Aplikasi Lagged Fibonacci Generator dan Bilangan Irrasional dalam Stream Cipher

Ismail Sunni 13508064
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ismailsunni@yahoo.co.id

Pada metode enkripsi dengan stream cipher, kekuatan enkripsi dipengaruhi oleh keystream. Keystream adalah bit-bit kunci yang digunakan untuk melakukan enkripsi maupun dekripsi pada stream cipher. Key stream ini dibangkitkan oleh keystream generator. Dengan melakukan operasi bit antara plainteksi dan keystream inilah, akan diperoleh cipherteks. Jadi, keamanan sistem cipher ini sangat bergantung pada keystream generator.

Semakin acak atau tidak periodik keystream yang digunakan, maka semakin sulit pula untuk dipecahkan. Seandainya, keystream yang digunakan truly random, maka tidak akan bisa dipecahkan, seperti pada one-time pad. Namun, ada masalah, yakni dengan onetime pad, maka kita harus menggunakan kunci yang memiliki panjang sama dengan plainteks. Hal ini tentu tidak efektif. Oleh karenanya, makalah ini akan membahas mengenai pemanfaatan bilangan irrational dan deret rekursif fibbonaci untuk "membangkitkan " bilangan yang acak tersebut.

Hal ini dimotivasi oleh sifat bilangan irrational yang tidak memiliki periode atau tidak periodik. Sehingga, dengan manipulasi sederhana, bisa digenerate sebuah keystream yang tidak periodik. Namun, memiliki kelemahan dalam hal struktur data.

Sedangkan bilangan pada deret fibonacci yang juga tidak memiliki periodik, akan dimanipulasi juga dengan metode yang sedikit berbeda, dan memanfaatkan sifat rekursivitas yang dimilikinya, untuk membangkitkan keystream yang memiliki periode yang cukup besar. Namun, memiliki kemudahan dalam menggeneratenya.

***Index Terms*—irrational numbers, stream cipher, fibonacci, Lagged Fibonacci Generator, one-time pad**

I. LATAR BELAKANG

Stream cipher merupakan salah satu jenis algoritma simetri modern. Stream cipher dapat didesain dengan sangat cepat, jauh lebih cepat daripada block cipher. Block cipher beroperasi pada sebuah blok data yang besar, tetapi stream cipher beroperasi pada ukuran yang lebih kecil, biasanya bit.

Stream cipher menggenerate apa yang disebut dengan keystream abuah deret bit yang dipergunakan sebagai key. Enkripsi dilakukan dengan cara mengkombinasikan keystream tersebut dengan plainteks, biasanya dengan operasi bit XOR.

Ketertarikan pada stream cipher pada masa sekarang adalah mengenai one-time pad atau kadang-kadang disebut dengan Vernam cipher. One-time pad menggunakan sebuah string yang memiliki panjang yang sama dengan plaintext dan digenerate dengan sangat random. Selanjutnya, dikombinasikan dengan plaintext menggunakan operasi bit-wise XOR, dan menghasilkan cipher text. Karena, keystream yang digunakan benar-benar acak, maka kriptanalis dengan kekuatan komputasiyang luar biasa pun hanya bisa menebak isi dari cipher text tersebut.

Karena memiliki keamanan yang sempurna inilah, analisis mengenai one-time pad ini menjadi konsentrasi ilmuwan masa kini.

Tetapi, ada masalah yang juga hadir di samping keamanan yang sempurna ini, yakni hanya bisa digunakan sekali dan fakta bahwa keystream ini memiliki panjang yang sama dengan plainteks. Selain itu, manajemen keystream ini juga menjadi masalah. Oleh karenanya, one-time pad ini tidak digunakan pada umumnya.

II. DASAR TEORI

Stream cipher

Stream cipher merupakan salah satu jenis algoritma modern. Algoritma ini menenkripsi satu per satu karakter atau biasanya per bit dari *plain text* dalam satu waktu, dan menggunakan transfomasi enkripsi yang berbeda setiap waktunya. *Stream cipher* biasanya lebih cepat daripada saudaranya, *block cipher*, dalam implementasi *hardware* dan lebih sederhana sirkuit *hardware*-nya. Algoritma ini juga memiliki keunggulan di beberapa kasus, misalnya di beberapa perangkat telekomunikasi, ketika *buffering* dibatasi dan ketika tiap karakter harus diproses secara terpisah pada saat diterima. Selain itu, karena *streamcipher* memiliki error yang tidak merambat atau terbatas, maka algoritma ini juga memiliki keunggulan di situasi dimana kesalahan transmisi mempunyai resiko tinggi.

