Penggunaan Fibonacci dan Josephus Problem dalam Algoritma Enkripsi Transposisi + Substitusi

Gregorius Ronny Kaluge / 13508019
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ronny.kaluge@gmail.com, if18019@students.if.itb.ac.id, miliknya_ronny@yahoo.co.id

***Abstrak*—Deret Fibonacci adalah salah satu deret matematika yang terkenal. Deret ini cukup menarik untuk digunakan dalam berbagai bidang, salah satunya untuk enkripsi. Dengan bilangan Fibonacci, bisa dibuat algoritma enkripsi transposisi maupun substitusi yang baru. Namun, akan lebih baik jika algoritma enkripsi yang digunakan tidak hanya bertumpu pada Fibonacci. Oleh karena itu, sebaiknya ide dari deret Fibonacci digabungkan dengan ide-ide lain. Salah satu ide yang bisa digunakan adalah menggabungkannya dengan Josephus Problem, salah satu permasalahan yang juga cukup terkenal di bidang matematika dan komputer sains. Melalui kedua ide tersebut, penulis akan mencoba merancang sebuah algoritma enkripsi yang baru.**

***Kata kunci*— Fibonacci, Josephus Problem, enkripsi substitusi, enkripsi transposisi**

I. PENDAHULUAN

Kriptografi memiliki sejarah yang panjang. Sejak ribuan tahun silam, kriptografi telah digunakan. Sudah banyak orang yang mendalami cabang ilmu ini sampai dengan sekarang.

Kriptografi dapat digunakan dalam berbagai hal, terutama yang berhubungan dengan keamanan informasi. Di zaman modern ini, informasi adalah sesuatu yang berharga, bahkan bisa dibilang sebuah harta. Ada beberapa informasi yang bisa disebarluaskan sesuka hati, namun ada pula informasi rahasia yang sebaiknya hanya diketahui beberapa pihak. Untuk informasi yang bersifat rahasia, salah satu cara menyampaikannya ke pihak lain adalah dengan menggunakan enkripsi.

Algoritma kriptografi memiliki 2 kategori, yaitu cipher substitusi dan cipher transposisi. Cipher substitusi mengubah karakter-karakter pada pesan sedangkan cipher transposisi mengubah letak karakter-karakter pada pesan. Dalam perkembangannya, kedua hal ini sering digabungkan dalam 1 algoritma enkripsi agar cipherteks yang dihasilkan sulit dipecahkan.

Saat ini, orang-orang masih berlomba dalam menciptakan algoritma enkripsi yang baik. Ada banyak algoritma yang telah dihasilkan dengan berbagai ide yang menarik. Setelah melakukan studi pustaka, penulis mendapat ide yang cukup menarik dan ingin mencoba membuat algoritma enkripsi yang baru.

II. DERET FIBONACCI

Dalam ilmu matematika, deret Fibonacci didefinisikan secara rekursif sebagai :

$F_n = F_{n-1} + F_{n-2}$

dengan $F_0 = 0$ dan $F_1 = 1$

Deret Fibonacci pertama kali diperkenalkan oleh Leonardo dari Pisa, yang dikenal sebagai Fibonacci. Fibonacci menulis sebuah buku pada tahun 1202 berjudul Liber Abaci dan melalui buku tersebut memperkenalkan deret Fibonacci kepada matematikawan di Eropa. Meskipun begitu, deret tersebut sempat disebutkan di India dan belum diketahui secara pasti siapa yang pertama kali menemukannya.

Deret Fibonacci memiliki banyak variasi, salah satunya adalah dengan mengubah nilai F_0 dan F_1 untuk menghasilkan nilai deret yang berbeda.

III. JOSEPHUS PROBLEM

Dalam bidang komputer sains dan matematika, Josephus Problem adalah sebuah permasalahan teoretis yang berhubungan dengan permainan counting-out. Deskripsi permasalahannya adalah sebagai berikut :

Ada beberapa orang yang berdiri melingkar, menunggu untuk dieksekusi. Setelah orang pertama dieksekusi, beberapa orang (sebut saja N) di-skip dan setelah itu satu orang setelahnya dieksekusi lagi. Lalu skip sebanyak N orang, lalu eksekusi orang setelahnya. Proses ini dilanjutkan sampai tersisa 1 orang saja, yang akan dibebaskan.

