n^i$ for every i = 1, 2, ... , k. The conversion of a to a residual system is defined as $n_1, n_2, \ldots, nk$ is carried out simply by $a \bmod n^i$ modular reduction. The reverse conversion from the residual representation to "standard numbers" is more difficult as required in the calculation of the Formula.

2.3 RSA-CRT

The RSA cryptographic system can be modified using the CRT theorem called RSA-CRT. It is proven that the RSA-CRT cryptography system has a shorter computation time than the usual RSA cryptographic system, which is about 4 times faster.

The RSA-CRT algorithm (Arief *et al*., 2016) is divided into 3 steps as follows:

2.3.1 RSA-CRT Key Generator

The RSA-CRT is the same as ordinary *RSA* but takes advantage of the CRT theorem to shorten the bit size of the exponent of *d* detection by hiding *d* in a congruent system to speed up the decryption time. Here is the RSA-CRT key generator algorithm:

1. Generate large prime numbers p and q
2. Calculate the modulus value n = p x q.
3. Calculate using Euler's function $\phi(n) = (p-1) \times (q-1)$.
4. Choose a random integer value e as the public key, provided it meets the Greater Common Divisor (GCD) $(e, \phi(n)) = 1$, $1 < e < \phi(n)$.
5. Compute the private key d such that $d \times e = 1 \pmod{\phi(n)}$.
6. dP = d mod (p – 1)
7. dQ = d mod (q – 1)
8. qInv = q-1 at Zp
9. Kpublic = (e, n), Kprivate = (dP, dQ, qInv, p, q)

2.3.2 RSA-CRT encryption

The RSA-CRT public key is the same as the RSA system, namely (*e* , *n*) so the encryption algorithm does not change.

2.3.3 RSA-CRT decryption

Given the ciphertext text and private key (dP, dQ, qInv, *p, q*), RSA-CRT decryption is as follows:

1. $m1 = C^{dP} \bmod p$
2. $m2 = C^{dQ} \bmod q$
3. $h = \text{qInv}.(m1 - m2) \bmod p$

$M = m2 + h.q$

3 Results and Discussion

In this study, a Crypto Chat application was built based on a mobile (mobile application) and Java application. Java language is also known as a portable programming language because it can be run as an operating system, provided the operating system has a JVM (Budiyanto, 2011). The algorithm was applied using Android Studio version 3.1.3. The research results comprise message testing and time testing of the encryption and decryption processes. The RSA and RSA-CRT cryptographic algorithms can be applied to Android-based instant messaging applications as message security, namely converting plain text to ciphertext and returning converted text (ciphertext) to plain text using three modulus *n* (1024 bits, 2048 bits, and 4096 bits).

Message testing is done to determine whether applications created by implementing the RSA algorithm and RSA-CRT algorithm with different modulus n can encrypt the original message into ciphertext properly and decrypt the ciphertext into plaintext correctly without changing a single character.

The results of message testing on the RSA 4096 bits algorithm, the sender of the message/plaintext on the first android smartphone, is shown in Figure 3. Then the results of message encryption in the form of ciphertext stored in the database are shown in Figure 4. While Figure 5 is the result of message testing on the algorithm. RSA 4096 bits, sender message/plaintext on android smartphone. And Figure 6 is a 4096 bits RSA-CRT ciphertext database.

Based on Figure 3, Figure 4, Figure 5, and Figure 6, the application that has been developed by implementing the RSA and RSA-CRT algorithms with modulus *n* 4096 bits has succeeded in encrypting the original message/plaintext into ciphertext, and then the ciphertext is stored in the database. After the ciphertext is stored in the database, the message in the form of ciphertext will be decrypted first into the original message/plaintext. Then the decrypted original message/plaintext is sent to the recipient without changing a single character.

Figure 3. Message sender/plaintext RSA 4096 bits

Figure 4. Chipertext database RSA 4096 bits

Figure 5. Message sender/plaintext RSA-CRT 4096 bits

Figure 6. RSA-CRT 4096 bits ciphertext database

In addition to testing messages, time testing is also carried out. The results of time testing of the encryption and decryption process of the RSA and RSA-CRT algorithms with modulus n 1024 bits, 2048 bits, and 4096 bits are presented in graphical form. Figure 7 shows a graph of the time complexity required to perform the first data text's encryption process.

