Extended-RSA for Encryption Process to Improve Application Server Availability

IOP Conf. Series: Materials Science and Engineering **598** (2019) 012059 doi:10.1088/1757-899X/598/1/012059

Extended-RSA for Encryption Process to Improve Application Server Availability

A Susanto[1], Herman[1], M I Putranto[1], D N Utama[1] and A Wibowo[1]

[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480

ditdit.utama@binus.edu

Abstract. Every enterprise stores and operates its transactions confidentially, therefore it has to encrypt the transaction to protect it from data intruders. Technically, most encryption processes are ranging from hundreds of milliseconds. If there is a lot of encrypting process, then the performance will be getting bottleneck and the transaction's speed is being slowed down. In this paper, a promising scheme by adding GZIP compression on the former RSA encryption method to increase speed is proposed; thus every transaction is going to require less times, therefore the number of failed transactions or requests will be reduced. Hence, this way makes the transaction easier for customers to do. The result of little experimental test shows that using XOR to stream cipher-text to replace the absent of padding into specific bytes shows a lot of performance increase resulting other factors such as availability also affected without sacrificing the security level.

1. Introduction

Security is the most valuable part in an application [1]. The main purpose is to protect data and information technologically. Data is a secretive entity, where only the permitted users are able to access. In recent years, smartphones are used to do online based transaction (e.g. shopping, buying food, money transferring, etc.). When the transaction process, there are a lot information or data sent; such as data regarding a sender, a receiver, money amount, information of the transaction, and so on. All these information/data must be hidden from third person who may steal them between sender and receiver.

To secure the data, cryptography is the best choice [2]. Cryptography is a technique or a method operated to secure the data between the sender and receiver. Cryptography works by changing an ordinary-text (plain-text) into a new text that called cipher-text. This process called encryption. After a cipher-text reached by receiver, the cipher-text will be technically re-turned into plain-text. This process called decryption. Based on [3], one of the state-of-the-art of encryption is public-key cryptography; where it needs two separate keys. One key for locking or encrypting the plain-text, the other is for unlocking or decrypting the cipher-text. Rivest, Shamir, and Adlemen (RSA) encryption is one of public-key encryption methods. It uses private and public key as process key to encryption and decryption. For an example, if **A** wants to send a secret message to **B**, then **A** will encrypt the message using **B**'s public key. After **B** received the message, he/she is going to decrypt the secret message using his/her private key. In transferring process, even third party has public key, he/she is still not able to decrypt the message.

PT. XYZ operates a similar way to rationally safeguard its transaction. Every transaction occurs between server and client must be surpassed through encryption and decryption process. The current

RSA method, which PT. XYZ uses, needs 400ms up to 600ms process time for encryption and 100ms up to 150ms for decryption process. This process time does not include the client's connection speed which may vary on recipient device networks. This makes clients sometime have to wait a bit longer, even for doing small transaction only. When the execution time takes so long, it is possible that the connection will be timed out, and the transaction is broken down or canceled.

Owing to this problem, we proposed a customized RSA using XOR logic that has been proven to be secure and stated "one of the most important history in cryptographic" on the patent [4] to replace the padding process on former method with GZIP to compress the text thus reducing the execution time. For the result, this method can compress the process time up to more than 70% faster for both encryption and decryption processes. This method increased a performance and availability of application programming interface (API) without give negative affect for data security. Other than the background this paper will also cover up the previous related work, proposed method, and results discussion of both former and proposed methods.

2. Related Works

Cryptography has been known since 600 BC that mostly used on war. It was used to keep an information in secret, so the information can not be used or stolen by the enemy. Various types of cryptography have been developed and known until now. RSA is one of most used cryptography methods in current days. RSA first made in 1977 by Rivest, Shamir, and Adlemen; where RSA is named from the first initial of the founders' name [5].

In 1996, [6] introduced a compression specification on GNU system called GZIP. Based on [6], the purpose of this specification is to define lossless compressed data format that is independent of CPU type, operation type, file system, character set, also can be used for interchange. The compressed data format can produce another data stream by compressing or decompressing a data stream. Then, the compressed data format can be used freely and it is compatible with the file format produced by the current widely used GZIP utility.

