Chill: Algoritma *Block Cipher*

Achmad Fahrurrozi Maskur-13515026[1] Diki Ardian Wirsandi-13515092[2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]fahrurrozi31@gmail.com [2]dikiwirasandi@gmail.com

Abstrak—Diajukan sebuah algoritma cipher blok baru yang dinamai Chill Cipher. Algoritma Chill Cipher melakukan pemrosesan data dengan ukuran blok sebesar 256 bit atau 32 *bytes*. Kelebihan dari algoritma Chill Cipher adalah operasi-operasi yang digunakan pada pemrosesannya cukup sederhana namun menghasilkan enkripsi yang aman berdasarkan beberapa eksperimen dan analisis yang dilakukan. Selain operasi yang sederhana, operasinya tergolong unik karena belum pernah dipakai di algoritma *block cipher* yang sudah ada. Operasi tersebut diantaranya SubX-, SubX+ dan L Transposition.

Kata Kunci—Cipher Blok, Pesan, Kunci, Cipherteks, Jaringan Feistel, Transposisi, Substitusi, Fungsi Putaran, Kunci Putaran

I. Pendahuluan

Di era modern ini, komunikasi digital sangatlah penting khususnya dalam pengiriman pesan rahasia yang tidak boleh diketahui oleh pihak lain. Hal tersebut bertentangan dengan jaringan internet yang merupakan saluran publik yang digunakan oleh jutaan orang. Dampak yang terjadi apabila pesan tersebut dapat diketahui oleh semua orang yaitu adanya serangan. Serangan tersebut dapat berupa pengumpulan data-data penting hingga melakukan perubahan pesan.

Salah satu upaya pencegahan pesan yang publik yaitu dengan menggunakan algoritma kriptografi. Dengan kriptografi, terlebih dahulu dilakukan enkripsi pesan yang akan dikirimkan. Enkripsi pesan digunakan dengan sebuah kunci simetris yang hanya diketahui oleh pengirim dan penerima pesan. Penerima pesan dapat melakukan pen-dekripsian pesan sehingga pesan kembali menjadi pesan asli.

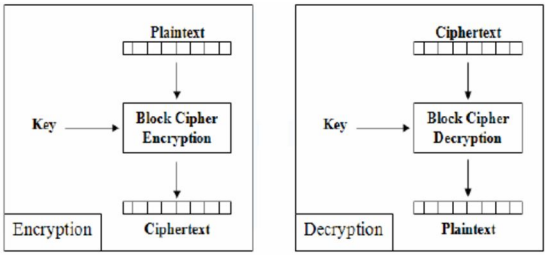
Seiring dengan perkembangan zaman, algoritma kriptografi juga semakin berkembang. Kriptografi yang memanfaatkan komputasi komputer disebut dengan kriptografi modern. Algoritma kriptografi modern memanfaatkan komputasi dari sebuah komputer dengan mengoperasikan pesan dalam satuan bit atau biner.

Contoh dari pengaplikasian kriptografi modern sangatlah banyak, diantaranya dalam komunikasi internet yang aman yang disebut dengan HTTP Secure, komunikasi perbankan, dan lain-lain. Salah satu metode yang digunakan pada algoritma kriptografi modern yaitu *block cipher*. *Block cipher* melakukan enkripsi dengan terlebih dahulu membagi pesan menjadi beberapa blok dengan panjang yang sama. Contoh algoritma yang menggunakan metode *block cipher* yaitu AES, DES, Twofish, dan lain-lain.

II. Dasar Teori

A. *Block Cipher*

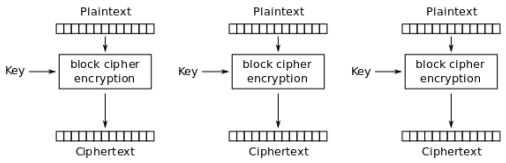
Block cipher merupakan suatu algoritma kriptografi modern yang melakukan operasi data per blok, dengan panjang blok yang sudah ditentukan oleh algoritma itu sendiri. Pemrosesan tiap blok dilakukan dengan kunci simetris. Apabila panjang pesan bukan merupakan kelipatan dari panjang blok, akan dilakukan *padding* sehingga panjang pesan merupakan kelipatan dari panjang blok. Hasil dari pemrosesan blok yaitu cipherteks dengan panjang yang sama seperti panjang blok pesan.