Biasanya yang dicari adalah posisi seseorang pada saat awal sehingga bisa tersisa sampai akhir.

Problem ini dinamai Josephus sesuai dengan nama seorang sejarawan Yahudi pada abad pertama. Menurut catatan Josephus pada saat pengepungan di Yodfat, beliau dan 40 orang pejuang terjebak di sebuah goa. Jalan keluar goa sudah dijaga oleh serdadu Romawi. Mereka lebih memilih bunuh diri daripada ditangkap dan memutuskan untuk berdiri melingkar dan mulai bunuh diri, satu per satu dengan N=3. Josephus menyebutkan bahwa entah karena beruntung atau karena kehendak Tuhan (beberapa orang menunjukkan bahwa Josephus adalah orang yang cukup cerdas dan sudah memprediksi hasilnya), dia dan salah satu temannya tersisa sampai akhir dan memutuskan menyerah kepada orang Romawi.

IV. ALGORITMA ENKRIPSI

Algoritma enkripsi yang dirancang adalah sebagai berikut :

1. Diketahui nilai S, f_0, f_1, M, X, serta string plainteks. Keterangan :
   a. Pada Josephus Problem, S bisa dianalogikan sebagai posisi yang pertama kali dieksekusi.
   b. f_0 dan f_1 bisa dibilang nilai suku pertama dan kedua dari deret Fibonacci.
   c. M digunakan untuk membatasi pergeseran nilai S.
   d. X digunakan untuk membatasi nilai bilangan Fibonacci.
2. Masukkan panjang plainteks ke dalam variabel N. Jika S > N, ubah nilai S menjadi S mod N.
3. Ambil karakter ke-S dari plainteks (sehingga panjang string berkurang 1), masukkan ke variabel c.
4. Geser nilai c sebanyak f_0, pergeseran bisa dilakukan seperti *Extended Vigenere Cipher* (melibatkan 256 karakter) maupun seperti *Standard Vigenere Cipher* (hanya melibatkan alphabet).
5. Tambahkan c ke string hasil enkripsi.
6. Kurangi nilai N sebanyak 1, lalu ubah nilai S menjadi (S+ (f_0 mod M)) mod N.
7. Ubah nilai f_0 dan f_1 menjadi nilai bilangan Fibonacci berikutnya. Jika f_1 lebih dari X, modulus nilainya dengan X.
8. Ulangi langkah 3-5 , jika N bernilai 0 maka proses dihentikan, jika tidak lanjut ke langkah 6-8.

Untuk memperjelas algoritma ini, bisa digunakan contoh berikut :
Diberikan masukan S = 2, f_0 = 1, f_1 = 3, M = 9, string awal = “abcdef”
Pada awalnya:

- S = 2, f_0 = 1, f_1 = 3, M = 9, X = 40, N = 6
- string awal = “abcdef”
- string hasil = “”

Tahap pertama :

- S = 2, f_0 = 1, f_1 = 3, M = 9, X = 40, N = 5
- c = “c” + 1
- string awal = “abdef”
- string hasil = “d”

Tahap kedua :

- S = 3, f_0 = 3, f_1 = 4, M = 9, X = 40, N = 4
- c = “e” + 3
- string awal = “abdf”
- string hasil = “dh”

Tahap ketiga :

- S = 0, f_0 = 4, f_1 = 7, M = 9, X = 40, N = 3
- c = “d” + 4
- string awal = “abf”
- string hasil = “dhh”

Tahap keempat :

- S = 1, f_0 = 7, f_1 = 11, M = 9, X = 40, N = 2
- c = “a” + 7
- string awal = “bf”
- string hasil = “dhhh”

Tahap kelima :

- S = 0, f_0 = 11, f_1 = 18, M = 9, X = 40, N = 1
- c = “f” + 11
- string awal = “b”
- string hasil = “dhhhq”

Tahap keenam :

- S = 0, f_0 = 11, f_1 = 18, M = 9, X = 40, N = 0
- c = “a” + 7
- string awal = “”
- string hasil = “dhhhqt”

