Enkripsi Block Cipher 16 bit dengan bantuan Beatty Sequence dari Bilangan Prima dan Implementasinya pada Mode ECB

Akbar Gumbira 13508106
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If18106@students.if.itb.ac.id

***Abstract*—Pada makalah ini, penulis menjelaskan tentang algoritma enkripsi baru block cipher 16 bit dengan bantuan dari beatty sequence dari bilangan prima. Enkripsi block cipher merupakan enkripsi yang termasuk kepada algoritma kriptografi modern. Algoritma cipher block ini beroperasi pada block bit. Enkripsi serta dekripsi dilakukan block per block. Fungsi dari algoritma baru yang dibuat oleh penulis menggunakan bantuan beatty sequence. Beatty sequence merupakan barisan bilangan pada ilmu matematika yang menerima masukan berupa bilangan irasional. Untuk mendapatkan bilangan irasional ini, penulis menggunakan akar dari bilangan prima. Beatty sequence ini digunakan oleh penulis untuk mengolah subkunci dari kunci masukan pengguna pada tiap putaran di jaringan feistel. Pada implementasinya, penulis menggunakan mode Electronic Code Book (ECB) dan menggunakan bahasa C#.**

***Index Terms*—beatty sequence, bilangan prima, block cipher, Electronic Code Book, jaringan feistel.**

I. PENDAHULUAN

Enkripsi block cipher merupakan enkripsi yang dilakukan dalam satuan blok. Blok-blok ini dibentuk dari bit-bit plainteks yang dibagi dengan panjang yang sama. Enkripsi dilakukan terhadap blok dari bit plainteks menggunakan bit-bit kunci. Algoritma enkripsi menghasilkan block cipherteks yang panjangnya sama dengan blok plainteks.

Terdapat empat mode operasi pada enkripsi cipher block ini, yaitu Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), dan Output Feedback. Fungsi enkripsi ini tidak membatasi implementasi mode karena semua mode dapat menggunakan fungsi enkripsi tanpa bergantung pada fungsi enkripsi yang dibuat.

Ilmu matematika merupakan ilmu yang mendukung kemajuan dari dunia kriptografi. Untuk memahami kriptografi sendiri diperlukan latar belakang matematika, khususnya matematika diskrit, dan lebih khususnya pada Teori Bilangan. Pada ilmu teori bilangan, bilangan prima merupakan bilangan yang sangat istimewa. Hal inipun yang menjadikan landasan dari dunia kriptografi. Pada serial film numb3rs (season 1, episode 5) disebutkan bahwa untuk melakukan enkripsi dari transaksi yang dilakukan pada internet digunakan bilangan yang sangat besar ($10^{230} - 10^{1000}$). Untuk melakukan enkripsi tersebut, bilangan besar tersebut tidak hanya sekedar digunakan, namun bilangan yang digunakan tersebut dibentuk dari perkalian bilangan prima. Dari bilangan prima inilah, banyak algoritma kriptografi dibentuk, sebut saja algoritma terkenal yang sering digunakan pada transaksi internet, RSA.

Pada makalah yang dibuat ini, penulis membuat suatu fungsi enkripsi block cipher dengan menggunakan suatu barisan bilangan pada matematika, yaitu beatty sequence, dengan menggunakan bilangan prima serta mengimplementasikannya pada mode ECB dengan bahasa pemrograman C#.

II. DASAR TEORI

1) LANDASAN MATEMATIKA

Dalam perancangan algoritma enkripsi ini, penulis menggunakan teori-teori matematika yang sudah ada. Teori-teori tersebut yaitu :

a. Bilangan Prima

Bilangan prima merupakan bilangan yang hanya memiliki dua faktor, yaitu 1 dan bilangan itu sendiri. Bilangan prima muncul pada interval yang random pada garis bilangan. Oleh karena itu sangat sulit untuk mengambail suatu bilangan yang besar dan memfaktorkannya menjadi perkalian bilangan prima. Hal inilah yang menjadi dasar bagaimana kemanan internet bekerja.