Gambar 1. Operasi enkripsi dan dekripsi *block cipher*.

Block cipher memiliki lima mode operasi yang beragam[2], diantaranya yaitu Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), dan mode Counter (CTR). Penjelasan tiap mode adalah sebagai berikut.

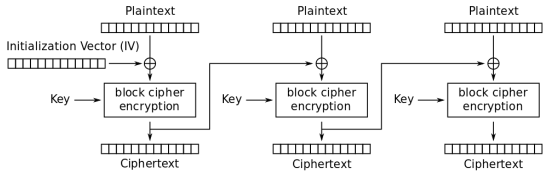
1. Electronic Code Book (ECB)
   Operasi ECB dilakukan dengan melakukan enkripsi serta dekripsi secara independen untuk tiap blok. Proses enkripsi serta dekripsi dilakukan dengan suatu fungsi yang menerima parameter berupa blok pesan serta kunci.



Gambar 2. Operasi *block cipher* dengan mode ECB.

Hasil dari proses tersebut tidak akan mempengaruhi hasil operasi blok yang lain. Karena hasil tersebut tidak memengaruhi blok lain, maka blok pesan yang sama akan menghasilkan cipherteks yang juga sama. Hal ini menyebabkan proses pemecahan pesan dapat dilakukan dengan metode kriptanalisis dengan mendeteksi kemunculan cipherteks yang sama.

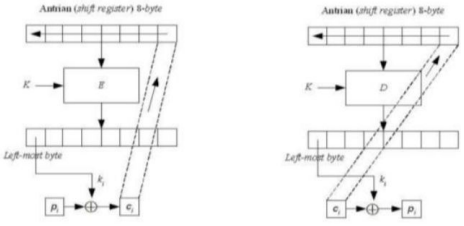
2. Cipher Block Chaining (CBC)
   Operasi CBC dilakukan dengan memanfaatkan hasil cipherteks operasi blok sebelumnya. Pada saat operasi blok yang pertama, digunakan sebuah *initialization vector* (IV) yang dibangkitkan secara acak. Cipherteks yang dihasilkan kemudian di-XORkan dengan blok pada proses selanjutnya. Hal ini dilakukan agar blok pesan yang sama akan menghasilkan cipherteks yang berbeda, sehingga proses kriptanalisis menjadi lebih sulit.



Gambar 3. Operasi *block cipher* dengan mode CBC.

Namun apabila terdapat kesalahan walau hanya satu bit pada proses enkripsi, hal tersebut juga akan berdampak pada proses enkripsi blok-blok selanjutnya.

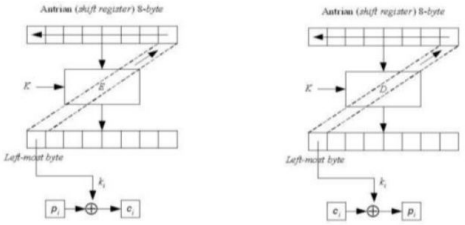
3. Cipher Feedback (CFB)
   Operasi CFB dilakukan dengan mengoperasikan unit yang lebih kecil dari ukuran blok, hal ini mengatasi kelemahan pada mode ECB dan CBC apabila ukuran blok yang diterima (pada proses komunikasi data) belum lengkap. Unit yang diproses dapat berupa n-bit dengan n tidak melebihi ukuran blok.



Gambar 4. Operasi *block cipher* dengan mode CFB.

Karena konsep yang dimiliki oleh CFB juga *chaining*, maka kesalahan satu bit pada blok pesan akan merambat pada blok-blok cipherteks selanjutnya pada proses enkripsi. Sebaliknya pada proses dekripsi.

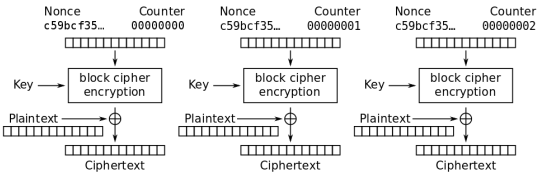
4. Output Feedback (OFB)
   Operasi OFB dilakukan mirip dengan operasi pada CFB. Pembedanya yaitu n-bit hasil enkripsi disalin menjadi elemen paling kanan antrian. Contoh OFB 8-bit pada blok 64-bit yaitu sebagai berikut.



Gambar 5. Operasi *block cipher* dengan mode OFB.