Untuk lebih jelasnya, perhatikan beberapa definisi berikut :
[D] Misalkan $\mathcal{K}$adalah ruang kunci utnuk himpunan transformasi enkripsi. Barisan simbol $e_1 e_2 e_3 \dots e_{i-1} e_i \in \mathcal{K}$, disebut *keystream*.

[D] misalkan $\mathcal{A}$ adalah alfabet dari q simbol dan misalkan pula E_e adalah subtitusi cipher sederhana dengan panjang blok 1 dimana $e \in \mathcal{K}$. Misalkan $m_1 m_2 m_3 \ldots$ adalah string *plaintext* dan misalkan $e_1 e_2 e_3 \ldots$ adalah *keystream* dari $\mathcal{K}$. Sebuah *streamcipher* mengambil string *plaintext* dan menghasilkan string *ciphertext* $c_1 c_2 c_3 \ldots$ dimana $c_1 = E_{e_i}(m_i)$. Jika d_i menyatakan inverse dari e_i, maka$m_1 = D_{d_i}(c_i)$ mendekripsikan *ciphertext*-nya.

Stream cipher menggunakan transformasi enkripsi sederhana berdasarkan *keystream* yang sedang digunakan. *Keystream* yang digunakan bisa digenerate secara random atau dengan menggunakan algoritma yang menggenerate *keystream* dari *keystream* mula-mula (atau biasa disebut *seed*), atau dari *seed* dan karakter *ciphertext* sebelumnya. Algoritma seperti ini biasa disebut *keystream generator*.

[D] *Vernam Cipher* adalah *stream cipher* yang didefinisikan pada alfabet $\mathcal{A} = \{0,1\}$. Sebuah pesan binary $m_1 m_2 m_3 \ldots m_t$ dioperasikan dengan *keystring* binary $k_1 k_2 k_3 \ldots k_t$ dengan panjang yang sama dan menghasilkan $c_1 c_2 c_3 \ldots c_t$ dimana $c_i = m_i \oplus k_i, 1 \leq i \leq t$.

Algoritma Ideal : One-time pad

Ada satu buah algoritma yang dinyatakan paling ideal dan tidak mungkin dipecahkan, kecuali tahu kuncinya. Algoritma tersebut adalah One-time pad. Definisi secara umum One-time pad adalah sebagai berikut :

[D] sebuah algoritma enkripsi simetri yang memiliki keamanan sempurna yang menggunakan kunci acak. Kunci memiliki panjang yang sama dengan plaintext.

Kunci acak di sini merupakan "kunci" dari kesempurnaan algoritma one-time pad. Hal ini dikarenakan, dengan acak-nya kunci, maka ada kemungkinan menghasilkan plaintext yang bermakna walau dengan kunci yang berbeda serta plainttext tersebut, bukan pliantext asli. Misal :

plaintext	S	E	N	I	N	P	A	G	I
Key(asli)	A	R	G	D	K	P	B	Y	Y
cipher	S	V	T	L	X	E	B	E	G
Key(palsu)	J	B	H	L	E	M	N	N	C
Plaintext(palsu)	J	U	M	A	T	S	O	R	E

Bisa dilihat, pada tabel di atas, plain text SENIN SORE yang dienkripsi dengan kunci ARGDKPBYY akan menghasilkan cipher SVTLXEBEG. Seandainya seorang kriptanalis mencoba untuk mendekripsi cipher tersebut dan memasukan kunci JBHLEMNNC, maka akan menghasilkan plain teks JUMAT SORE, yang juga bermakna. Di sini, seorang kriptanalis tidak lagi mempunyai modal untuk mendekripsikan lebih lanjut.

Namun, one-time pad memiliki kelemahan yang sangat besar, yakni mengenai kerahasiaan kunci. Karena kunci ini terdiri atas elemen-elemen yang random, maka kunci ini tidak bisa digenerate secara terpisah. Dengan kata lain, kunci ini harus dikirimkan juga ke penerima pesan. Sehingga cost untuk melakukan pengiriman menjadi 2 kali lipat.

