OZ: Algoritma Cipher Blok Kombinasi Lai-Massey dengan Fungsi *Hash* MD5

Fahziar Riesad Wutono
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
fahziar@gmail.com

Ahmad Zaky
Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
ahmadzaky003@gmail.com

***Abstract*—Makalah ini membahas algoritma kriptografi simteris OZ yang berbasis berbasis block cipher. Algoritma ini menggunakan kunci yang berukuran 32 bit dan ukuran blok 32 bit. Algoritma ini menggunakan skema Lai-Massey dalam proses enkripsi dan dekripsinya.**

Keywords—Block cipher, Cipher, Skema Lai-Massey

I. PENDAHULUAN

Saat ini penggunaan internet sebagai media komunikasi semakin populer. Data yang ditransmisikan dari komputer pengirim pesan ke komputer penerima pesan perlu melewati beberapa komputer di jaringan internet sebagai perantara. Jika data yang ditransmisikan berupa *plaintext*, data dapat dapat disadap saat data ditransmisikan di internet. Untuk melindungi data yang ditransmisikan dari penyadapan, *plaintext* tersebut perlu dienkripsi dengan menggunakan algoritma *cipher* sebelum ditransmisikan di internet. *Paper* ini membahas mengenai algoritma *cipher* baru yang berbasis *block cipher* yaitu **OZ**. Nama OZ diperoleh dari akronim pembuatnya, yaitu Oji (nama panggilan akrab Fahziar) dan Zaky.

Algoritma OZ menggunakan skema yang sama dengan algoritma IDEA. Algoritma IDEA menggunakan skema Lai-Massey dalam enkripsi dan dekripsinya [1]. Algoritma OZ juga menggunakan skema Lai-Massey dalam proses enkripsi dan dekripsi

II. DASAR TEORI

A. Prinsip Diffusion dan Confusion

Algoritma yang aman memenuhi prinsip *Confusion* dan *Diffusion*. Prinsip *confusion* dan *diffusion* mempersulit analisa statistik pada *ciphertext* [2].

Prinsip *confusion* berarti menghilangkan pola-pola statistik pada *ciphertext*. Prinsip *confusion* dapat diimplementasi dengan menggunakan subtitusi.

Prinspi *diffusion* berarti menyebarkan pengaruh satu bit pada *plaintext* dan kunci ke sebanyak mungkin bit *ciphertext*. Perubahan satu bit pada *plaintext* dapat mengubah banyak bit pada *ciphertext*. Prinsip ini dapat diimplementasi dengan menggunakan permutasi.

B. Skema Lai-Massey

Skema Lai-Massey merupakan skema yang mirip dengan skema Feistel. Keduanya sama-sama dijalankan dalam beberapa ronde.

Terdapat dua buah fungsi di skema Lai-Massey yang perlu didesain, yaitu *round function* (fungsi F) dan *half round function* (fungsi H). Fungsi H harus memiliki invers. Berbeda dengan fungsi H, fungsi F tidak perlu memiliki invers.

Terdapat dua buah fungsi di skema Lai-Massey yang perlu didesain, yaitu fungsi F dan fungsi H. Fungsi H harus memiliki invers. Berbeda dengan fungsi H, fungsi F tidak perlu memiliki invers.

Setiap ronde memiliki kunci sendiri. Kunci ini digunakan sebagai masukan fungsi F. Fungsi H tidak diwajibkan untuk menggunakan kunci ronde sebagai masukan.

Confusion dan *diffusion* ditentukan oleh fungsi F dan fungsi H. Fungsi F dan fungsi H yang bagus memberikan efek *confusion* dan *difusion* yang bagus.

