64-DIB : 64-Bit Difusing Invers Block

Moch Ginanjar Busiri

13513041

Teknik Informatika

Institut Teknologi Bandung

gbusiri@gmail.com

Wiwit Rifa'i

13513073

Teknik Informatika

Institut Teknologi Bandung

wiwitrifai@gmail.com

Abstrak – Pada paper ini diajukan sebuah algoritma yang merupakan pengembangan dari algoritma blok cipher. Algoritma baru ini dinamakan Difusing Invers Block (DIB), yang dapat mengenkripsi data dengan terlebih dahulu membalikkan (invers) bit yang sedang diproses baru kemudian dioperasikan. Algoritma ini akan diterapkan dengan beberapa metode diantaranya metode Electronic Codebook (ECB) dan metode Cipher Block Chaining (CBC). Algoritma ini memiliki tingkat keamanan yang cukup baik dan memenuhi syarat dari kriptografi dengan kategori paling aman yaitu difusion dan confusion.

Kata Kunci : Block Cipher, Invers, Transposisi, Key, Enkripsi, Dekripsi, Difusing, Confusing.

I. Pendahuluan

Perkembangan teknologi selalu beriringan dengan bertambahnya pengguna dan peredaran data. Meningkatnya kedua hal tersebut memicu tingginya resiko akan kejahatan dari para peretas yang mecoba mencuri atau menyalahgunakan data tersebut. Untuk mengatasi itu, para kriptografer mencoba mengembangkan suatu metode-metode kriptografi untuk mengamankan data dari para peretas sehingga muncul algoritma krigtografi yang sampai sekarang terus berkembang dari mulai kriptografi klasik sampai modern.

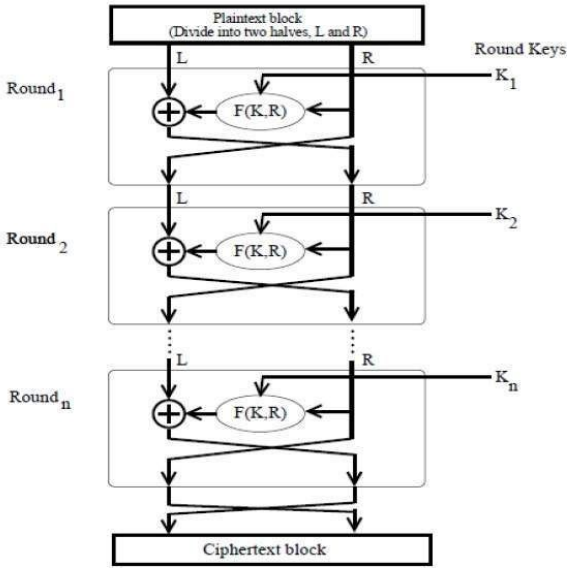
Salah satu algoritma kriptografi modern adalah algoritma yang menggunakan proses komputasi pada operasi bit. Setelah itu dikembangkan kembali menjadi operasi bit yang dibaca secara blok per blok yang sekarang ini lebih dikenal dengan Block Cipher.

Ciri khas dari algoritma modern block cipher adalah dengan melakukan operasi XOR pada setiap proses enkripsi dan deskripsinya. Algoritma yang termasuk kedalam block cipher diantaranya adalah Cipher Block Chaining.

II. Dasar Teori

1. Jaringan Feistel

Jaringan Feistel adalah struktur simetris yang biasa digunakan dalam mengkonstruksi *block cipher*. Jaringan Feistel memiliki skema enkripsi seperti pada gambar 1.



