A Comparison of One Time Pad Random Key Generation using Linear Congruential Generator and Quadratic Congruential Generator

D Apdilah\textsuperscript{1}, M K Harahap\textsuperscript{2}, and N Khairina\textsuperscript{2}, A M Husein\textsuperscript{3}, M Harahap\textsuperscript{3}

\textsuperscript{1}Lecturer of Amik INTeL Com GLOBAL INDO, Medan, Indonesia
\textsuperscript{2}Lecturer of Politeknik Ganesha Medan, Medan, Indonesia
\textsuperscript{3}Lecturer of Universitas Prima Indonesia, Medan, Indonesia

E-mails: dicky@nusa.net.id, choir.harahap@yahoo.com, nurulkhairina27@gmail.com, amirmahmud@unprimdn.ac.id, mawaddah@unprimdn.ac.id

Abstract. One Time Pad algorithm always requires a pairing of the key for plaintext. If the length of keys less than a length of the plaintext, the key will be repeated until the length of the plaintext same with the length of the key. In this research, we use Linear Congruential Generator and Quadratic Congruential Generator for generating a random number. One Time Pad use a random number as a key for encryption and decryption process. Key will generate the first letter from the plaintext, we compare these two algorithms in terms of time speed encryption, and the result is a combination of OTP with LCG faster than the combination of OTP with QCG.

1. Introduction
Research on data security is very interesting. This study is also always discussed to improve the convenience of using confidential data. Cryptography depends on key strengths, by using a strong key on the process of encryption and decryption, we can improve the security of confidential data [1][2]. The One Time Pad (OTP) algorithm is characterized, in that the key is used only once in every encryption process [3][4]. The OTP algorithm uses random numbers that can be constructed from different types of random number generator algorithms. An OTP algorithm key will be established along the plaintext to be encrypted. If the plaintext is longer than the key, the key will be repeated throughout the plaintext [5][6][7]. Because of this unique key, the OTP algorithm can be considered as the most powerful algorithm in cryptography [8].

Muhammad Iqbal et al [5] implements the OTP algorithm in the SMS app to keep the security of confidential messages. XOR operators are used in encryption and decryption processes. Key selection is done manually by selecting some characters that will be converted into ASCII code. Messages generated from this application are messages that can be guaranteed confidentiality because the message can only be read by the sender and the recipient. The OTP algorithm is the latest version of the Vernam Cipher algorithm [9]. The keys used in the OTP Algorithm can also be generated through a random number generator algorithm. The first random number to be referred to is called Seed [10]. Jhessica Clawdia et al [11] uses the Linear Congruential Generator (LCG) algorithm to generate
random numbers. This algorithm is used to avoid generating repetitive keys. From the key generation process can be seen that the longer the range of m values the less likely the key will be repeated.

Using the LCG algorithm to generate keys, can also generate dynamic keys. As has been done by Zeenat Mahmood et al [12] it appears that the cryptographic algorithm is already difficult to solve by cryptanalysis. Random number generator algorithms have an important role in cryptography because they have resistance to cryptanalysis attacks. Mina Mishra et al [13] he did some research on testing some random number generator algorithms, one of them is LCG. The LCG algorithm has resistance to cryptanalysis attacks and has a good key sensitivity.

Various combinations of algorithms are also performed to generate random numbers. Massoud Sokouti et al [14] he generates random numbers with a genetic algorithm. The key that has been formed will be used by the OTP algorithm. The results of this study, OTP algorithm has a strong resistance and able to maintain data security from cryptanalyst attacks.

In addition to the LCG random number generator algorithm, there is also a Quadratic Congruential Generator (QCG) algorithm. This algorithm generates random numbers with mathematical formulas and quadratic ranks as written by Jurgen Eichenauer et al [15].

From some previous research, researchers are still conducting trials to generate random numbers that aim to form a strong cryptographic key. In this study we propose and compare the combination of OTP with LCG algorithm and OTP with QCG algorithm. And we also compare these two algorithms in terms of time speed encryption. The author will also modify the OTP algorithm. Differences exist in the encryption and decryption processes that do not involve the first plaintext characters.

