A Cryptocompression System based on H-Rabin Public Key Encryption Algorithm and Elias Gamma Codes

Dian Rachmawati[1] and Mohammad Andri Budiman[1]

[1]*Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Medan, Indonesia*

Keywords: Public Key Cryptography, Compression, Encryption, Cryptocompression, H-Rabin, Elias Gamma Codes.

Abstract: A public key encryption scheme is usually used to encrypt a short message such as a secret key of another encryption algorithm. This is because a public key encryption creates a ciphertext that is many times longer than the original message. H-Rabin public key encryption algorithm is a variant of Rabin public key encryption algorithm. While Rabin uses two prime numbers to construct its modulus, H-Rabin uses three prime numbers. The longer the size of the modulus, the securer the cryptosystem becomes, but the longer the resulting ciphertext grows into. In this work, Elias Gamma compression algorithm is used in order to compress the ciphertext of the H-Rabin. As a result, one experiment shows that the size of the ciphertext can be reduced with the compression ratio of 1.94 to 1 and the space savings of around 48%.

1 INTRODUCTION

The public key encryption scheme (Diffie and Hellman, 1976) is a breakthrough concept that uses two keys to secure a message. One key is used to encrypt the message into ciphertext, and the other key is used to decrypt the ciphertext into the original message. The encryption key is made public and it is called ‘public key’, while the decryption key is kept private and it is called ‘private key’. One of the first algorithms to implement the public key encryption scheme is the Rivest-Shamir-Adleman (RSA) algorithm (Rivest, Shamir, and Adleman, 1978).

The RSA has many variants; Rabin algorithm (Rabin, 1979) is one of them. The Rabin algorithm also has many variants, one of them is H-Rabin algorithm which was created in 2014 (Hashim, 2014). The RSA and the Rabin algorithms depend their security on the hardness of factoring one big integer into its own two prime factors. Meanwhile, the H-Rabin algorithm rely its security on the hardness of factoring one big integer into its own three prime factors. Since it is harder to factor an integer into three prime factors than to factor an integer into two prime factors, the H-Rabin may have higher security level than the RSA and the Rabin algorithms (given the same size of all the prime factors).

The security of H-Rabin algorithm is not without drawbacks. One of the drawbacks is that the resulting H-Rabin ciphertext usually has bigger size than that of the RSA and the Rabin. This drawback is due to the size of H-Rabin modulus n, which is made by multiplying three large prime factors. In our previous study which implements the H-Rabin algorithm in the three-pass protocol (Rachmawati and Budiman, 2017), we have shown that with n = 216055288303296063383965054760089912266609 7797443665910587 (58 digits) and message m = 75 (2 digits), the resulting ciphertext c is 93605625 (8 digits). Even though the security of the ciphertext is high, the size of the ciphertext which has six more digits than the original message is still a trade-off since the transmission cost will be high. Therefore, public key encryption scheme is usually used to secure short data such as session key or a key of a symmetric encryption algorithm rather than the actual message (Smart, 2016). The actual message is then encrypted with the symmetric encryption algorithm. This whole system that makes use of both public key encryption algorithm and symmetric encryption algorithm is called hybrid cryptosystem (Rachmawati, Budiman, and Siburian, 2018; Rachmawati, Sharif, and Budiman, 2018).

It is indeed very ideal if one could use a public key encryption algorithm alone to encrypt large messages without setting it in a hybrid cryptosystem. One way

Rachmawati, D. and Budiman, M.
A Cryptocompression System based on H-Rabin Public Key Encryption Algorithm and Elias Gamma Codes.
DOI: 10.5220/0010092419101914
In *Proceedings of the International Conference of Science, Technology, Engineering, Environmental and Ramification Researches (ICOSTEERR 2018) - Research in Industry 4.0*, pages 1910-1914
ISBN: 978-989-758-449-7

to attain this is by compressing the ciphertext before transmitting it to the other party. To ensure the integrity of the ciphertext, only lossless compression algorithms should be used. Lossless compression algorithms produce statistics of the data and make use of this statistics to associate the data with a sequence of bits so that the data now have shorter representative bits (Saikumar and Sivaranjani, 2018).

Variable-length codes (VLC) or variable-size codes are codes that have different length which are commonly used in data compression (Salomon, 2008). One example of VLC is Elias gamma codes (Elias, 1975). An Elias gamma code uses binary representation of a symbol and its length to create shorter representative bits of that symbol. The codes increase in size slowly, making it useful to compress integer data (Salomon, 2008).