V. IMPLEMENTASI

Penulis telah mencoba mengimplementasikan algoritma tersebut dalam bentuk program C++. Penulis membuat 2 versi program dan keduanya akan ditampilkan di makalah ini. S*ource code* di bawah ini menggunakan algoritma enkripsi dengan pergeseran 256 karakter :

```
//Mode 256 karakter
#include <iostream>
using namespace std;

#define MOD 256
#define S 0

string Encrypt(int start, int f1, int f2, int M, int X, string s)
{
	int i, j, k, l, length, awal, akhir, Length, temp, f3;
	char c;
	string hasil = "";
	length = s.length();
	int next[length];

	start = ((start % length) + length) % length;
	for (i=1; i<length-1; ++i)
		next[i] = i+1;
	if (length>=2)
	{
		next[0] = 1;
		next[length-1] = 0;
	}
	else
		next[0] = 0;
	awal = 0;
	akhir = length-1;

	Length=length-1;
	for (i=0; i<length; ++i)
	{
		temp = start;
		if (i<length-1)
		{
			start = ((start + f1 % M) %
Length + Length) % Length;
```

```
                        Length=(length-i-2);
                }
                k = awal;
                for (j=0; j<temp; ++j)
                {
                        l = k;
                        k = next[k];
                }
                if (k==awal)
                {
                        next[akhir] = next[k];
                        awal = next[k];
                }
                else if (k==akhir)
                {
                        next[l] = awal;
                        akhir = l;
                }
                else
                        next[l] = next[k];
                c = ((s[k] - S + f1) % MOD ) % MOD + S;
                hasil += c;
                f3 = (f1 + f2) % X;
                f1 = f2;
                f2 = f3;
        }
        return hasil;
}

string Decrypt(int start, int f1, int f2, int M, int X, string s)
{
        int i, j, k, l, length, awal, akhir, Length, temp, f3;
        string hasil = "  ";
        length = s.length();
        int next[length];

        start = ((start % length) + length) % length;
        for (i=1; i<length-1; ++i)
        {
                next[i] = i+1;
                hasil += ' ';
        }
        if (length>=2)
        {
                next[0] = 1;
                next[length-1] = 0;
        }
        else
                next[0] = 0;
        awal = 0;
        akhir = length-1;

        Length=length-1;
        for (i=0; i<length; ++i)
        {
                temp = start;
                if (i<length-1)
                {
                        start = ((start + f1 % M) % Length + Length) % Length;
                        Length=(length-i-2);
                }
                k = awal;
                for (j=0; j<temp; ++j)
                {
                        l = k;
                        k = next[k];
                }
                if (k==awal)
                {
                        next[akhir] = next[k];
                        awal = next[k];
                }
                else if (k==akhir)
                {
                        next[l] = awal;
                        akhir = l;
                }
                else
                        next[l] = next[k];
                hasil[k] = ((s[i] - S - f1) % MOD ) % MOD + S;
                f3 = (f1 + f2) % X;
                f1 = f2;
                f2 = f3;
        }
        return hasil;
}

int main()
{
        int start, f1, f2, M, X;
        string s, s2;

        printf("Masukkan nilai awal Josephus : ");
        scanf("%d", &start);
        printf("Masukkan nilai 2 deret awal Fibonacci : ");
        scanf("%d %d", &f1, &f2);
        printf("Masukkan nilai M (untuk membatasi pergeseran S) : ");
        scanf("%d", &M);
        printf("Masukkan nilai X (untuk membatasi nilai Fibonacci) : ");
        scanf("%d", &X);
        printf("Masukkan string yang akan dienkripsi : ");
        cin >> s;
        printf("--------------------------------------------------\n");

        s2 = Encrypt(start, f1, f2, M, X, s);
        cout << "hasilnya(bukan dekripsi) : " << s2 << endl;
        cout << "hasilnya(bukan enkripsi) : " <<
```

```
Decrypt(start, f1, f2, M, X, s2) << endl;
	return 0;
}
```

