Analysis of the formation of a dynamic brief key algorithm RC4+ for file security

Siddik Karo-karo, Tulus[2], M Zarlis[3]

[1]Student, Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia
[2]Department of Information Technology, Faculty of Mathematics, Universitas Sumatera Utara, Indonesia
[3]Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia

siddikzidane85@gmail.com[1] , tulus_jp@yahoo.com[2], m.zarlis@usu.ac.id [3]

Abstract. The development of technology, makes it easy for us to communicate and exchange information and can also have an impact on the development of cryptographic algorithms. An algorithm that is used to do encryption and decryption must not only be correct, but also must be used efficiently. An algorithm is said to be efficient if the algorithm uses relatively smaller memory and has a fast processing time. So far, many studies have discussed the efficiency of an algorithm. Regarding the measurement of time and memory space needed is known as algorithm complexity. The complexity of the algorithm itself can be interpreted as a number of computational steps carried out by an algorithm to solve problems. The RC4 + algorithm is a symmetric algorithm that is used to encrypt and decrypt messages, and use the same key in the encryption-decryption process. The key used is the RC4 + session key key. In the result, the amount of memory needed by the modified RC4 + Algorithm is more. Execution time or runtime The modified RC4 + algorithm for encryption and decryption is much faster than the original algorithm.

1. INTRODUCTION

The development of technology, makes it easy for us to communicate and exchange information and can also have an impact on the development of cryptographic algorithms. An algorithm that is used to do encryption and decryption must not only be correct, but also must be used efficiently. An algorithm is said to be efficient if the algorithm uses relatively smaller memory and has a fast processing time. So far, many studies have discussed the efficiency of an algorithm. Regarding the measurement of time and memory space needed is known as algorithm complexity. The complexity of the algorithm itself can be interpreted as a number of computational steps carried out by an algorithm to solve problems [1]. In the symmetrical algorithm RC4 + (Ron Code or Rivest's Cipher), it provides very secure message security and is an efficient stream cipher [2]. The RC4 + algorithm is a symmetric algorithm that is used to encrypt and decrypt messages, and use the same key in the encryption-decryption process. The key used is the RC4 + session key key. The RC4 + algorithm has a weakness in its use to encrypt a message. The disadvantage is the RC4 + algorithm can only do the encryption and decryption process for short messages only. This is because the number of plaintext characters and keys is the same, it will take a lot of memory and a long time in the process of encryption and decryption [3]. This weakness makes RC4 + no longer a practical algorithm to use. Many

previous studies only discussed the combination of the RC4 + algorithm with an asymmetrical algorithm in key generation, not the effectiveness and complexity of using the algorithm itself. There needs to be a method and propose a new method to make the RC4 + algorithm more effective, that is by making keys shorter or not depending on the number of plaintext lengths.

2. PROPOSED METHOD

In the process of modifying RC4+ algorithm, the sender of the message makes the plaintext to be sent, then determines the number of keys. The process of making keys is based on the scheme of taking even numbers in descending and there should be no repeat characters. Next is the system will make the modified plaintext using a retrieval scheme that says odd with ascending sequences and no similar characters. After that the system processes the XOR algorithm RC4 + by using the modified key and plaintext. After encrypting XOR, the sender of the message will get a ciphertext. This ciphertext will later be submitted to the recipient of the message

2.1. RC4+ Algorithm

RC4 + Cipher is a type of RC4 algorithm. Where the RC4 algorithm is one of the symmetrical key algorithms in the form of stream ciphers that do the encryption / decryption process in one byte and use the same key. Stream ciphers are used to encrypt plaintext into ciphertext bits per bit (1 bit each time transformation) or bytes per byte (1 character = 1 byte). RC4 uses key length variables from 1 to 256 bits which are used to initialize 256-bit tables. RC4 + successfully handles the weaknesses of the RC4 algorithm [4]. RC4 + is a symmetric key algorithm in the form of synchronous stream ciphers, which are ciphers that have symmetric keys and encrypt or decrypt plainteks in digits per digit or bits per bit. Stream ciphers are perfect for communicating and processes that run in real-time [5]. RC4 + uses a structure like RC4 and adds several operations to strengthen the cipher. This structure tries to utilize good points in RC4 then provides some additional features for better security limits [6]. The RC4 + algorithm consists of 2 components, namely Key Scheduling Algorithm (KSA +) and Pseudo Random Generation Algorithm (PRGA +).

