loremCipher

Tugas II IF4020 Kriptografi

Nicholas Chen - 13519029
Cynthia Rusadi - 13519118
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail:
13519029@std.stei.itb.ac.id
13519118@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas mengenai usulan perancangan sebuah *block cipher* baru bernama loremCipher. Algoritma ini merupakan rancangan lebih lanjut dari Algoritma DES. Beberapa prinsip seperti *confusion* dan *diffusion* oleh Shannon serta konsep Jaringan Feistel digunakan dalam pembangunan algoritma ini. Ukuran blok dan panjang kunci yang digunakan oleh algoritma ini adalah 128 bit, dengan melakukan proses putaran sebanyak 16 kali. Berdasarkan hasil analisis dan eksperimen yang telah dilakukan, algoritma loremCipher aman untuk digunakan karena memiliki ketahanan terhadap analisis frekuensi, memiliki efek longsoran yang baik, serta memiliki ruang kunci yang besar.**

Keywords—block cipher; DES; loremCipher; Prinsip Confusion dan Diffusion; Jaringan Feistel

I. PENDAHULUAN

Dewasa ini bidang ilmu teknologi terutama pada bagian penyebaran dan pengolahan informasi semakin disruptif terhadap perkembangan berbagai aspek dalam kehidupan manusia. Hal ini tidak hanya membuka alternatif baru bagi manusia dalam bertukar informasi, namun juga membuka alternatif baru bagi para oknum untuk menyadap, mencuri, maupun menyabotase informasi-informasi tersebut. Untuk itu, suatu mekanisme agar pertukaran informasi dapat berlangsung dengan aman sangat dibutuhkan.

Kriptografi adalah bidang ilmu yang mempelajari proses enkripsi dan dekripsi suatu pesan berupa teks, citra, suara, video, dan berbagai media informasi lainnya. Berbagai algoritma kriptografi sudah dikembangkan, mulai dari yang sederhana seperti algoritma *Caesar Cipher*, sampai dengan algoritma DES dan AES yang digunakan sebagai standar kriptografi saat ini.

Pada makalah ini, sebuah algoritma kriptografi simetri yang baru dikembangkan berdasarkan fundamental cara kerja dari algoritma DES. Algoritma ini menerapkan prinsip *diffusion* dan *confusion* dari Shannon, perulangan *cipher* dengan jumlah putaran sebanyak 16 kali, serta memiliki ukuran blok pesan maupun panjang kunci sebesar 128 bit.

II. STUDI PUSTAKA

A. Block Cipher

Pada *block cipher*, plainteks terbagi menjadi blok-blok bit yang memiliki panjang yang sama. Pada umumnya, ukuran bloknya adalah 64 bit, 128 bit, 256, bit, dsb. Untuk setiap blok plainteks, dilakukan enkripsi dengan menggunakan bit-bit kunci. Panjang blok cipherteks sama dengan panjang blok plainteks, sedangkan panjang kunci eksternal tidak harus sama dengan panjang blok plainteks.

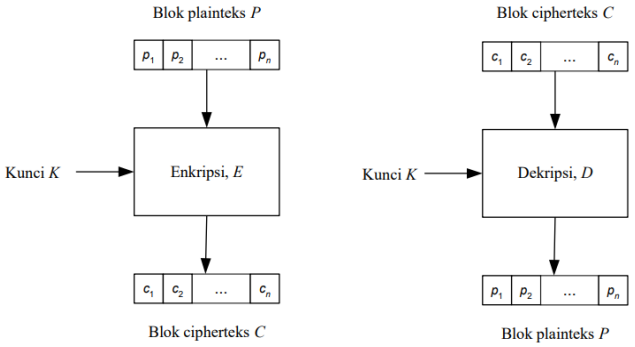


Fig. 1. Enkripsi dan Dekripsi *Block Cipher*

Sejauh ini, sudah ada berbagai algoritma *block cipher* yang diteliti dan/atau ditemukan, seperti Lucifer, DES, GOST, AES, dst. Selain algoritma *block cipher*, terdapat mode operasi *block cipher* yang dapat diperhatikan juga. Mode operasi *block cipher* terbagi menjadi lima, yaitu *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB), dan *Counter Mode*.

B. Prinsip Confusion dan Diffusion Shannon

Pada makalah "*Communication theory of secrecy systems"*, oleh Claude Shannon pada tahun 1949, Prinsip *Confusion* dan *Diffusion* dikenalkan untuk mempersulit ciphertext untuk dipecahkan menggunakan serangan berbasis statistik. Kedua prinsip ini sering diimplementasikan secara bersamaan pada algoritma *block cipher* modern karena sifatnya yang saling melengkapi.