Berikut ini adalah implementasi algoritma enkripsi dengan pergeseran 26 karakter :

```
//Mode 26 Huruf
#include <iostream>
using namespace std;

#define MOD 26

string Encrypt(int start, int f1, int f2, int M, int X, string s)
{
	int i, j, k, l, length, awal, akhir, Length, temp, f3;
	char c;
	string hasil = "";
	length = s.length();
	int next[length];

	start = ((start % length) + length) % length;
	for (i=1; i<length-1; ++i)
		next[i] = i+1;
	if (length>=2)
	{
		next[0] = 1;
		next[length-1] = 0;
	}
	else
		next[0] = 0;
	awal = 0;
	akhir = length-1;

	Length=length-1;
	for (i=0; i<length; ++i)
	{
		temp = start;
		if (i<length-1)
		{
			start = ((start + f1 % M) % Length + Length) % Length;
			Length=(length-i-2);
		}
		k = awal;
		for (j=0; j<temp; ++j)
		{
			l = k;
			k = next[k];
		}
		if (k==awal)
		{
			next[akhir] = next[k];
			awal = next[k];
		}
		else if (k==akhir)
		{
			next[l] = awal;
			akhir = l;
		}
		else
			next[l] = next[k];
		c = s[k];
		if (c>='A' && c<='Z')
			c = ((c - 65 + f1) % MOD + MOD) % MOD + 65;
		else if (c>='a' && c<='z')
			c = ((c - 97 + f1) % MOD + MOD) % MOD + 97;
		hasil += c;
		f3 = (f1 + f2) % X;
		f1 = f2;
		f2 = f3;
	}
	return hasil;
}

string Decrypt(int start, int f1, int f2, int M, int X, string s)
{
	int i, j, k, l, length, awal, akhir, Length, temp, f3;
	char c;
	string hasil = "  ";
	length = s.length();
	int next[length];

	start = ((start % length) + length) % length;
	for (i=1; i<length-1; ++i)
	{
		next[i] = i+1;
		hasil += ' ';
	}
	if (length>=2)
	{
		next[0] = 1;
		next[length-1] = 0;
	}
	else
		next[0] = 0;
	awal = 0;
	akhir = length-1;

	Length=length-1;
	for (i=0; i<length; ++i)
	{
		temp = start;
		if (i<length-1)
		{
			start = ((start + f1 % M) % Length + Length) % Length;
			Length=(length-i-2);
		}
		k = awal;
		for (j=0; j<temp; ++j)
		{
			l = k;
			k = next[k];
```

```
                }
                if (k==awal)
                {
                        next[akhir] = next[k];
                        awal = next[k];
                }
                else if (k==akhir)
                {
                        next[l] = awal;
                        akhir = l;
                }
                else
                        next[l] = next[k];
                c = s[i];
                if (c>='A' && c<='Z')
                        hasil[k] = ((c - 65 - f1) % MOD + MOD) % MOD + 65;
                else if (c>='a' && c<='z')
                        hasil[k] = ((c - 97 - f1) % MOD + MOD) % MOD + 97;
                else
                        hasil[k] = c;
                f3 = (f1 + f2) % X;
                f1 = f2;
                f2 = f3;
        }
        return hasil;
}

int main()
{
        int start, f1, f2, M, X;
        string s, s2;
        printf("Masukkan nilai awal Josephus : ");
        scanf("%d", &start);
        printf("Masukkan nilai 2 deret awal Fibonacci : ");
        scanf("%d %d", &f1, &f2);
        printf("Masukkan nilai M (untuk membatasi pergeseran S) : ");
        scanf("%d", &M);
        printf("Masukkan nilai X (untuk membatasi nilai Fibonacci) : ");
        scanf("%d", &X);
        printf("Masukkan string yang akan dienkripsi : ");
        cin >> s;
        printf("--------------------------------------------------\n");

        s2 = Encrypt(start, f1, f2, M, X, s);
        cout << "hasilnya(bukan dekripsi) : " << s2 << endl;
        cout << "hasilnya(bukan enkripsi) : " << Decrypt(start, f1, f2, M, X, s2) << endl;
        return 0;
}
```

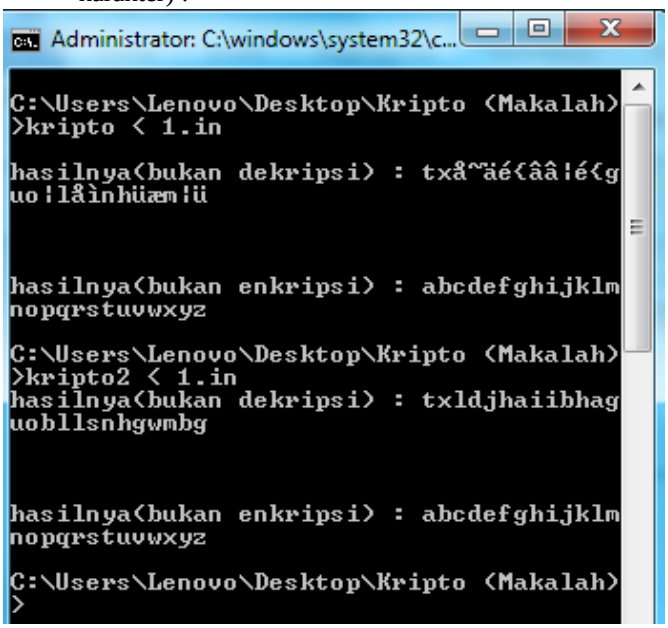
VI. PENGUJIAN

Pada bab pengujian ini, penulis akan menguji kedua source code yang telah dibuat tersebut.