III. METODE YANG DIUSULKAN

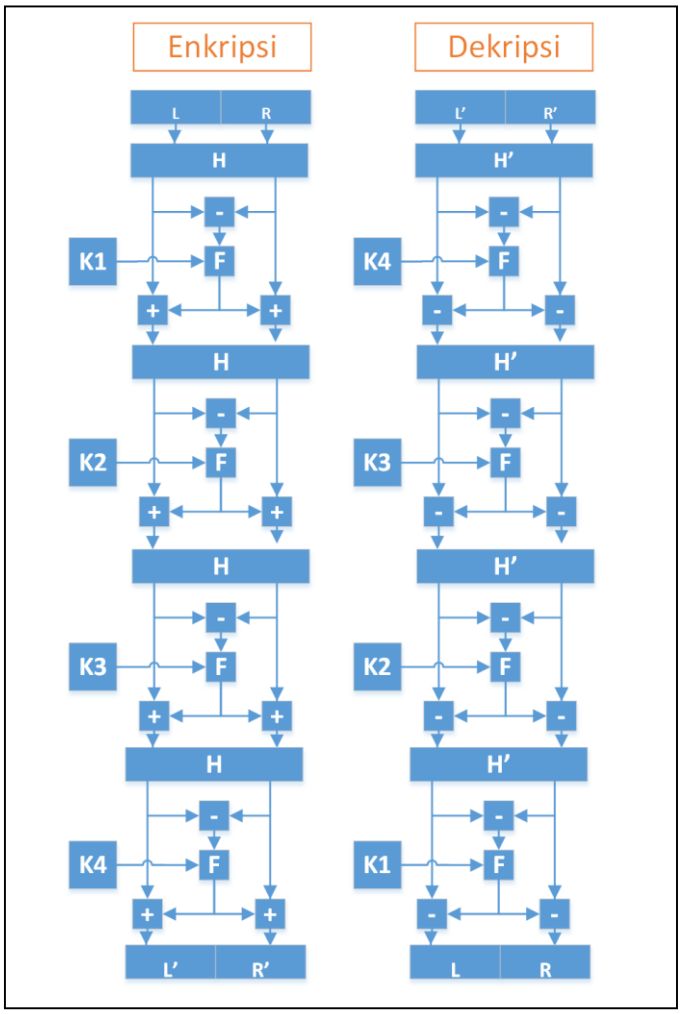
Prinsip konfusi Shannon dapat dicapai dengan menggunakan struktur Feistel atau yang sejenis. Dalam OZ, digunakan skema Lai-Massey sebanyak 4 putaran. Dalam *round function*, algoritma ini memakai fungsi *hash* MD5 dan P-Box yang akan dijelaskan di bawah ini.

Besar blok plainteks dan cipherteks adalah 32 Byte dan panjang kunci juga sebesar 32 Byte. Pemilihan besar blok didasarkan pada dua hal, yaitu

- Hasil *hash* MD5 adalah sepanjang 128 bit atau 16 Byte. Oleh karena itu, panjang blok harus dua kali panjang tersebut.
- P-Box memanfaatkan kunci internal sebagai indeks yang akan ditukarkan. Selain itu, untuk mempermutasikan sebuah larik sepanjang *n*, diperlukan paling tidak *n* buah pertukaran agar hasilnya benar-benar acak.

A. Lai-Massey Schema

Berbeda dengan Feistel, skema Lai-Massey menambahkan suatu fungsi H yang disebut dengan *half-round function*. Fungsi H ini harus dapat diinversikan karena dibutuhkan dalam proses dekripsi.

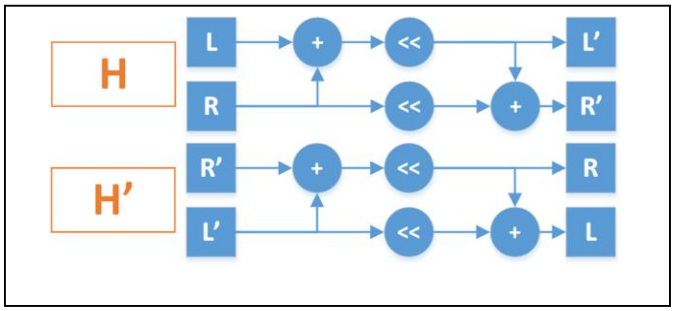


Gambar 1. Skema Enkripsi dan Dekripsi

B. *Half-round Function*

Fungsi H menerima dua masukan dan menghasilkan dua keluaran. Tiap masukan dan keluaran adalah sepanjang setengah dari panjang blok yang digunakan. Fungsi H bisa dikatakan sebagai pengganti prosedur pertukaran pada struktur Feistel. Oleh karena itu, kedua masukan harus saling mempengaruhi satu sama lain.