Recent years, [7] conducted the research on RSA algorithm. They aimed to optimize the security, integrity, and availability of the information. Their customized RSA uses a matrix (Cod) to mix the character in the message, and the indexes of the rows in the matrix are randomly generated. The array has m rows by 221 columns, where m is an integer between 1 and 221. All rows represent chains mixed randomly. As the result showed the efficiency and functionality of the RSA algorithm in term of information security. Besides that, time, processor, and network performance when performing encryption and decryption were lower than the other RSA solutions.

Two years after, [8] proposed a hybrid topology, where they have merged the technique of artificial neural network (ANN) with RSA algorithm for an efficiency and to secure algorithm for data communication. The plain-text first affected with RSA algorithm. The cipher-text from RSA algorithm then will be affected with ANN architecture which has three consecutive layers as input, hidden, and output layers. The encrypted cipher-text from RSA algorithm is obtained from output layer that is in public network. As the result, they gained better result than ordinary RSA algorithm where the experiment for minimum file size (6 kb), ANN Based RSA Algorithm took 1.29261 sec less and 3.58185 sec for maximum file size (24) less than ordinary RSA algorithm for encryption process. While for decryption process, they experiment gained 0.50242 sec for minimum file size and 3.58185 sec for maximum file size less than ordinary RSA algorithm.

And the last research we found, [9] did their research that using general-purpose graphic processing units (GPGUs) for low cost, and high performance implementation of RSA algorithm. The experiment result showed that their method speed up the encryption process up to 6 x and 2 x for decryption was obtained using OpenCL over low-end GPU of desktop computer.

3. Research Methods

We started by finding out the general background of the encryption problem, as well as the current state-of-the-art to secure message sent during transactions. Next, we tried to identify the current problems and what could have causes it. We then analyzed the details of the current method and implemented the proposed method covering the problems that had been identified based on the studies about encryption. The implementation operated on a computer with following specifications: Intel ® Core (TM) i5-3317U CPU @ 1.70GHz, 8 GB of RAM running with Windows 8 Pro-64 bit. For the Java environment we used Java Development Kit version 1.8.0 and Spring Tool Suite IDE version 3.8.2. After testing out our proposed method, we evaluated how much is the performance and availability increased using 13 kinds of sentences with different length of words generated from lorem ipsum website (https://www.lipsum.com/), which is: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, and 100 [7] using the equation (1).

$$Reduced\ execution\ time\ =\ (new\ -\ old)\ /\ old\ *\ 100\% \quad (1)$$

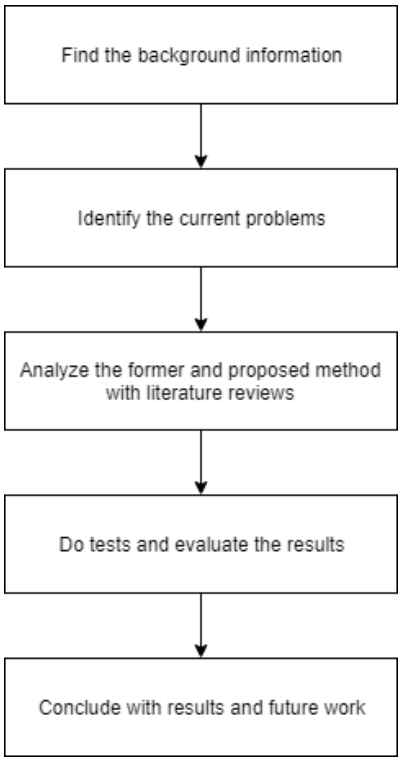

Figure 1. Research steps.

Finally we concluded the research with the end results reports of how much improvement is. The flowchart of our research steps can be seen on Figure 1. RSA encryption consists of 2 keys, public and private keys between sender and recipients. The idea of RSA is the sender only needs to hold one public key that is used to encrypt the data, while there can be multiple private keys for each recipients. We annotated the steps of RSA method concluded from [5] and old method:

1. Generate a public key and private key
    a. Generate by deciding 2 prime number as p and q
    b. Multiple p and q to get the value of n
    c. Find out the ϕn by using the $(p-1)(q-1)$ formula
    d. Decide the value of e, while $e = 1 < e < \phi n$, and must be coprime and the greatest common distribution of ϕn
    e. The value of d can be calculated by the following formula $ed\ (mod\ \phi n\ ) = 1$
    f. Results public key and private key value will be (n, e) and (n, d) respectively