Prinsip *Confusion* digunakan untuk menyembunyikan hubungan sebuah / sebagian huruf pada *plaintext* terhadap sebuah / sebagian huruf pada *ciphertext* yang berkoresponden. Prinsip ini diterapkan dengan maksud untuk mempersulit kriptanalis dalam mencari pola-pola statistik pada *ciphertext*. Teknik substitusi nirlanjar (*non-linear*) dapat digunakan untuk merealisasikan prinsip ini.

Prinsip *Diffusion* digunakan untuk menyebarkan pengaruh dari masing-masing bit pada *plaintext* atau kunci terhadap sebanyak mungkin bit pada *ciphertext*. Algoritma *block cipher* yang menerapkan prinsip ini akan memiliki perubahan drastis yang sulit diprediksi pada *ciphertext* jika ada sebagian karakter pada *plaintext* atau kunci yang berubah. Teknik permutasi pada level bit dapat digunakan untuk merealisasikan prinsip ini.

C. *Jaringan Feistel*

Jaringan Feistel merupakan struktur *enciphering* yang digunakan untuk setiap putaran dan digunakan oleh berbagai algoritma *block cipher*, seperti DES, GOST, Feal, Lucifer, RC5, RC6, dsb. Dengan menggunakan jaringan Feistel, blok plainteks dibagi menjadi dua bagian, dengan masing-masing blok akan dilakukan proses transformasi menjadi upa-bagian untuk putaran selanjutnya.

Struktur dari jaringan ini bersifat *reversible*, yang berarti enkripsi dilakukan dari "atas" ke "bawah" dan dekripsi dilakukan dari "bawah" ke "atas". Sifat inilah yang memudahkan perancang *block cipher* untuk tidak membuat algoritma baru untuk melakukan dekripsi cipherteks menjadi plainteks. Sifat ini tidak bergantung pada fungsi *f*, maka fungsi *f* dapat dirancang serumit mungkin.

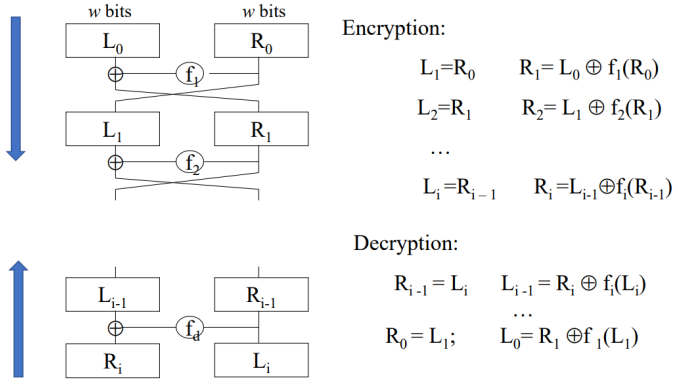


Fig. 2. Jaringan Feistel

D. *Data Encryption Standard (DES)*

Pada tahun 1974, Horst Feistel di IBM merancang *block cipher* Lucifer (panjang kunci dan panjang blok 128 bit) dan kemudian berkembang menjadi DES dengan berbagai masukan dari NSA (*National Security Agency*), yaitu panjang kunci 56 bit, ukuran blok 64 bit, dan dilakukan sebanyak 16 putaran. Pada tahun 1976, NBS (*National Bureau of Standard*) menyetujui DES dan semenjak itu, DES diimplementasikan baik *software* maupun *hardware*. Tetapi pada tahun 1997-1998, DES berhasil dipecahkan dengan menggunakan *brute force attack* dan pada tahun 2001, DES diganti oleh AES (*Advanced Encryption Standard)*.

Enkripsi pada algoritma ini dilakukan dengan 16 putaran *enciphering* untuk setiap blok plainteks, dengan setiap putarannya menggunakan kunci internal/kunci putaran yang berbeda, yang panjangnya 48 bit dan dibangkitkan dari kunci eksternal. Alur dari DES adalah setiap blok plainteks mengalami permutasi awal (IP), 16 putaran *enciphering*, dan inversi permutasi awal (IP^{-1}).

