BeRez Cipher

Fancy Modified DES Algorithm, BeRez (Bervianto Reza) Cipher

Bervianto Leo Pratama
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
bervianto.leo@gmail.com

Muhammad Reza Ramadhan
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
rezaramadhan.m@gmail.com

***Abstract*—BeRez Cipher merupakan algoritma kriptografi berupa *cipher* blok menggunakan kunci simetrik. BeRez cipher memiliki ukuran blok masing-masing 128-bit untuk blok kiri dan blok kanan dan memiliki kunci sebesar 256-bit. Algoritma BeRez menggunakan Feistel Network, fungsi f yang dimodifikasi dan substitusi dengan S-Box sebagai komponen penyusunnya. Algoritma BeRez memiliki jumlah putaran yang dinamis dan diturunkan berdasarkan nilai kunci. Algoritma BeRez dapat dijalankan pada berbagai metode seperti EBC, CBC, CFB, OFB, dan Counter (CTR). Hasil pengujian terhadap algoritma BeRez memberikan hasil distribusi frekuensi yang cukup merata pada *ciphertext* serta penggunaan prinsip confusion dan diffusion yang cukup baik.**

Keywords—BeRez, block cipher, dekripsi, enkripsi, kriptografi

I. Introduction

BeRez cipher merupakan sebuah algoritma *block cipher* dengan kunci simetrik. *Block cipher* merupakan suatu fungsi yang memetakan n-bit blok plainteks menjadi n-bit blok cipherteks, dengan n merupakan panjang blok [1]. Terdapat beberapa kriteria untuk mengevaluasi blok cipher, yaitu 1) perkiraan tingkat keamanan, 2) besar atau panjang kunci, 3) *throughput*, 4) besar atau panjang blok, 5) kompleksitas dari pemetaan kriptografi (*cryptographic mapping)*, 6) *data expansion*, dan 7) *error propagation [1].*

Blok cipher melakukan enkripsi pesan panjang yang melebihi n-bit dengan membagi-bagi pesan menjadi n-bit blok dan melakukan enkripsi pada pesan secara terpisah antar pesan yang terbagi. Terdapat empat mode yang umum digunakan untuk blok *cipher* yaitu ECB, CBC, CFB, dan OFB [1].

Algoritma blok *cipher* yang terkenal yaitu DES dan AES. Algoritma DES merupakan algoritma blok *cipher* yang hadir pada pertengahan tahun 1970-an sebagai algoritma kriptografi modern yang pertama dengan detail implementasi yang terbuka dan.sepenuhnya dispesifikasikan [1]. Algortima DES dikenalkan oleh American standard FIPS 46-2 [1]. Secara umum algoritma DES terdiri dari dua konsep yaitu *product ciphers* dan *Feistel ciphers.* Konsep dasar dari *product cipher* yaitu membangun fungsi enkripsi yang kompleks dengan melakukan beberapa operasi yang mudah yang memberikan perlindungan dengan saling melengkapi namun tidak memadai secara individu [1]. Operasi dasar yang dilakukan termasuk transposisi, translasi dan transformasi linear, operasi aritmatika, perkalian modular, dan subsitusi yang simpel [1]. Algoritma DES merupakan Feistel cipher yang memproses blok *plaintext* dengan n = 64-bit dan menghasilkan 64-bit blok *ciphertext.* Ukuran efektif untuk kunci yaitu 56-bit, lebih tepatnya masukan kunci dispesifikasikan 64-bit dengan 8-bit digunakan sebagai bit paritas. Proses enkripsi dilakukan sebanyak 16 *stages* atau *round* [1]. Setiap putaran mendapat masukan dari hasil enkripsi putaran sebelumnya dan dilakukan pertukaran yang sebelumnya dari kiri menjadi kanan, dari yang kanan menjadi yang kiri. Setiap putaran dalam algoritma DES dilakukan pembangkitan atau penjadwalan kunci dari kunci eksternal menjadi beberapa kunci internal. Secara lengkap mengenai alur dari algoritma DES dapat dilihat pada Gambar 1.

Algorithm Data Encryption Standard (DES)

INPUT: plaintext $m_1 \dots m_{64}$; 64-bit key $K = k_1 \dots k_{64}$ (includes 8 parity bits).
OUTPUT: 64-bit ciphertext block $C = c_1 \dots c_{64}$. (For decryption, see Note 7.84.)