Apabila terdapat kesalahan satu bit pada blok pesan, hanya akan memengaruhi blok cipherteks yang berkoresponden saja, sama halnya pada proses dekripsi.

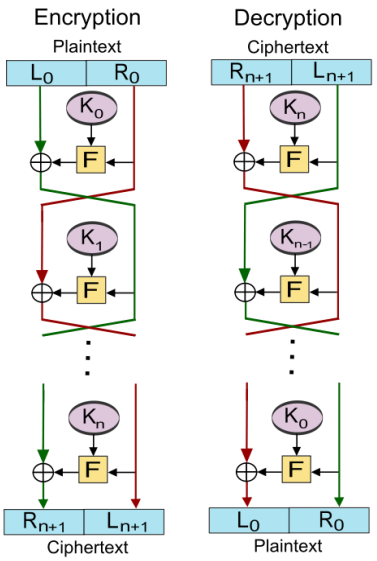
5. Counter (CTR)
   Operasi CTR dilakukan mirip dengan ECB, hanya saja operasi CTR menggunakan sebuah *nonce* dan juga nilai *counter*. Counter merupakan sebuah nilai blok bit yang ukurannya sama dengan ukuran blok pesan. Nilai tersebut harus berbeda pada tiap blok yang dienkripsi, yaitu dengan menaikkan nilainya satu.



Gambar 6. Operasi *block cipher* dengan mode CTR.

B. Struktur Feistel

Struktur Feistel, atau disebut juga Feistel Cipher, merupakan salah satu struktur yang digunakan untuk membangun *block cipher*. Struktur ini merupakan struktur yang simetris. Sifat simetris ini memberikan keuntungan dalam pembangunan *block cipher* dimana proses enkripsi dan dekripsi yang dilakukan tidak berbeda jauh atau bahkan sama. Proses enkripsi dan dekripsi yang hampir sama tentunya akan meminimalkan kode program pada implementasinya[1].



Gambar 7. Enkripsi dan dekripsi pada struktur Feistel.

Secara garis besar, proses enkripsi pada struktur Feistel adalah:

i. Pisahkan blok pesan menjadi dua bagian, yaitu blok kiri (L) dan blok kanan (R)

ii. Untuk setiap putaran $i = 0,1,2,..n$, hitung:

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

dimana **F** merupakan fungsi putaran dan $\oplus$ merupakan operasi XOR.

iii. Jika seluruh putaran sudah dioperasikan, maka dihasilkan blok cipherteks (R_{n+1}, L_{n+1}).

iv. Pemrosesan diulang dari tahap (i) untuk blok pesan selanjutnya. Cipherteks akhir merupakan penggabungan dari semua blok cipherteks secara terurut.

Proses dekripsi merupakan invers dari proses enkripsi baik secara alur dan pengoperasiannya. Perlu diketahui bahwa salah satu keuntungan dari struktur Feistel yaitu fungsi putaran **F** tidak harus dapat dibalik atau *invertible*.

C. Konfusi dan Difusi

Konfusi dan difusi merupakan properti yang digunakan dalam dunia kriptografi dalam penjaminan keamanan dari suatu cipherteks. Prinsip ini dipublikasikan dan diperkenalkan oleh Claude Shannon pada tahun 1949.

Prinsip konfusi merupakan prinsip yang menyembunyikan keterhubungan antara plainteks serta cipherteks. Hal ini dapat terjadi dengan menggunakan algoritma substitusi yang kompleks. Sedangkan prinsip difusi merupakan prinsip yang menyebarkan pengaruh satu bit plainteks ataupun kunci ke sebanyak mungkin cipherteks.

III. Rancangan Algoritma

Algoritma Chill ini dibangun dengan menerapkan struktur Feistel. Struktur yang digunakan merupakan struktur Feistel normal tanpa modifikasi. Algoritma ini beroperasi dalam satuan *byte*. Terdapat beberapa tahapan yang diimplementasi yang dijelaskan sebagai berikut.

A. Pemrosesan Awal

Terdapat beberapa pemrosesan awal terhadap pesan dan kunci sebelum dilakukan enkripsi terhadap pesan. Pemrosesan tersebut adalah sebagai berikut.