2.1.1. Key Schedulling Algorithm (KSA)

KSA + is a modified KSA RC4. KSA + consists of three layers of randomization. The KSA + of the RC4 + algorithm is as follows of Figure 1, Figure 2, Figur 3 & Figur 4 [6]:

Initialization
For $i = 0, \ldots, N-1$ $S[i] = i;$ $j = 0;$

Figure 1. Array Initialization

Layer 1: Basic Scrambling
For $i = 0, \ldots, N-1$ $j = (j + S[i] + K[i]);$ $\text{Swap}(S[i], S[j]);$

Figure 2. Basic Randomization

Layer 2: Scrambling with IV

For $i = \frac{N}{2} - 1$ down to 0
    $j = (j + S[i]) \oplus (K[i] + IV[i])$;
    Swap$(S[i], S[j])$;

For $i = \frac{N}{2}, \ldots, N - 1$
    $j = (j + S[i]) \oplus (K[i] + IV[i])$;
    Swap$(S[i], S[j])$;

Figure 3. Basic Randomization

Layer 3: Zigzag Scrambling

For $y = 0, \ldots, N - 1$
    If $y \equiv 0 \bmod 2$ then
        $i = \frac{y}{2}$;
    Else
        $i = N - \frac{y+1}{2}$;
    $j = (j + S[i] + K[i])$;
    Swap$(S[i], S[j])$;

Figure 4. Zig-zag Randomization

2.1.2. Pseudo-Random Generation Algorithm (PRGA)

Pseudo Random Generation Algorithm (PRGA +) from RC4 + algorithm is different from RC4 algorithm. The scheme from Pseudo-Random Generation Algorithm (PRGA) is shown as follow of Figure 5[6]:

Input: Key-dependent scrambled permutation array $S[0 \ldots N - 1]$.
Output: Pseudo-random keystream bytes z.

Initialization:
$i = j = 0$;

Output Keystream Generation Loop:
$i = i + 1$;
$j = j + S[i]$;
Swap$(S[i], S[j])$;
$t = S[i] + S[j]$;
$t' = (S[i_R^3 \oplus j_L^5] + S[i_L^5 \oplus j_R^3]) \oplus \ 0xAA$;
$t'' = j + S[j]$;
Output $z = (S[t] + S[t']) \oplus S[t'']$;

Figure 5. Pseudo Random Generation Algorithm (PRGA$^+$)

2.2. Complexity Of Algorithm

The complexity of algorithms can be said to be a measure of the number of computational steps required by an algorithm to solve a problem. In general, an algorithm that can perform short computational steps (less time needed) has a low complexity. In contrast, algorithms

that require many stages and long completion times to solve problems have high complexity [7]. The complexity of an algorithm can at least be categorized into two types. That is time complexity and space complexity. Time complexity, usually expressed by, is calculated based on the number of computational steps needed to run an algorithm as a function of input size, where input size is the amount of data processed by an algorithm. The complexity of space, expressed by, is measured from the memory used by the data structure contained in the algorithm as a function of input. By using time complexity or space complexity, it can be determined the rate of increase in time or space required by the algorithm, along with the increase in input size.

2.2.1. Time Complexity

The truth of an algorithm must always pass the testing stage with a certain number of inputs, the purpose is to ensure the performance of the algorithm in the form of time needed to execute an algorithm and see the memory space used for the use of its data structure [8]. Determining the time complexity of an algorithm, it takes a measure of the input and running time of the algorithm. So that the running time of an algorithm can be expressed as a function of n. The running time algorithm for a particular input n is the number of computational steps executed. A pseudocode line can have a different amount of time than another line. We can assume that every implementation of the i-line takes as much time as ci, where ci is a constant.

2.2.2. Space Complexity

The complexity of space can be interpreted as the amount of memory needed by a computer to run a computational or algorithmic phase until it is finished. The complexity of space can be measured from the memory used by the data structure contained in the algorithm as a function of input size n.

3. RESULTS AND ANALYSIS

The system was built using SharpDevelop 4.4. with the programming language is C#. This system is tested with Personal Computer with 1.0 GHz processor specification AMD C-70 APU, 2 GB Memory.

The test results of the number of characters that input for each algorithm both the original version and the modified version produce different memory and runtime requirements. At 500 characters, the original RC4+ algorithm requires 373,1200 bits of memory and 23 ms runtime. Whereas in the modified version of the algorithm with the same number of characters, 500 characters require 6483968 bits of memory and 10 ms can be illustrated in a graph of Fig. 6.