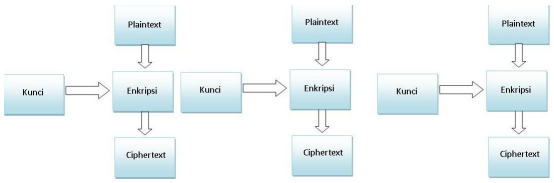
Gambar 1 Skema Jaringan Feistel
Sumber gambar: www.tutorialspoint.com

Jaringan Feistel membagi blok yang akan dienkripsi menjadi 2 bagian terlebih dahulu yaitu bagian L dan R. Bagian R ini menjadi menjadi

parameter dari suatu fungsi enkripsi yang juga memiliki kunci sepanjang setengah ukuran blok. Hasil dari fungsi ini kemudian diXOR dengan bagian L. Kemudian bagian L dan R saling bertukar posisi. Proses tersebut dilakukan secara berulang-ulang sehingga blok tersebut menjadi semakin terlihat acak. Satu perulangan biasa disebut sebagai satu *Round*. Dalam setiap Round kunci yang dipakai dalam fungsi enkripsi bisa berbeda-beda. Semakin banyak jumlah Round maka keamanannya pun semakin tinggi, namun hal tersebut juga membuat komputasi yang semakin banyak dan kurang efisien.

Untuk melakukan dekripsi, Jaringan Feistel cukup melakukan hal yang berkebalikan dari proses enkripsi. Karena fungsi enkripsi hanya diXOR dengan bagian L, maka kita hanya perlu menghitung fungsi enkripsi tersebut dan kemudain diXORkan kembali dengan L untuk membatalkan efek transformasi dari fungsi enkripsi tersebut. Hal tersebut akan mengembalikan nilai L kembali seperti semula ketika L belum diXOR dengan fungsi enkripsi tadi. Sehingga dalam pemilihan fungsi enkripsi ini, kita tidak perlu mempermasalahkan fungsi dekripsi yang perlu digunakan ketika mendekripsi *cipher text* menggunakan Jaringan Feistel ini.

2. Algoritma Electronic Codebook (ECB)

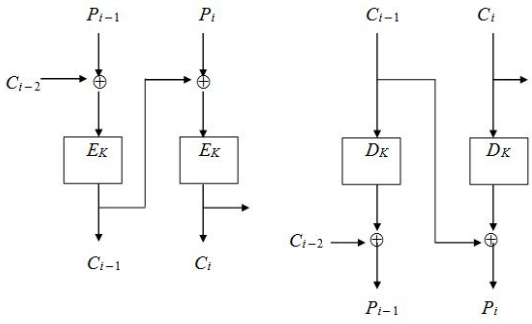


Algoritma ECB adalah suatu algoritma blok cipher yang mengoperasikan bit bit dari suatu plainteks yang kemudian dilakukan operasi XOR untuk mendapatkan cipherteksnya. Pada algoritma ini antar cipherteks tidak saling terkait atau terpengaruhi (independent) sehingga jika terjadi noise pada salah satu plainteks tidak akan mempengaruhi yang lain. Keunggulan dari algoritma ini dapat digunakan sebagai dictionary suatu data karena setiap plainteks yang sama akan menghasilkan cipherteks yang sama. Akan tetapi algoritma ini sangat mudah terserang oleh para peretas dengan menggunakan pendekatan statistika.

3. Algoritma Cipher Block Chaining (CBC)

Algoritma CBC adalah suatu algoritma dimana suatu blok dan blok lain saling terkait (chained). Saling terkait disini maksudnya adalah enkripsi dan dekripsi suatu blok data selalu melibatkan ciphertext (hasil enkripsi) blok sebelumnya. Setiap blok plaintext di-XOR dengan ciphertext hasil enkripsi blok plaintext sebelumnya baru kemudian hasil operasi XOR ini dienkrip untuk menghasilkan blok ciphertext. Begitu pula sebaliknya ketika dekripsi. Dekripsi yang dilakukan terhadap suatu blok ciphertext tidak langsung menghasilkan blok plaintext, hasil dekripsi tersebut harus di-XOR dulu dengan blok ciphertext sebelumnya untuk menghasilkan blok plaintext. Jadi enkripsi maupun dekripsi selalu melibatkan blok ciphertext sebelumnya.