1. (key schedule) Compute sixteen 48-bit round keys K_i from K using Algorithm 7.83.
2. $(L_0, R_0) \leftarrow \text{IP}(m_1 m_2 \dots m_{64})$. (Use IP from Table 7.2 to permute bits; split the result into left and right 32-bit halves $L_0 = m_{58} m_{50} \dots m_8$, $R_0 = m_{57} m_{49} \dots m_7$.)
3. (16 rounds) for i from 1 to 16, compute L_i and R_i using Equations (7.4) and (7.5) above, computing $f(R_{i-1},\ K_i) = P(S(E(R_{i-1}) \oplus K_i))$ as follows:
   (a) Expand $R_{i-1} = r_1 r_2 \dots r_{32}$ from 32 to 48 bits using E per Table 7.3: $T \leftarrow E(R_{i-1})$. (Thus $T = r_{32} r_1 r_2 \dots r_{32} r_1$.)
   (b) $T' \leftarrow T \oplus K_i$. Represent T' as eight 6-bit character strings: $(B_1, \dots, B_8) = T'$.
   (c) $T'' \leftarrow (S_1(B_1), S_2(B_2), \dots S_8(B_8))$. (Here $S_i(B_i)$ maps $B_i = b_1 b_2 \dots b_6$ to the 4-bit entry in row r and column c of S_i in Table 7.8, page 260 where $r = 2 \cdot b_1 + b_6$, and $b_2 b_3 b_4 b_5$ is the radix-2 representation of $0 \le c \le 15$. Thus $S_1(011011)$ yields $r = 1$, $c = 13$, and output 5, i.e., binary 0101.)
   (d) $T''' \leftarrow P(T'')$. (Use P per Table 7.3 to permute the 32 bits of $T'' = t_1 t_2 \dots t_{32}$, yielding $t_{16} t_7 \dots t_{25}$.)
4. $b_1 b_2 \dots b_{64} \leftarrow (R_{16}, L_{16})$. (Exchange final blocks L_{16}, R_{16}.)
5. $C \leftarrow \text{IP}^{-1}(b_1 b_2 \dots b_{64})$. (Transpose using IP^{-1} from Table 7.2; $C = b_{40} b_8 \dots b_{25}$.)

Gambar 1 Algoritma DES

Sumber: [1]

Algoritma AES merupakan algoritma blok *cipher* dengan panjang blok dan panjang kunci secara independen dispesifikasikan 128, 192, 256 bit [2]. Setiap putaran kecuali putaran terakhir dilakukan beberapa transformasi yaitu ByteSub, ShiftRow, MixColumn dan AddRoundKey, sedangkan putaran terakhir tidak dilakukan MixColumn. ByteSub merupakan subsitusi non-linear dengan tabel subsitusinya dapat berkebalikan (invertible) dan dibangun dari komposisi dua transformasi, dapat disebutkan juga transformasi ini menggunakan S-box. ShiftRow merupakan

transformasi yang melakukan pergeseran pada baris 1, 2 dan 3 sedangkan baris 0 tidak digeser. Jumlah pergeseran berdasarkan ukuran blok misalkan blok berukuran 4 akan memiliki pergeseran sebesar 1, 2, dan 3 berturut-turut untuk baris 1, 2 dan 3. Tranformasi MixColumn melakukan perkalian pada *state* dengan matriks yang sudah dispesifikasikan pada algoritma tersebut. Transformasi selanjutnya yaitu AddRoundKey merupakan transformasi yang melakukan xor terhadap *state* dengan kunci.

BeRez cipher merupakan algoritma yang memanfaatkan jaringan Feistel dan S-box sebagai subsitusi. Secara umum algoritma BeRez mirip dengan algoritma DES namun dengan modifikasi pada jumlah putaran dan operasi yang terdapat pada fungsi f.

II. Proposed Block Cipher

Algoritma BeRez menggunakan ukuran blok dan kunci sebesar 256-bit. 256-bit yang terdapat dalam sebuah blok akan dibagi menjadi 128-bit blok kiri dan 128-bit blok kanan. Pembagian blok kiri dan kanan ini sesuai dengan proses yang terjadi dalam sebuah feistel network. Setiap blok kiri dan blok kanan tersebut akan dioperasikan dalam bentuk matriks 4x4 byte. Sebelum memasukkan blok dalam feistel network, blok kiri dan kanan akan ditransposisi dengan cara mengubah posisi matriks dari posisi sebagai berikut