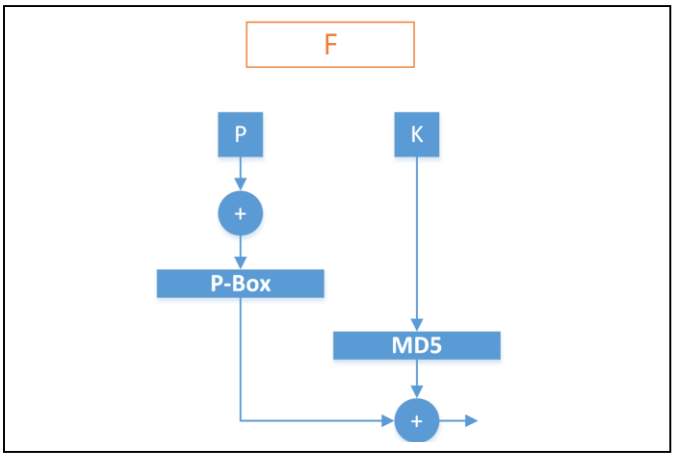
Skema fungsi H dan inversnya dapat dilihat di gambar 3



Gambar 2. Skema *half round function*

C. *Round Function*

Salah satu keunggulan dari *round function* pada skema Lai-Massey adalah fungsi ini tidak harus mempunyai invers. Proses enkripsi dan dekripsi menggunakan fungsi yang sama. Oleh karena itu, enkripsi satu arah seperti *hash* dapat diaplikasikan di sini. Kami menggunakan MD5 sebagai fungsi *hash*. Selain itu, keamanan ditingkatkan dengan menggunakan P-Box yang akan dijelaskan pada bagian selanjutnya.



Gambar 3. Skema *round function*

D. *P-Box*

Inti dari P-Box adalah mempermutasikan masukan sehingga menjadi acak. Namun, berbeda dengan permutasi pada algoritma cipher blok seperti DES, di mana bagaimana mempermutasikan maskan telah ditentukan sebelumnya, OZ menggunakan kunci internal untuk menentukan indeks mana yang akan dipermutasikan.

Masukan dapat direpresentasikan sebagai sebuah larik dengan indeks 0 sampai $n - 1$, di mana n adalah banyak Byte dalam masukan. Kunci juga dapat dianggap demikian.

Tiap elemen kunci dapat direpresentasikan dalam bentuk heksadesimal, misalnya 100 dalam desimal sama dengan 64 pada heksadesimal. Kedua digit pada representasi heksadesimal kunci inilah yang menentukan indeks yang akan dipertukarkan. Pertukaran akan dilakukan sebanyak panjang kunci internal.

Karena terdapat 16 buah digit heksadesimal, yaitu dari 0 hingga F, maka hal ini menyugestikan kita bahwa panjang masukan beserta kunci yang digunakan adalah sebesar 16 Byte. Angka ini sangat tepat karena:

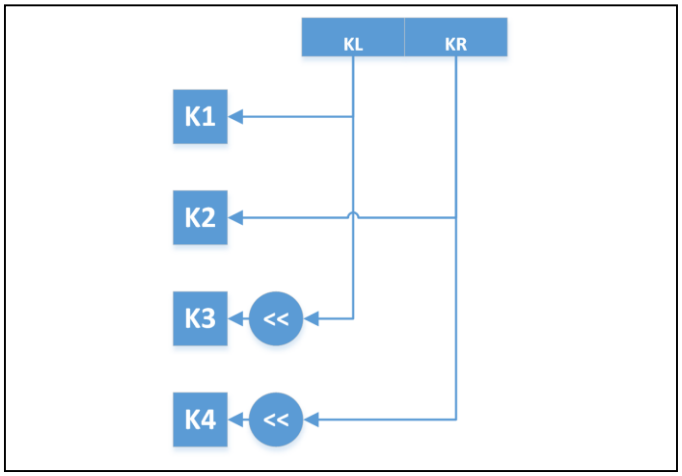
- Panjang larik sesuai dengan besar indeks yang dimungkinkan.
- Untuk menukarkan n buah elemen larik, diperlukan paling tidak n buah proses pertukaran agar memastikan isinya benar-benar acak. Kurang dari n buah pertukaran berarti tidak semua elemen saling berpengaruh satu sama lain, sedangkan lebih dari n buah pertukaran tidak diperlukan. Karena panjang kunci adalah 16 Byte, maka hal ini sesuai dengan konsep tersebut.

- Fungsi *hash* MD5 juga menghasilkan keluaran sebesar 16 Byte.

E. Pembangkitan Kunci Internal

Panjang kunci adalah sebesar 32 Byte. Kunci ini kemudian dibagi menjadi dua bagian, masing-masing sebesar 16 Byte. Bagian pertama menentukan kunci internal pertama, sedangkan bagian kedua menentukan kunci internal kedua hingga keempat.

Skema lengkap pembangkitan kunci internal diilustrasikan di gambar 4.



Gambar 4. Skema pembangkitan kunci internal

IV. EKSPERIMEN DAN PEMBAHASAN

OZ diimplementasikan dalam bahasa Java. Diimplementasikan pula berbagai mode cipher blok, yaitu ECB, CBC, dan CFB. Algoritma ini akan dicoba untuk berbagai kasus kunci dan plainteks. Pada bagian terakhir, terdapat analisis performansi dari algoritma ini.

A. Hasil Enkripsi dan Dekripsi Plainteks Acak

Berikut adalah contoh hasil enkripsi tiga buah plainteks acak dengan kunci yang sama. Masing-masing sepanjang satu blok, yaitu 32 Byte. Tiap data direpresentasikan per Byte, dalam bentuk heksadesimal.

```
Key    : d3 fb 63 8e 4c 22 ff 3f 25 63 73 ee
55 c6 b4 02 d6 ec 8d 42 d2 80 70 9e e5 7d 6f
6c 2a 97 79 a7

Plain1 : 03 9f 29 e9 8f 6f 22 80 b1 71 2a 5d
68 4c 57 ad b1 7b 36 cb 8e 74 17 76 06 5a 3b
e6 ef d3 b4 84

Cipher1: c5 57 ae 36 4d c5 b1 3a 29 71 4e 38
24 17 1c 5e 32 84 75 09 79 e5 fe 5c 0d d9 3e
e0 45 fe 92 51

Plain2 : 15 42 05 7b 90 6c 8f a1 a4 2e e8 30
0e b6 1c 6d d4 7f f1 f2 0c da c4 eb 3d 00 6e
8a 28 a6 78 28

Cipher2: 77 18 50 b6 f8 55 36 e4 80 96 0f d1
86 3e 2b d2 1a 32 6d d2 2a 19 76 25 30 6b 99
0d db 60 4f ee

Plain3 : d9 7e e1 f8 76 a0 81 6a 69 cc 43 72
93 f9 e2 14 68 bc 76 9f c4 af 2d 0b dc 1c 22
33 c3 28 de 09

Cipher3: 83 74 ed c6 f1 8c 9c 50 36 1e 61 38
5b 0d 8e ea 07 cc 11 6b 43 b4 7f 1b 82 91 8b
f7 7c 56 e7 a2
```