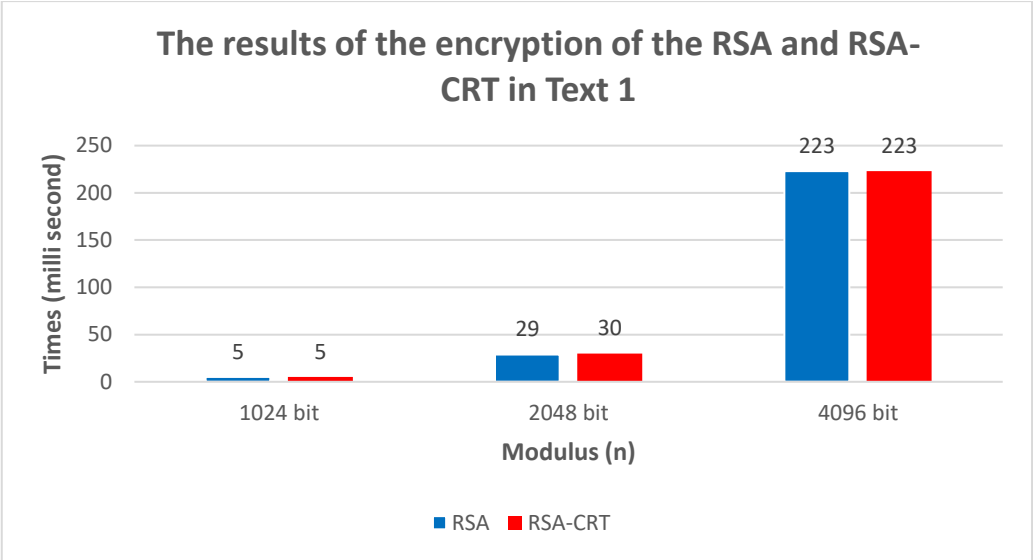

Figure 7. Comparison results of RSA and RSA-CRT encryption times

The graph from Figure 7 shows that the encryption process of the RSA and RSA-CRT algorithms does not experience a significant difference even though the modulus n used is different. Likewise, the second data text's encryption process to the fifth data text also did not experience a significant difference in the encryption process. Figure 8 is a graph of the complexity of the time required for the decryption process on the first data text presented.

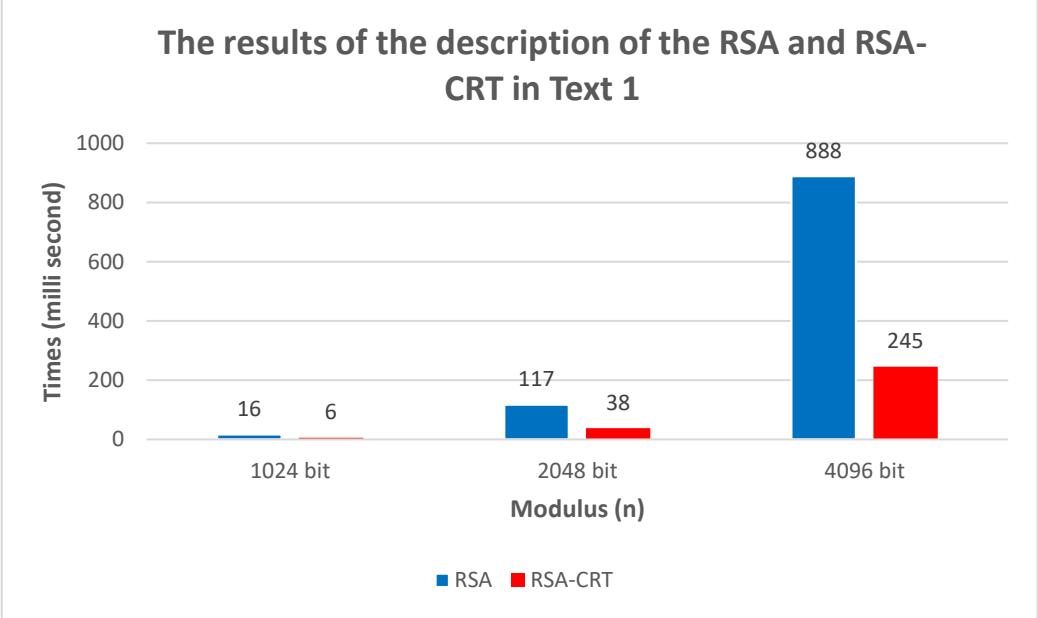

Figure 8. Comparison results of RSA and RSA-CRT decryption time

Based on the graph in Figure 8, the CRT method on RSA effectively accelerates the message decryption process even though the modulus n used is different. In the RSA-CRT 1024 bits, if calculated, the decryption speed will increase about 2.6 times faster than the 1024 bits RSA. In the 2048 bits RSA-CRT, if calculated, the decryption speed will increase nearly three times more quickly than the 2048 bit RSA. Whereas in the RSA-CRT 4096 bits, if calculated, the decryption speed will increase about 3.6 times faster than the RSA 4096 bits. Decryption speed increase also occurs in the decryption process of second text data to fifth text data. The longer or greater the modulus n used, the speed of the decryption process in the RSA-CRT algorithm will increase compared to the RSA

algorithm. Still, the key generation process will be longer along with the length or size of the modulus n used.