1. Pengujian pertama:
   S = 9
   $f_0 = 10$, $f_1 = 3$
   M = 20
   X = 30
   Plainteks :

abcdefghijklmnopqrstuvwxyz

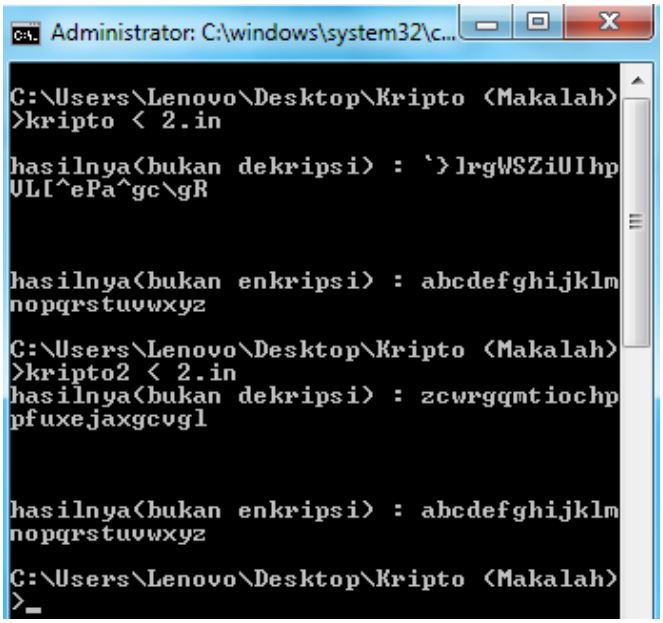
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



2. Pengujian kedua:
   S = 9
   $f_0 = -10$, $f_1 = 3$
   M = 20
   X = 30
   Plainteks :

abcdefghijklmnopqrstuvwxyz

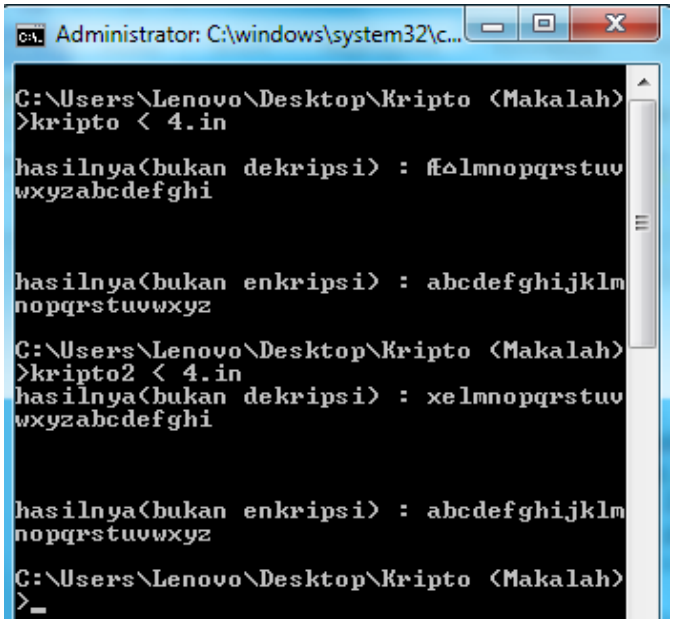
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