Berikut adalah hasil dekripsi ketiga cipherteks yang diperoleh di atas dengan kunci yang sama.

```
Key    : d3 fb 63 8e 4c 22 ff 3f 25 63 73 ee
55 c6 b4 02 d6 ec 8d 42 d2 80 70 9e e5 7d 6f
6c 2a 97 79 a7

Cipher1: c5 57 ae 36 4d c5 b1 3a 29 71 4e 38
24 17 1c 5e 32 84 75 09 79 e5 fe 5c 0d d9 3e
e0 45 fe 92 51

Plain1 : 03 9f 29 e9 8f 6f 22 80 b1 71 2a 5d
68 4c 57 ad b1 7b 36 cb 8e 74 17 76 06 5a 3b
e6 ef d3 b4 84

Cipher2: 77 18 50 b6 f8 55 36 e4 80 96 0f d1
86 3e 2b d2 1a 32 6d d2 2a 19 76 25 30 6b 99
0d db 60 4f ee

Plain2 : 15 42 05 7b 90 6c 8f a1 a4 2e e8 30
0e b6 1c 6d d4 7f f1 f2 0c da c4 eb 3d 00 6e
8a 28 a6 78 28

Cipher3: 83 74 ed c6 f1 8c 9c 50 36 1e 61 38
5b 0d 8e ea 07 cc 11 6b 43 b4 7f 1b 82 91 8b
f7 7c 56 e7 a2

Plain3 : d9 7e e1 f8 76 a0 81 6a 69 cc 43 72
93 f9 e2 14 68 bc 76 9f c4 af 2d 0b dc 1c 22
33 c3 28 de 09
```

Dapat dilihat bahwa plainteks yang dihasilkan pada proses dekripsi sama dengan plainteks awal. Jadi, algoritma tersebut telah mengenkripsi dan mendekripsi dengan benar.

B. Hasil Enkripsi Plainteks atau Kunci yang Mirip

Sekarang akan dilakukan enkripsi dua plainteks yang berbeda hanya satu bit. Akan dilihat perbedaan hasil enkripsi kedua plainteks tersebut.

```
Key    : d5 9d a5 de b5 d8 e6 dd 3d db 71 47
13 66 01 9b 56 84 79 28 3f 90 46 38 de b2 c1
59 d2 53 98 1d

Plain1 : ac 67 0f 33 eb 82 5a 7f 55 b0 1d fe
af 1a 04 d0 7c f8 a7 8a fa 9e 3e 17 dc 36 8e
d9 37 6d 63 4e

Cipher1: 03 1a 8e 89 fc b6 39 e0 04 f6 9d ea
b7 cd 3e b7 ea 39 18 ca 09 87 74 b8 6f b0 6d
11 dc 9a 3f 8d
```

```
Plain2 : ad 67 0f 33 eb 82 5a 7f 55 b0 1d fe
af 1a 04 d0 7c f8 a7 8a fa 9e 3e 17 dc 36 8e
d9 37 6d 63 4e

Cipher2: 4d e0 97 8c c4 80 38 e4 70 03 93 16
c2 5e f1 b7 a1 e6 3f ce a3 c4 30 1f 11 b4 bb
3a a9 2a 07 8f
```

Dapat dilihat bahwa kedua cipherteks berbeda jauh. Jika sekarang dilakukan enkripsi plainteks yang sama, tapi dengan kunci yang berbeda satu bit, maka berikut adalah hasilnya.

```
Key1   : e8 d2 c5 32 b5 31 9b ea 7c 10 d1 01
54 73 86 c2 3d 1c d2 60 4e 1c 94 fb 44 ec ab
46 4d 81 cf ae

Plain1 : de 12 b3 37 c8 0c b8 9c 1b cc 16 ec
65 68 15 0b 85 05 8b 46 8e 99 b6 d6 50 c4 3e
34 d7 73 6e 01

Cipher1: 1c 71 1e dd 2c 2e 0e c9 7f cf a2 89
a3 95 c7 e1 f8 4b 17 37 27 db f4 08 d7 57 01
04 18 b9 4b cb
```

```
Key2   : e8 d2 c5 33 b5 31 9b ea 7c 10 d1 01
54 73 86 c2 3d 1c d2 60 4e 1c 94 fb 44 ec ab
46 4d 81 cf ae

Plain2 : de 12 b3 37 c8 0c b8 9c 1b cc 16 ec
65 68 15 0b 85 05 8b 46 8e 99 b6 d6 50 c4 3e
34 d7 73 6e 01

Cipher2: 5d 3e 9b 5c bf d9 82 80 d2 e3 9f df
ce c6 0c 3a 0c 17 0b 61 04 f7 5a 96 f7 6f ab
d7 f1 de 4e 62
```