b. Beatty Sequence

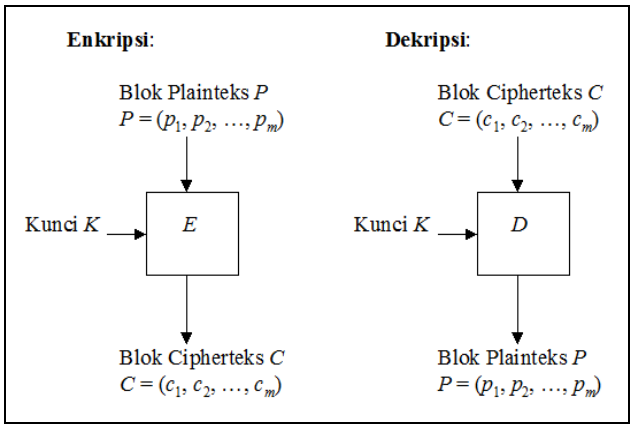
Beatty sequence merupakan barisan bilangan dengan suaru parameter bilangan irasional a, sehingga dihasilkan suatu barisan $\lfloor a \rfloor, \lfloor 2a \rfloor, \lfloor 3a \rfloor, \lfloor 4a \rfloor, \ldots$. Dengan fungsi $\lfloor \_ \rfloor$ merupakan fungsi flooring.

Karena bilangan prima hanya terdiri dari dua faktor yaitu 1 dan bilangan itu sendiri, maka akar dari suatu bilangan prima pasti merupakan bilangan irasional. Dengan memanfaatkan sifat dari bilangan prima ini, kita dapat menggenerate beatty sequence untuk parameter akar dari suatu bilangan prima.

2) PRINSIP DASAR PERANCANGAN BLOCK CIPHER

Enkripsi block cipher dilakukan per satuan block. Block ini dibentuk dari bit-bit plainteks. Panjang block hasil enkripsi akan sama dengan panjang block hasil dekripi, begitu juga dengan jumlah blocknya. Misal jika, suatu block plainteks yang berukuran *m* bit P={p_1, p_2, p_3, …., p_m), dengan $p_i \in \{0,1\}$, maka block cipher yang dihasilkan akan berukuran *m* bit juga, C = {c_1, c_2, c_3,…., c_m), dengan $c_i \in \{0,1\}$.

Secara umum, enkripsi block cipher yang dilakukan ini yaitu seperti pada gambar berikut ini :



Gambar 1 Skema Enkripsi Dekripsi Cipher Block

Prinsip-prinsip dasar dari perancangan block cipher ini yaitu :

a. Prinsip Confusion dan Diffusion Shannon

Prinsip confusion digunakan untuk menyembunyikan hubungan antara plainteks, cipherteks, dan kunci. Confusion yang baik yaitu confusion yang membuat hubungan statistik antara plainteks, cipherteks, dan kunci menjadi sangat rumit.

Sedangkan prinsip diffusion merupakan prinsip yang menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Pengubahan kekcil pada plainteks sebanyak satu atau dua bit akan menghasilkan perubahan pada cipherteks yang tidak dapat diprediksi. Mode CBC dan CFB menggunakan prinsip ini.

b. Cipher Berulang

Cipher berulang merupakan fungsi transformasi sederhana yang mengubah plainteks menjadi cipherteks yang diulang sejumlah kali. Pada setiap putaran digunakan subkunci atau kunci putaran yang dikombinasikan dengan plainteks.

Secara formal, cipher berulang dinyatakan sebagai :

$C_i = f(C_{i-1}, K_i)$,

Dengan :

i = 1, 2, 3,…., r (r merupakan jumlah putaran)

K_i = subkunci pada putaran ke-*i*

f = fungsi transformasi

c. Jaringan Feistel

Hampir semua algoritma cipher block yang terkenal bekerja pada model jaringan feistel. Jaringan feistel ditemukan oleh Horst Feistel pada tahun 1970.