Seandainya biaya bukan masalah, ada masalah yang lebih fundamental. Yakni bagaimana merahasiakan kunci tersebut. Pengrahasiaan di sini termasuk dalam menyimpan dan mengirimkannya. Hal ini tentu rentan. Kita tidak seharusnya mengirim pesan sekaligus kunci untuk membukannya, karena sanga beresiko disadap.

Seandainya tidak dikirim, maka penerima tidak akan bisa menggenerate kunci tersebut, karena bersifat random.

Sehingga, one-time pad tidak banyak diimplementasikan. Namun, one-time pad bisa dijadikan model algoritma yang ideal dari segi keamanannya. Dan kita cukup mendekati one-time pad dengan metode atau algoritma lainnya.

Bilangan Irrasional

[D] *bilangan irrational* adalah bilangan yang tidak dapat direpresentasikan sebagai bentuk desimal berhingga atau sebagai pecahan $\frac{m}{n}$ dimana, $GCD(m,n) = 1;\ m,n \in Z$.

Seperti definisi di atas, bilangan seperti *pi* $\pi = 3{,}14 \ldots$ termasuk bilangan irrasional karena memiliki banyak digit tak hingga yang tidak memiliki periode. Namun, bilangan seperti $0.3333 \ldots$ tida termasuk bilangan-bilangan irrational karena memiliki digit tak hingga yang periodi, selain bisa direpresentasikan ke dalam pecahan, yakni $\frac{1}{3}$.

Barisan Rekursif

[D] barisan rekursif $\{f(n)\}_n$, atau bisa disebut juga barisan rekurens, adalah bariasn bilangan $f(n)$ yang memiliki index bilangan bulat n dan digenerate oleh persamaan rekurens.

Bilangan Fibonacci

[D]bilangan fibonacci adalah barisan bilangan $\{F_n\}_{n=1}^{\infty}$ yang didefinisikan sebagai persamaan rekurens linier

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

Dimana $F_1 = F_2 = 1$. Dan selanjutnya, bisa diturunkan bahwa $F_0 = 0$.

Bilangan fibonacci untuk $n = 1,2, \ldots$ adalah 1,1,2,3,5...

Lagged Fibonacci Generator

[D] S_n adalah Lagged Fibonacci Generator, dimana $S_n \equiv S_{n-j} \blacksquare S_{n-k} (mod\ m)$, dimana $0 < j < k$.

Bentuk ini merupakan generalisasi dari barisan Fibonacci. Dimana kotak hitam bisa diganti dengan operater binary apapun, misalnya penjumlahan, pengurangan, atau bahkan XOR. Dan, biasanya, m merupakan bilangan yang berbentuk 2^n.

III. Aplikasi Stream Cipher

Aplikasi Bilangan Irrasional di Cipher Stream

Pada bagian ini, akan dilakukan pendekatan one-time pad dengan memanfaatkan bilangan irrasional. Seperti yang sudah dijelaskan sebelumnya, bilangan irrasional memiliki digit desimal yang banyaknnya tak hingga dan tidak memiliki periode. Sifat ini, bisa dimanfaatkan untuk menggenerate kunci yang dipergunakan untuk mengenkripsi plaintext di streamcipher.

Algoritma enkripsi stream cipher dengan bilangan irrasional, di level bit :

1. Masukan plain teks, dapatkan panjangnya.
2. Masukan kunci, yang berupa bilangan irrasional
3. Generate representasi bilangan irrasional tersebut dalam desimal
4. Ambil bagian desimal (digit di belakang koma) sepanjang plain teks
5. Ubah ke dalam representasi bit
6. Lakukan operasi XOR antara bit ke-i pada plainteks dan bit ke-i pada key
7. Cipher text telah dihasilkan

Catatan :

1. Pengubahan dari representasi desimal ke bit dilakukan dengan metode sederhana. Yakni mengubah tiap digit menjadi bilangan basis 2. Misal 8 = 1001, 9 = 1001, dan seterusnya.

