A cryptocompression system with Multi-Factor RSA algorithm and Levenstein code algorithm

M A Budiman, P Sihombing, and I A Fikri

Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

mandrib@usu.ac.id, poltak@usu.ac.id, ihyainunfikri123@gmail.com

Abstract. Exchanging digital messages by electronic means has a major security concern. The data being sent can be accessed and read by some unintended parties if there is no security system involved. Cryptography obscures the messages being sent and they can only be read by intended recipients by using the right key. Using asymmetric cryptography algorithm such as Multi-Factor RSA may help to obscure a message with high security; however, the ciphertext is usually much larger than the message so it takes much more time to transmit. In order to solve this problem, this research combines the Multi-Factor RSA with the Levenstein Code algorithm in a cryptocompression system to compress the size of the message being sent. This research also compares encrypt-then-compress scheme with compress-then-encrypt scheme. The result shows that encrypt-then-compress scheme has a better compression ratio than the compress-then-encrypt scheme. On the contrary, in terms of actual time performance, the compress-then-encrypt scheme is faster than encrypt-then-compress scheme.

1. Introduction
Cryptography is a technique based on mathematics and it is related to data security such as confidentiality, data integrity, and authentication [1]. The purpose of cryptography is to build a secure system that has a difficulty level which is not easy to break by unintended parties [2]. There are two types of cryptographic algorithms, namely symmetrical algorithms and asymmetric algorithms. Multi-Factor RSA is an asymmetric cryptographic algorithm which is the result of the development of the ordinary RSA algorithm. It was found that the processing time results were 2.25 times faster than the ordinary RSA algorithm [3]. The security of ordinary Multi RSA is high level due to the hardness of factoring of large integers into prime numbers [3].

Data compression is a technique created to convert the source stream into another stream to reduce the size of the initial stream length [4]. Compression types are divided into two types namely lossy compression and lossless compression [5] [6]. Levenstein code is a type of lossless algorithm [5] in which this algorithm processes the overall coding of non-negative integers, and the process has certain stages for coding and decoding.

In this research, we compare compares encrypt-then-compress scheme with compress-then-encrypt scheme to determine which one is better in terms of compression ratio and time performance.
2. Method
Here we will explain how the Multi-Factor RSA algorithm works step-by-step as can be seen in [3]. Starting from generating some prime numbers using the Agrawal-Biswas algorithm, followed by the data encryption process and also the final process, namely decrypting the obscured data back into the original one.

2.1. Multi-Factor RSA Algorithm

2.1.1. Key Generation. This process is the initial process which is indispensable for the encryption process and also the decryption process. It begins with the generation of prime numbers of at least three numbers. Choosing larger prime numbers can increase the level of security. The key generation is done by the recipient of the message. The steps are as follows:

1. Generate a value of $b$ where $b$ is the number of prime numbers and $b > 3$.
2. Generate prime numbers as much as $b$ using the Agrawal-Biswas algorithm, the steps are:
   a. Generate $p$ value.
   b. Take the $z$ value randomly, with $2 \leq z < n-2$.
   c. If the result is $(1 + z^p \mod p) = (1 + z^p \mod p)$, then $p$ is probably a prime number.
   d. If not then, $p$ is a composite number.
3. Calculate the value of $n = \prod_{i=1}^{b} p_i$
4. Find the value of $\varphi(n) = \prod_{i=1}^{b}(p_i - 1)$
5. Determine the value of $e$ according to the standard RSA public key, namely $e = 65537$.
6. Generate the value of $d$ which is the inverse of $e \mod \varphi(n)$.

2.1.2. Encryption. In this process, the text will be encrypted as follows:

1. Take the public key, namely $N$ and $e$.
2. Encrypt the message with the formula $c = m^e \mod n$
3. Send $c$ to the intended recipient.

2.1.3. Decryption. After the message is received, what must be done is the decryption so that the message returns to the original message, the steps are as follows:

1. Take the previously encrypted message which is $c$, and prepare the private keys that have been generated $(d, p_1, p_2, p_3, ..., p_n)$.
2. Calculate the value of $d$ using CRT (Chinese Remainder Theorem), $d_i = d \mod (p_i - 1)$, where $1 \leq i \leq b$.
3. Find the value $B = p_1 * p_2 * p_3 * ... * p_n$
4. Obtain the value for $B_i = B/p_i$.
5. Calculate $S_i = B_i - 1 \mod p_i$.
6. Get the final result with the formula $m = \sum_{i=1}^{b} a_i \cdot B_i \cdot S_i \mod B$.

2.2. Levenstein Code
The coding process of a number $n$ with the Levenstein Code algorithm is as follows [5]:

1. Initialize the value $C = 1$
2. Take the number $n$ and calculate its binary value, then delete the number 1 in the first index
3. Initialize $M$ which is the number of bits in the second step.
4. If $M \neq 0$, then add $C$ with $I$, then repeat step 2 where $n$ is $M$.
5. If $M = 0$, then prepend $I$’s $C$ times followed by the number 0 into the code.
Table 1. Levenstein Code for each $N$ value

| N | Levenstein Code | N | Levenstein Code |
|---|----------------|---|----------------|
| 0 | 0              | 9 | 1110 1 001     |
| 1 | 10             | 10| 1110 1 010     |
| 2 | 110 0          | 11| 1110 1 011     |
| 3 | 110 1          | 12| 1110 1 100     |
| 4 | 1110 0 00      | 13| 1110 1 101     |
| 5 | 1110 0 01      | 14| 1110 1 110     |
| 6 | 1110 0 10      | 15| 1110 1 111     |
| 7 | 1110 0 11      | 16| 11110 0 00 0000|
| 8 | 1110 1 000     | 17| 11110 0 00 0001|