Model jaringan feistel yaitu sebagai beikut :

i) Bagi block yang panjangnya n bit menjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya n/2 (dari sini, disyaratkan bahwa n haruslah genap).

ii) Definisikan cipher block berulang dimana hasil dari putaran ke-*i* ditentukan dari hasil putaran sebelumnya, yaitu :

$L_i = R_{i+1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

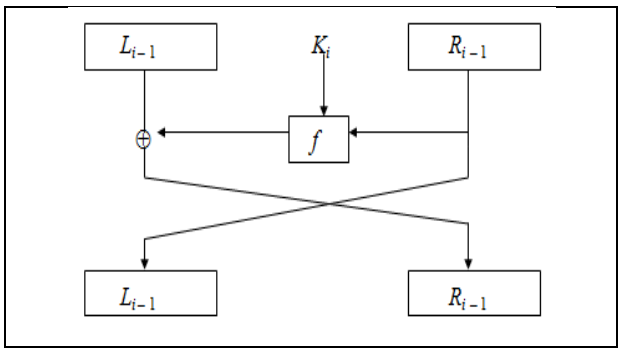
Dengan :

i = 1, 2, 3,…., r (r merupakan jumlah putaran)

K_i = subkunci pada putaran ke-*i*

f = fungsi transformasi

Model jaringan feistel dapat digambarkan sebagai berikut :



Gambar 2 Skema Jaringan Feistel

Jaringan feistel banyak digunakan pada algoritma kriptografi DES, LOKI, GOST, FEAL, Lucifer, Blowfish, Khufu, Khafre, dan lain-lain karena model jaringan feistel ini bersifat reversible untuk proses enkripsi dan dekripsi. Sifat reversible ini membuat kita tidak perlu membuat algoritma baru untuk mendekripsi cipherteks menjadi plainteks. Karena operator XOR mengkombinasikan setengah bagian kiri dengan hasil dari fungsi transformasi *f*, maka kesamaan berikut pasri benar :

$L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) = L_{i-1}$

Sifat reversible ini tidak bergantung pada fungsi *f* sehingga fungsi *f* dapat dibuat serumit mungkin.

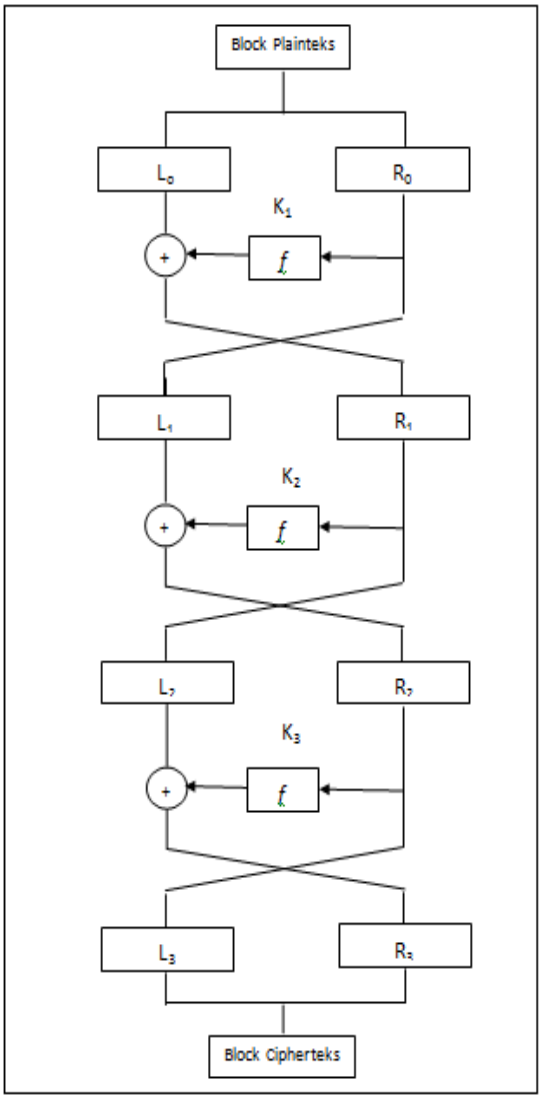
III. RANCANGAN ALGORITMA BLOCK CIPHER

Pada rancangan algoritma block cipher yang penulis

buat, berikut bagian-bagian dari algoritma tersebut :

1) Plainteks Schedule

Pada bagian ini, plainteks masukan diubah menjadi block-block berukuran 16 bit. Jika jumlah bit bukan merupakan kelipatan 16, maka block terakhir harus dipadding. Karena block ini berukuran 16 bit, sementara pada implementasinya digunakan file (dalam satuan byte atau kelipatan 8 bit), maka dapat diketahui dengan mudah bahwa jumlah padding yang mungkin yaitu sebanyak 0 bit atau sebanyak 8 bit. Selanjutnya setiap block akan dimasukkan ke jaringan feistel. Jaringan feistel yang dirancang terdiri dari 3 putaran. Rancangan jaringan feistel tersebut yaitu sebagai berikut :