2. Pad the text to 100-bit
    a. Get bytes value from text that want to be encrypted
    b. If the bytes length lower than 100, then padding it to 100
3. Encrypt
    a. Retrieve public key and use it to encrypt
    b. The encrypted text is $c = m^e \ mod \ n$, where m is the padded bytes text

These are the former method to decrypt:

1. Encrypt text; read the encrypted text received by the recipient device
2. Retrieve the private key; access the private key stored
3. Decrypt; decrypt the encrypted text using the stored private key using the formula $m' = c^d \ mod \ n$

The method we proposed will follow these steps:

1. Generate a public key and private key
    a. Generate by deciding 2 prime number as p and q
    b. Multiple p and q to get the value of n
    c. Find out the ϕn by using the $(p-1)(q-1)$ formula
    d. Decide the value of e, while $e = 1 < e < \phi n$, and must be coprime and the greatest common distribution of ϕn
    e. The value of d can be calculated by the following formula $ed \ (mod \ \phi n \ ) = 1$
    f. Results public key and private key value will be (n, e) and (n, d) respectively
2. Define custom key; define a custom key that will be used to create new key with XOR logic
3. Generate key with public key XOR custom key
    a. XOR custom key with public key
    b. The custom key is defined once and will be used again for next encryption
4. Encrypt; encrypt the string with $c = m^e \ mod \ n$ formula using the new generated key before
5. Compress Gzip; the encrypted string will be compressed with GZIP and encoded to base64 used to send to recipient device
6. Send the encoded zipped file get bytes value from zipped file and send it to recipient

These are the proposed method for decryption:

1. Encode zipped file; read the encrypted text received by the recipient device
2. Create zipped file with the encoded text; generate the zipped file from the encrypted encoded text received
3. Retrieve the private key; the recipient device will convert the base64 to GZIP file back and extract the string
4. Private key XOR custom key; generate the decrypt key using XOR logic between the private key and custom key
5. Decrypt; decrypt the encrypted text using the key generated before using the formula $m' = c^d \ mod \ n$

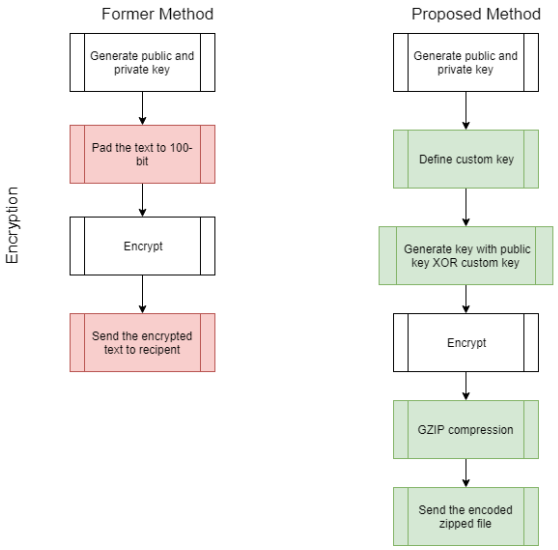

Figure 2. Encryption flowchart.

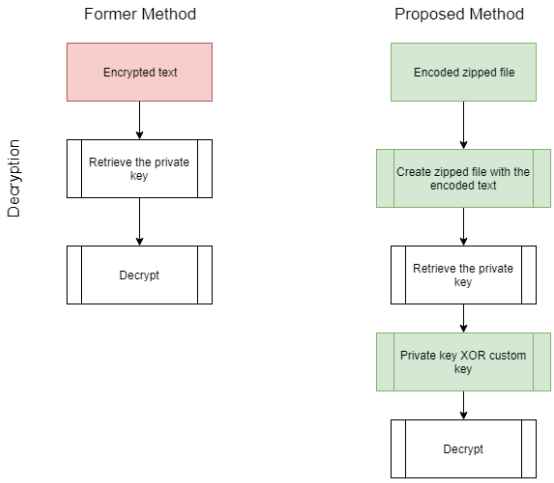

Figure 3. Decryption flowchart.

4. Result

The result for this research is quite pleasant. We got improvement both in encryption and decryption processes. We could decrease execution time in encryption process for 99.04% and 71.58% for decryption process. The detail data can be seen in Figure 4 and Figure 5. The average execution time for extended RSA encryption process is 3.83 ms and average execution time for extended RSA decryption process is 4.5 ms.