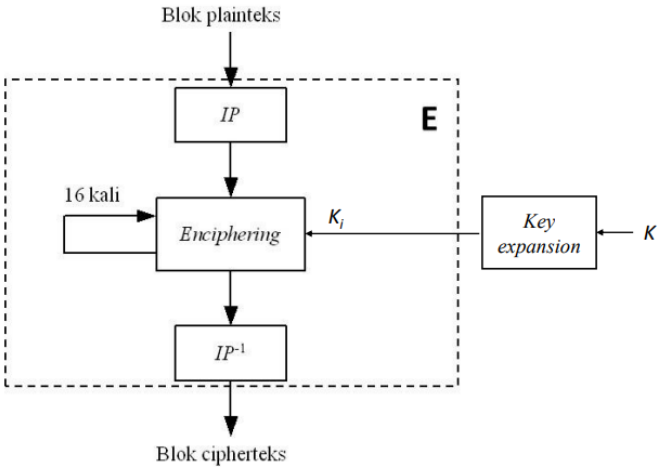


Fig. 3. Skema Algoritma DES

III. Rancangan Block Cipher Baru

Algoritma loremCipher adalah algoritma *block cipher* yang dirancang dengan algoritma DES sebagai referensinya. Algoritma ini dapat mengenkripsi blok pesan berukuran 128 bit menggunakan kunci dengan panjang 128 bit.

Pada Algoritma loremCipher, proses enkripsi ataupun deskripsi dibagi menjadi 2 bagian, yaitu bagian pembangkitan kunci dari kunci eksternal, serta proses enkripsi / dekripsi itu sendiri.

A. *Rancangan Algoritma Pembangkitan Kunci*

Algoritma loremCipher membutuhkan sebuah kunci sepanjang 128 bit untuk melakukan proses enkripsi maupun dekripsi. Kunci awal ini disebut juga sebagai kunci eksternal. Kunci eksternal ini digunakan untuk membangkitkan kunci-kunci internal yang selanjutnya akan digunakan pada proses enkripsi / dekripsi pada masing-masing putaran.

Pada awalnya, kunci eksternal yang berukuran 128 bit akan dipermutasi menggunakan sebuah tabel permutasi. Isi dari tabel permutasi yang digunakan dapat dilihat pada gambar di bawah.

118	45	25	72	102	22	18	46	76	33	98	88	34	96	2	35
106	24	50	56	63	121	51	103	53	57	58	37	64	90	128	89
61	105	26	95	28	62	107	123	32	92	100	113	111	68	114	42
6	23	49	9	48	11	80	78	16	116	66	85	39	93	77	59
120	86	5	30	17	19	119	36	47	15	99	12	10	38	29	69
14	108	7	54	73	81	71	109	126	21	8	79	112	3	117	104
1	20	55	84	97	31	125	127	52	82	41	4	91	13	87	94
75	40	65	101	110	124	43	67	70	44	27	115	60	122	74	83

Fig. 4. Tabel Permutasi I

Hasil dari permutasi yang telah dilakukan selanjutnya akan dibagi menjadi 4 sub-blok sama besar berukuran 32 bit (blok A, B, C, dan D). Masing-masing sub-blok tersebut akan dilakukan operasi XOR dengan sub-blok kanannya seperti yang terlihat pada skema di bawah.

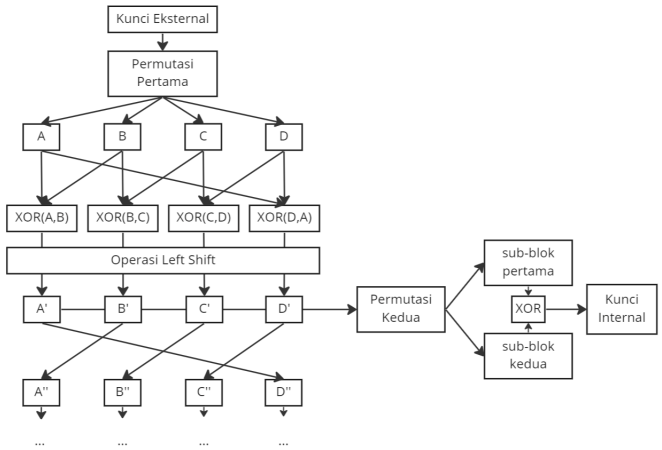


Fig. 5. Skema Pembangkitan Kunci

Berikutnya, pada masing-masing sub-blok dilakukan left shift dengan ketentuan berikut: sub-blok A di-left-shift sebesar 1 bit, sub-blok B di-left-shift sebesar 2, sub-blok C di-left-shift sebesar 3 bit, dan sub-blok D di-left-shift sebesar 4 bit. Hasil dari operasi left shift akan digunakan untuk membangkitkan kunci internal pada putaran tersebut.