4 Conclusion

Based on the test results, the applications that have been developed by implementing the RSA and RSA-CRT algorithms with modulus *n* have successfully encrypted and decrypted messages. The results of time testing carried out in this study by comparing the time complexity of the encryption and decryption process of the RSA and RSA-CRT algorithms with *n* modulus of 1024 bits, 2048 bits and 4096 bits. The results revealed that for the comparison of the time complexity of the RSA algorithm encryption process and RSA-CRT did not experience a significant difference, although the modulus *n* used was different. Then, to compare the time complexity of the RSA and RSA-CRT algorithms' decryption process, the CRT method in RSA is very effective in accelerating the message decryption process even though the modulus *n* used is different. In RSA-CRT 1024 bits, if calculated, the decryption speed will increase about 2.6 times faster than RSA 1024 bits. In the 2048 bit RSA-CRT, if calculated, the decryption speed will grow about three times faster than the 2048 bit RSA. Whereas in the RSA-CRT 4096 bits, if calculated, the decryption speed will increase about 3.6 times faster than the RSA 4096 bits.

References

Arief, A., & Saputra, R. (2016). Implementasi Kriptografi Kunci Publik dengan Algoritma RSA-CRT pada Aplikasi Instant Messaging [Public Key Cryptography Implementation with RSA-CRT Algorithm in Instant Messaging Applications]. *Scientific Journal of Informatics*, 3(1), 46-54. doi:10.15294/sji.v3i1.6115

Budiyanto, U. (2011). Rancang Bangun Aplikasi Mobile Dosen Penasihat Akademik: Studi Kasus Universitas Budi Luhur Jakarta [Mobile Applications Designing for Academic Advisors: A Case Study of Universitas Budi Luhur, Jakarta]. *Jurnal TELEMATIKA MKOM*, 3(2), 75-79. Retrieved from http://journal.budiluhur.ac.id/index.php/telematika/article/view/191

Busran. B., & Putra, N. A. (2014). Rekayasa Perangkat Lunak Kriptografi menggunakan Algoritma RSA pada Sistem Keamanan File berbasis Java [Cryptograph Software Engineering using RSA algorithm in Java-Based File Security System]. *Jurnal TEKNOIF*, 2(1), 7-17. doi:10.21063%2Fjtif.2014.V2.1.

Dhakar, R. S., Gupta, A. K., Sharma, P. (2012). Modified RSA Encryption Algorithm (MREA). In *Second International Conference on Advanced Computing & Communication Technologies*, 426-429. IEEE. doi:10.1109/ACCT.2012.74

Ekodeck, S. G. R., & Ndoundam, R. (2016). PDF Steganography Based on Chinese Remainder Theorem. *Journal of Information Security and Applications*, 29, 1-15. doi:10.1016/j.jisa.2015.11.008

Muslim, M. A., Prasetiyo, B., & Alamsyah. (2016). Implementation Twofish Algorithm for Data Security in a Communication Network using Library Chilkat Encryption Activex. *Journal of Theoretical and Applied Information Technology*, 84(3), 370-375. Retrieved from https://lib.unnes.ac.id/33055/

Sharma, S., Yadav, J. S., & Sharma, P. (2012). Modified RSA Public Key Cryptosystem using Short Range Natural Number Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, 2(8), 134-138. Retrieved from http://www.ajer.org/papers/v4(01)/S040101430149.pdf

Sukarno, A. S. (2013). Pengembangan Aplikasi Pengamanan Dokumen Digital memanfaatkan Algoritma Advance Standard, RSA Digital Signature dan Invisible Watermarking [Digital Document Security Applications Development using Advance Standard Algorithms, RSA

Digital Signature and Invisible Watermarking]. *Seminar Nasional Aplikasi Teknologi Informasi*, 1-8. Retrieved from https://journal.uii.ac.id/index.php/Snati/article/view/3053

Wibowo, I., Susanto, B., & Junius, K. T. (2009). Penerapan Algoritma Kriptografi Asimetris RSA untuk Keamanan Data di Oracle [Application of the RSA Asymmetric Cryptographic Algorithm for Data Security at Oracle]. *Jurnal Informatika*, 5(1), 6-12. doi:10.21460/inf.2009.51.68

Wulansari, D., Muslim. M. A., & Sugiharti, E. (2016). Implementation of RSA algorithm with Chinese Remainder Theorem for Modulus N 1024 Bit and 4096 Bit. *International Journal of Computer Science and Security (IJCSS)*, 10(5), 186-194. Retrieved from http://www.cscjournals.org/manuscript/Journals/IJCSS/Volume10/Issue5/IJCSS-1289.pdf

Zuliarso, E., & Februariyanti, H. (2013). Pemanfaatan Instant Messaging untuk Aplikasi Layanan Akademik [The Use of Instant Messaging for Academic Service Applications]. *Jurnal Teknologi Informasi DINAMIK*, Vol. 18(2), 112-121. Retrieved from https://www.unisbank.ac.id/ojs/index.php/fti1/article/view/1699