In this work, we use the Elias gamma codes to reduce the size of ciphertext of the H-Rabin public key encryption algorithm. In this way, the number of bits needed to represent the ciphertext is cut down. Therefore, the transmission cost of the ciphertext will be reduced and the H-Rabin algorithm can be efficiently used for encrypting larger data without having to put it in a hybrid cryptosystem scheme.

2 METHODS

In this section we elaborate the H-Rabin public key encryption, the Elias gamma codes, and the cryptocompression system.

2.1 H-Rabin Public Key Encryption Algorithm

Being a public key encryption algorithm, the H-Rabin algorithm has three stages which are key generation, encryption, and decryption. These stages can be found in Hashim (2014) and Rachmawati and Budiman (2017).

The key generation stage, which is done by the message recipient, is as follows:

1. Generate three large random primes: p, q, and r. All of them must be equivalent to 3 in modulo 4.
2. Compute n = pqr.
3. Let the primes p, q, and r be the private key.
4. Publish n as the public key.

The encryption stage, which is done by the message sender, is as follows:

1. Obtain the value of the public key n.
2. Prepare the message m. For decryption purpose, some redundant information can be added to m.
3. Compute the ciphertext $c = m^2 \bmod n$.
4. Send the value of c to the recipient.

The decryption stage, which is done by the message recipient, is as follows:

1. Get the value of c from the sender.
2. Compute the values of yp and yq by solving the Linear Diophantine Equation (LDE): yp × p + yq × q = 1. The LDE can be solved by using the Extended Euclidean algorithm.
3. Compute these three parameters:
   a. $mp = c^{((p + 1) / 4)} \bmod p$
   b. $mq = c^{((q + 1) / 4)} \bmod q$
   c. $mr = c^{((r + 1) / 4)} \bmod r$
4. Compute these six parameters:
   a. plus_mp = mp mod p
   b. minus_mp = -mp mod p
   c. plus_mq = mq mod q
   d. minus_mq = -mq mod q
   e. plus_mr = mr mod r
   f. minus_mr = -mr mod r
5. Compute b1, b2, and b3 as follows:
   a. $b1 \equiv (n / p)^{-1} \pmod p$
   b. $b2 \equiv (n / q)^{-1} \pmod q$
   c. $b3 \equiv (n / r)^{-1} \pmod r$
6. Compute x1 to x8 as follows:
   a. x1 = (plus_mp × b1 × n / p + plus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n
   b. x2 = (minus_mp × b1 × n / p + plus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n
   c. x3 = (plus_mp × b1 × n / p + minus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n
   d. x4 = (plus_mp × b1 × n / p + plus_mq × b2 × n / q + minus_mr × b3 × n / r) mod n
   e. x5 = n - x1
   f. x6 = n - x2
   g. x7 = n - x3
   h. x8 = n - x4
7. The original message is in one of the x's in step 6. This x is simply the one with the redundant information.

   For example, in the key generation stage, let p = 1034071, q = 1035163, and r = 1039667. Thus, n = pqr = 1112892866247075191.

In the encryption stage, the sender wants to encrypt the letter 'C', so she look into the ASCII table and get the value of 67. She converts it to binary and gets 1000011. She adds some redundant information, by appending the binary with itself and gets 10000111000011. The new binary is converted to decimal, and she get the value of m = 8643. She then computes the ciphertext c = m^2 mod n = 74701449. The c is sent to the recipient.

In the decryption stage, the recipient solves the Linear Diophantine Equation yp × p + yq × q = 1 by using Extended Euclidean algorithm, and he gets yp = 97639 and yq = -97536. Then, he computes mp = $c^{((p+1)/4)}$ mod p = 8643, mq = $c^{((q+1)/4)}$ mod q = 8643, and mr = $c^{((r+1)/4)}$ mod r = 1031024. Next, he computes plus_mp = mp mod p =8643, minus_mp = -mp mod p = 1025428, plus_mq = mq mod q = 8643, minus_mq = -mq mod q = 1026520, plus_mr = mr mod r = 1031024, and minus_mr = -mr mod r = 8643. Next, he computes b1 ≡ $(n / p)^{-1}$ (mod p) ≡ 203988, b2 ≡ $(n / q)^{-1}$ (mod q) ≡ 247091, and b3 ≡ $(n / r)^{-1}$ (mod r) ≡ 586409. Next, he calculates x1 = (plus_mp × b1 × n / p + plus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n = 93423036598505791, x2 = (minus_mp × b1 × n / p + plus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n = 142436456928361573, x3 = (plus_mp × b1 × n / p + minus_mq × b2 × n / q + plus_mr × b3 × n / r) mod n = 1063879435917210766, x4 = (plus_mp × b1 × n / p + plus_mq × b2 × n / q + minus_mr × b3 ×n / r) mod n = 8643, x5 = n - x1 = 1019469839648569400, x6 = n - x2 = 970456409318713618, x7 = n - x3 = 49013430329864425, and x8 = n - x4 = 1112892866247066548. All of the x's are then converted to binary. In binary, x4 is 10000111000011. Its first half of the bits (1000011) is exactly the same as the second half of the bits. Thus, x4 contains the redundant information, so the message is contained in x4. The message m is simply the decimal of 1000011, so m = 67, and with the ASCII table he gets the letter 'C' which is the original message.