Gambar 3 Skema Feistel

Dari skema feistel diatas, dapat kita tulis persamaan yang memudahkan kita dalam implementasi. Persamaan tersebut yaitu :

a. Enkripsi Block

Pada enkripsi block, dari gambar 3 didapat persaman sebagai berikut :

- $L_1 = R_0$
- $R_1 = L_0 \oplus f(R_0, K_1)$

- $L_2 = R_1$
- $R_2 = L_1 \oplus f(R_1, K_2)$

- $L_3 = R_2$
- $R_3 = L_2 \oplus f(R_2, K_3)$

Gambar 4 Skema Feistel Enkripsi

b. Dekripsi Block

Pada dekripsi block, dari gambar 4 didapat persaman sebagai berikut (Perhatikan bahwa karena jaringan feistel bersifat reversible, maka kita tidak perlu membuat fungsi dekripsi, cukup menggunakan fungsi enkripsi) :

- $R_2 = L_3$
- $L_2 = R_3 \oplus f(R_2, K_3)$

- $R_1 = L_2$
- $L_1 = R_2 \oplus f(R_1, K_2)$

- $R_0 = L_1$
- $L_0 = R_1 \oplus f(R_0, K_1)$

Gambar 5 Skema Feistel Dekripsi

Untuk mendekripsi block, sebenarnya kita tidak perlu membuat algoritmanya lagi. Pada implementasinya, karena sifat jaringan feistel yang reversible kita cukup memanggil kembali fungsi enkripsi pada block yang telah dienkripsi.

2) Key Schedule

Pada bagian ini, key yang berupa string diubah menjadi bit-bit. Pada tiap putaran dari jaringan feistel, subkunci di*generate* dengan bantuan beatty sequence. Karena Li dan Ri dari block (lihat pada jaringan feistel) terdiri dari 8 bit, maka subkunci pada tiap putaran pun terdiri dari 8 bit. Misal bit kunci disimbolkan sebagai Key (Array of bit). Panjang kunci tersebut disimbolkan sebagai Key.Length (banyak bit dari Key).

Pembangkitan subkunci tersebut, dapat dirumuskan sebagai berikut

- Bit ke-*i* dari K*j* = Key[$\lfloor i \; x \; \sqrt{Prime(((\sum key)\%256)) + j} * 256 \rfloor \% (Key.Length)$].
  Dengan i ϵ {0, 1 ,… ,7 }, j ϵ {1, .. ,3}, dan *Prime(i)* merupakan bilangan prima ke-i.

3) Fungsi *f*

Fungsi f dalam setiap putaran didefinisikan sebagai berikut :

$$f(R_i, K_{i+1}) = R_i \oplus \mathrm{K}_{i+1} \oplus \mathrm{MSB}_8(Key) \oplus \mathrm{LSB}_8(Key)$$

Dengan $\mathrm{MSB}_8(Key)$ yaitu 8 bit dari Most Significant Bit Key dan $\mathrm{LSB}_8(Key)$ adalah 8 bit dari Least Significant Bit Key. Oleh karena itu, algoritma ini mensyaratkan bahwa kunci minimal terdiri dari satu karakter (8 bit) karena dibutuhkan LSB serta MSB untuk mengenkripsi.

IV. IMPLEMENTASI ALGORITMA DENGAN MENGGUNAKAN MODE ECB

Pada implementasi yang penulis buat, penulis menggunakan mode Electronic Code Book. Pada mode ini, setiap block plainteks *Pi* dienkripsi secara individual dan independen menjadi block cipherteks *Ci*. Penulis mengenkripsi sebuah file. File yang dapat dienkripsi adalah file dengan semua jenis ektensi. Berikut source code file yang berisi fungsi utama enkripsi dan dekripsi yang penulis buat (file source code lain tidak ditampilkan) :

