Algoritma Rubik Cipher

Khoirunnisa Afifah
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
k.afis3@rocketmail.com

Yollanda Sekarrini
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
yollandasekarrini@gmail.com

***Abstract—Block Cipher* merupakan algoritma kriptografi yang melakukan enkripsi dengan membagi plaintext menjadi blok-blok. Satu blok berukuran 64 bit atau lebih. Dibandingkan enkripsi tiap satu byte, algoritma block cipher menjadi algoritma yang lebih baik karena pola enkrip yang lebih beragam sehingga sulit untuk diserang atau dipecahkan. Inilah yang membuat kami tertarik untuk membuat algoritma *Block Cipher* versi lain. Algoritma yang kami tawarkan adalah algoritma Rubik Cipher yang menggunakan media rubik dalam pengenkripsian. Tiap bit plaintext dimasukkan sebagai elemen rubik, kemudian elemen-elemen tersebut diputar sesuai dengan rumus yang terdapat pada rubik yang berasal dari kunci. Fungsi rubik tersebut dibungkus dengan menggunakan jaringan Feistel dan dilakukan berulang-ulang sehingga menambah efek difusi pada algoritma ini. Ukuran blok pada algoritma ini yakni 12 byte.**

Kata Kunci--block chiper; ciphertext; jaringan feistel; plaintext; rubik;

I. PENDAHULUAN

Block Cipher adalah algoritma kriptografi modern yang banyak digunakan pada saat ini. Banyak implementasi algoritma *Block Cipher*, salah satunya algoritma Rijndael yang pada tahun 2001, ditetapkan menjadi *Advance Encryption Standard* (AES). Algoritma ini lebih mengandalkan kerahasiaan pada kunci daripada algoritma sehingga siapa saja dapat mengetahui proses enkripnya. Tetapi tanpa kunci, ciphertext tidak akan dapat dipecahkan. Hal itulah yang menyebabkan algoritma *Block Cipher* banyak diimplementasikan saat ini.

Jaringan Feistel adalah suatu metode yang dapat mentransformasikan fungsi apapun ke dalam permutasi. Jaringan Feistel memiliki sifat *reversible* sehingga proses enkripsi dan dekripsi, tidak perlu membuat algoritma baru. Disamping itu, Jaringan Feistel bisa diimplementasikan pada fungsi secara independen, sehingga dapat membuat fungsi serumit apapun.

Rubik adalah mainan teka-teki berbentuk kubus yang terdiri dari enam sisi dengan warna berbeda dan setiap sisi terdapat delapan kotak yang bisa diputar. Kotak tersebut dapat diputar pada bagian kiri, kanan, depan, belakang, atas, bawah, kolom tengah dan baris tengah. Akibat perputaran kotak tersebut, pola rubik yang mungkin terbentuk adalah 43,252,003,274,489,856,000. Dengan mengambil keuntungan banyaknya kemungkinan pola rubik yang dapat terbentuk dan algoritma *Block Cipher* yang andal, kami menawarkan algoritma Rubik Cipher yang mengimplementasikan *Block Cipher* dan menggunakan Rubik sebagai media enkripsi plaintext. Enkripsi tersebut kemudian dibungkus dengan Jaringan Feistel.

II. DASAR TEORI

A. Block Cipher

Block Cipher adalah algoritma kriptografi yang melakukan enkripsi dengan membagi-bagi plaintext menjadi blok-blok dengan ukuran tertentu (64 bit atau lebih). Blok-blok yang telah terbentuk dienkripsi dengan menggunakan kunci yang sama. Penggunaan enkripsi terhadap blok akan membuat pola ciphertext menjadi lebih rumit dibandingkan enkripsi terhadap satu bit. Implementasi *Block Cipher* adalah *Advance Encryption Standard* (AES) dan *Data Encryption Standard* (DES).