1. *Padding* pesan

Ukuran dari blok pesan yaitu 256 bit. Ukuran pesan secara keseluruhan harus merupakan kelipatan dari ukuran blok pesan, yaitu 256 bit. Jika ukuran pesan bukan merupakan kelipatan dari 256 bit, maka terdapat mekanisme penambahan terhadap isi pesan (*padding* pesan). Penambahan dilakukan hingga ukuran pesan menjadi kelipatan dari 256 bit terdekat. Metode *padding* yang digunakan adalah metode PKCS5Padding[4]. Metode PKCS5Padding menambahkan perulangan representasi heksadesimal dari jumlah *byte* yang harus ditambahkan. Sebagai contoh pada pesan berikut:

```
Pesan = "Hello"
Pesan (heksadesimal) = 48656C6C6F
```

Pesan di atas berukuran 5 *bytes*, sehingga diperlukan 27 *bytes* pesan tambahan agar menjadi kelipatan dari 256 bit (32 *bytes*). Representasi heksadesimal dari 27 adalah 1B, sehingga pesan akan di-*padding* menjadi:

```
Pesan (setelah padding):
48656C6C6F1B1B1B1B1B1B1B1B
1B1B1B1B1B1B1B1B1B1B1B1B1B
1B1B1B1B1B1B
```

2. *Padding* kunci

Ukuran kunci minimal 128 bit. Jika ukuran kunci kurang dari minimal, maka akan dilakukan *padding* atau penambahan isi dari kunci. *Padding* dilakukan hingga panjang kunci mencapai 128 bit. Metode *padding* terhadap kunci adalah dengan menambahkan karakter-karakter dari karakter kunci itu sendiri. Karakter akan dipilih secara acak yang dibangkitkan dengan *seed*. *Seed* merupakan jumlah dari nilai desimal (ascii) karakter-karakter ganjil dari kunci, yang dirumuskan sebagai berikut.

```
seed = ascii(k[1]) + ascii(k[3])+ ...
```

Jika ukuran kunci lebih dari 128 bit, maka akan dilakukan pembuangan karakter. Karakter kunci yang dibuang akan dipilih secara acak yang dibangkitkan

dengan *seed*. *Seed* merupakan jumlah dari nilai desimal (ascii) karakter-karakter genap dari kunci yang dapat dirumuskan sebagai berikut.

```
seed = ascii(k[2]) + ascii(k[4])+ …
```

B. Pembangkitan Kunci Putaran

Setelah dilakukan pemrosesan awal terhadap pesan dan kunci, sebelum melakukan pemrosesan pada struktur Feistel, dibutuhkan kunci putaran K_i yang menjadi input dari fungsi putaran ke-*i* (F) bersama dengan blok R_i. Pada algoritma Chill, jumlah putaran pada satu tahap Feistel ditentukan dengan rumus sebagai berikut.

```
jumlah putaran = 5 + (length(kunci_asli) mod 6)
```

Oleh karena jumlah putaran tidak hanya satu kali, kunci putaran harus dibangkitkan sebanyak jumlah putaran. Untuk setiap putaran ke-*i* akan digunakan kunci putaran K_i. Untuk kunci putaran berikutnya (K_{i+1}) dapat dibangkitkan dari kunci putaran K_i yang dioperasikan dengan fungsi Transformasi ke Matriks, RotMod, SubX- dan XorCol.

1. Transformasi ke Matriks
   Transformasikan kunci putaran K_i dalam bentuk matriks 4x4 dari representasi heksadesimal. Tiap elemen matriks berukuran 1 *byte*, sehingga tiap elemen terdiri dari 2 digit heksadesimal. Urutan transformasi ke matriks adalah sebagai berikut.

x_1	x_3	x_4	x_{10}
x_2	x_5	x_9	x_{11}
x_6	x_8	x_{12}	x_{15}
x_7	x_{13}	x_{14}	x_{16}

2. RotMod
   Rotasikan matriks K_i sebanyak `length(kunci_asli) mod 4` searah jarum jam.

3. SubX-
   Setelah ditransformasikan menjadi bentuk matriks dari representasi heksadesimal, kemudian substitusikan tiap elemen ke-*i* dari matriks K_i dengan rumus sebagai berikut.