Sama seperti sebelumnya, kedua cipherteks yang dihasilkan berbeda jauh. Maka, prinsip konfusi telah dicapai oleh algoritma ini.

C. *Enkripsi Plainteks atau Kunci Null*

Plainteks atau kunci null adalah kondisi di mana seluruh plainteks atau kunci seluruhnya berisi Byte 0 atau null. Berikut adalah hasil enkripsinya.

```
Key1   : bf d0 b1 22 91 bf 5c b6 f6 62 31 20
67 5d 28 1d 0d 96 b1 d9 d0 33 62 e6 7d ba 3f
f5 f7 93 35 ab

Plain1 : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00

Cipher1: 5c 42 72 57 c7 09 81 63 cd 00 19 06
30 62 dc c3 a8 d0 83 ea 92 23 42 11 ee ff 15
64 15 fe 88 ba
```

```
Key2   : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00

Plain2 : e5 4f c9 45 12 0e 54 20 84 de e2 d7
6c 91 82 88 2b ea 25 72 f9 65 ab 20 f0 30 a7
07 46 57 ba 92

Cipher2: 62 a4 73 50 42 3a a2 8e f2 fe 64 fb
e3 ff 6a 26 af 21 2f cb bd 5b 51 23 c1 4b 34
0c 59 b9 9e 50
```

```
Key3   : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00

Plain3 : 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00

Cipher3: 1d 60 d1 56 de 64 88 ed e2 87 b1 c8
84 35 ba 78 45 47 3f 66 bd 68 77 0b 58 eb c9
7f da ea bf 88
```

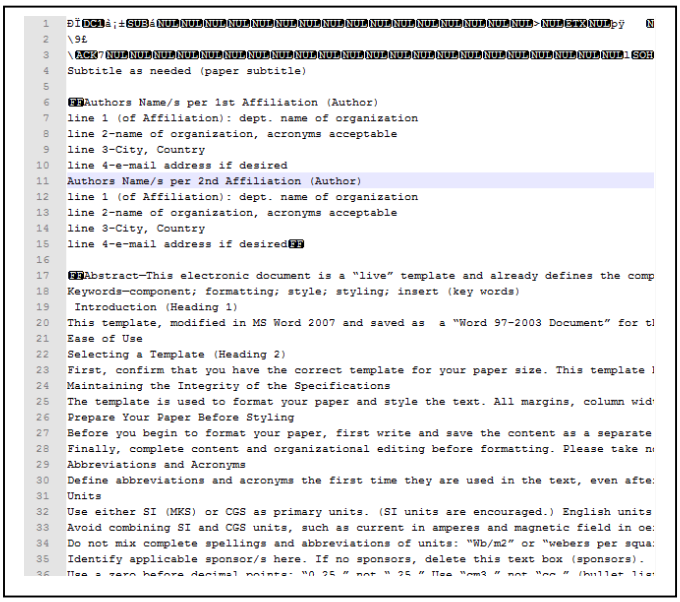
Saat plainteks null, cipherteks yang dihasilkan tetap acak. Saat kunci null, cipherteks tetap berbeda jauh dengan plainteks. Bahkan saat plainteks dan kunci keduanya null, cipherteks yang dihasilkan tetap acak. Hal ini karena adanya fungsi *hash* MD5 yang menyebabkan keluaran akan acak apapun kunci atau plainteksnya.