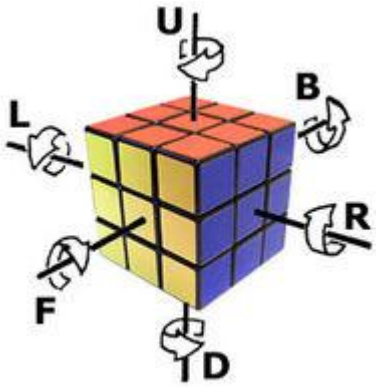
B. Rubik

Rubik adalah sebuah permainan teka-teki 3D berbentuk kubus yang terdiri dari 27 bagian kecil yang berputar terhadap porosnya. Satu sisi kubus terdiri dari sembilan kotak dengan warna yang berbeda untuk tiap sisinya. Untuk menyelesaikan permainan ini, pemain bertugas untuk memutar bagian kecil tadi agar tiap kotak berada pada warna yang sama dengan warna kotak tengah yang berada ditiap sisi sehingga sembilan kotak dengan warna yang sama berada dalam satu sisi. Rubik ditemukan oleh pemahat dan arsitektur Hungaria, Ernő Rubik, pada tahun 1974. Rubik menjadi mainan dengan penjualan terbanyak di dunia yakni sekitar 300 juta buah.

Pada rubik setiap kotak disebut sebagai *piece*. Jenis *piece* ada tiga buah. Yang pertama disebut *center* yaitu *piece* yang ada di tengah setiap sisi. Yang kedua adalah edge yaitu piece yang mewakili rusuk pada rubik, dan terakhir adalah corner yang mewakili titik sudut pada rubik.

Pergerakan dalam rubik pada dasarnya ada 8 jenis berdasarkan bagian yang diputar. Notasi untuk pergerakan yang digunakan pada makalah ini adalah R untuk sisi kanan, L untuk sisi kiri, F untuk sisi depan, B untuk sisi belakang, U untuk sisi atas, D untuk sisi bawah, M untuk sisi tengah verikal dan E

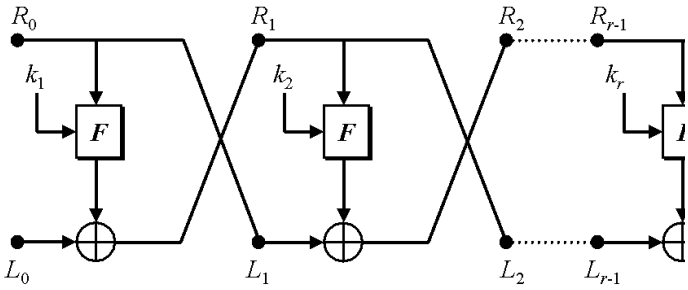
untuk sisi tengah horizontal. Perputaran normal dengan notasi diatas dilakukan searah dengan jarum jam, sedangkan jika notasi tersebut diberi tambahan tanda petik ia dilakukan berlawanan arah jarum jam. Misalnya R adalah perputaran sisi kanan searah jarum jam dan R' adalah perputaran sisi kanan berlawanan arah jarum jam.



Gambar 1 Notasi Rotasi Rubik[2]

C. *Feistel*

Feistel Cipher atau biasa disebut *Feistel Network* adalah suatu struktur simetris yang sering digunakan dalam pembangunan *Block Cipher*. Ditemukan oleh lelaki kelahiran Jerman, seorang ahli fisika dan kriptografer, bernama Horst Feistel. Jaringan Feistel memiliki sifat *reversible* karena strukturnya yang simetris sehingga enkripsi dan dekripsinya sangat mirip bahkan dekripsi hanya perlu memutar kembali penggunaan kunci. Contoh algoritma yang menggunakan Jaringan Feistel adalah DES, LOKI dan *Blowfish.*



Gambar 2 Jaringan Feistel[3]

III. DESKRIPSI ALGORITMA

Algoritma *Block Cipher* yang dibuat menggunakan jaringan Fiestel dan Rubik 3x3. Pesan dienkripsi setiap 96 bit (12 karakter). Apabila masukkan pesan kurang dari 96 bit, pesan akan ditambahi bit 0 di akhir sampai panjang pesan 96 bit. Jumlah iterasi pada Fiestel sama dengan jumlah byte pada pesan yaitu 12 kali. Kunci yang digunakan juga merupakan string sepanjang 12 karakter.

