Implementation of Super-Encryption with Trithemius Algorithm and Double Transposition Cipher in Securing PDF Files on Android Platform

M A Budiman, D Rachmawati and Jessica
Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

Email: mandrib@usu.ac.id, dian.rachmawati@usu.ac.id, jessica13@student.usu.ac.id

Abstract. This study aims to combine the trithemius algorithm and double transposition cipher in file security that will be implemented to be an Android-based application. The parameters being examined are the real running time, and the complexity value. The type of file to be used is a file in PDF format. The overall result shows that the complexity of the two algorithms with super encryption method is reported as \( \Theta(n^2) \). However, the processing time required in the encryption process uses the Trithemius algorithm much faster than using the Double Transposition Cipher. With the length of plaintext and password linearly proportional to the processing time.

1. Introduction

The word ‘cryptography’ comes from Greek. According to the language, the word cryptography is divided into two, namely ‘crypt’ and ‘graphia’. ‘Crypt’ means secret and ‘graphia’ means writing. According to its terminology, cryptography is the science and art of keeping messages safe when messages are sent from one place to another [1]. Thus, cryptography can help us in preventing the arrival of important information, into unauthorized hands. The purpose of cryptography is 4, ie confidentiality, integrity, authentication, and non-repudiation. Confidentiality focuses on data confidentiality, integrity focuses on received data is true and genuine data, authentication is done to prevent false data, and non-repudiation aims to prevent the shielding of the data from the sender. Cryptography algorithm divided 2 that is, symmetric algorithm and asymmetric algorithm. However, in this study the algorithm used is a classical algorithm, namely Trithemius Algorithm and double transposition cipher.

Trithemius Cipher is one of the polyalphabetic codes designed to be easier to use. Instead of using a random combination of letters of the alphabet, Trithemius uses a special table of Trithemius Cipher which has 24 rows and columns 24 of 24 letters in the Latin alphabet except the letter j and v [2]. However, in order to make the system more useful, in this study we used a trithemius table consisting of 65 rows and columns. Where, all characters have entered into the table. Both lowercase and uppercase. Plus numbers 0 through 9, and some punctuation.

Double Transposition Cipher consists of two rounds of transition columns, using 2 different keys. After the key in the first round is determined, then each letter in the key is numbered in alphabetical order, and the plaintext is read per column based on the sequence of numbers. Where, the direction of
the sequence should be determined first whether left-to-right, or right-to-left, to facilitate when there is the same letter on the key. Ciphertext generated from the first round, encrypted again in the second round in the same way, but using a different key [3].

File to be used in this research is a file with PDF format. PDF files are considered safe for static file formats. The advantage of a document created in PDF format is that it cannot be changed directly by the recipient of a document unlike a document file with the format .doc, .txt or the other [4].

The purpose of this study are using Trithemius Algorithm and Double Transposition Cipher in super encryption scheme to avoid theft of files/data, calculates the super encryption complexity in Big Theta (Big-$\Theta$), and measure Real Running Time super encryption in millisecond (ms).

2. Method
On the encryption process, Trithemius algorithm will be used first. In the encryption process, the next letter on the plaintext will be replaced by the relevant letter of the next row in the table. After using the last line, one should move back to the first line. This means that all letters on the plaintext change according to the number of positions specified by the actual row. Therefore the first letter is encrypted without a shift, the second letter with the shift is determined by the second row (so by one position), the third letter with the shift determined by the third row (so by two positions) and so on. Or, can also be calculated by the formula:

\[ C \equiv P + K(\text{mod } n) \]  \hspace{1cm} (1)
\[ P \equiv C - K(\text{mod } n) \]  \hspace{1cm} (2)

Where:
- $C$: Ciphertext
- $P$: Plaintext
- $K$: Key
- $N$: number of letters used

This cipher consists of two rounds of transition columns, using 2 different keys. After the key in the first round is determined, then each letter in the key is numbered in alphabetical order, and the plaintext is read per column based on the sequence of numbers. Where, the direction of the sequence should be determined first whether left-to-right, or right-to-left, to facilitate when there is the same letter on the key.

Ciphertext generated from the first round, encrypted again in the second round in the same way, but using a different key. Ciphertext generated from the second round, then divided into blocks. Where, each block consists of 5 letters to facilitate the transmission process. One of the requirements of encryption process using this cipher is, the table we use must be fully loaded. In order for the table used to be fully charged, padding character is used.

While in the decryption process by using Double Transposition Cipher, start by using the second key first. Where, the number of rows in the table is determined by the division of ciphertext length by key. (Salomon, 2005). Then, the ciphertext is inserted into the table based on the number of fragments in alphabetical order in the key used.

3. Result and Discussion
The system was built using Android Studio 1.3.2. with the programming language is Java. This system is tested with Personal Computer with processor intel core i7 – 3632QM CPU @ 2.20 GHz, Memory 4 GB RAM.

Table 1. Complexity of Trithemius Algorithm Encryption Method
1. public String enkripsi_ttm(String plaintext, String key) {
   char[] arraykey = createkey(key, plaintext);
2. char[] arrayplaintext = plaintext.toCharArray();
3. char[] arrayciphertext = new char[arrayplaintext.length];
4. int p, k, c;
5. for (int i = 0; i < arrayplaintext.length; i++) {
6.   if (getindex(arrayplaintext[i]) == -1) {
7.     arrayciphertext[i] = arrayplaintext[i];
8.     continue;
9.   }
10.   p = getindex(arrayplaintext[i]);
11.   k = getindex(arraykey[i]);
12.   c = (p + k) % index.length();
13.   arrayciphertext[i] = getcharacter(c);
14. }
15. String hasil = new String(arrayciphertext);
16. return hasil;
17. }

\[ T(n) = \sum_{i} C_i \#_i \]

\[ = 5C_1 + 5C_1n + C_2n + C_3n + C_4n + C_5 \]

\[ = (5C_1 + C_5)n^0 + (5C_1 + C_2 + C_3 + C_4)n^1 \]

\[ \theta(n) \]