1	8	9	16
2	7	10	15
3	6	11	14
4	5	12	13

menjadi matriks baru yang memiliki posisi sebagai berikut,

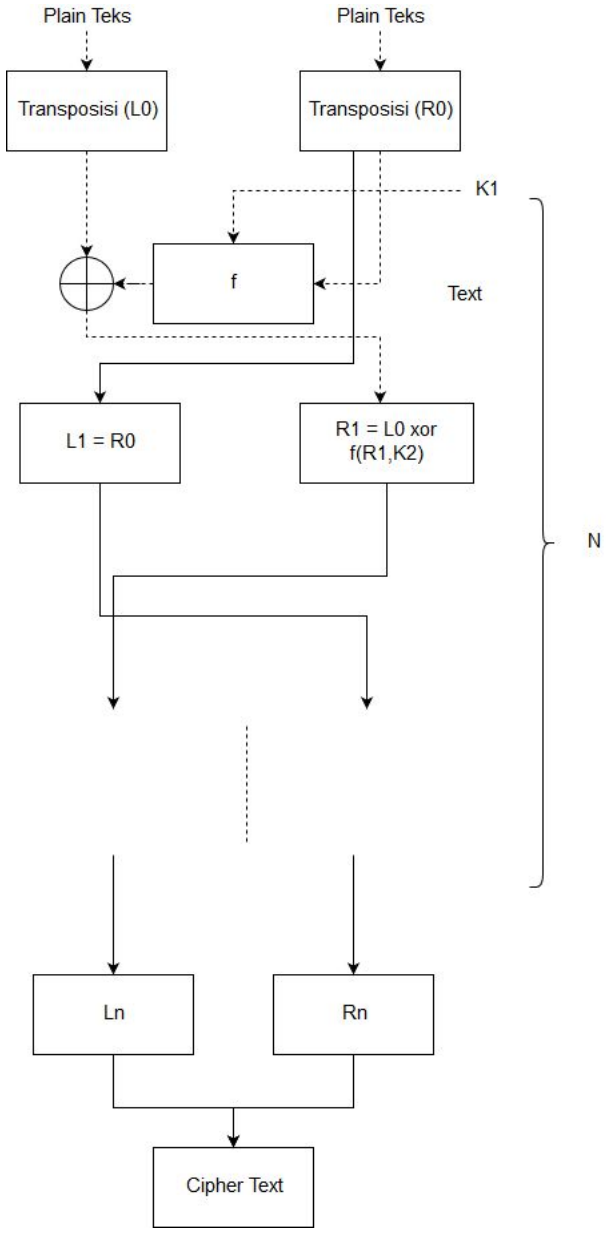
4	3	2	1
5	6	7	8
12	11	10	9
13	14	15	16

Transposisi dilakukan pada bagian awal dari proses enkripsi. Sementara itu dalam dekripsi transposisi dilakukan pada akhir proses. Transposisi yang dilakukan dalam dekripsi adalah kebalikan dari transposisi pada proses enkripsi.

Jumlah putaran yang digunakan dalam BeRez cipher berjumlah dinamis. Jumlah putaran yang mungkin terdapat dalam rentang 7 hingga 25. Jumlah putaran bergantung dari algoritma pembangkitan bilangan random yang dibangkitkan dengan *seed* berbasis kunci.

Operasi feistel network yang dilakukan pada algoritma ini dapat dideksripsikan sebagai berikut; Blok kanan akan dimasukan pada fungsi *f* beserta kunci internal untuk round tersebut. Hasil dari fungsi *f* akan dilakukan XOR pada blok kiri. Hasil dari operasi tersebut digunakan sebagai blok kanan pada round berikutnya, sedangkan blok kanan menjadi blok kiri pada round berikutnya. Pada blok terakhir tidak perlu dilakukan pertukaran.

Ilustrasi dari feistel network yang digunakan pada BeRez cipher dapat dilihat pada Gambar 2.



Gambar 2 Algoritma BeRez

A. Pembangkitan Jumlah Putaran

Jumlah putaran akan diturunkan dari kunci yang dimasukan, dengan demikian penggunaan kunci yang sama pasti akan mendapatkan jumlah putaran yang sama. Jumlah putaran dihitung berdasarkan algoritma pembangkitan bilangan random tertentu. Bilangan random yang akan dibangkitkan akan berada dalam rentang 7 hingga 25. Pembangkitan bilangan random akan menggunakan *seed* yang bergantung pada kunci yang dimasukkan.