Dalam bentuk notasi matematika, bisa dilihat di bawah ini:



$$C_1 = E_k(P_1 \oplus IV) \qquad P_1 = D_k(C_1) \oplus IV$$
$$C_2 = E_k(P_2 \oplus C_1) \qquad P_2 = D_k(C_2) \oplus C_1$$
$$C_3 = E_k(P_3 \oplus C_2) \qquad P_3 = D_k(C_3) \oplus C_2$$
$$C_4 = E_k(P_4 \oplus C_3) \qquad P_4 = D_k(C_4) \oplus C_3$$

$$C_n = E_k(P_n \oplus C_{n-1}) \qquad P_n = D_k(C_n) \oplus C_{n-1}$$

Variabel yang dipakai dalam formula di atas:
⊕ = Notasi untuk eXclusive OR
P = Plaintext
C = Ciphertext
IV = Initialization Vector
Ek = Enkripsi dengan kunci k
Dk = Dekripsi dengan kunci k
P1 = Plaintext blok ke-1
P2 = Plaintext blok ke-2
Pn = Plaintext block ke-n
C1 = Ciphertext block ke-1
C2 = Ciphertext block ke-2
Cn = Ciphertext block ke-n

4. Difusion

Difusion secara harfiah adalah sebar atau penyebaran. Kaitannya dengan algoritma block cipher adalah bahwa metode atau sistem yang dibuat harus bersifat tersebar dalam hal komputasi sehingga untuk meretas cipherteks sangat sulit karena komputasi yang kompleks dan tidak monoton. Misalnya untuk mengenkripsi suatu plainteks, hasil dari perhitungan atau salah satu prosesnya dipakai kembali pada langkah berikutnya. Hal ini akan mengakibatkan plainteks yang sama tidak akan menghasilkan cipherteks yang sama pula sehingga cukup sulit untuk menentukan atau meretas cipherteks tersebut.

5. Confusion

Confusion secara harfiah adalah bingung atau membingungkan. Kaitannya dengan algoritma block cipher adalah bahwa metode atau sistem yang dibuat harus mempunyai sifat membingungkan bagi kriptanalis atau peretas yang ingin memecahkan cipher tersebut. Untuk membuat sistem yang bersifat confusion dapat dilakukan dengan berbagai macam misalnya melakukan substitusi huruf dengan huruf yang ada pada tabel khusus yang bersifat privasi milik kita, melakukan komputasi berulang sehingga hasil yang diambil adalah hasil iterasi terakhir, dan sebagainya. Hal ini akan membuat kriptanalis kebingungan meskipun fungsi atau komputasi yang dipakai sangat sederhana, apalagi yang bersifat kompleks.

III. Rancangan Block Cipher

Algoritma 64-Bit Difusing Invers Block (64-DIB) merupakan pengembangan dari algoritma blok cipher dimana setiap plain yang diproses akan diinverskan terlebih dahulu sebelum dienkrip atau didekrip. Pada rancangan ini akan dijelaskan 2 metode yang termasuk kedalam algoritma block cipher yaitu ECB dan CBC dan penerapannya terhadap algoritma 64-DIB. Untuk mempermudah ilustrasi, akan dijelaskan dengan memakai operasi 8 bit karena 64 bit terlalu banyak. Meskipun begitu algoritma yang diimplementasikan tetap akan berbasis 64 bit perbloknya.

1. Proses Enkripsi

Pertama, terdapat sebuah kunci yang dimasukkan oleh user untuk komputasi awal. Kunci ini harus merupakan deretan kombinasi dari biner (0 dan 1) sebanyak blok yang didefiniskan (untuk ilustrasi ini memakai 8 bit) karena pada dasarnya algoritma modern merupakan algoritma yang melakukan operasi pada level binary sehingga asumsi untuk kunci kami deskripsikan sebagai berikut.

Contoh, key = 11111111

Kedua, terdapat suatu plain teks yang ingin di enkripsi. Pada plain teks tersebut berdasarkan algoritma 64-DIB ini akan dibaca blok per blok dan diubah dalam format bit yang telah didefinisikan (untuk ilustrasi ini 8 bit).

Contoh, plainteks = “Saya suka kriptografi”

Pembacaan : bit ‘S’ = 01010011

bit ‘a’ = 01100001

bit ‘y’ = 01111001

bit ‘a’ = 01100001

dst..