**Table 2. Complexity of Trithemius Algorithm Decryption Method**

| No | Program Code | C | # | C# |
|----|--------------|---|---|----|
| 1. | public String dekripsi_ttm(String ciphertext, String key) { | C1 | 1 | C1 |
|    | char[] arraykey = createkey(key, ciphertext); | | | |
| 2. | char[] arrayciphertext = ciphertext.toCharArray(); | C1 | 1 | C1 |
| 3. | char[] arrayplaintext = new char[arrayciphertext.length]; | C1 | 1 | C1 |
| 4. | int p, k, c; | C1 | 1 | C1 |
5. for (int i=0; i<arrayciphertext.length;i++){  
6.     if(getindex(arrayciphertext[i])==1){  
7.         arrayplaintext[i]=arrayciphertext[i];  
8.         continue;  
9.     }  
10.    c= getindex(arrayciphertext[i]);  
11.    k= getindex(arraykey[i]);  
12.    p= (c-k);  
13.    if(p<0) {  
14.        p=p+index.length();  
15.    }  
16.    else{  
17.        p=p%index.length();  
18.    }  
19.    arrayplaintext[i]=getcharacter(p);  
20. }  
21. String hasil = new String(arrayplaintext);  
22. return hasil;  
23. }

\[ T(n) = \sum C_i \#_i \]
\[ = 5C_1 + 7C_1n + C_2n + 2C_3n + C_4n + C_5 \]
\[ = (5C_1 + C_5)n^0 + (7C_1 + C_2 + 2C_3 + C_4)n^1 \]
\[ = \Theta(n) \]

From the above calculation, the result of complexity of Trithemius algorithm is \( \Theta (n) \).

**Table 3. Complexity of Double Transposition Cipher Encryption Method**

| No | Program Code | C | # | C# |
|----|--------------|---|---|----|
| 1. | public String enkripsi_double(String plain, String key1, String key2) {  
|    | String k1 = key1, k2 = key2, cipher1 = "", cipher2 = "";  
|    | Integer i1 = (int) Math.ceil(plain.length() * 1.0 /  
|    | k1.length()); | CLI | 1 | CLI |
| 2. | | CLI | 1 | CLI |
3. Integer j1 = k1.length();
4. char[][] tabel1 = new char[i1][j1];
5. int k = 0;
6. }
7. for (int i = 0; i < i1; i++)
8.     for (int j = 0; j < j1; j++) {
9.         if (k < plain.length())
10.             tabel1[i][j] = plain.charAt(k);
11.         else
12.             tabel1[i][j] = 'Ā';
13.         k++;
14. }
15. Integer[] runtutan = urutan(k1);
16. for (int i = 0; i < runtutan.length; i++)
17.     for (int j = 0; j < i1; j++)
18.         cipher1 += tabel1[j][runtutan[i]];
19. Integer i2 = (int) Math.ceil(cipher1.length() * 1.0 / k2.length());
20. char[][] tabel2 = new char[i2][j2];
21. k = 0;
22. for (int i = 0; i < i2; i++)
23.     for (int j = 0; j < j2; j++) {
24.         if (k < cipher1.length())
25.             tabel2[i][j] = cipher1.charAt(k);
26.         else
27.             tabel2[i][j] = 'Ī';
28.         k++;
29. }
30. runtutan = urutan(k2);
31. for (int i = 0; i < runtutan.length; i++)
32.     for (int j = 0; j < i2; j++)
33.         cipher2 += tabel2[j][runtutan[i]];
34. return cipher2;
35. }

\[ T(n) = \sum C_i \#_i \]

\[ = 11c_1 + 4c_2 n + 8c_1 n^2 + 4c_2 n^2 + 2c_3 n^2 + c_4 \]

\[ = (11c_1 + c_4)n^0 + 4c_2 n + + (8c_1 + 4c_2 + 2c_3) n^2 \]

\[ = \Theta(n^2) \]

From the above calculation, the result of complexity for Double Transposition Cipher is \( \Theta(n^2) \). Thus, it can be concluded that the complexity of the super-encryption method between the Trithemius algorithm and the Double Transposition Cipher is \( \Theta(n^2) \). It means, the length of plaintext and password proportional to linear processing time. The longer the plaintext and password the longer the time it takes.

4. Conclusion

- The process of encryption and decryption using a combination of Trithemius algorithm and Double Transposition Cipher in super encryption method, successfully done.
- The value of super encryption complexity between the two algorithms used is Trithemius and Double Transposition Cipher algorithm yield value \( \Theta(n^2) \)
- In the encryption process, the longer the plaintext and password the longer the time it takes. In other words, the length of the plaintext and the password is linearly proportional to time.

5. References

[1] Ariyus, Doni. 2008. *Pengantar ilmu Kriptografi, Teori, analisis dan Implementasi*. Yogyakarta: Penerbit Andi.

[2] Mollin, Richard A.2005. *Codes: the guide to secrecy from ancient to modern times*: United State of America.

[3] Salomon, David.2005. *Coding for Data and Computer Communications*: United State of America.

[4] Kurnanda, Rahman. 2016. *Perbandingan Algoritma 3DES dan Rinjdael Dalam Perancangan Aplikasi Keamanan Data*. Skripsi. Medan: Universitas Sumatera Utara.

Alghazzawi, Daniyal M., Hasan, Syed Hamid. & Trigui, Mohamed Salim. 2013. *Stream Ciphers: A Comparative Study of Attacks and Structures*. *International Journal of Computer Applications* 1 27-29.