   $$x_i = |(x_{i[0]} - 16) \bmod 256 - (x_{i[1]} - 1) \bmod 16|$$

   $x_{i[0]}$: digit ke-1 dari heksadesimal elemen matriks ke-i

   $x_{i[1]}$: digit ke-2 dari heksadesimal elemen matriks ke-i

4. XorCol
   Pada setiap elemen dari matriks K_i lakukan operasi XOR dengan aturan sebagai berikut.

A_1	B_1	C_1	D_1
A_2	B_2	C_2	D_2
A_3	B_3	C_3	D_3
A_4	B_4	C_4	D_4

$$A_i = A_i \oplus B_i$$
$$B_i = B_i \oplus C_i$$
$$C_i = C_i \oplus D_i$$
$$D_i = D_i \oplus A_i$$

C. Fungsi Putaran pada Struktur Feistel

Setelah kunci putaran dibangkitkan sebanyak jumlah putaran, langkah selanjutnya yaitu pemrosesan pada struktur Feistel. Pada awal pemrosesan struktur Feistel, ditentukan terlebih dahulu pemisahan blok pesan untuk setiap blok menjadi blok kanan (R) dan blok kiri (L). Ukuran dari blok kanan dan blok kiri yaitu setengah dari ukuran blok pesan (16 *bytes*). Pada algoritma Chill cipher, blok kanan awal (L_0) dari struktur Feistel adalah setengah blok bagian awal dari hasil pemisahan blok pesan, sedangkan blok kanan awal (L_0) dari struktur Feistel adalah setengah blok bagian akhir dari hasil pemisahan blok pesan. Beberapa fungsi yang diimplementasi pada fungsi putaran struktur Feistel yaitu Transformasi ke Matriks, SubX+, L Transposition, ShiftCol dan XOR with Key.

1. Transformasi ke Matriks
   Setelah ditentukan blok input dari struktur Feistel (blok L_i dan blok R_i) untuk suatu tahap, maka untuk setiap putaran ke-*i,* transformasikan L_i dan R_i dalam bentuk matriks 4x4 dari representasi heksadesimal. Tiap elemen matriks berukuran 1 *byte*, sehingga tiap elemen terdiri dari 2 digit heksadesimal. Aturan transformasi sama seperti fungsi Transformasi ke Matriks pada pembangkitan kunci putaran.

2. SubX+
   Setelah ditransformasikan menjadi bentuk matriks dari representasi heksadesimal, kemudian substitusikan tiap elemen ke-*i* dari matriks R_i dengan rumus sebagai berikut.

   $$x_i = |(x_{i[0]} - 16) \bmod 256 + (x_{i[1]} - 1) \bmod 16|$$

   $x_{i[0]}$: digit ke-1 dari heksadesimal elemen matriks ke-i

   $x_{i[1]}$: digit ke-2 dari heksadesimal elemen matriks ke-i

3. L Transposition
   Pada fungsi transposisi ini, dilakukan pertukaran posisi elemen pada matriks R_i dengan aturan sebagai berikut.

A	B	C	D
E	G	H	F
F	H	G	E
B	A	D	C

   Tukarkan posisi elemen dengan karakter yang sama seperti aturan di atas. Sebagai contoh tukarkan elemen pada posisi A ($X_{1,1}$) dengan A ($X_{4,2}$), tukarkan elemen pada posisi B ($X_{1,2}$) dengan B ($X_{4,1}$) dan seterusnya.

4. ShiftCol
   ShiftCol melakukan pergeseran elemen tiap kolom ke-*i* pada matriks R_i sebanyak modulo 4 dari hasil penjumlahan elemen masing-masing kolom.

$$putaran_{i,} = (X_{1,i} + X_{2,i} + X_{3,i} + X_{4,i}) \bmod 4$$

   Aturan arah pergeseran tiap kolom adalah sebagai berikut.

↑	↓	↑	↓
↑	↓	↑	↓
↑	↓	↑	↓
↑	↓	↑	↓

   Pergeseran bersifat sirkular, artinya elemen jika pada baris ke-1 digeser ke arah atas akan pindah ke baris ke-4 dan seterusnya.

5. XOR with Key
   Operasi terakhir pada suatu putaran ke-*i* yaitu melakukan operasi XOR terhadap blok kanan (R_i) dan kunci putaran ke-*i* (K_i). Karena R_i dan K_i direpresentasikan dalam matriks, maka operasi XOR dilakukan untuk setiap elemen matriks dengan posisi yang sama dari kedua matriks tersebut.

Untuk putaran selanjutnya, akan dilakukan pembaruan nilai dari blok kanan serta blok kiri sesuai dengan struktur Feistel.