Di sini, kita tidak perlu menggenerate secara langsung keseluruhan key. Kita bisa melakukan pekerjaan tersebut secara paralel. Pekerjaan tersebut adalah :

1. Menggenerate representasi desimal bilangan irrasional
2. Mengubah representasi desimal key ke dalam representasi bit
3. Melakukan operasi XOR
4. Mengirim pesan yang sudah terenkripsi (untuk pengirim) atau menerima pesan yang sudah terenkripsi (untuk penerima)

Jadi, di sini bisa dilihat, sifat stream cipher masih dipertahankan. Yakni kemengaliran data atau plainteks.
Pen-generate-an key, bisa dilakukan dengan ilmu matematika numerik. Misal saja seperti pada algoritma berikut ini :

1. Masukan bilangan bulat sembarang non kuadrat sempurna
2. Cari akar dari bilagan tersebut dengan menggunakan metode pencarian akar, misal metode Newton.

Untuk melakukan dekripsi, cukup membalik alogritma enkripsi di atas, karena stream cipher merupakan algoritma simetris.

Analisis Penggunaan Bilangan Irrasional di Stream Cipher

Setelah dijelaskan penggunaan bilagan irrasional pada stream cipher, pada bagian ini akan dibahas mengenai penggunaan tersebut.

1. Key
   Key yang menjadi pilihan bagi pengguna ada tak hingga banyaknya. Hal ini dkarenakan bahwa ada tak hingga buah bilangan bulat non kuadrat sempurna. Sehingga, ada tak hingga buah pula bilangan irrational yang bisa digenerate. Itu baru satu metode, yakni pencarian akar. Metode lain misalnya :
   a. Penggunaan akar pangkat n, misal akar pangkat 3, 4, dst
   b. Penggunaan fungsi logaritma, misal log key
   c. Operasi pada bilangan irrational, misal kali, tambah, dan seterusnya.
   d. Manipulasi pada *well-known* irrational-number, seperti *e*(bilangan euler), *phi* (rasio emas), *pi* (perbandingan keliling dan diameter lingkaran)
   e. Penggunaan fungsi matematika lainnya, misal sinus, kosinus, dan lainnya.
2. Pengiriman key
   Karena menggunakan key-generator, maka yang perlu dikirmkan adalah kunci untuk membangkitkan key tersebut. Dan kalau perlu, algoritma untuk membangkitkannya.
3. Efisiensi
   Efisiensi yang ditawarkan metode ini, sama seperti algoritma stream cipher lainnya. Karena, pekerjaan bisa dilakukan secara paralel.
4. Keamanan
   Keamanan yang ditawarkan metode ini cukup menjanjikan. Tanpa tahu algoritmanya, maka ada kemungkinan terlihat seperti one-time pad. Karena, key yang dibangkitkan memiliki panjang yang sama dengan plaintext.
5. Pemecahan dengan *brute-force*
   Pemecahan dengan *burte-force* hampir tidak mungkin dilakukan. Karena, ada sebanyak 2^n kemungkinan, dimana *n* adalah panjang plaintext dalam mode bit jika ingin melakukan *burte force* pada mode bit. Jika mencoba mencoba-coba *brute-force* dalam pencarian kunci, semakin tidak mungkin lagi. Ada tak hingga banyaknya bilangan irrational.
6. Kunci mudah diingat.
   Pengguna tidak perlu mengahafalkan representasi desimal dari bilangan irrational. Tetapi, cukup mengingat representasi yang lebih singkat dari bilangan irrational tetrsebut, misal $\sqrt{7}, e^2$, atau yang lainnya.
7. Pembangkitan key
   Pembangkitan key memakan resource yang cukup besar. Bahkan, perlu mendefinisikan sendiri struktur datanya.

Aplikasi Barisan Fibonacci pada Cipher Stream

Setelah melihat penggunaan bilangan irrational pada cipher stream, pada bagian ini akan diperlihatkan penggunaan barisan Fibonacci pada strream cipher. Yang akan dibahas adalah Lagged Fibonacci Generator (LFG), karena lebih umum.

$S_n \equiv S_{n-j} \blacksquare S_{n-k} (mod\ m)$

Berikut, algoritma secara umumnya :

1. Masukan k buah bilangan sebagai basis.
2. Bangkitkan suku-suku berikutnyadari barisan fibonacci
3. Ubah masing-masing suku menjadi bit, dengan panjang m.

Sementara algoritma untuk melakukan enkripsi sebuah pesan adalah sebagai berikut :

1. Masukan plaintext
2. Generate key
3. Lakukan operasi XOR antara bit pada plaintext dan key pada index yang bersesuaian

Sama seperti pada pemanfaatan bilangan irrasional, pada pemnafaatan LFG ini juga ada beberapa pekerjaan yang bisa dilakukan secara parale, yakni :

1. Menggenerate key
2. Mengubah representasi desimal key ke dalam representasi bit
3. Melakukan operasi XOR
4. Mengirim pesan yang sudah terenkripsi (untuk pengirim) atau menerima pesan yang sudah terenkripsi (untuk penerima)

Untuk melakukan dekripsi, cukup membalikkan algoritma diatas, karena stream cipher merupakan algoritma enkripsi simetris.