Ketiga, setiap plainteks tersebut diproses dengan jaringan Feistel yang ingin diproses lakukan invers terlebih dahulu, lalu lakukan XOR dengan kunci yang telah dimasukkan di awal.

Contoh : ‘S’ = 01010011 → 11001010

Key = 11111111

Result = 00110101

Keempat, pada hasil XOR terebut tukarkan setengah bit pertama (4 bit pada ilustrasi ini) dengan setengah bit terakhir dengan tujuan agar memenuhi konsep confusion.

Contoh : result = 00110101

4-bit pertama = 0011

4-bit kedua = 0101

final = 01010011

Kelima, setelah dilakukan operasi pertukaran tersebut, maka bit inilah yang menjadi cipherteks dari plainteks yang kita enkripsi.

Untuk meningkatkan keamanan dari algoritma ini, kita mencoba mengkombinasikan algoritma ini dengan Jaringan Feistel. Algoritma 64-DIB ini digunakan sebagai fungsi enkripsi pada Jaringn Feistel. Karena 64-DIB menggunakan blok berukuran 64 bit, maka ukuran blok yang akan diproses dalam Jaringan Feistel ini menjadi 2 kali lipatnya yaitu 128 bit. Kunci yang digunakan untuk round pertama adalah kunci masukan dari user di awal, dan untuk round selanjutnya dibuat dengan mengalikan kunci sebelumnya dengan sebuah bilangan prima dan di modulo 2^{64} (untuk memotong panjang kunci menjadi 64 bit). Dan banyaknya round yang digunakan adalah 16, karena 16 dianggap nilai yang cukup ideal agar keamanannya meningkat namun komputasinya juga tidak terlalu banyak.

Pada proses selanjutnya akan dibedakan berdasarkan metode yang kita gunakan:

- ECB

Setelah didapatkan cipherteks hasil dari blok yang pertama, maka untuk enkripsi blok kedua dilakukan hal yang sama pada proses sebelumnya yaitu pemakaian kunci yang sama yang dimasukkan user akan digunakan kembali pada operasi XOR.

- CBC

Setelah didapatkan hasil cipherteks, pada proses berikutnya dilakukan langkah yang sama akan tetapi kunci pertama yang digunakan diganti dengan bit hasil proses sebelumnya. Hal ini dilakukan agar memenuhi konsep dari difusion.

2. Proses Dekripsi

Untuk proses dekripsi pada algoritma 64-DIB, langkah-langkah yang dilakukan adalah melakukan hal yang sama pada saat enkripsi hanya saja prosesnya yang dibalik mulai dari cipherteks yang diperoleh ditukar terlebih dahulu setengah bit pertama dengan setengah bit terakhir. Setelah itu dilakukan XOR dengan kunci awal yang kita punya. Lalu setelah itu hasilnya diinverskan dan plainteks pun akan kembali seperti semula. Begitu seterusnya sampai selesai.

Namun karena algoritma 64-DIB ini dikombinasikan dengan Jaringan Feistel maka proses dekripsi sebenarnya menjadi tidak terlalu penting sebab untuk dekripsi pada Jaringan Feistel tidak membutuhkan fungsi dekripsi. Sedangkan proses dekripsi Feistel dilakukan dengan cara kebalikan dari proses enkirpsi Feistel seperti yang sudah dijelaskan sebelumnya.

IV. Eksperimen dan Pembahasan Hasil

Berikut ini adalah hasil eksperimen yang dilakukan untuk menguji algoritma yang telah kami buat. Untuk teks uji coba yang dilakukan untuk menguji algoritma ini akan digunakan teks Lorem ipsum. Teks ini dipilih karena teks ini mudah didapatkan dan merupakan teks umum yang digunakan untuk uji coba.