Seed dihitung berdasarkan kunci yang digunakan. Setiap karakter karakter pada kunci akan diubah representasinya menjadi sebuah integer, kemudian seluruh representasi tersebut dijumlahkan. Perubahan representasi kunci menjadi integer dapat dilakukan dengan metode apapun, salah satu contoh yang mungkin digunakan adalah dengan menggunakan coding ASCII.

Sebagai contoh, jika kunci adalah 'APA', maka perhitungan *seed* dapat dituliskan sebagai berikut:

$$\begin{aligned} seed &= int('A') + int('P') + int('A') \\ &= 65 + 80 + 65 \\ &= 210 \end{aligned}$$

B. Pembangkitan Kunci

Algoritma BeRez melakukan pembangkitan kunci dengan menggunakan lima langkah sebagai berikut,

1. Membagi kunci eksternal 256-bit menjadi dua buah blok 128-bit berdasarkan byte ganjil-genap pada kunci tersebut.
2. **K1** didapatkan dari hasil XOR dua blok tersebut.
3. Buat sebuah blok baru yang berisi hasil substitusi **K1** dengan menggunakan S-Box.
4. **K2** didapatkan dari hasil XOR **K1** dan blok baru tersebut.
5. Untuk **K3**, **K4**, dan seterusnya, ulangi langkah nomor 3 dan 4.

C. Fungsi f

Fungsi f pada algoritma BeRez terdiri dari beberapa operasi yaitu XORRoundKey, ShiftByKey, XORShifted, dan Subsitution. Operasi XORRoundKey dilakukan dengan melakukan XOR antara blok dengan kunci internal untuk round tersebut. Operasi XORRoundKey ini mirip dengan yang dilakukan AES.

ShiftByKey melakukan pergeseran terhadap setiap baris blok plaintext berdasarkan blok kunci internal. ShiftByKey merupakan keunikan dari BeRez cipher. Operasi ini dilakukan untuk meningkatkan prinsip confusion pada BeRez cipher. Dalam ShiftByKey setiap kunci internal yang berbeda akan memberikan jumlah pergeseran dalam tiap baris yang berbeda. Baris genap pada blok akan digeser ke kiri, sementara pada baris ganjil akan digeser ke kanan. Jumlah pergeseran berdasarkan nilai setiap baris pada kunci. Sebagai contoh jika pada suatu putaran memiliki kunci internal sebagai berikut,

4F	4A	4D	4F
3F	3F	5F	AE
4D	5A	77	46
99	71	AA	90

Pada baris pertama akan didapatkan nilai geser sebesar 79 + 74 + 77 + 79 = 309. Pergeseran sebesar 309 akan sia-sia jika dilakukan sepenuhnya mengingat pergeseran lebih dari 4 akan memiliki hasil yang sama dengan pergeseran sebelumnya. Sehingga untuk mendapat pergeseran yang efektif jumlah pergeseran akan di-mod-kan dengan 4. Dengan demikian baris pertama akan digeser sebesar 309 mod 4 = 1.

XORShifted dilakukan dengan melakukan XOR dari matriks hasil operasi ShiftByKey dengan sebuah matriks baru. Matriks baru merupakan plainteks awal sebelum dilakukan operasi XORRoundKey yang sudah dilakukan permutasi dan pergeseran. Permutasi dilakukan untuk mengubah baris pada matriks menjadi kolom, dan begitu pula sebaliknya. Dengan kata lain, Permutasi mengubah matriks yang memiliki posisi sebagai berikut

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Menjadi matriks baru dengan posisi

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Setelah itu, setiap baris pada matriks hasil permutasi akan digeser seperti pada operasi ShiftByKey, dimana baris genap akan digeser ke kiri dan baris ganjil akan digeser ke kanan. Perbedaannya adalah jumlah pergeseran dilakukan berdasarkan jumlah per baris pada blok plainteks, bukan berdasar pada kunci. Operasi XORShifted dilakukan untuk menambah prinsip diffusion mengingat posisi matriks akan ditentukan dari hasil perhitungan baris pada plainteks.

Subsitution dilakukan pada blok hasil operasi XORShifted dengan menggunakan S-Box. S-Box yang kami gunakan adalah S-Box AES yang dibangkitkan berdasarkan multiplicative inverse untuk angka tertentu dalam GF(2^8).