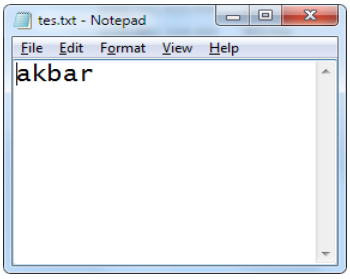
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace makalahstima
{
  static class Function
  {
    public static int Prime(int input)
    {
      int[] prime =
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,8
9,97,101,103,107,109,113,127,131,137,139,149,151,157,163,167,17
3,179,181,191,193,197,199,211,223,227,229,233,239,241,251,257,2
63,269,271,277,281,283,293,307,311,313,317,331,337,347,349,353,
359,367,373,379,383,389,397,401,409,419,421,431,433,439,443,44
9,457,461,463,467,479,487,491,499,503,509,521,523,541,547,557,5
63,569,571,577,587,593,599,601,607,613,617,619,631,641,643,647,
653,659,661,673,677,683,691,701,709,719,727,733,739,743,751,75
7,761,769,773,787,797,809,811,821,823,827,829,839,853,857,859,8
63,877,881,883,887,907,911,919,929,937,941,947,953,967,971,977,
983,991,997,1009,1013,1019,1021,1031,1033,1039,1049,1051,106
1,1063,1069,1087,1091,1093,1097,1103,1109,1117,1123,1129,115
1,1153,1163,1171,1181,1187,1193,1201,1213,1217,1223,1229,123
1,1237,1249,1259,1277,1279,1283,1289,1291,1297,1301,1303,130
7,1319,1321,1327,1361,1367,1373,1381,1399,1409,1423,1427,142
9,1433,1439,1447,1451,1453,1459,1471,1481,1483,1487,1489,149
3,1499,1511,1523,1531,1543,1549,1553,1559,1567,1571,1579,158
3,1597,1601,1607,1609,1613,1619};
      return prime[input-1];
    }

    public static BitArray keyGenerator(string key, int keyi)
    {
      BitArray result = new BitArray(8);
      int keysum=0;
      for (int i = 0; i < key.Length; ++i)
      {
        keysum+=(int)key[i] ;
      }
      BitArray keyBitArray =
Converter.convertStringToBitArray(key);
     for (int i = 0; i < 8; ++i)
      {
        result[i] = keyBitArray[(int)(Math.Floor(i *
Math.Sqrt(Prime(keysum%256) +
keyi*256))%(keyBitArray.Length) )];
      }
      return result;
    }

    public static BitArray MSB_8(BitArray key)
    {
      BitArray result = new BitArray(8);
      for (int i = 0; i < 8; ++i)
      {
        result[i] = key[i];
      }
      return result;
    }

    public static BitArray LSB_8(BitArray key)
    {
      int n = key.Length;
      int mulai = n - 8;
      BitArray result = new BitArray(8);
      for (int i = mulai; i < n; ++i)
      {
        result[i-mulai] = key[mulai];
      }
      return result;
    }

    public static BitArray f(BitArray R, BitArray Kround, BitArray
KeyBitArray)
    {
      return
(R.Xor(Kround).Xor(MSB_8(KeyBitArray))).Xor(LSB_8(KeyBitArray)
);


    }
    public static Block_Builder Encrypt(Block_Builder input, string
key)
    {
      Block_Builder result = new Block_Builder(input.Blocks.Count);
      BitArray keyBitArray =
Converter.convertStringToBitArray(key);
      for (int i = 0; i < input.Blocks.Count; ++i)
      {
        BitArray block = new BitArray(input.Blocks[i]);
        BitArray L0 = new BitArray(8);
        for (int j=0; j<8;++j)
        {
          L0[j] = block[j];
        }
        BitArray R0 = new BitArray(8);
        for (int j=0; j<8;++j)
        {
          R0[j] = block[j+8];
        }

        BitArray L1 = new BitArray(R0);
        BitArray K1 = new BitArray(keyGenerator(key, 1));
BitArray R1 = new BitArray(L0.Xor(f(R0, K1, keyBitArray)));

        BitArray L2 = new BitArray(R1);
        BitArray K2 = new BitArray(keyGenerator(key, 2));
       BitArray R2 = new BitArray(L1.Xor(f(R1, K2, keyBitArray)));


        BitArray L3 = new BitArray(R2);
        BitArray K3 = new BitArray(keyGenerator(key, 3));
       BitArray R3 = new BitArray(L2.Xor(f(R2, K3, keyBitArray)));

        BitArray resBlock = new BitArray(16);
        for (int j = 0; j < 16; ++j)
        {
          if (j < 8)
```