Plain text	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Plain text (Hex)	4c6f 7265 6d20 6970 7375 6d20 646f 6c6f 7220 7369 7420 616d 6574 2c20 636f 6e73 6563 7465 7475 7220 6164 6970 6973 6963 696e 6720 656c 6974 2c20 7365 6420 646f 2065 6975 736d 6f64 0a74 656d 706f 7220 696e 6369 6469 6475 6e74 2075 7420 6c61 626f 7265 2065 7420 646f 6c6f 7265 206d 6167 6e61 2061 6c69 7175 612e 2055 7420 656e 696d 2061 6420 6d69 6e69 6d20 7665 6e69 616d 2c0a 7175 6973 206e 6f73

	7472 7564 2065 7865 7263 6974 6174 696f 6e20 756c 6c61 6d63 6f20 6c61 626f 7269 7320 6e69 7369 2075 7420 616c 6971 7569 7020 6578 2065 6120 636f 6d6d 6f64 6f0a 636f 6e73 6571 7561 742e 2044 7569 7320 6175 7465 2069 7275 7265 2064 6f6c 6f72 2069 6e20 7265 7072 6568 656e 6465 7269 7420 696e 2076 6f6c 7570 7461 7465 2076 656c 6974 2065 7373 650a 6369 6c6c 756d 2064 6f6c 6f72 6520 6575 2066 7567 6961 7420 6e75 6c6c 6120 7061 7269 6174 7572 2e20 4578 6365 7074 6575 7220 7369 6e74 206f 6363 6165 6361 7420 6375 7069 6461 7461 7420 6e6f 6e0a 7072 6f69 6465 6e74 2c20 7375 6e74 2069 6e20 6375 6c70 6120 7175 6920 6f66 6669 6369 6120 6465 7365 7275 6e74 206d 6f6c 6c69 7420 616e 696d 2069 6420 6573 7420 6c61 626f 7275 6d2e

A. Menggunakan Mode ECB

Cipher text (hex)	2605 d567 8989 d18a d89e 4902 a28d 3cba 2687 550f b1c9 d16a fe1f fa7e 1d9d 3e3e 2c25 5d2f b929 e93a caef 7dc6 a549 932e 847d 7f9d 8999 238a 2563 d7e3 82c2 06aa 9415 55f9 7bf1 d1a2 a947 1179 9eed 908d

	8cb7 55df f989 2382 f517 84f3 1272 ae24 d485 8d8f c9bb 81e2 cf84 4839 be3f d20a 5635 d527 7b91 8da8 faa6 55b8 7e17 a647 b4c5 ed1f d9d1 231a d65a cadf a172 a402 54b7 b53f 3191 e95a d250 6408 0989 7025 0c35 553f 7bc9 d13a c2d5 95ca 6755 9c5d d4f5 fd9f 7b71 b1e2 e7f2 86d1 beeb e1f5 ac25 4d2f 7bb1 b102 fa07 2d17 b933 8a47 0445 cd1f 8979 775a c6cc a3d2 51b8 338a b41d 59ad d139 2362 a347 690f cf7a b3b8 1445 5dad e9bb 695a bbc5 1b52 0148 da44 8495 0d0f 7b91 b1e2 ce85 b203 acbb 6e4d 849d fd87 4919 49a8 17c0 d0e3 e6ac 07f9 1485 ed6f 59bb 9102 c8c8 90ea edbf bac9 c41d 1dad 7b91 a13a 6335 9181 0fb3 24e5 0c35 ede7 7bcb 6902 49fa 1d12 3b08 7ec5 b47d 7fe7 89bb 09da d943 bfdb 08a6 5a02 b4bd d56f f991 a18a c856 64bf a83b a6c4 8495 ed5f 7b8b 09da 94ba b25b e80e 2835 2c85 d56f d131 232a 40a6 10ae ad22 b312 8495 7f2f d1f1 8162 0a73 b2e3 2dd7 b070 5435 7f87 e9bb 913a cf81 9d08 372f 42fc 8cfd 8def 7fbf 531a a16c a443 236c 504d
Decry-pted text (hex)	4c6f 7265 6d20 6970 7375 6d20 646f 6c6f 7220 7369 7420 616d 6574 2c20 636f 6e73 6563 7465 7475 7220 6164 6970 6973 6963