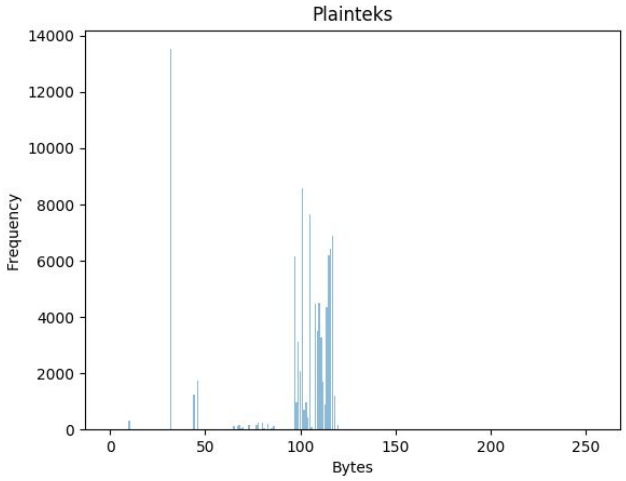
III. Simulation and Testing

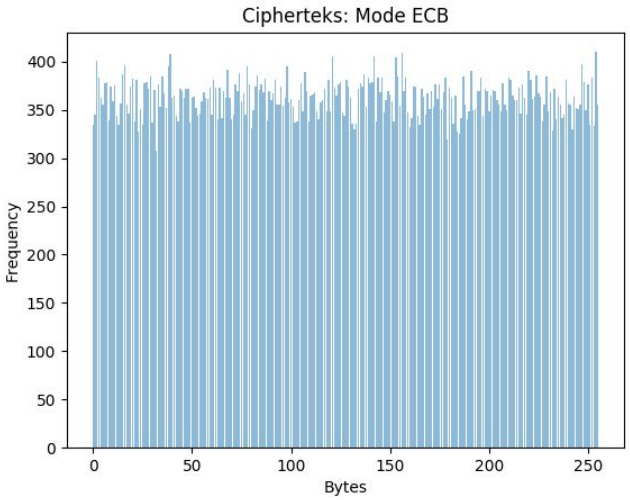
Pengujian dilakukan pada teks lorem ipsum yang dapat dilihat pada pranala berikut https://github.com/berviantoleo/fancy-DES-algorithm/blob/master/samples/lorem-ipsum.txt. Implementasi dari algoritma BeRez juga dapat dilihat pada pranala https://github.com/berviantoleo/fancy-DES-algorithm, algoritma BeRez diimplementasikan pada lingkungan bahasa pemrograman Python. Pengujian dilakukan pada setiap metode ECB, CBC, CFB, OFB dan counter (CTR). Hasil pengujian pada seluruh metode tersebut dapat dilihat pada pranala berikut https://github.com/berviantoleo/fancy-DES-algorithm/tree/master/samples/output

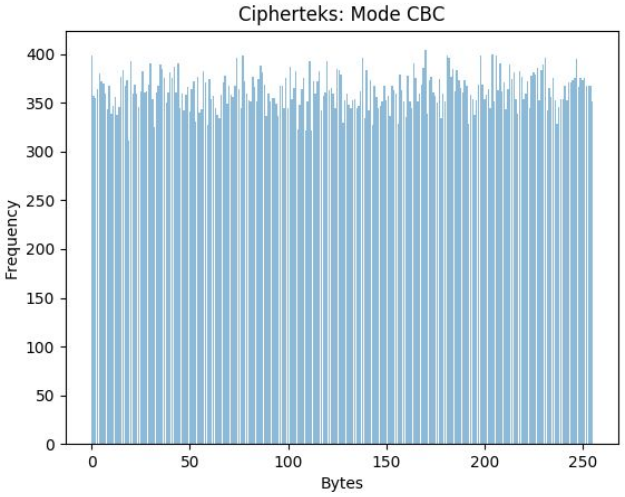
A. Analisis Frekuensi

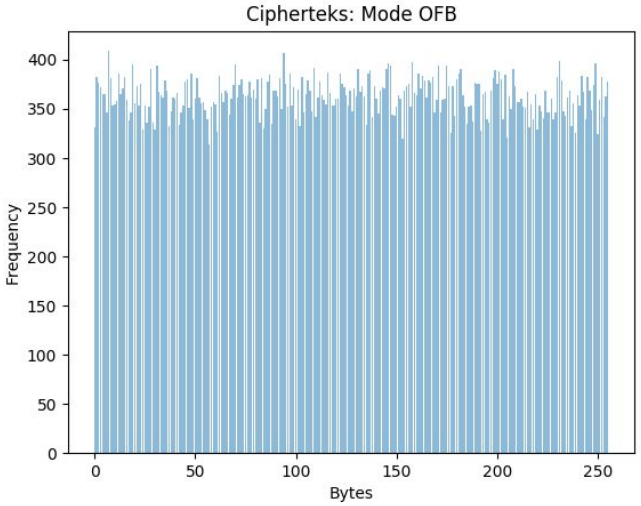
Analisis frekuensi dilakukan pada tiga dari lima mode pada BeRez cipher yaitu mode ECB, CBC, dan OFB. Analisis frekuensi dilakukan untuk mengetahui apakah BeRez cipher akan rentan terhadap kriptanalisis berbasis frekuensi. Plainteks yang digunakan pada analisis ini adalah plainteks lorem-ipsum yang terdiri dari 149 paragraf, 13.680 kata, dan 92.561 karakter. Cipherteks yang dihasilkan oleh masing-masing mode dapat dilihat pada pranala yang diberikan sebelumnya.