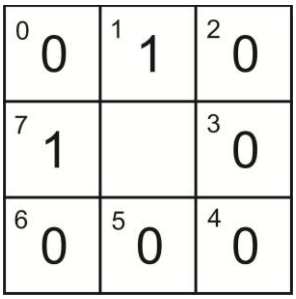
A. *Pembangkitan kunci internal*

Langkah pertama untuk melakukan enkripsi dengan algoritma ini adalah pembangkitkan semua kunci internal yang akan digunakan untuk menentukan permutasi pada setiap iterasi fiestel. Kunci eksternal sepanjang 12 karakter yang didapat dari masukkan pengguna akan digunakan untuk membangkitkan kunci internal pertama, kemudian kunci internal berikutnya dibangkitkan dari kunci internal yang sudah ada sekarang, begitu seterusnya. Langkah pembangkitan kunci internalnya dengan cara menjumlahkan perkarakter kunci dengan karakter pertama dari kunci sebelumnya. Potongan kode berikut merupakan implementasi dari pembangkitan kunci internal Algoritma Rubik Cipher.

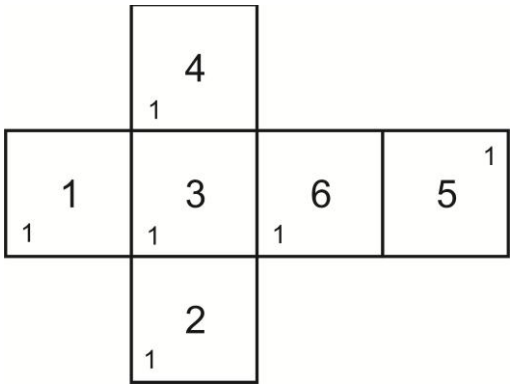
```
key = new String[numIteration];
String temp = eKey;
String res;
for (int j=0; j<numIteration;j++){
   res = "";
   for (int i=0; i<temp.length();i++) {
   res += (char) ((temp.charAt(i) +
(int) temp.charAt(0) % 255);
   }
   key[j] = res;
   temp = key[j];
}
```

B. *Fungsi Rubik*

Pada algoritma ini rubik 3x3 digunakan sebagai fungsi yang dipanggil pada setiap iterasi Fiestel. Fungsi ini bertujuan mengacak urutan bit. Fungsi ini menerima masukan berupa 48 bit pesan dan kunci internal sejumlah n bit. Setiap bit masukan akan diletakkan pada setiap *piece* dari rubik kecuali bagian *center* dari rubik. Bit-bit tersebut dimasukkan dan dibaca dengan urutan yang sama sehingga orientasi rubik tidak boleh berubah. Urutan penyisispan bit adalah sisi atas, kiri, depan, kanan, belakang, lalu bawah. Pada setiap sisinya urutan peletakan bit dimulai dari pojok kiri atas dan searah jarum jam. Diagram penyusunan bit dapat dilihat dari gambar berikut.



Gambar 3 penyusunan bit untuk nilai 01000001



Gambar 4 urutan sisi rubik dengan 1 adalah atas, 2 adalah kiri, 3 adalah depan dan angka satu ditiap sudut sisi sebagai petunjuk awalan id

Algoritma penyusunan bit plaintext ke dalam elemen-elemen rubik yaitu:

```
for (int face=0; face<6; face++) {
 for (int id=0; id<8; id++) {
  elmt[face][id]=
           binaryString.charAt(idx);
  idx++;
 }
}
```

Setelah bit-bit tersebut diletakkan di rubik, langkah selanjutnya adalah menentukan perputaran yang harus diterapkan. Perputaran ditentukan oleh kunci masukkan dengan cara mencocokkan nilai kunci dengan matriks permutasi yang telah ditentukan. Matriks yang digunakan adalah matriks seperti berikut :

$$\begin{bmatrix} R & Ri & L & Li \\ U & Ui & D & Di \\ F & Fi & B & Bi \\ M & Mi & E & Ei \end{bmatrix}$$

Gambar 5 matriks perputaran rubik

Nilai kunci internal yang digunakan untuk membaca matriks permutasi adalah nilai dari empat bit terakhir dari setiap karakter pada kunci. Empat bit tersebut dibagi menjadi dua bagian dengan bagian pertama merupakan nilai baris pada matriks dan dua nilai terakhir merupakan nilai kolom pada matriks. Algoritma mendapatkan nilai permutasi matriks dari kunci yang di masukkan adalah

```
String binary = Integer.toBinaryString(ch);
if (binary.length()<8) {
  String zero="";
  for (int j=0; j<8-binary.length();j++) {
       zero += "0";
  }
  binary = zero+binary;
}
binary = binary.substring(4);
```