	696e 6720 656c 6974 2c20 7365 6420 646f 2065 6975 736d 6f64 0a74 656d 706f 7220 696e 6369 6469 6475 6e74 2075 7420 6c61 626f 7265 2065 7420 646f 6c6f 7265 206d 6167 6e61 2061 6c69 7175 612e 2055 7420 656e 696d 2061 6420 6d69 6e69 6d20 7665 6e69 616d 2c0a 7175 6973 206e 6f73 7472 7564 2065 7865 7263 6974 6174 696f 6e20 756c 6c61 6d63 6f20 6c61 626f 7269 7320 6e69 7369 2075 7420 616c 6971 7569 7020 6578 2065 6120 636f 6d6d 6f64 6f0a 636f 6e73 6571 7561 742e 2044 7569 7320 6175 7465 2069 7275 7265 2064 6f6c 6f72 2069 6e20 7265 7072 6568 656e 6465 7269 7420 696e 2076 6f6c 7570 7461 7465 2076 656c 6974 2065 7373 650a 6369 6c6c 756d 2064 6f6c 6f72 6520 6575 2066 7567 6961 7420 6e75 6c6c 6120 7061 7269 6174 7572 2e20 4578 6365 7074 6575 7220 7369 6e74 206f 6363 6165 6361 7420 6375 7069 6461 7461 7420 6e6f 6e0a 7072 6f69 6465 6e74 2c20 7375 6e74 2069 6e20 6375 6c70 6120 7175 6920 6f66 6669 6369 6120 6465 7365 7275 6e74 206d 6f6c 6c69 7420 616e 696d 2069 6420 6573 7420 6c61 626f 7275 6d2e 0000 2af8 de19 b995 09bf 95de cd44 35e4 fdfe

B. Menggunakan Mode CBC

Cipher text (hex)	fe45 d567 2c8e b00b 189e 4b19 9f08 d810 be55 2c17 b9d2 c193 002a 1318 5298 971a 34ed 092f e1c0 f070 22f1 c4a3 3e52 4fa9 415e f0d9 1c6b 69f6 f302 a46c b29f 4a84 a230 1536 5aa3 28ef c14a cf97 43e4 0c31 6544 075c 75b9 0440 58f5 4721 a682 674f 5067 2295 3b5d c087 ad91 2dbe 1a9d 28eb 2b81 5c92 ac85 34f0 fe73 9e0c 8589 3470 84bc 2360 d7fd b2bb 6113 d507 2bd7 3147 b41c 7db9 d31d 028e b587 8c2e ff13 fe89 7804 b492 eab6 19c5 eaec 21c9 e906 da1f 4771 cac8 832a d175 4433 8706 f475 56bd ccc4 810d c6db 1f2d ad67 24f3 e3cd 883b cb61 2baa 5568 c49d db2a a412 068d 60d2 fc38 0d76 9a3f 9202 2db9 4bce 70d4 c5c0 6797 c019 ea18 4254 f8cc 4ca3 31d5 e3dd 41a7 3e10 c056 1a6e ae4a 1746 b72c 95c7 e675 aff2 aab0 7d45 b1f6 2227 27d8 f83d f0c1 82e2 e5a4 8ae6 36cb 1caa 68d9 4fa1 9125 cec1 fe63 3a2c 12bf 0e24 003b 4ee3 2845 10af bcb9 b502 d836 b7bf 0f6f f659 4990 13fc 13d4 ff2a f47d aa70 4764 3166 bae8 6b40 9f1d 8768 03e3 c75d bba0 bc6a 3e76 2a9f 2db6 0c07 958f 625b 454f 2ca6 f6c3 24c6 b405 d188 2c7d 1c89 085d