Berikut merupakan grafik batang yang menggambarkan distribusi frekuensi dari plainteks serta cipherteks yang dihasilkan oleh BeRez cipher









Terlihat bahwa ciphertext memiliki hasil distribusi yang relatif mirip, hal ini terjadi karena kami menggunakan fungsi f yang cukup kompleks, maka akan membuat hasil fungsi f cukup random sehingga ketika dilakukan XOR antara hasil fungsi f dan plaintext maka hasilnya juga akan random. Distribusi frekuensi yang cukup merata pada ciphertext akan

membuat kriptoanalisis berbasis frekuensi akan sulit dilakukan pada cipher BeRez buatan kami ini.

B. Analisis Prinsip Confusion dan Diffusion

Pada bagian ini kami akan menguji hasil dari BeRez cipher dengan menggunakan kunci yang berbeda satu bit, plaintext yang berbeda satu bit, serta ciphertext yang berbeda satu bit. Pengujian dilakukan pada implementasi BeRez cipher implementasi CBC. Pada pengujian ini, digunakan kunci, plainteks, serta cipherteks standar sebagai berikut:

Plainteks standar	`Argentina's Federal Congress is debating a unique new intermediary liability law, which would protect users of platforms from having their content removed without a court order.`
Ciphertext standar (hex)	`8c2c61673588150b9254e34f026566496ce9f1fa0b07a336e5f85a13b887b6c04e778f5f3d3158087aaf3a990f171cc5507852dcf83cd2e4e30b1b1cb3f640f1f2eab7985fcf07ba6a4bb9db51f4705dfccd7592457641ed4bdd8fe59416031d1ed075f57f49b7a27c649f018bcd22fd5cd43988a217253cb1245e81b2e6a094b7ffb80cf29c8742dd63e78fbd9b4f921ea29aa649fa06a0ab13b5f0bdec60a1fd0b6fc59bf3840f410c9fee244d27a06de6a2c94e805c690c5a6323c1328987`
Kunci standar	`HELLO WORLD! HAHAHHA`

1) Pengujian Kunci berbeda: Enkripsi

Pada pengujian ini, kami melakukan dua buah enkripsi terhadap sebuah plainteks, yaitu plainteks standar dengan menggunakan dua kunci yang berbeda. Kunci pertama merupakan kunci standar, sementara kunci kedua adalah kunci standar dimana byte ke-4 pada kunci ditambahkan nilainya dengan satu bit. Kunci modifikasi yang digunakan adalah

```
HELMO WORLD! HAHAHHA
```

Dengan penggunaan kunci yang berbeda satu bit, didapatkan cipherteks yang berubah dengan cukup signifikan. Cipherteks yang didapat adalah

```
7d0573e77922657de91284fc6027475d33811a12e3696c4df
56aed6bac066585aaea56b07e89a3ae4279f52252848c9a75
f35c5b3cb151edc4285509ecfe88dbfee82925e4edf29dc2c
3e4c8431a9af24cb12fe3b574a433060fe101e4b1a2d79a35
48d4c772599366bee93a1d9aa61b9c3debccdbc0ef1353b42
acc710d0688903cd58c409039674be3238e49d15c48a4f0f4
ec028be5d7152bcd5fd7553cd8f0c1bff862f945506ffa211
d10e138baad033ffe38cccdda1597178c152cd5fe
```

Dari hasil pengujian, terlihat bahwa cipherteks yang dihasilkan sangat bergantung pada kunci yang digunakan. Hal ini terjadi karena penggunaan jumlah putaran yang dinamis membuat hasil dari feistel network dari dua buah kunci yang berbeda akan berbeda cukup jauh. Dengan demikian, BeRez cipher sudah menggunakan prinsip confusion dengan cukup baik.