3. Pengujian ketiga:
   S = 9
   $f_0 = 10$, $f_1 = 3$
   M = 20
   X = 13
   Plainteks :

abcdefghijklmnopqrstuvwxyz

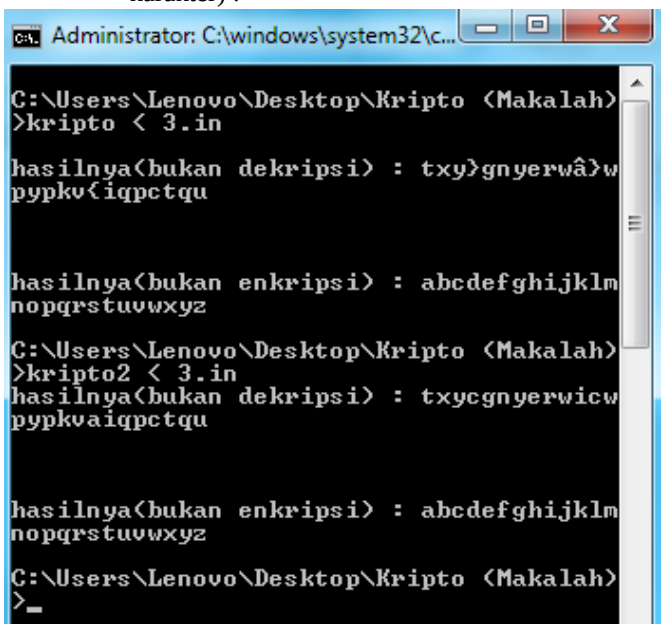
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



5. Pengujian kelima:
   S = 9
   $f_0 = 10$, $f_1 = 3$
   M = 20
   X = 30
   Plainteks :

4bcdefghijklmnopqrstuvwxyz

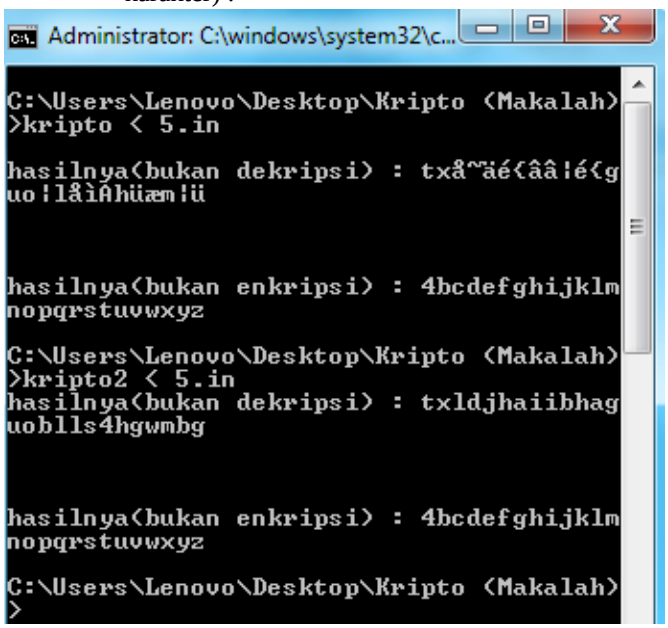
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



4. Pengujian keempat:
   S = 9
   $f_0 = 40$, $f_1 = 20$
   M = 20
   X = 20
   Plainteks :

abcdefghijklmnopqrstuvwxyz

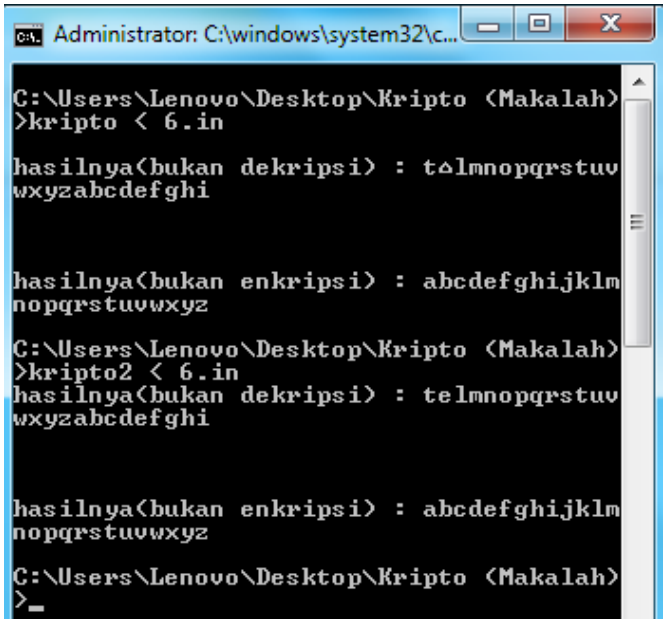
   ipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