```
            {
              resBlock[j] = L3[j];
            }
            else
            {
              resBlock[j] = R3[j - 8];
            }
          }
          result.Blocks.Insert(i, resBlock);
        }
        return result;

      }

      public static Block_Builder Decrypt(Block_Builder input, string
key)
      {
        return Encrypt(input,key);
      }
    }
}
```

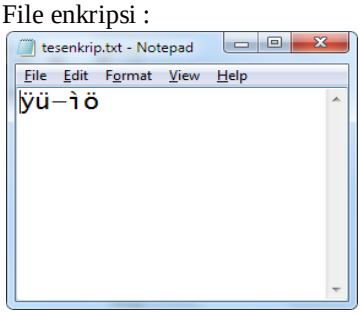
V. HASIL PENGUJIAN IMPLEMENTASI PROGRAM

Beberapa file dengan ekstensi yang berbeda telah penulis coba. File-file tersebut diantaranya yaitu :

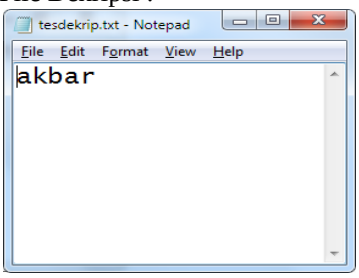
1) File teks (tes.txt) :
   - o File Asli :



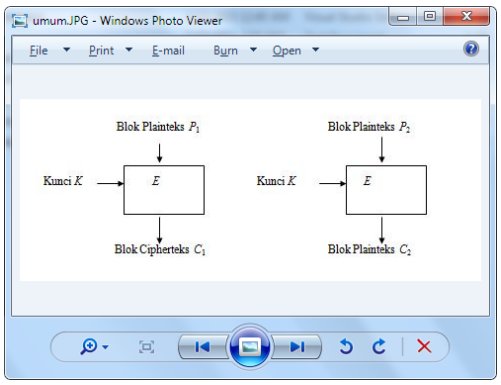
   - o File enkripsi :



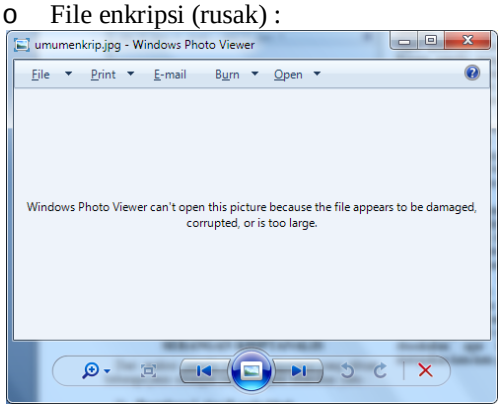
   - o File Dekripsi :



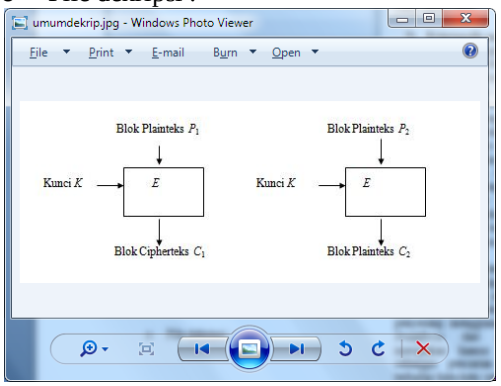
2) File Image (umum.jpg)
   - o File asli :



   - o File enkripsi (rusak) :



   - o File dekripsi :



VI. ANALISIS ALGORITMA DARI SERANGAN KRIPTANALIS

Dari analisis penulis terhadap algoritma yang dibuat, beberapa jenis serangan yang mungkin dilakukan yaitu :

1) Bruteforce L dan R pada block

Dari suatu block terenkripsi, terdapat L_3 dan R_3. Perhatikan pada Gambar 5 (Skema Feistel Dekripsi), untuk mendapatkan L_0 dan R_0, penyerang dapat mencoba semua kemungkinan $f(R_2,K_3)$ untuk mendapatkan L_2, dari L_2 penyerang akan mendapatkan R_1, kemudian penyerang mencoba semua kemungkinan $f(R_1, K_2)$, dari L_1 penyerang akan mendapatkan R_0, kemudian untuk mendapatkan L_0, penyerang mencoba semua kemungkinan $f(R_0, K_1)$. Untuk algoritma ini sendiri, karena L dan R terdiri dari 8 bit, maka untuk percobaan semua kemungkinan $f(R_2, K_3)$ dilakukan 2^8 kali. Begitu juga dengan percobaan kemungkinan $f(R_0, K_1)$ dan $f(R_2, K_3)$. Sehingga

dibutuhkan total $2^8 x 2^8 x 2^8 = 2^{24} = 16777216$ kali percobaan (dilakukan perkalian karena ketiga hasil ini saling mempengaruhi). Mungkin untuk komputer modern ini percobaan sebanyak ini masih mungkin dilakukan dengan cepat. Untuk mengatasi masalah serangan ini, algoritma dapat diubah dengan cara menambah jumlah putaran pada feistel serta menambah ukuran block. Jika jumlah putaran sebanyak *k* kali dan ukuran block plainteks adalah *n* bit, maka dibutuhkan percobaan sebanyak $(2^{n/2})^k$. Perhatikan bahwa jika algoritma ini dibuat untuk block dengan 128 bit misalnya (bukan 16 bit) serta jumlah putaran pada feistel sebanyak 10 kali, maka dibutuhkan percobaan sebanyak $(2^{64})^{10} = 2^{640}$. Percobaan sebanyak ini tentu akan memakan waktu yang sangat lama.