2.2 Elias Gamma Codes

Constructing an Elias gamma code of the nth symbol is as follows (Salomon, 2008; Elias, 1975):

1. Let b(n) be the binary form of n.
2. Let M be the length of b(n).
3. Let u(M) be M – 1 zeros followed by 1.
4. Follow each bit of b(n) by each bit of u(M), and let it be e(n).
5. Discharge the leftmost bit of e(n).
6. The representative of the nth symbol is e(n).

For example, let n = 9. Thus, b(n) = 1001 and M = 4. Then, u(M) = 0001. So, e(n) =10000011. Dropping its leftmost bit, e(n) = 0000011. Therefore, the representative of the 9th symbol is 0000011.

2.3 A New Cryptocompression System

Our cryptocompression system works as follows:

1. Input plaintext as a string.
2. Get the list of m (in decimal form) which is the ist of the ASCII number of each character in the plaintext.
3. Encrypt each m with H-Rabin encryption algorithm as the list of c (in decimal form).
4. Convert the list of c into a string s, separating each value of c with a " " (space).
5. Count the frequency of each symbol "0", "1", "2", "3", "4", "5", "6", "7", "8", and "9" in the string s.
6. Sort the symbols by the frequencies in descending order.
7. Construct the Elias gamma code for each symbol.
8. Construct sb, the string-bit representation of the string s by using the Elias gamma code.
9. Convert each 8-bit of sb with the corresponding ASCII symbol and put it into a string cc.
10. We now have the compressed ciphertext cc whose length is shorter than the length of s, the uncompressed ciphertext.

3 RESULTS AND DISCUSSIONS

Let us compare the ciphertext created by H-Rabin public key encryption algorithm and the compressed ciphertext created by our cryptocompression system. Suppose we have plaintext = "CRYPTO COMPRESSION SYSTEM". Using the same parameters as in Section 2.1 (p = 1034071, q = 1035163, r = 1039667, and n = 1112892866247075191), we encrypt the plaintext with H-Rabin public key encryption, and we get the computer generated result as follows.

```
'C'  =>   67   =>   74701449
'R'  =>   82   =>   111894084
'Y'  =>   89   =>   131813361
'P'  =>   80   =>   106502400
'T'  =>   84   =>   117418896
'O'  =>   79   =>   103856481
' '  =>   32   =>   4326400
'C'  =>   67   =>   74701449
```

```
'O'  =>   79   =>   103856481
'M'  =>   77   =>   98664489
'P'  =>   80   =>   106502400
'R'  =>   82   =>   111894084
'E'  =>   69   =>   79227801
'S'  =>   83   =>   114639849
'S'  =>   83   =>   114639849
'I'  =>   73   =>   88679889
'O'  =>   79   =>   103856481
'N'  =>   78   =>   101243844
' '  =>   32   =>   4326400
'S'  =>   83   =>   114639849
'Y'  =>   89   =>   131813361
'S'  =>   83   =>   114639849
'T'  =>   84   =>   117418896
'E'  =>   69   =>   79227801
'M'  =>   77   =>   98664489
```

The ciphertext numbers on the right are then put in the string s, separated by a " " (space). So, the string s = "74701449 111894084 131813361 106502400 117418896 103856481 4326400 74701449 103856481 98664489 106502400 111894084 79227801 114639849 114639849 88679889 103856481 101243844 4326400 114639849 131813361 114639849 117418896 79227801 98664489 ". The length of s is 239. If each character in the string s is encoded by 8-bit ASCII, then we need 239 × 8 = 1912 bits, and this is the size of the uncompressed ciphertext.