2. Literature Review

2.1 One Time Pad Algorithm
One Time Pad Algorithm (OTP) is a symmetric key cryptography algorithm with encryption and decryption process using the same key [1]. In this research, we use 256 characters from the table of ASCII so we will do encryption and decryption process in mod 256.

OTP encryption formula is [16]:

$$C_i = (P_i + K_i) \mod 256$$  \hfill (1)

And OTP decryption formula is [16]:

$$P_i = (C_i - K_i) \mod 256$$  \hfill (2)

Description of formula :

$C_i$ = Ciphertext
$P_i$ = Plaintext
$K_i$ = Key
Mod 255 is mean total of ASCII numbers

2.2 Linear Congruential Generator Algorithm
This is the formula of Linear Congruential Generator [1]:

$$X_n = (a \times X_{n-1} + b) \mod m$$  \hfill (3)

Description of formula :

$X_n$ = Random number to n of the series
$X_{n-1}$ = previous random number
$a$ = multiplier
2.3 Quadratic Congruential Generator Algorithm

This is the formula of Cubic Congruential Generator [1]:

\[ X_n = (a X_{n-1}^2 + b X_{n-1} + c) \mod m \] (4)

Description of formula:

- \( X_n \) = Random number to i of the series
- \( X_{n-1} \) = previous random number
- \( a, b, c \) = multiplier
- \( d \) = increment
- \( m \) = modulus

3. Proposed Method

3.1 One Time Pad Algorithm Modified with LCG

In this chapter, we perform the encryption and decryption process with a combination of the modified OTP and LCG. The modified OTP algorithm is not different from the original OTP algorithm. Differences exist in the encryption and decryption processes that do not involve the first plaintext characters. In generating the keys we use \( a = 3; b = 17; \) and \( m = 101. \)

3.1.1 Encryption Process:

There are several steps to perform the encryption process:

- Step 1: Retrieve the plaintext to be encrypted
- Step 2: Convert the first letter into ASCII code
- Step 3: Make the number in Step -2 as Z0 (seed) to generate a random number
- Step 4: Leaving the first letter unencrypted
- Step 5: Key = Z
- Step 6: Perform encryption process on next letter
- Step 7: Generate a random number Zn using LCG
- Step 8: Return to step 5 until all letters are encrypted

Here is a table for performing the encryption process:

| No | Px | ASCII Px | Kn | Encryption Results | Ciphertext in Char |
|----|----|----------|----|--------------------|-------------------|
| 1  | K  | 75       | 75 | C1 = 75 (first letter unencrypted) | K                 |
| 2  | H  | 72       | 42 | C2 = (72 + 42) mod 256 = 114 | r                 |
| 3  | O  | 79       | 44 | 123                | {}                |
| 4  | T  | 73       | 50 | 123                | {}                |
| 5  | R  | 82       | 68 | 150                | –                 |

3.1.2 Decryption Process:

There are several steps to perform the decryption process:

- Step 1: Retrieve the ciphertext to be decrypted

\[
\begin{align*}
\text{Table 1. Encryption Process.} \\
\text{No} & \quad \text{Px} & \text{ASCII Px} & \text{Kn} & \text{Encryption Results} & \text{Ciphertext in Char} \\
1 & \text{K} & 75 & 75 & C_1 = 75 \text{ (first letter unencrypted)} & \text{K} \\
2 & \text{H} & 72 & 42 & C_2 = (72 + 42) \mod 256 = 114 & \text{r} \\
3 & \text{O} & 79 & 44 & 123 & \{} \\
4 & \text{T} & 73 & 50 & 123 & \{} \\
5 & \text{R} & 82 & 68 & 150 & \{}
\end{align*}
\]
Step – 2 : Convert the first letter into ASCII code
Step – 3 : Make the number in Step -2 as Z₀ to generate a random number
Step – 4 : Leaving the first letter undecrypted
Step – 5 : Key = Z₀
Step – 6 : Perform decryption process on next letter
Step – 7 : Generate a random number Zₙ
Step – 8 : Return to step 5 until all letters are decrypted