2) Kriptanalis membuat Code Book

Karena pada implementasinya penulis menggunakan mode Electronic Code Book, maka untuk satu kunci yang sama, block terenkripsi yang sama akan memetakan block dekripsi yang sama pula. Sehingga, kriptanalis mungkin dapat membuat statistik dari block yang sering muncul. Karena penulis menggunakan block 16 bit, maka pada satu code book (untuk satu kunci) akan terdapat 2^{16} entri block. Hal ini dapat diatasi dengan cara mengganti kunci untuk setiap transaksi yang berbeda, sehingga code book akan berubah-ubah. Selain itu, agar code book tersebut tidak mungkin disimpan, cara lain yang dapat digunakan yaitu memperbesar ukuran block. Jika ukuran block adalah *k*, maka pada satu code book akan terdapat 2^k masukan. Semakin besar *k* maka akan semakin besar pula code book yang harus disimpan.

3) Bruteforce kunci

Semua jenis algoritma enkripsi yang ada dapat dilakukan serangan jenis ini. Serangan dilakukan dengan cara mencoba semua kemungkinan kunci. Biasanya penyerang menggunakan *dictionary*, sehingga kunci di-*bruteforce* dari *dictionary* tersebut. *Dictionary* merupakan kamus kata-kata pada bahasa tertentu, sehingga pencarian lebih efektif, karena dilakukan terhadap kata-kata yang ada dari kamus tersebut.

Cara yang mudah untuk menghindari serangan ini yaitu diusahakan agar kunci yang dimasukkan bukan merupakan kata-kata pada bahasa tertentu.

VII. KESIMPULAN

Dari semua yang telah dipaparkan sebelumnya, ada beberapa kesimpulan yang dapat ditarik dari makalah ini, yaitu :

1) Fungsi block cipher tidak mempengaruhi mode yang digunakan. Mode block cipher yang digunakan mungkin akan memperkuat algoritma block cipher secara keseluruhan.

2) Matematika merupakan landasan yang sangat penting dari ilmu kriptografi. Dengan landasan ini, kita dapat membuat algoritma-algoritma baru. Salah satu yang paling disukai dari kriptografer yaitu penggunaan dari bilangan prima.

3) Implementasi dari block cipher yang penulis buat dapat diaplikasikan pada berbagai jenis file. Jenis file yang telah penulis coba antara lain file gambar dan file teks.

4) Jenis-jenis serangan yang mungkin terhadap algoritma yang penulis buat diantaranya yaitu bruteforce L dan R dari block, kelemahan mode Electronic Code Book, serta Bruteforce semua kemungkinan string kunci.

REFERENSI

[1] Munir, Rinaldi. Slide-slide kuliah Kriptografi.

[2] http://mathworld.wolfram.com/BeattySequence.html. Diakses pada tanggal 27 Februari 2011.

[3] Serial Numb3rs, episode 5. “Prime Suspect”.

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

Akbar Gumbira (13508106)