2) Pengujian Kunci berbeda: Dekripsi

Pada pengujian ini, kami melakukan dua kali dekripsi terhadap sebuah cipherteks, yaitu cipherteks standar dengan menggunakan dua kunci yang berbeda. Kunci pertama merupakan kunci standar, sementara kunci kedua adalah kunci standar dimana byte ke-4 pada kunci ditambahkan nilainya dengan satu bit. Kunci modifikasi yang digunakan adalah

```
HELMO WORLD! HAHAHHA
```

Dengan penggunaan kunci yang berbeda satu bit, didapatkan plainteks hancur dan tidak dapat diketahui lagi makna sebenarnya. Plainteks yang didapat adalah

```
CX\x83Q\x8a\ts\xdc\xef\x12\xfbI\xb1\xcdOQ\x1f:t\x
acD\xe2`P\xa9\xfa\xbb\x0cK\x91\xfb\xaa\xb2r\x84{\
x83c:\x83k\xe2\xde\x82h\xdc\xcb\x8dq\t&\xc5\xafZ-
/\xaf\x9f?\x15\xe3\xbb\xc72\x82\xb1\x13q[\xcav\xb
3\xcb{\r\x12/\xf9\xbfzQ\x87\x8d>\xd2\x96a\x92\xab
\xac\xb4\x011M6a\x1a\x86\x9e\xe8\xe4\x91t\xf4\xc2
K\xf8^L\x9d\x13\xa7\xa6\xbcxB?\xb8-\xe3\x191\x15\
xc7\n\x9c\x8f_\xfc\xe0\xffb\x95\xc5\xa0[\xce\x8e#
\xbb=\x05\x0b$\x08\x196\xee\x8a\xe7\xcc\xfdn\xba\
xd1\x9f2\xbc|\xdf\xbdy\x8f\xfd{Wh\x12\x0bF\xde<\x
e0\x9f\xe9n\x85\x91}\xdb~\x81\xb1z\x11z\x8b\xb7\x
d8\xbbp{
```

Pada BeRez cipher, perbedaan satu bit pada kunci dapat membuat proses plaintext pada hasil dekripsi menjadi sejumlah kumpulan byte tanpa makna. Penggunaan jumlah round yang dinamis membuat perubahan sedikit saja pada kunci dapat mengubah hasil dari feistel network pada dua kunci yang berbeda menjadi jauh berbeda.

3) Pengujian Plaineks berbeda

Pada pengujian ini, kami melakukan enkripsi terhadap dua plainteks, yaitu plainteks standar serta plainteks baru yang merupakan plainteks standar dengan nilai byte ke-33 yang sudah ditambah satu. Kunci yang digunakan pada kedua plainteks tersebut merupakan kunci yang sama, yaitu kunci standar. Byte ke-33 ini terdapat pada blok ke-2 pada enkripsi metode CBC. Modifikasi plainteks terdapat pada bagian

```
... Federal Congress is dfbating a unique ...
```

Perubahan satu bit pada plainteks tersebut membuat cipherteks yang dihasilkan menjadi cukup berbeda dibandingkan cipherteks standar. Bagian cipherteks yang sama hanyalah terdapat pada blok pertama, dimana perubahan bit masih belum terjadi. Cipherteks yang dihasilkan dari pengujian ini adalah.

```
8c2c61673588150b9254e34f026566496ce9f1fa0b07a336e
5f85a13b887b6c067e833d4557826fa9602efdf1f4ae530077
9b0e95248439a2dd21cebd099f470b968eaf7727c6418010cf7
```

```
3ec57d075af17ad160f1416437090d336f8c7bf49275c4c9063
e58ac5a33f5bc72a3137ee50c41b5b8415fb141f764ca138d3c
c9a04f9854f3ad717cebc9d8fb8dee4fd734c2f2edb9d21fec7
7c40b2ee08126bcc93a2239c8c3642ebda4b3d41bcaa927c2cc
6c23b4c5564df77e7c6567eb8b537
```

Dari pengujian, terlihat bahwa sebagian cipherteks sangat bergantung pada plainteks. Perubahan satu bit pada plainteks dapat mengubah cipherteks menjadi cukup signifikan. Efek ini juga ditingkatkan dengan penggunaan metode CBC, dimana setiap blok akan bergantung pada hasil enkripsi blok sebelumnya. Dengan demikian, dapat dikatakan bahwa BeRez cipher telah menggunakan prinsip diffusion dengan cukup baik.