6. Pengujian keenam:
   S = 9
   $f_0 = 10$, $f_1 = 20$
   M = 10
   X = 10
   Plainteks :

4bcdefghijklmnopqrstuvwxyz

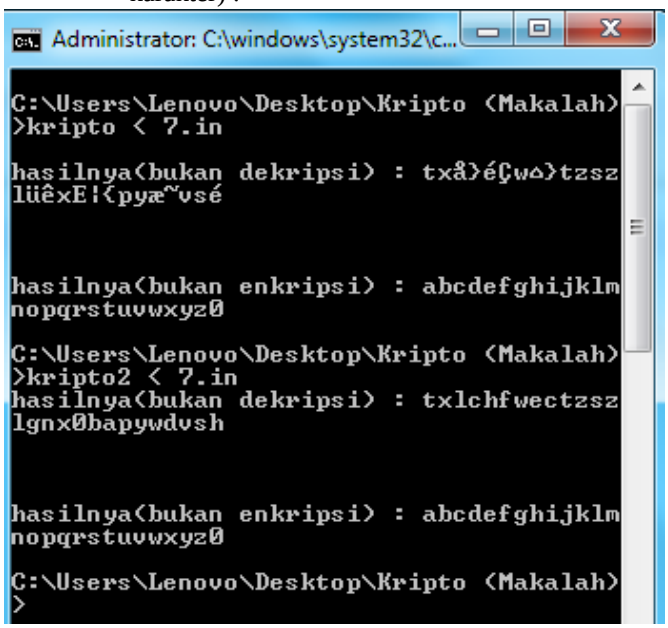
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



7. Pengujian ketujuh:
   S = 9
   f_0 = 10, f_1 = 3
   M = 20
   X = 30
   Plainteks :

abcdefghijklmnopqrstuvwxyz1

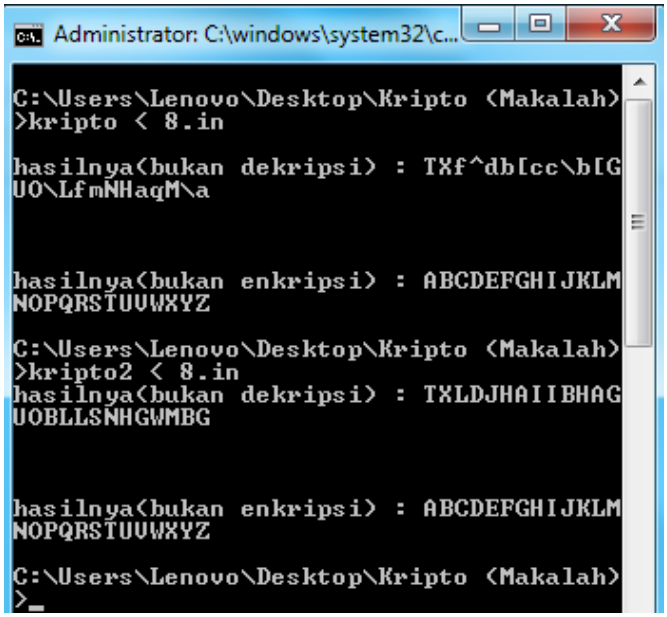
   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



8. Pengujian kedelapan:
   S = 9
   f_0 = 10, f_1 = 3
   M = 20
   X = 30
   Plainteks :

ABCDEFGHIJKLMNOPQRSTUVWXYZ

   Cipherteks (dengan mode pergeseran 256 karakter dan dengan mode pergeseran 26 karakter) :



VII. ANALISIS

A. Analisis Kompleksitas

Berdasarkan deskripsi algoritma pada bab 4, dapat dilihat bahwa kompleksitas algoritma ini adalah $O(N^2)$. Jika diamati, proses pengambilan sekaligus penghapusan elemen ke-S dari sebuah string membutuhkan O(N) operasi(pada kasus terburuknya). Proses tersebut dilakukan sebanyak N-1 kali sehingga kemungkinan terburuk algoritma ini adalah O(NxN) = $O(N^2)$.