Pembagian per karakter ini mengakibatkan jumlah permutasi yang dilakukan pada setiap iterasinya akan sejumlah panjang kunci. Misal kunci internal *'yola'* perputaran yang dilakukan adalah :

y – 1001 : baris 2, kolom 1 = F'

o – 1111: baris 3, kolom 3 = E'

l – 1100 : baris 3, kolom 0 = M

a – 0001 : baris 0, kolom 1 = R'

C. Jaringan Fiestel

Pesan awal yang telah diubah menjadi bit akan dibagi menjadi dua dan dijadikan L dan R awal untuk iterasi Fiestel. Seperti jaringan Fiestel pada umumnya, nilai L selanjutnya sama dengan nilai R pada iterasi saat ini. Nilai R pada iterasi saat ini juga akan menjadi masukan untuk fungsi rubik dengan kunci internal yang telah dibangkitkan. Hasil keluaran fungsi rubik tersebut akan di XOR kan dengan nilai L saat ini untuk mendapatkan nilai R di iterasi selanjutnya. Proses ini diulangi terus sampai 12 kali. Perhitungan nilai R dan L untuk setiap iterasi pada Fiestel dapat dituliskan sebagai berikut :

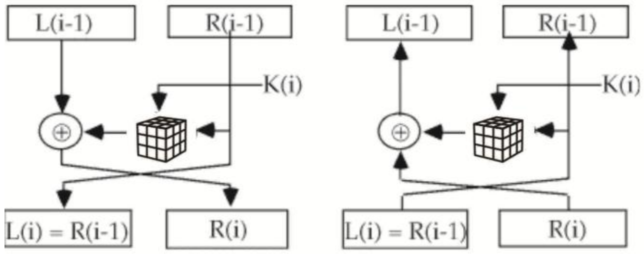
$$L_i = R_{i-1}$$

$$Ri = L_{i-1}\ XOR\ (R_{i-1} \rightarrow \text{▦})$$

Untuk dekripsi pesan dapat dilakukan dengan membalik urutan fiestel dari mulai paling bawah. Penentuan L dan R setiap iterasi dapat dilakukan menggunakan cara :

$$R_i = L_{i+1}$$

$$L_i = R_{i+1}\ XOR\ (R_{i+1} \rightarrow \text{▦})$$



Gambar 6 Jaringan Feistel yang dipakai (menggunakan fungsi Rubik)

Algoritma untuk mendapatkan semua bit pada elemen rubik yaitu

```
String retval="";
for (int face=0; face<6; face++) {
   for (int id=0; id<8; id++) {
      retval+=elmt[face][id];
   }
}
```

IV. EKSPERIMEN DAN PEMBAHASAN HASIL

Algoritma telah diimplementasikan menjadi sebuah program dalam bahasa Java dan setelah diujikan ke beberapa kasus uji didapatkan hasil yang baik. Pesan dapat terenkripsi dan terdekripsi dengan baik.

Struktur data yang digunakan dalam penerapan algoritma Rubik Cipher ini adalah:

```
private char elmt[][] = new char[6][8];
```

Struktur data di atas mewakili nilai bit pesan yang akan dimasukkan ke dalam enam sisi rubik dengan satu sisi bisa memuat delapan elemen.

Contoh pengujian menggunakan plainteks berukuran 24 byte adalah sebagai berikut :

Plaintext =
01001100011011110111001001100101011011010010000001
10100101110000011100110111010101101101001000000110
01000110111101101100011011110111001000100000011100
1101101001011101000010000001100001011011010101100

Kunci = haloapakabar

Ciphertext =

01110110101100010110000101000111001100110001010111
01001110001000000010010000110111110001100011101101
01001111011000111101010100101110000110110111110011
0110001111111001000110101010100001111001100001 1

Dari pengujian dapat diketahui bahwa tidak ada hubungan antara kunci, plaintext, dan ciphertext yang dapat terlihat sehingga prinsif confussion dari cipher block sudah terpenuhi. Prinsip ini dapat tercapai karena susunan bit pesan diacak terus di setiap iterasi Fiestel menggunkan rubik. Satu kali permutasi rubik mengubah 8 atau 20 susunan bit yang ada. Sedangkan pada algoritma ini setiap block akan diacak menggunakan 16 jenis rotasi.