	f200 15ad c11b d1be 3b72 452a 6d05 3401 e25d f483 2b25 53fc 1116 5228 45a5 4df2 dd2a 295f 99a5 2b04 1ca8 5790 20c9 0616 b2ad
Decry-pted text (hex)	4c6f 7265 6d20 6970 7375 6d20 646f 6c6f 7220 7369 7420 616d 6574 2c20 636f 6e73 6563 7465 7475 7220 6164 6970 6973 6963 696e 6720 656c 6974 2c20 7365 6420 646f 2065 6975 736d 6f64 0a74 656d 706f 7220 696e 6369 6469 6475 6e74 2075 7420 6c61 626f 7265 2065 7420 646f 6c6f 7265 206d 6167 6e61 2061 6c69 7175 612e 2055 7420 656e 696d 2061 6420 6d69 6e69 6d20 7665 6e69 616d 2c0a 7175 6973 206e 6f73 7472 7564 2065 7865 7263 6974 6174 696f 6e20 756c 6c61 6d63 6f20 6c61 626f 7269 7320 6e69 7369 2075 7420 616c 6971 7569 7020 6578 2065 6120 636f 6d6d 6f64 6f0a 636f 6e73 6571 7561 742e 2044 7569 7320 6175 7465 2069 7275 7265 2064 6f6c 6f72 2069 6e20 7265 7072 6568 656e 6465 7269 7420 696e 2076 6f6c 7570 7461 7465 2076 656c 6974 2065 7373 650a 6369 6c6c 756d 2064 6f6c 6f72 6520 6575 2066 7567 6961 7420 6e75 6c6c 6120 7061 7269 6174 7572 2e20 4578 6365 7074 6575 7220 7369 6e74 206f 6363 6165 6361 7420 6375 7069 6461 7461 7420 6e6f 6e0a 7072 6f69 6465 6e74 2c20 7375 6e74 2069 6e20 6375 6c70 6120 7175 6920 6f66 6669 6369 6120 6465 7365 7275 6e74 206d 6f6c 6c69 7420 616e 696d 2069 6420 6573 7420 6c61 626f 7275 6d2e 0000 03a7 47bc 9291 1517 c24e ed8d 33f2 4f53

Dari hasil eksperimen tersebut dapat disimpulkan bahwa algortima ini berhasil untuk melakukan enkripsi dan dekripsi dengan baik. Pada cipher text dan decrypted text terdapat tambahan nilai yang merupakan nilai-nilai tambahan yang diperlukan untuk menggenapkan jumlah bloknya.

V. Analisis Keamanan

A. Analisis terhadap sejumlah serangan

Jika seserorang mencoba untuk menyerang algoritma ini dengan Bruteforce Attack, maka orang tersebut membutuhkan uji coba sebanyak 2^{64} jenis kunci. Dan jika dianggap dalam waktu satu detik orang tersebut bisa melakukan percobaan sebanyak 1 juta maka waktu yang diperlukan untuk menyelesaikan semua percobaan adalah sekitar 600 ribu tahun. Sehingga tidak mungkin orang tersebut memecahkan algoritma ini dengan cukup cepat dengan bruteforce attack.

Kemudian jika ada orang yang mengetahui pasangan plain text dan cipher text, dan orang tersebut mencoba memecahkan algoritma ini, orang tersebut juga masih kesulitan untuk mendapatkan kunci yang digunakan. Hal tersebut karena dalam jaringan Feistel dilakukan 16 kali round, sehingga akan sulit untuk menganalisis keterhubungannya.

VI. Kesimpulan dan Saran

Jadi kesimpulannya adalah algoritma blok cipher yang kami kembangkan berhasil untuk untuk melakukan enkripsi dan dekripsi dengan baik.

Dengan menggunakan kunci 64-bit, algoritma kami sudah cukup bagus untuk menjaga keamanan data dengan mengenkripsinya.

Untuk mengembangkan algortima yang kami buat, sebaiknya ditambahkan operasi substitusi pada kotak S-Box atau semacamnya yang berguna untuk membuat algoritma lebih susah untuk dipecahkan. Selain itu operasi blok bisa ditambahkan untuk membuat algoritma ini menjadi semakin sukar, yang asalnya 64 bit per blok maka ditambah misal nya 256 bit atau 512 bit per bloknya.

VII. Daftar Referensi

- http://www.ilmuhacking.com/cryptography/padding-oracle-attack/
- Rinaldi. Bahan Kuliah IF4020 Kriptografi: Algoritma Kriptografi Modern
- https://ilmukriptografi.wordpress.com/2012/04/25/kriptografi-simetrik-dasar-block-cipher/