Pada contoh source code pada bab 5, bisa dilihat juga bahwa pada saat diimplementasikan, algoritma ini juga memiliki kompleksitas $O(N^2)$, namun untuk alasan yang berbeda. Pada implementasi, penulis menggunakan array next untuk menyimpan indeks setelah posisi sekarang. Awalnya, next[i] = (i+1)mod N, namun saat elemen ke-X dihapus, next[X-1] berubah menjadi next[X]. Oleh karena itu, proses "penghapusan" (karena tidak benar-benar menghapus karakter dari string) hanya membutuhkan O(1) operasi. Yang menjadikan N^2 adalah operasi pencarian indeks yang akan dihapus. Jika yang dihapus adalah elemen terakhir, maka program akan melakukan iterasi sebanyak N-1 kali sebelum melakukan "penghapusan" pada indeks yang dituju.

B. Analisis Keamanan

Dari 8 pengujian yang sudah dilakukan, ada banyak hal yang bisa diamati dari algoritma ini. Berikut adalah hal-hal yang teramati oleh penulis :

1. Pada pengujian 1, 2, 3, dan 7, penulis hanya mengubah salah satu variabel input. Namun perubahan yang terjadi pada cipherteks cukup drastis. Hanya 2 karakter pertama pada cipherteks yang sama pada keempat cipherteks tersebut.
2. Pada pengujian 4 dan 6, dapat terlihat dengan jelas bahwa selain 2 karakter pertama pada

cipherteks, karakter lainnya adalah hasil pergeseran posisi dari plainteks.

3. Pada pengujian 1 dan 5, hanya terdapat 1 perbedaan karakter pada plainteks, yaitu karakter pertama plainteks. Pada cipherteks 1 dan 5, hanya ada 1 karakter yang berbeda, yaitu karakter pada posisi 20.
4. Pada pengujian 1 dan 8, hanya terdapat perbedaan plainteks yaitu pada pengujian 1, seluruh plainteks berupa huruf kecil dan pada pengujian 8, seluruh plainteks berupa huruf kapital. Hasil enkripsi dengan pergeseran 26 karakter menunjukkan hasil yang sama jika dieja, namun hasil enkripsi dengan pergeseran 256 karakter menunjukkan hasil yang sangat berbeda.

VIII. KESIMPULAN DAN SARAN

Dari analisis yang telah dilakukan, ada beberapa kesimpulan yang diperoleh :

1. Sampai saat ini, penulis belum menemukan cara implementasi algoritma ini agar kompleksitasnya bisa lebih baik daripada $O(N^2)$.
2. Secara umum, algoritma ini cukup baik karena dengan sedikit perubahan pada nilai variabelnya, cipherteks yang dihasilkan bisa berbeda jauh. Selain itu, jika plainteks ditambah atau dikurangi jumlah karakternya, cipherteks yang dihasilkan juga dapat berbeda jauh.
3. Jika nilai X = nilai terkecil di antara f_0 dan f_1, dan nilai f_0 merupakan kelipatan dari f_1 atau nilai f_1 merupakan kelipatan dari f_0 maka cipherteks yang dihasilkan kurang kuat karena sebagian besar karakternya hanya mengalami pergeseran posisi.
4. Enkripsi yang dilakukan pada suatu karakter tidak berpengaruh pada enkripsi yang dilakukan pada karakter selanjutnya. Oleh karena itu, pengubahan 1 karakter pada plainteks akan menghasilkan cipherteks yang hampir sama.
5. Pergeseran 256 karakter lebih sulit untuk dianalisis daripada pergeseran 26 karakter.

Saran dari penulis bagi yang ingin menggunakan algoritma ini :

1. Gunakan nilai X yang tidak memenuhi kesimpulan nomor 3 agar cipherteks yang dihasilkan sulit dipecahkan.
2. Karena kompleksitas algoritma ini adalah $O(N^2)$, sebelum mengenkripsi pesan yang berukuran besar akan lebih baik jika pesan tersebut dipotong-potong menjadi beberapa bagian untuk dienkripsi secara independen. Jika pesan berukuran 10.000 karakter dan pesan tersebut dipecah menjadi 200 bagian, maka dengan perhitungan sederhana diperoleh total operasinya = $(10.000/200)^2 * 200 * k + c = 500.000 k + c$. Jika pesan tersebut tidak dipecah, maka total operasinya adalah $(10.000)^2 * k + c = 100.000.000 k + c$.

DAFTAR PUSTAKA

http://en.wikipedia.org/wiki/Josephus_problem
http://en.wikipedia.org/wiki/Fibonacci_number

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Maret 2011

ttd

Gregorius Ronny Kaluge / 13508019