D. *Enkripsi dan Dekripsi Berkas*

Akan dilakukan enkripsi dan dekripsi berkas dengan format tertentu dengan ketiga mode cipher blok. Kunci yang digunakan sama untuk ketiganya, yaitu "halohalobandung". Karena panjang kunci kurang dari 32 Byte, maka kunci tersebut ditambahkan dengan Byte null hingga sepanjang 32 Byte.

1) *Berkas Asli*

Berkas yang akan digunakan adalah template makalah IEEE, "IEEE Paper Template.doc". Panjang berkas adalah 57.344 Byte. Berikut adalah bagaimana isi berkas tersebut jika dibuka dengan *notepad*.



Gambar 5. *Screenshot* berkas asli dibuka dengan aplikasi notepad++.

2) Enkripsi dengan Mode ECB

Berikut adalah hasil enkripsi berkas tersebut dengan mode ECB (*Electronic Code Book).*

Gambar 6. *Screenshot* berkas yang dienkripsi dengan mode ECB

3) Enkripsi dengan Mode CBC

Berikut adalah hasil enkripsi berkas tersebut dengan mode CBC (*Cipher Block Chaining).*

Gambar 7. *Screenshot* berkas yang dienkripsi dengan mode CBC

4) Enkripsi dengan Mode CFB

Berikut adalah hasil enkripsi berkas tersebut dengan mode CFB (*Cipher Feedback)*

Gambar 8. *Screenshot* berkas yang dienkripsi dengan mode CFB

Hasil dekripsi ketiga cipherteks tersebut berhasil mengembalikan dokumen awal. Dapat dilihat bahwa secara sekilas, hasil enkripsi sama sekali tidak menggambarkan plainteks atau kunci yang dipakai.

E. Analisis Performansi

Pada analisis ini dilakukan enkripsi dan dekripsi data sebesar 1 MB sebanyak 100 kali. Diambil waktu rata-rata dari seluruh percobaan tersebut.

TABEL 1. KECEPATAN ENKRIPSI DAN DEKRIPSI

Mode	Enkripsi		Dekripsi	
	Waktu	Kecepatan	Waktu	Kecepatan
ECB	289 ms	3.46 MB/s	298 ms	3.36 MB/s
CBC	289 ms	3.45 MB/s	295 ms	3.90 MB/s
CFB	295 ms	3.39 MB/s	290 ms	3.45 MB/s

Performa di atas relatif cepat karena implementasi menggunakan bahasa Java, yang cukup lambat dibandingkan bahasa lain seperti C atau C++.

V. ANALISIS KEAMANAN

Analisis akan didasarkan pada kriptanalisis yang umum, yaitu *brute force attack, frequency analysis*, dan *known plaintext attack.*

A. Brute Force Attack

Lamanya pencarian kunci dengan *brute force attack* sebagian besar tergantung oleh panjang kunci. Karena panjang kunci sudah ditetapkan sepanjang 32 Byte atau 256 bit, maka ada sebanyak 2^{256} buah kunci berbeda, atau sebanyak 1.1579 ×

10^{77} kunci berbeda. Tabel berikut menggambarkan berapa lama kunci dapat dipecahkan jika seluruh kunci harus dicoba.