Analisis Penggunaan Barisan Fibonacci di Stream Cipher

Pada bagian ini, akan dibahas mengenai pemanfaatan barisan Fibonacci pada stream cipher :

1. Key
   Key yang menjadi pilihan bagi pengguna tidak sebanyak bilangan irrasional. Namun, sudah cukup besar untuk memusingkan kriptanalus. Jadi, pendeketan ke one-time pad tidak sedekat bilangan irrational.
2. Pengiriman key
   Sama seperti pada bilangan irrasional, tidak perlu mengirim sejumlah besar key, seperti pada one-time pad. cukup mengunakan key generato saja.
3. Efisiensi
   Efisiensi yang ditawarkan metode ini, sama seperti algoritma stream cipher lainnya. Karena, pekerjaan bisa dilakukan secara paralel.
4. Keamanan
   Keamanan yang ditawarkan oleh LFG tidak sehebat bilangan irrational. Hal ini dikarenakan, masih muncul periode ketika menggunakan beberapa pasangan j dan k. Tetapi, jika menggunakan operasi XOR, LFG bisa memiliki periode sebanyak : $(2^k - 1)k$. Namun, besar periode ini bergantung pada pasangan j dan k seperti yang telah disebutkan sebelumnya. Untuk suatu j dan k yang kecil, maka makin kecil pula periodenya. Beberapa pasangan j dan k yang sudah dianggap populer misalnya $(5,7), (5,17), (24,55), (65,71)$ dan lainnya. Selain itu, ada juga yang mengatakan bahwa lebih baik menggunakan pasangan j dan k yang mempunya rasio mendekati golden rasio
5. Pemecahan dengan *brute-force*
   Karena masih memiliki periode, maka ada kemungkian bisa dipecahkan. Tapi, seiring makin besarnya periode, makin susah pula untuk dipecahkan.
6. Kunci mudah diingat.
   Jika diharuskan untuk tidak mencatat key, penggunaan LFG ini sulit dilakukan. Hal ini dikarenakan susahnya mengingat basis LFG yang terdiri dari bilangan sebanyak k.
7. Pembangkitan key
   Pembangkitan key tidak terlalu memakan resource yang cukup besar. Hal ini dikarenakan, LFG bisa dibangkitkan dengan mudah secara rekursif.

IV. IMPLEMENTASI

Pada bagian ini, penulis akan mengimplementasikan LFG dan bilangan irrasional pada stream cipher. Impelementasi menggunakan bahasa C#, dengan kakas Visual Studio 2010, dan antarmuka berupa console.

Implemetasi Bilangan Irrasional

Pada implementasi ini, akan digunakan fungsi sederhana yakni mencari akar dari bilangan bulat. Namun, karena hanya prototipe, maka struktur data digunakan adalah strukur data default, yakni Int64 pada bahasa C#. Antar muka yang dipergunakan pun hanya console. Dan inputan yang dibolehkan hanya string. Dan dioperasikan tidak per-bit namun per byte, karena akan menghasilkan hasil yang sama ketika dipergunakan fungsi XOR.

Berikut kutipan kode penting yang dipergunakan :