4) Pengujian Cipherteks berbeda

Pada pengujian ini, kami melakukan dekripsi terhadap dua cipherteks, yaitu cipherteks standar serta cipherteks hasil modifikasi yang merupakan cipherteks standar dengan nilai byte ke-12 yang sudah dikurangi satu. Kunci yang digunakan pada kedua cipherteks tersebut merupakan kunci yang sama, yaitu kunci standar. Modifikasi ini digunakan untuk mensimulasikan cipherteks yang corrupt dalam proses pengiriman melalui jaringan. Modifikasi cipherteks yang digunakan terdapat pada bagian

```
... 8c2c61673588150b9254e34f03 6566496ce9f1 ...
```

Perubahan satu bit pada cipherteks tersebut membuat plainteks yang dihasilkan corrupt pada blok tempat terjadinya perubahan tersebut. Namun, bagian plainteks pada blok lain tidak terpengaruh pada perubahan ini. Plainteks yang didapat pada pengujian ini adalah

```
\x94\xad\x00\x8aB\xb21\x9a\x17\xbdp\xf5XJk[\x93\x81
.m\xaa\xac<\xe1"[\xa7&\x7f\x08\xb6debating a
uoique new intermediary liability law, which
would protect users of platforms from having
their content removed without a court order.
```

Dapat dilihat hasil dekripsi pada ciphertext yang mengalami kerusakan hanya merusak blok tersebut saja, tidak pada keseluruhan plaintext. Dengan demikian BeRez cipher dapat mengatasi kasus dimana ciphertext corrupt pada saat pengiriman data melalui jaringan.

C. Analisis Ruang Kunci

Kunci yang digunakan dalam BeRez cipher memiliki ukuran sebesar 256-bit, dengan setiap bit-nya memiliki peranan yang penting untuk melakukan proses enkripsi dan dekripsi. Perubahan satu bit pada kunci dapat menghasilkan cipherteks ataupun plainteks yang dihasilkan BeRez cipher menjadi sekumpulan byte tidak bermakna, seperti telah disimulasikan pada subbab sebelumnya.

Dengan penggunaan kunci sepanjang 256-bit, serangan bruteforce pada BeRez cipher hampir mustahil untuk dilakukan. Terdapat lebih dari 10^{77} kemungkinan kunci yang mungkin digunakan pada cipher ini. Apabila terdapat sebuah mesin sangat cepat yang dapat melakukan percobaan bruteforce terhadap BeRez cipher sebanyak 10^{10} kali per detik, maka mesin itu membutuhkan waktu selama $3.16 * 10^{59}$ tahun untuk dapat mencoba seluruh kunci yang mungkin pada BeRez cipher ini.

IV. Conclusion and Future Works

BeRez cipher memiliki tingkat keamanan yang baik dengan ukuran kunci sebesar 256 bit. Percobaan dengan kunci sebagian diketahui tidak akan memberikan plain teks yang bermakna sehingga hanya kunci tersebut yang akan memberikan pesan bermakna. Percobaan dengan kunci yang berbeda sedikit tidak memberikan hasil yang sedikit berbeda sehingga kunci akan semakin sulit ditemukan. BeRez cipher dalam hal menghasilkan cipherteks sudah memiliki sifat *diffusion* dan *confusion*.

BeRez cipher dapat dikembangkan menjadi lebih efisien pada kapasitas yang disediakan misalkan memberikan variasi ukuran blok. Algoritma BeRez juga dapat dikembangkan dengan menambah operasi dalam fungsi *f* untuk memperunik algoritma ini. Selain itu, dapat juga ditambahkan mekanisme sehingga perubahan plaintext dapat mengubah ciphertext dengan signifikan, mungkin seperti adanya permutasi plaintext secara random dengan seed dari plaintext awal.

Acknowledgment

Puji syukur kepada Tuhan Yang Maha Esa sehingga tulisan ini dapat diselesaikan dengan baik. Terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. sebagai dosen mata kuliah kriptografi yang sudah memberikan ilmu dan pengetahuan mengenai kriptografi terutama *cipher* blok. Terima kasih kepada penyusun dan pencetus algoritma DES dan AES yang memberikan inspirasi pengembangan algoritma *cipher* blok.

References

[1] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography," 5th ed., CRC Press, pp. 223–282, 1996.

[2] J. Daemen and V. Rijmen, "AES Proposal: Rijndael", version 2, September 1999.