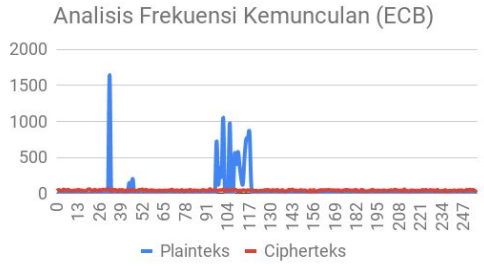
$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

Kemudian diterapkan kembali operasi fungsi putaran di atas dimulai dari fungsi SubX+ dan seterusnya (Transformasi ke Matriks tidak perrlu dilakukan kembali karena L_{i+1} dan R_{i+1} sudah dalam bentuk matriks..

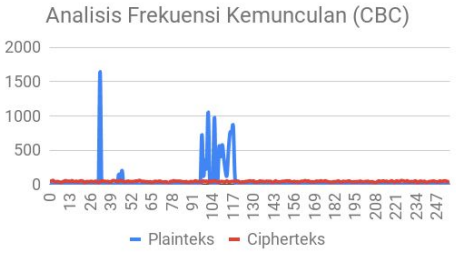
IV. Percobaan dan Analisis

Pada algoritma Chill yang telah dirancang dengan bahasa pemrograman Python, dilakukan beberapa percobaan serta analisis untuk mengetahui seberapa efektif dan efisien algoritma ini. Percobaan dilakukan dengan mode ECB, CBC, CFB, OFB, dan CTR. Proses pengujian menggunakan data besar dengan ukuran 11323 bytes.

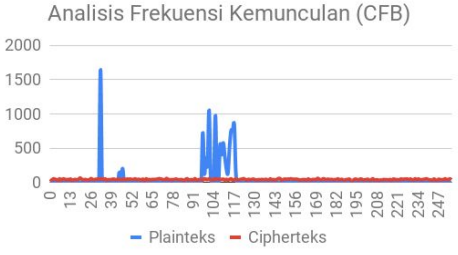
Untuk setiap mode enkripsi, akan dilakukan analisis frekuensi kemunculan pada pesan dan cipherteks. Hasil cipherteks yang didapatkan relatif merata, dibandingkan dengan plainteks yang terdapat kata A, T, E, spasi, yang sangat banyak. Sehingga dapat dikatakan algoritma Chill sudah menerapkan properti konfusi.



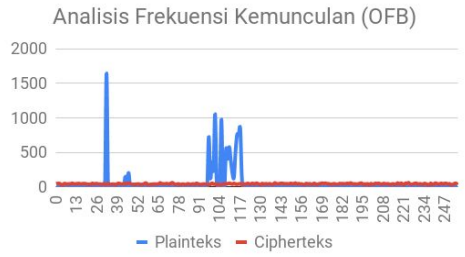
Gambar 8. Analisis Frekuensi Kemunculan (ECB).



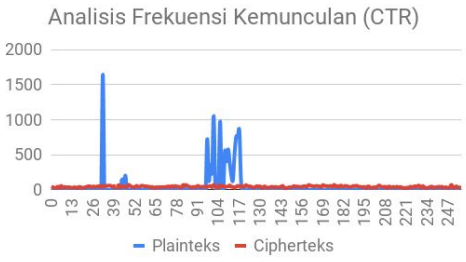
Gambar 9. Analisis Frekuensi Kemunculan (CBC).



Gambar 10. Analisis Frekuensi Kemunculan (CFB).



Gambar 11. Analisis Frekuensi Kemunculan (OFB).



Gambar 12. Analisis Frekuensi Kemunculan (CTR).

Selain analisis frekuensi kemunculan, juga dilakukan percobaan perubahan kunci sebanyak 1 bit. Hal ini ditujukan untuk mengamati perubahan cipherteks dari kunci yang hanya berbeda 1 bit. Kunci yang diganti adalah "itbganeshasepulu" menjadi "itbganeshasepulv". Diperoleh hasil yang cukup memuaskan karena hampir tidak ada byte yang sama antara kedua cipherteks untuk tiap mode yang digunakan. Sehingga dapat dikatakan algoritma Chill sudah menerapkan properti difusi.

```
69 74 62 67 61 6e 65 73 68 61 73 65 70 75 6c
75
```

Representasi hex kunci semula.

```
69 74 62 67 61 6e 65 73 68 61 73 65 70 75 6c
76
```

Representasi hex kunci setelah diganti.