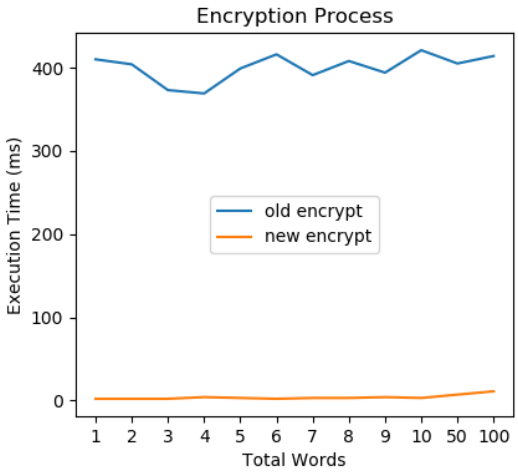

Figure 4. Encryption comparison.

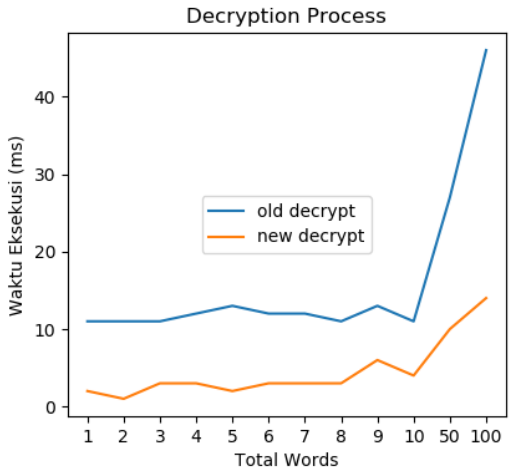

Figure 5. Decryption comparison.

5. Discussion

Based on the testing phase, we could decrease execution time in encryption and decryption processes into 99.04%. This means, we already achieved improvement in execution time, which is larger than 70.00%. By reducing the execution time by 99.04% means that we also improved the availability of our application. We know that, one of the problem in availability is slow response time affect to outages of system [10]; where the objectives of availability are to optimize the readiness of the system. Therefore, by reducing response time in our application, we also improved availabilty criteria, where it is able to downgrade practically the outages of the system.

In the previous RSA algorithm, every character will be extended with the padding character will be encrypted using generated public key.Therefore, in the old RSA algorithm we found that every encrypted text is very long and need more time to encrypt it. But, in our new RSA algorithm, we only encrypt the needed character and zip it into GZIP and get the byte array from the zipped file. Therefore, the length of encrypted text is depends on text that want to be encrypt.

6. Conclusion & Future Work

This research is made to make new RSA algorithm to support faster execution encryption time. In order to achieve this, we designed and implemented new RSA algorithm that did not pad the text that will be encrypted. We encrypted the text just as it is, zipped it and got the value of encoded byte array from zipped file. We used Spring tool suite IDE v 3.8.2 as our tool and Java Development Kit v 1.8.0. The results of this research showed that we created faster new RSA algorithm and already tested quantitatively. As for the future work, we planned to improve this algorithm in the security criteria.

7. References

[1] Rachmawati D, Budiman M A, and Lubis R S. 2018 *IOP Conference Series: Materials Science and Engineering* **434,** 1-6.
[2] Baheti A, Singh L, and Khan A U. 2014 *IEEE - Fourth International Conference on Communication Systems and Network Technologies* 664-8.
[3] Ahmed M, Sazzad T M S, and Mollah M E. 2012 *Int J Comput Sci Issues* **9,** 583-6.
[4] Klein M. 2012. *The TSEC / KW-26* (1st ed.) CreateSpace Independent Publishing Platform.
[5] Kelly M D. 2009. The RSA algorithm: A mathematical history of the ubiquitous cryptological algorithm 1–14.
[6] Deutsch P. 1996. *GZIP file format specification version 4.3 Status* RFC.
[7] Meneses F, Fuertes W, Sancho J, Salvador S, Flores D, Aules H, Castro F, Torres J, Miranda A, and Nuela D. 2016 *IJCSNS – Int J Comput Sci Netw Secur* **16,** 55-62.
[8] Chakraborty M, Jana B, Mandal T, and Kule M. 2018 *9th International Conference on Computing, Communication and Networking Technologies* 1–5.
[9] Srivatsa P S S and Rao P V R R B. 2019 *Emerging Technologies in Data Mining and Information Security.* **1,** 725–35.
[10] Schiesser R. 2010. *IT systems management* Pearson Education.