Here is a table for performing the decryption process:

| No | Px | ASCII Px | Kn | Encryption Process | Plaintext in Char |
|----|----|----------|----|--------------------|-------------------|
| 1  | K  | 75       | 75 | P₁ = 75 (first letter undecrypted) | K                 |
| 2  | r  | 114      | 42 | P₂ = (114 - 42) mod 256 = 72  | H                 |
| 3  | {  | 123      | 44 | 79                  | O                 |
| 4  | {  | 123      | 50 | 73                  | I                 |
| 5  | –  | 150      | 68 | 82                  | R                 |

3.2 One Time Pad Algorithm Modified with QCG
In this chapter, we perform the encryption and decryption process with a combination of the modified OTP and QCG. The modified OTP algorithm is not much different from the original OTP algorithm. Differences exist in the encryption and decryption processes that do not involve the first plaintext characters. In generating the keys we use a = 3; b = 17; c = 19, and m = 101.

3.2.1. Encryption Process
There are several steps to perform the encryption process:
Step – 1: Retrieve the plaintext to be encrypted
Step – 2: Convert the first letter into ASCII code
Step – 3: Make the number in Step -2 as Z₀(seed) to generate a random number
Step – 4: Leaving the first letter unencrypted
Step – 5: Key = Z₀
Step – 6: Perform encryption process on next letter
Step – 7: Generate a random number Zₙ using QCG
Step – 8: Return to step 5 until all letters are encrypted

Here is a table for performing the encryption process:

| No | Px | ASCII Px | Kn | EncryptionResults | Ciphertext in Char |
|----|----|----------|----|-------------------|-------------------|
| 1  | K  | 75       | 75 | C₁ = 75 (first letter unencrypted) | K                 |
| 2  | H  | 72       | 90 | C₂ = (72 + 90) mod 256 = 162 | ø                 |
| 3  | O  | 79       | 94 | 173               | -                 |
| 4  | I  | 73       | 47 | 120               | x                 |
| 5  | R  | 82       | 72 | 154               | §                 |

3.2.2. Decryption Process :
There are several steps to perform the decryption process:

Step 1: Retrieve the ciphertext to be decrypted
Step 2: Convert the first letter into ASCII code
Step 3: Make the number in Step 2 as $Z_0$ to generate a random number
Step 4: Leaving the first letter undecrypted
Step 5: Key = $Z$
Step 6: Perform decryption process on next letter
Step 7: Generate a random number $Z_n$
Step 8: Return to step 5 until all letters are decrypted

Here is a table for performing the decryption process:

| No | Cx | ASCII Cx | Kn | Decryption Process | Plaintext in Char |
|----|----|----------|----|-------------------|-------------------|
| 1  | K  | 75       | 75 | $P_1 = 75$ (first letter undecrypted) | K                 |
| 2  | e  | 162      | 90 | $P_2 = (162 - 90) \mod 256 = 72$ | H                 |
| 3  | -  | 173      | 94 | 79                | O                 |
| 4  | x  | 120      | 47 | 73                | I                 |
| 5  | $\$| 154      | 72 | 82                | R                 |

4. Result and Discussion
The testing is done in the text file (.txt) thus get all the characters that have been symbolized on ASCII code. In reality, there are still many file types that need to be tested. Testing is also done on text files with large size and test results are really stable. This is influenced by the use of modulo 255 in the encryption process and decryption process. So all of ASCII characters can be encrypted and decrypted properly.

| No | Length | Time Test | Encryption Time Average | Time Difference |
|----|--------|-----------|-------------------------|-----------------|
|    |        |           | OTP + LCG               | OTP + QCG       |                 |
| 1  | 100    | 10        | 0,000200033             | 0,000200033     | 0               |
| 2  | 1000   | 10        | 0,00079999              | 0,000999975     | 0,000199986     |
| 3  | 10000  | 10        | 0,01079998              | 0,012199974     | 0,001399994     |
| 4  | 100000 | 10        | 0,098200035             | 0,118799973     | 0,020599937     |
| 5  | 1000000| 10        | 1,238600016             | 1,416599989     | 0,177999973     |

The process through which Encryption and Decryption are the same, then decryption test is not performed.