1. Electronic Code Book (ECB)

```
e6 31 e3 0b 43 62 3c ba 0f 24 d8 93 de b3 60 50
e3 12 af e8 79 15 05 65 e3 5b e9 86 6b 67 ad b6
0e b1 9a 74 00 b3 ab c3 f3 a2 f1 a2 22 2a e2 ee
4a 0b 22 0d 16 87 71 a9 67 ef 79 5b f4 ff eb cb
```

Representasi hex 64 bytes cipherteks (ECB) kunci semula.

```
44 b2 3b 28 12 67 c4 d7 32 ef 15 d1 79 2d 87 ec
91 91 d8 f7 c0 cf d8 b5 8a 0b 0b 7c ea 7c a0 fa
83 8f 93 fc 5e 87 bd 44 4d 5a 80 4b ce e9 92 ae
61 97 9b c6 cc 6e a9 6f 78 89 12 9d 36 3b d1 ca
```

Representasi hex 64 bytes cipherteks (ECB) kunci setelah diganti.

2. Cipher Block Chaining (CBC)

```
75 7a 46 4d ea 18 20 b7 dc d2 2b b1 47 50 dd c3
8f a8 ef e9 02 08 da 2c b5 a8 12 1c 0e 26 9c 68
06 d4 ee 1b 22 b2 16 60 7d 76 6f 4e 94 6c ff 1d
bb 3c 8d b8 f7 a9 19 74 02 14 94 44 6f 31 b0 ad
```

Representasi hex 64 bytes cipherteks (CBC) kunci semula.

```
79 dd 29 2c 2e 01 a6 c3 8b bd 34 30 d0 cb 59 a1
23 07 71 ff 9d 25 cd 0a 14 2a d1 48 f6 12 e6 a2
8b 4e 5e d3 7d da 91 43 da 30 d1 a7 60 d2 08 ec
a7 6c ba fb 8d b6 ef 25 1b bf 71 75 c4 6a 9c 68
```

Representasi hex 64 bytes cipherteks (CBC) kunci setelah diganti.

3. Cipher Feedback (CFB)

```
aa 6e 4f 5b f6 3a 7a 20 87 91 bb fa a3 85 04 9a
9a 31 d6 f8 cc 0c 17 27 ba f2 67 36 51 50 08 84
32 02 2a 49 81 7d d5 33 2d 0f c0 6b c4 67 8d ee
e2 b4 5a 1f 9b 83 66 8c fa 98 ed e4 02 c6 27 d0
```

Representasi hex 64 bytes cipherteks (CFB) kunci semula.

```
21 02 e8 c3 0a 5a e5 df 10 7f e5 91 c9 08 c0 6e
8e 9c 19 4c 1d 59 10 6f fa d2 ea be 03 07 01 bf
9b 4b 56 8f 46 c8 05 05 8b 48 f1 33 75 f2 71 a5
8f d0 ec 8c c2 a8 fe 4a 13 ac cf 43 c7 f1 79 14
```

Representasi hex 64 bytes cipherteks (CFB) kunci setelah diganti.

4. Output Feedback (OFB)

```
44 25 94 f3 73 b6 68 d5 74 a5 c6 20 dd 9a 39 07
9d af e1 23 56 be 1c 0e 17 09 6d e0 f2 73 79 00
7e 24 84 b9 b9 eb fd 7e a9 db 7d ca 24 77 f5 2a
af cc da 74 fb 3a ca bf f5 a9 e0 65 6c 45 88 70
```

Representasi hex 64 bytes cipherteks (OFB) kunci semula.

```
bd db f4 20 06 ba 69 14 dc 8e 13 c2 a4 56 c0 b8
71 77 4b 92 ba 40 f0 80 ed 8c 99 dc 52 d9 e7 83
75 2e 68 50 b3 80 52 02 a8 6c 27 a0 a4 3d b1 5a
14 8a be 4f cd 72 62 95 3f 4b 50 db 7b c7 36 ff
```

Representasi hex 64 bytes cipherteks (OFB) kunci setelah diganti.