Prinsip diffusion dari *Block Cipher* juga sudah dipenuhi oleh algoritma ini karena apabila salah satu bit dari kunci internal yang dibangkitkan salah maka nilai permutasi rubik yang dibaca dari matriks permutasi akan salah dan bisa menyebabkan permutasi diterapkan pada sisi yang berbeda dan mengubah susunan 20 bit yang salah. Karena pembangkitan kunci internal menggunakan kunci internal pada iterasi sebelumnya maka kesalahan akan menjalar ke semua iterasi Fiestel.

Berikut adalah tabel waktu hasil pengujian dengan berbagai file teks yang berukuran berbeda pada komputer dual core @2.3 GHz :

Ukuran input	Waktu untuk enkripsi
3 Kb	148 ms
6 Kb	274 ms
21 Kb	1067 ms
68 Kb	4280 ms

Tabel 1 hasil pengujian algoritma Rubik Cipher

Dari pengujian waktu dapat dilihat bahwa waktu komputasi dari algoritma ini masih dalam batas wajar. Fungsi rubik yang digunakan tidak menggunakan banyak iterasi karena yang dilakukan hanya menukar posisi rubik sesuai masukkan input.

Algoritma ini juga dapat diimplementasikan dengan baik dalam mode *Electronic Code Block* (ECB), *Chaining-Cipher Block* (CBC), dan *Cipher Feedback* (CFB).

V. ANALISIS KEAMANAN

Serangan yang mungkin dilakukan kriptanalisis terhadap algoritma ini adalah dengan cara pencarian kunci. Jika dilakukan pencarian dengan exhaustive search, maka akan terdapat 2^{96} $^{(=}$ 7.9228163e$^{28)}$ kemungkinan untuk susunan kunci. Untuk plaintext berukuran 68Kb dengan waktu komputasi satu percobaan 4280 ms dikali 2^{96} kemungkinan, diperlukan waktu bertahun-tahun untuk memecahkannya sehingga pemecahan kunci pada algoritma ini sangat tidak efektif. Selain itu, penggunaan jaringan Feistel pada algoritma ini juga menambah efek difusi karena pembangkitan kunci internal diawali dengan menggunakan kunci masukan pengguna kemudian dilanjutkan dengan kunci internal yang sudah ada.

VI. KESIMPULAN DAN SARAN

Dari hasil percobaan dan analisis terhadap Algoritma Rubik Cipher didapatkan beberapa hasil sebagai berikut :

1. Algoritma Rubik Cipher ini telah memenuhi kriteria confession dan diffusion dari block cipher.
2. Algoritma Rubik Cipher dapat diterapkan pada mode block cipher ECB, CBC, dan CFB.
3. Waktu komputasi algoritma Rubik Cipher masih dalam batas wajar.
4. Dengan panjang kunci 96 bit tidak mungkin dilakukan *exhaustive search* terhadap kunci algoritma.

Saran untuk pengembangan selanjutnya dari algoritma ini adalah mencari cara pembangkitan kunci internal yang lebih baik, misal dengan melakukan operasi yang melibatkan plaintext. Hal tersebut bertujuan agar sifat diffusion menjadi lebih tinggi. Selain itu dengan menggunkan kunci internal dari plaintext jumlah kemungkinan kasus untuk dilakukan exhaustive search juga akan lebih banyak.

REFERENSI

[1] http://kodu.ut.ee/~lipmaa/crypto/link/block/cryptanalysis.php

[2] http://vignette2.wikia.nocookie.net/rubiks/images/9/96/Rubik's_cube_notation.jpg

[3] http://www.emc.com/emc-plus/rsa-labs/images/feistel.gif

[4] http://www.scribd.com/doc/97613764/Ringkasan-Materi-Block-Cipher#scribd

[5] Michael Shanks (May 8, 2005). "History of the Cube". Stanford University.

[6] http://www.cse.wustl.edu/~jain/cse567-06/ftp/encryption_perf/#2_5

[7] http://ilmu-kriptografi.blogspot.com/2009/05/algoritma-des-data-encryption-standart.html

[8] Slide kuliah kriptografi IF4020