5. Conclusions
Based on the above experiments it is concluded that the LCG process is faster than CCG. Future work needs to be tested about how the treatment of documents that have special codes such as the use of font size, italic, bold, underline, and also the use of color. Thing need to be tested on the spreadsheet.
file and how it affects the formula on the spreadsheet and also against the existing table. In the future are still many more need to be tested with OTP using this dynamic key.

References
[1] B. Schneier, Applied Cryptography Protocol, Algorithms, and Source Code in C, 2nd ed., California: Jhon Wiley & Sons, Inc, 1996, p. 17.
[2] P. P. Kumar, J. M. Kumar, and M. Neelima, "A new method for generating a key for cryptography," International Journal of Engineering Trends and Technology (IJETT), vol. 4, no. 9, pp. 3915-3920, 2013.
[3] C. Matt and U. Maurer, "The One-Time Pad Revisited," in International Symposium on Information Theory, Istanbul, 2013.
[4] E. H. A. Mendrofa., E. Y. Purba, B. Y. Siahaan and R. W. Sembiring, "Collaborative Encryption Algorithm Between Vigenere Cipher, Rotation of Matrix (ROM), and One," Advances in Science, Technology and Engineering Systems Journal Vol. 2, No. 5, pp. 13-21, 2017.
[5] M. Iqbal, M. A. S. Pane and A. P. U. and Siahaan, "SMS Encryption Using One-Time Pad Cipher," IOSR Journal of Computer Engineering (IOSR-JCE), vol. 18, no. 6, pp. 54-58, 2017.
[6] R. Shukla, H. O. Prakash, R. Bhushan, S. Venkataraman and G. Varadan, "Unconditionally Secure and Authenticated One Time Pad Cryptosystem," in International Conference on Machine Intelligence and Research Advancement, India, 2013.
[7] E. B. B. Samah M.H. Alwahbani, "Speech Scrambling Based on Chaotic Maps and," in International Conference on Computing, Electrical and Electronic Engineering (ICCEEE), Sudan, 2013.
[8] B. J. Saha, K. K. Kabi, Arun and C. Pradhan, "Non-Blind Watermarking Technique using Enhanced," in 5th ICCCNT, Hefei, 2014.
[9] N. Khairina and M. K. Harahap, "Analisis Algoritma One Time Pad Dengan Algoritma Cipher Transposisi Sebagai Pengamanan Pesan Teks," SinkrOn (Jurnal & Penelitian Teknik Informatika), pp. 58-62, 2017.
[10] R. Maheshwari, S. Gupta, V. Sharma and V. Chauhan, "VRS algorithm A Novel Approach to Generate Pseudo," in International Advance Computing Conference (IACC), Gurgaon, 2014.
[11] J. Clawdia, N. Khairina and M. K. Harahap, "Implementasi Algoritma Kriptografi One Time Pad (OTP) dengan Dynamic Key Linear Congruential Generator (LCG)," Komik, vol. 1, no. 1, pp. 12-14, 2017.
[12] Z. Mahmood, J. Rana, and A. Khare, "Symmetric Cryptography using Dynamic Key and Linear Congruential Generator (LCG)," International Journal of Computer Applications, vol. 50, no. 19, pp. 7-11, 2012.
[13] M. Mishra and V. Mankar, "Text Encryption Algorithms based on Pseudo-Random Number Generator," International Journal of Computer Applications, vol. 111, no. 2, pp. 1-6, 2015.
[14] M. Sokouti, B. Sokouti, S. Pashazadeh, M. R. F. Derakhshi and S. Haghipour, "Genetic-based random key generator (GRKG): a new method," Neural Comput & Applpp. 1667 - 1675, 2013.
[15] J. Eichenauer and Herrmann, "Quadratic congruential pseudorandom numbers;," Journal of Computational and Applied Mathematics,pp. 75-85, 1997.
[16] M. K. Harahap, "Analisa Perbandingan Algoritma Kriptografi Klasik Vigenere Cipher dan One Time Pad," Info Tekjar, vol. 1, no. 1, pp. 61-64, 2016.