Fungsi-fungsi penting

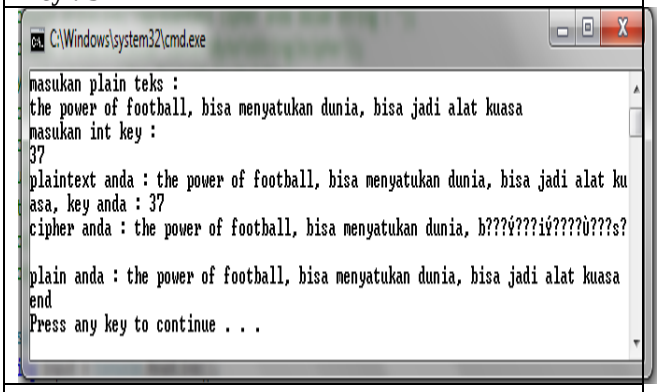
```
public static byte[] getRoot(Int64 a, Int64 length)
        {
            byte[] retval = new byte[length];
            double dRoot = Math.Sqrt(a);
            Int64 intRoot = (Int64)(dRoot * (Math.Pow(10, length)));
            Int64 temp;
            for (int i = 0; i < length; ++i)
            {
                temp = intRoot % 10;
                retval[length - i - 1] = (byte)temp;
                intRoot = intRoot / 10;
            }
            return retval;

        }

        public static byte[] encryptRoot(byte[] plainByte, int key)
        {
            byte[] byteRoot = getRoot(key, plainByte.Length);
            byte[] retval = new byte[plainByte.Length];
```

```
            for (int i = 0; i < plainByte.Length; ++i)
            {
                retval[i] = (byte)(byteRoot[i] ^ plainByte[i]);
            }

            return retval;
        }

        public static byte[] decryptRoot(byte[] cipherByte, int key)
        {
            byte[] byteRoot = getRoot(key, cipherByte.Length);
            byte[] retval = new byte[cipherByte.Length];
            for (int i = 0; i < cipherByte.Length; ++i)
            {
                retval[i] = (byte)(byteRoot[i] ^ cipherByte[i]);
            }

            return retval;
        }
```

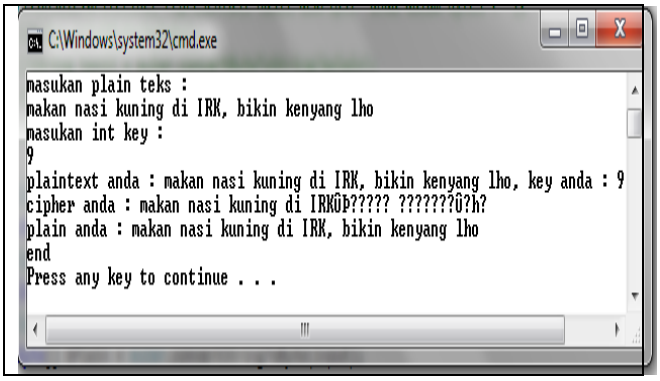
Sedangkan percobaan program, dilakukan sebanyak 2 kali, sebagai berikut :

Pada percobaan pertama, dimasukan kunci bilangan non-kuadrat sempurna. Sehingga menghasilkan cipher yang sempurna. Namun, ketika dimasukan key yang merupakan bilangan kuadrat sempurna, hanya sebagian saja yang ternkripsi. Karena, memiliki bagian periodik, yakni 0.

Input : the power of footbal, bisa menyatukan dunia, bisa jadi alat kuasa
Key : 37



Input : makan nasi kuning di IRK, bikin kenyang lho
Key : 9
Contoh ketika bilangan kuadrat yang dimasukan sebagai key



Implemetasi Lagged Fibonacci Generator

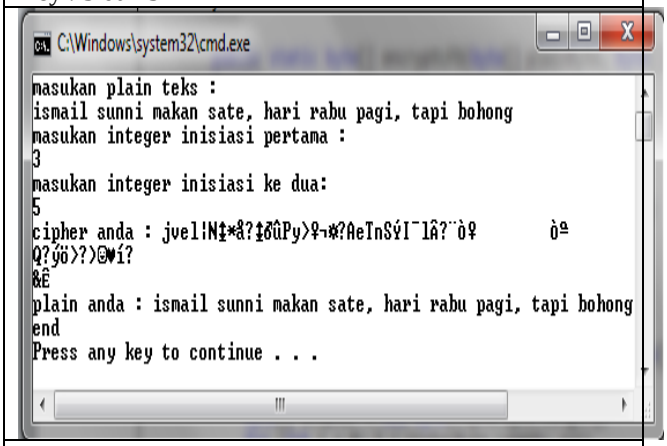
Untuk implementasi LFG, di sini digunakan j dan k statik, yakni 1 dan 2. Sehingga cukup memasukan 2 buah integer sebagai ke nya.