3. Results and Discussions

The sample calculation of the cryptography system is as follows:

A. Key Generation (Recipient)
1. Choose $b = 3$.
2. Generate $p_1 = 73, p_2 = 167, p_3 = 199$ and do all the checks for each $p$ value with Agrawal-Biswas algorithm. To keep this step short, here we confirm that all the three numbers are prime numbers.
3. Calculate the value of $n$ with the formula $n = p_1 * p_2 * p_3 = 73 * 167 * 199 = 2426009$.
4. Calculate the value of $\sqrt{n} \leftarrow \prod_{i=1}^{b}(p_i - 1) = (73 - 1) * (167 - 1) * (199 - 1) = 2366496$
5. Let $e = 65537$.
6. Calculate the value of $d = e^{-1} \mod \sqrt{n}$. Then the result of the $d$ value is:
   - $d = 1052767 \mod 2366496$
   - $d = 1313729$

B. Encryption (Sender)
1. Input the message, for example $m = 'A' = 65$ (in ASCII Code).
2. Calculate ciphertext with the formula $c = m^e \mod n$
   - $c = 65^{65537} \mod 2426009$
   - $c = 203825$.
3. Send $c = 203825$ to the intended recipient.

C. Decryption (Recipient)
1. Take $c$ that has been sent by the sender, $c = 203825$, and prepare the private key ($d = 1313729, p_1 = 73, p_2 = 167, p_3 = 199$)
2. Calculate the value $d_i = d \mod (p_i - 1)$ where $1 \leq i \leq b$ using CRT, and observe:
   - $d_1 = 1313729 \mod (72) = 17$
   - $d_2 = 1313729 \mod (166) = 5$
   - $d_3 = 1313729 \mod (198) = 197$
3. Calculate the value of $B = p_1 * p_2 * p_3$,
   - $B = 73 * 167 * 199$
B = 2426009

4. Obtain the value for $B_i = B/p_i (B_1 = 33233, B_2 = 14527, B_3 = 12191)$. 
5. Calculate $S_i = B_i - 1 \pmod{p_i}$ so that the results are $S_1 = 69, S_2 = 83, S_3 = 111$. 
6. Find the value of $m = \sum_{i=1}^{k} a_i B_i S_i \pmod{B}$
   
   $m = a_1 B_1 S_1 + a_2 B_2 S_2 + a_3 B_3 S_3 \pmod{B}$
   
   $m = 149050005 + 78373165 + 87958065 \pmod{2426009}$
   
   $m = 65$ which is the letter ‘A’

In this study, some text files in the Canterbury Corpus are used to measure the performance.

| File         | Size (Bytes) | Encrypt-then-Compress | Compress-then-Encrypt |
|--------------|--------------|------------------------|-----------------------|
|              | $C_R$ | $S_S$ | Running Time (second) | $C_R$ | $S_S$ | Running Time (second) |
| asyoulik.txt | 125179 | 65%  | 35%  | 2,356 | 84%  | 16%  | 1,033             |
| fields.c     | 11150  | 65%  | 35%  | 0,281 | 87%  | 13%  | 0,123             |
| cp.html      | 24603  | 66%  | 34%  | 0,118 | 81%  | 19%  | 0,051             |
| grammar.lsp  | 3721   | 66%  | 34%  | 0,118 | 81%  | 19%  | 0,051             |
| lcet10.txt   | 426754 | 65%  | 35%  | 4,319 | 84%  | 16%  | 1,421             |

Table 2. The comparison of encrypt–then–compress and compress–then–encryption schemes

Table 2 shows that encrypt-then-compress scheme has a lower (or, better) compression ratio which results in better space savings than the compress-then-encrypt scheme. Conversely, in terms of time performance, the compress-then-encrypt scheme is quicker than encrypt-then-compress scheme.

4. Conclusion

Both the compress-then-encrypt scheme and the encrypt-then-compress scheme in the Multi-Factor RSA and Levenstein code cryptocompression system result in reduced data size. However, the encrypt-then-compress scheme is better than the compress-then-encrypt scheme in terms of compression ratio. On the flipside, the compress-then-encrypt scheme is faster than encrypt-then-compress scheme if the actual time performance is considered.

References

[1] Menezes A J, van Oorschot P C, and Vanstone S A 1997 Chapter 8.4 ElGamal Public Key Encryption. Handbook of Applied Cryptography. CRC Press
[2] Diffie W dan Hellman M. 1976. New directions in cryptography. IEEE Transactions on Information Theory
[3] Boneh D and Shacham H 2007 Fast Variants of RSA.
[4] Nelson M and Jean-Loup G 1996. The data compression book. (New York: M & t Books)
[5] Salomon, D and Motta G 2010. Handbook of Data Compression. Fifth Edition. (London: Springer)
[6] Budiman M A, Herriyance, and Khairunnada 2020 Performance Analysis of Start-Step-Stop Codes and Gopala-Hemachandra Codes2 (GH-2(n)) Ascomression Algorithms on Text Files Journal of Physics: Conference Series 1566 012073