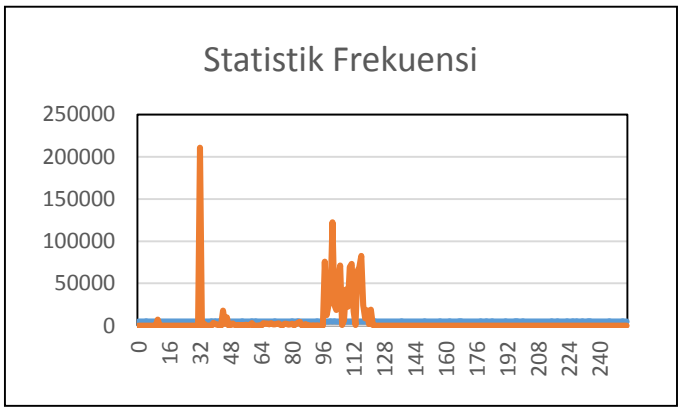
TABEL 2. WAKTU YANG DIBUTUHKAN OLEH SERANGAN BRUTE FORCE

Kecepatan	**Waktu**
1 kunci / detik	3.6717×10^{69} tahun
10^3 kunci / detik	3.6717×10^{66} tahun
10^6 kunci / detik	3.6717×10^{63} tahun
10^9 kunci / detik	3.6717×10^{60} tahun
10^{12} kunci / detik	3.6717×10^{57} tahun

Waktu yang dibutuhkan tentu saja tidak memungkinkan. Jadi, algoritma ini aman dari *brute force attack* naif.

B. *Frequency Analysis*

Berikut akan dilakukan analisis frekuensi terhadap enkripsi plainteks berbahasa Inggris yang diperoleh dari Project Gutenberg [3]. Berikut adalah statistik frekuensi kemunculan tiap Byte pada plainteks dan cipherteks hasil enkripsi.



Gambar 9. Grafik statistik frekuensi

Angka pada sumbu x adalah byte (0 – 255), dan angka pada sumbu y adalah kemunculannya. Frekuensi pada plainteks ditunjukkan oleh garis oranye, dan frekuensi pada cipherteks ditunjukkan oleh garis biru.

Pada plainteks, kemunculan tertinggi adalah pada karakter spasi dan karakter-karakter yang umum muncul pada bahasa Inggris seperti E, T, dan A. Sedangkan pada cipherteks, kemunculan tiap byte merata. Kemunculan minimum adalah 4767 kali dan kemunculan maksimum adalah 5196, dengan standar deviasi sebesar 75,7096. Hal ini mengakibatkan analisis frekuensi menjadi sulit dilakukan.

C. *Known Plaintext Attack*

Serangan dengan jenis ini memerlukan analisis yang lebih mendalam yang berbeda untuk setiap cipher blok. Keamanan OZ ditentukan oleh struktur utamanya, yaitu skema Lai-Massey dan fungsi *round* dan *half-round* yang terdapat di dalamnya. Bagaimana membangkitkan kunci internal juga menjadi faktor dalam hal ini.

Fungsi *half-round* dan pembangkit kunci internal OZ sangatlah sederhana, hanya melibatkan operasi XOR dan *binary shifting* sederhana. Kekuatan utama OZ adalah pada fungsi *round*, yang ditopang oleh P-Box dan fungsi *hash* MD5.

Walaupun seseorang mengetahui masukan dan keluaran dari satu *round* pada skema utama Lai-Massey, menemukan kuncinya sangatlah sulit. Walaupun belakangan ini MD5 sudah dikenal tidak aman lagi karena menemukan *collision* dari MD5 mudah, tetapi keluaran dari tiap *round function* telah di-XOR-kan dengan hasil dari P-Box. P-Box di sini juga tidak seperti P-Box pada umumnya, di mana pasangan yang ditukarkan telah ditentukan sebelumnya. P-Box di sini ditentukan oleh kunci internal.

VI. KESIMPULAN DAN SARAN

Terdapat banyak sekali algoritma cipher blok yang telah dibuat oleh para kriptografer. OZ adalah salah satunya. Bahkan algoritma *hashing* yang sudah dikenal tidak aman seperti MD5 bisa dipakai untuk menghasilkan algoritma cipher blok yang aman setelah dikombinasikan dengan struktur Lai-Massey dan P-Box.

REFERENCES

[1] Xuejia Lai, “On the design and security of block ciphers”, Zurich:Swiss Federal Institute of Technology, 1992

[2] Claude E. Shannon, "Communication theory of secrecy systems", Bell System Technical Journal, vol. 28-4, page 656–715, 1949

[3] Long, William J., “English Literature”. http://www.gutenberg.org/files/10609/10609-h/10609-h.htm. Diakses pada 17 Maret 2015 pukul 21.08