Fungsi-fungsi penting

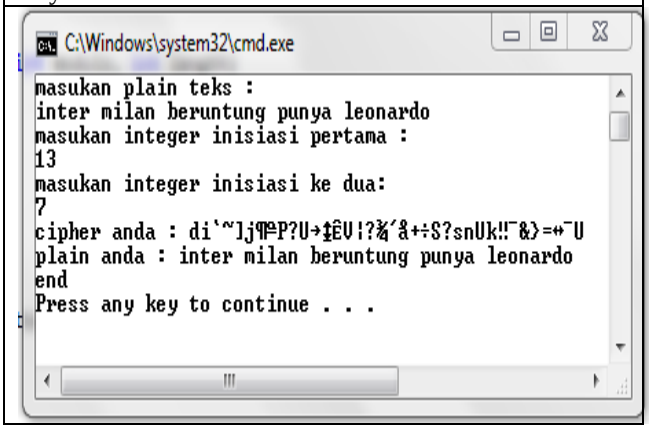
```
public static byte[] getByteLFG(int a, int b, int modulo, int length)
        {
            byte[] retval = new byte[length];
            retval[0] = (byte)a;
            retval[1] = (byte)b;

            for (int i = 2; i < length; ++i)
            {
                retval[i] = (byte)((retval[i - 1] + retval[i - 2]) % modulo);
            }

            return retval;
        }

        public static byte[] encryptLFG(byte[] plainByte, byte[] keyByte)
        {
            byte[] bCipher = new byte[plainByte.Length];
            for (int i = 0; i < plainByte.Length; ++i)
            {
                bCipher[i] = (byte)(plainByte[i] ^ keyByte[i]);
            }
            return bCipher;
        }

        public static byte[] decryptLFG(byte[] cipherByte, byte[] keyByte)
        {
            byte[] bCipher = new byte[cipherByte.Length];
            for (int i = 0; i < cipherByte.Length; ++i)
            {
                bCipher[i] = (byte)(cipherByte[i] ^ keyByte[i]);
            }
            return bCipher;
        }
```

Percobaan dilakukan 2 kali. Namun, karena masih prototipe, belum terlihat periodenya. Hal ini dikarenakan, periode baru muncul setelah sekian banyak digit. Modulo yang digunakan adalah 256. Kotak hitam yang digunakan adalah operator binary XOR.

Input : ismail sunni makan sate, hari rabu pagi, tapi bohong
Key : 3 dan 5



Input : lontong adalah sumber karbohidrat, sate adalah sumber protein
Key : 7 dan 5



VII. KESIMPULAN

Berikut ini adalah kesimpulan yang bisa diambil dari makalah ini :

1. Bilangan irrational dapat dipergunakan untuk menggenerate key untuk stream cipher, dan bisa mendekati sifat one-time pad, namun memiliki kelemahan pada proses generatingnya, karena membutuhkan resource yang cukup besar , dan perlu membuat struktur data sendiri
2. Barisan fibonacci, atau lebih umumnya Lagged Fibonacci Generator, bisa dipergunakan untuk menggenerate key untuk stream cipher. Namun, kekuatan enkripsi bergantung pada nilai j dan k yang digunakan untuk memperbesar periode.
3. Prototipe aplikasi telah dibuat dengan skala sederhana.

DAFTAR PUSTAKA

[1] http://math.about.com/library/bli.htm tanggal akses 22 Maret 2011

[2] http://mathworld.wolfram.com/FibonacciNumber.html tanggal akses 22 Maret 2011

[3] http://en.wikipedia.org/wiki/Lagged_Fibonacci_generator. tanggal akses 22 Maret 2011

[4] http://jproc.ca/crypto/terms.html tanggal akses 22 Maret 2011

[5] http://www.rsa.com/rsalabs/node.asp?id=2174 tanggal akses 2 Maret 2011

[6] http://www.kremlinencrypt.com/algorithms.htm tanggal akses 2 Maret 2011

[7] http://www.rossbach.to/tech/crypto/IrratCrypto.htm tanggal akses 2 Maret 2011

[8] http://kryptosfan.wordpress.com/speculations/is-k4-enciphered-using-a-one-time-pad/ tanggal akses 2 Maret 2011

[9] Ghodosi, Houssei, 1995, Pseudorandom Sequences obtained from Expansions of Irrational Numbers, Department of Computer Science Center for Computer Security Research, University of Wollongong Australia,

[10] Menezes, A, 1996, Handbook of Applied Cryptography,

[11] Munir, Rinaldi, Ir.,M.T. 2007. Diktat Kuliah IF-3058Kriptografi. Bandung : Informatika ITB

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, 23 Maret 2011

Ismail Sunni 13508064