Now, let us compress the size of the ciphertext using our cryptocompression system as in Section 2.3. The computer generated result of the cryptocompression system is as follows.

```
s   =   "74701449  111894084  131813361
106502400  117418896  103856481  4326400
74701449  103856481  98664489  106502400
111894084  79227801  114639849  114639849
88679889  103856481  101243844  4326400
114639849 131813361 114639849 117418896
79227801 98664489"
|s| = 239

Char = ['1', '4', '8', ' ', '0', '9',
'6', '3', '7', '2', '5']
|Char| = 11

Char    Freq    Elias Gamma Code
'1'     42      1
'4'     36      001
'8'     31      011
' '     25      00001
'0'     22      00011
'9'     22      01001
'6'     20      01011
'3'     16      0000001
'7'     11      0000011
'2'     9       0001001
'5'     5       0001011

String-bits             sb             =
0000011001000001100011100100101001000001
1110110100100100011011001000011000000011
0111000000100000010101110000110001101011
1000101100011000100100100011000110000011
1000001100110110110100101011000011000111
0000001011000101101011001011100001001000
0000100010010101100100011000110000100000
0110010000011000111001001010010000110001
1100000010110001011010110010111000010101
0101101011010110010010110100100001100011
1010110001011000110001001001000110001101
0001111011010010010001101100100001000000
1101001000100100010010000011011000111000
0011100101011000000101001011001010010000
0111001010110000001010010110010100100000
1011011010110000011010010110110100100000
1100011000000101100010110101100101110000
0110001110001001001000000010110010010000
1001000000100010010101100100011000110000
0111001010110000001010010110010100100000
1100000011011100000010000001010111000011
1100101011000000101001011001010010000111
1000001100110110110100101011000010000011
1010010001001000100100000110110001110000
0101001011010110101100100101101001000011
000000001

|compressed_bits| = 984
|uncompressed_bits| = 1912

Compression Ratio = 1.94308943089 : 1
Space Savings = 48.5355648536 %
```

The length of the uncompressed ciphertext is 1912 bits and the length of the compressed ciphertext is only 984 bits. Therefore, we have a compression ratio of 1912 / 984 = 1.94 to 1, and the space savings of (1 – 984 / 1912) × 100 % = 48.53 %.

4 CONCLUSIONS

In the final analysis we have seen that our cryptocompression system has successfully compressed the ciphertext with a compression ratio of 1.94 to 1 and space savings around 48%. Overall, this result is quite promising since it means that: (1) the transmission time of the ciphertext will be reduced; and (2) one could see a prospect of using a public key encryption algorithm (such as H-Rabin) alone to encrypt large messages without setting it in a hybrid cryptosystem by compressing the ciphertext using a compression algorithm (such as Elias gamma codes) before transmitting it.

ACKNOWLEDGEMENTS

We gratefully acknowledge that this research is funded by Lembaga Penelitian Universitas Sumatera Utara. The support is under the research grant TALENTA USU of Year 2018 Contract Number: 77/UN5.2.3.1/PPM/KP-TALENTA USU/2018.

REFERENCES

Diffie, W., Hellman, M., 1976. New directions in cryptography *IEEE Transactions on Information Theory* 22 6 pp 644-654

Elias, P., 1975. Universal codeword sets and representations of the integers *IEEE Transactions on Information Theory* 21 194–203

Hashim H, R., 2014. H-Rabin Cryptosystem *Journal of Mathematics and Statistics* 10 304–8

Rabin M, O., 1979. Digital Signatures and Public-Key Encryptions as Intractable as Factorization *Technical Report MIT/LCS/TR-212* (MIT Laboratory for Computer Science)

Rachmawat,i D., Budiman, M, A and Siburian, W, S, E., 2018. Hybrid cryptosystem implementation using fast data encipherment algorithm (FEAL) and Goldwasser-Micali algorithm for file security *Journal of Physics: Conference Series* 1013 012159

Rachmawati, D., Sharif,A., Jaysilen and Budiman M, A., 2018. Hybrid Cryptosystem Using Tiny Encryption Algorithm and LUC Algorithm *IOP Conference Series: Materials Science and Engineering* 300 012042

Rachmawati D and Budiman M A., 2017. An implementation of the H-Rabin algorithm in the Shamir three-pass protocol 2017. *2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-¬Mechanical System, and Information Technology (ICACOMIT)*

Rivest, R, L., Shamir, A and Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems *Communications of the ACM* 21 2 pp 120-126

Saikumar, R and Sivaranjani, M., 2018. A Review on Compression and Encryption of Data for Secure Transmission *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 4 3

Salomon, D., 2008. *Variable-length Codes for Data Compression* (Springer London)

Smart N, P., 2016. *Cryptography Made Simple* (Cham: Springer International Publishing)