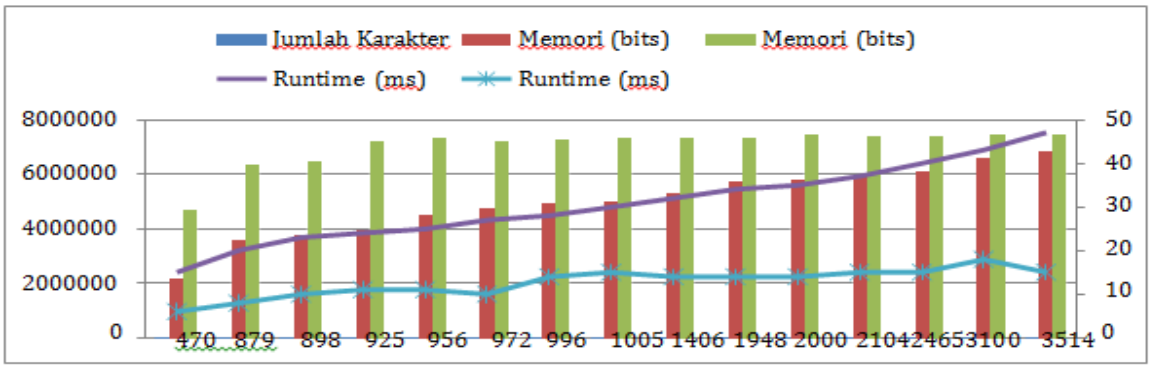

Figure 6. Graph of memory needs and runtime needs in the encryption process

The test results of the same number of characters memory usage and runtime are different for the original algorithm and modifications. For the number of 500 memory and runtime characters needed to perform the decryption process on the original RC4+ algorithm are 2200974 bits and 15 ms. While the modified algorithm with the number of 500 characters of memory and runtime needed is 3293128 bits and 7 ms. The amount of memory needed depends on the number of characters and processes carried out in the decryption process. In addition, the number of keys also determines the amount of memory and the speed of a decryption process can be illustrated in a graph of Figure 7:

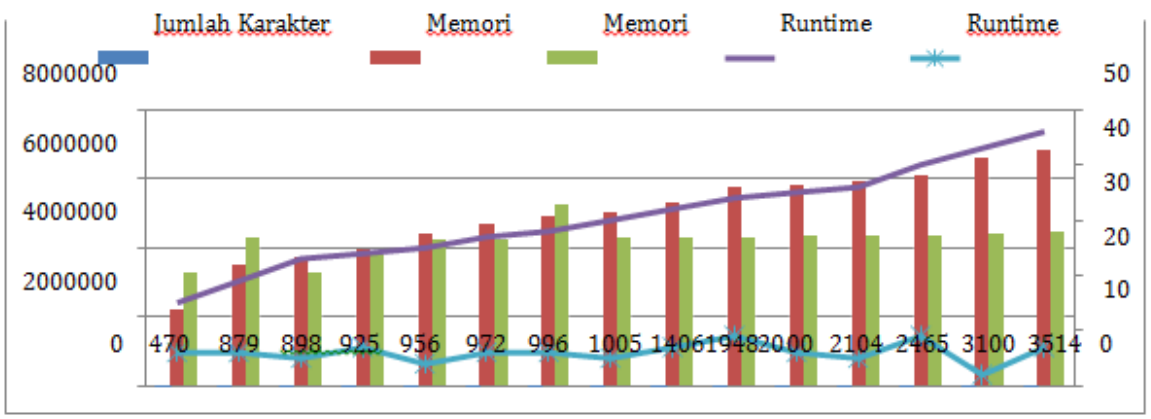

Figure 7. Graph of memory needs and runtime needs in the decryption process

4. CONCLUSION

- The RC4+ algorithm can be modified to get a key that is shorter than the number of characters. The rule that is applied is the selection of the number of keys determined by an even number with a descending sequence and there must not be same or repeated characters.
- The amount of memory needed by the RC4+ Algorithm is modified more.
- The execution time or the One Time Pad Algorithm runtime for encryption and decryption is much faster than the original algorithm.

Acknowledgments

The author would like thank to Prof. Dr. Tulus and Prof. Dr. Muhammad Zarlis for the guidance given until the research was completed well.

References

[1] Berthold, Vocking. Helmut, Alt. Martin, Dietzfelbinger. Rudiger, Reischuk. Christian, Scheideler. Heribert, Vollmer. Dorothea, Wagner. *Algorithm Unplugged*. Springer: New York.

[2] Banik, Subhadeep, Santanu Sarkar, and Raghu Kacker. "Security analysis of the RC4+ stream cipher." *International Conference on Cryptology in India*. Springer, Cham, 2013.

[3] Zulfizar. 2014. Analisis Kombinasi Algoritma *One Time Pad* dan Algoritma Elgamal dalam Pengamanan Pesan. Tesis. Universitas Sumatera Utara.

[4] Jindal, P. & Singh, B. 2014. RC4 Encryption-A Literature Survey. *International Conference on Information and Communication Technologies (ICICT 2014)*, pp. 697 –

705.
[5] Taqieddin, E., Abu-Rjei, O., Mhaidat, K., Bani-Hani, R. 2015. Efficient FPGA implementation of the RC4 stream cipher using block RAM and pipelining. *The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015)*, pp. 8 – 15.
[6] Paul, G. & Maitra, S. 2012. *RC4 Stream Cipher and Its Variants*. CRC Press Taylor & Francis Group: London.
[7] Kumalasari, D. 2017. Analisa Perbandingan Kompleksitas Algoritma *Bubble Sort*, *Coktail Sort*, dan *Comb Sort*, dengan Bahasa Pemrograman C++. *Journal Speed* **9**2:, 1 - 5.
[8] Nugraha, D. W. 2012. Penerapan Kompleksitas Waktu Algoritma *Prime* untuk Menghitung Kemampuan Komputer dalam Melaksanakan Perintah. *Jurnal Ilmiah Foristek* **2**(2): 195 - 207.