5. Counter (CTR)

```
e3 e0 e6 e8 2b e1 7a 74 78 8b 5c f6 a3 fd ce 13
71 c3 eb 80 06 e3 d8 b9 28 fb 3e 78 66 6f c0 30
e7 6c 08 d3 9f 60 6f 54 c4 54 08 d0 c1 e9 34 5e
cd 2d fb 10 8c a5 6e e1 5d eb be 20 79 7f b7 f9
```

Representasi hex 64 bytes cipherteks (CTR) kunci semula.

```
1d ea 62 49 75 27 70 44 8b 6e 07 2c 33 eb 16 6c
06 35 3c b0 51 63 2b 81 40 fd 69 1b 66 e8 58 ac
03 15 27 d3 3e cf 42 4a 8e 27 8d 4a fd 6f c2 30
43 2d 1d 18 07 87 4c b1 9b 66 9d d3 c2 35 84 fb
```

Representasi hex 64 bytes cipherteks (CTR) kunci setelah diganti.

V. Analisis Keamanan

Keamanan dari algoritma Chill dapat diukur dengan beberapa teknik serangan yang sering digunakan seperti teknik *brute force* dan kriptanalisis seperti analisis frekuensi kemunculan[3]. Pada bagian ini akan dibahas tiap teknik.

A. *Brute force*

Teknik *brute force* dilakukan dengan menebak dan mencoba semua kemungkinan kunci satu persatu sehingga didapatkan kunci yang sesuai. Teknik ini menggunakan *exhaustive search*, sehingga pasti akan ditemukan kunci yang sesuai. Namun permasalahannya yaitu seberapa lama waktu yang dibutuhkan untuk menemukan kunci yang sesuai.

Algoritma Chill menggunakan ukuran kunci sebanyak 16 bytes atau 128 bit sehingga kemungkinan kunci yang dapat dibentuk yaitu 2^{128}. Digunakan asumsi *brute force* yang dapat dilakukan yaitu 10^6 kunci per detik, maka waktu yang diperlukan untuk menguji seluruh kemungkinan kunci yaitu sekitar 3.4×10^{32} detik atau sekitar 1.075×10^{25} tahun.

Biaya dan waktu yang dibutuhkan untuk memecahkan 128 bit dapat dikatakan cukup mahal. Sehingga dapat disimpulkan algoritma Chill cukup aman dari serangan dengan menggunakan teknik *brute force*.

B. Analisis frekuensi kemunculan

Analisis frekuensi kemunculan dilakukan dengan memerhatikan frekuensi kemunculan huruf-huruf dari cipherteks, bisa juga bigram atau n-gram lainnya yang banyak digunakan pada sebuah pesan. Dari hal tersebut dapat ditarik keterhubungan dari suatu huruf cipherteks dan pesan. Contohnya pada pesan dengan Bahasa Inggris, huruf yang sering muncul yaitu E, T, A, dst.

Pada percobaan algoritma Chill pada Bab IV, dapat diperoleh hasil bahwa persebaran cipherteks lebih merata jika dibandingkan dengan pesan. Hal tersebut berlaku pada untuk setiap mode yang digunakan. Sehingga teknik analisis frekuensi kemunculan dapat dikatakan tidak mungkin untuk dilakukan.

VI. Kesimpulan dan Saran

Algoritma Chill cipher merupakan algoritma *block cipher* alternatif yang mudah digunakan karena menggunakan operasi cukup sederhana. Namun dengan tetap mempertahankan prinsip-prinsip keamanan seperti konfusi dan difusi sehingga cipherteks yang dihasilkan sulit untuk dipecahkan baik dengan menggunakan teknik *brute force* maupun analisis frekuensi kemunculan.

Saran untuk pengembangan selanjutnya yaitu jumlah blok yang digunakan lebih beragam dan tidak hanya 256 bit. Sehingga ukuran panjang kunci serta jaringan feistel juga dapat beragam menyesuaikan panjang blok.

References

[1] Menezes, Alfred J.; Oorschot, Paul C. van; Vanstone, Scott A. (2001). Handbook of Applied Cryptography (Fifth ed.). p. 251

[2] Munir, Rinaldi. 2019. Slide Kuliah IF4020 Kriptografi: Algoritma Kriptografi Modern.

[3] Munir, Rinaldi. 2018. Slide Kuliah IF4020 Kriptografi: Serangan Terhadap Kriptografi.

[4] Yang, Herong. 2018. *What is PKCS5Padding?* http://www.herongyang.com/Cryptography/DES-JDK-What-Is-PKCS5Padding.html. [Diakses 22-Februari-2019].

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, 13 Maret 2019

Achmad Fahrurrozi M. (13515026)

Diki Ardian W. (13515092)