Untuk membangkitkan kunci internal, hasil sub-blok pada operasi sebelumnya akan digabungkan (*merge*) kembali menjadi satu untuk dilakukan proses permutasi menggunakan tabel permutasi dibawah ini. Setelah itu hasil permutasi tersebut dibagi menjadi 2 sub-blok berukuran 64 bit. Kedua sub-blok tersebut akan dilakukan operasi XOR untuk menghasilkan kunci internal untuk putaran tersebut. Kunci internal pada setiap putaran memiliki panjang 64 bit.

77	74	102	45	16	118	29	99	32	53	17	65	57	75	64	6
103	50	49	94	8	2	13	40	24	106	10	76	31	63	35	1
96	114	62	78	14	61	104	124	95	30	100	46	56	39	97	70
82	93	47	18	15	66	122	126	123	125	34	27	23	25	105	79
73	101	88	67	60	52	59	127	109	7	110	121	4	87	72	90
54	12	108	115	98	43	36	113	85	20	68	44	112	38	21	69
26	33	117	19	91	111	84	41	120	128	116	28	11	51	37	92
5	83	80	81	86	119	55	42	48	107	3	22	71	89	9	58

Fig. 6. Tabel Permutasi II

Sebelum masuk ke putaran selanjutnya, urutan sub-blok A, B, C, D dari operasi sebelumnya akan diubah terlebih dahulu. Perubahan dilakukan dengan cara menggeser masing-masing sub-blok kekanan, sehingga sub-blok B menempati posisi sub-blok A sebelumnya, sub-blok C menempati posisi sub-blok B sebelumnya, dan seterusnya. Setelah proses ini dilakukan, sub-blok A, B, C, dan D yang baru siap untuk diproses pada putaran selanjutnya.

B. Rancangan Algoritma Enkripsi

Algoritma enkripsi dilakukan pada beberapa tahapan:

1. Mengubah Plainteks

   Plainteks yang diterima merupakan sebuah *string* dan akan diubah menjadi biner basis 8.

2. Permutasi Awal

   Permutasi awal pada Algoritma DES dilakukan dengan melakukan permutasi plainteks dengan matriks IP saja.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	14	5	63	55	47	39	31	23	15	7

Fig. 7. Matriks IP

Algoritma loremCipher melakukan permutasi awal dengan menggunakan matriks IP dan juga matriks IP^{-1}.

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Fig. 8. Matriks IP^{-1}

Hal ini dilakukan karena kedua matriks berukuran 64 bit saja, sedangkan ukuran blok pesan adalah sebesar 128 bit. Maka dari itu, 64 bit pertama pada blok pesan akan dipermutasikan dengan matriks IP dan 64 bit kedua pada blok pesan akan dipermutasikan dengan matriks IP^{-1}, dengan begitu, hasil dari permutasi awal akan tetap menjadi 128 bit. Alasan mengapa tidak menggunakan matriks IP saja untuk melakukan permutasi pada 64 bit pertama dan 64 bit kedua adalah untuk menghasilkan acakan yang lebih unik. Hasil dari permutasi awal akan terbagi menjadi dua, untuk 64 bit pertama akan dinamakan sebagai L dan 64 bit kedua akan dinamakan sebagai R.

3. Fungsi Feistel

   Alur pada jaringan Feistel pada awalnya R akan dipermutasikan dengan matriks permutasi ekspansi dan hasil tersebut akan di-XOR-kan dengan kunci yang telah dibangkitkan sebelumnya, lalu dilakukan substitusi dengan kotak-S.

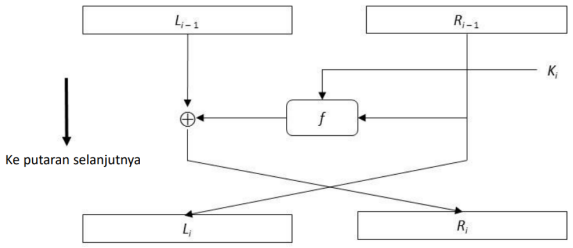
32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Fig. 9. Matriks Permutasi Ekspansi

Pada Algoritma loremCipher, urutan dari jaringan Feistel tersebut dimodifikasi, sehingga fungsi XOR akan dijalankan terlebih dahulu dan tahapan selanjutnya adalah hasil dari XOR tersebut akan

dipermutasikan dengan matriks permutasi ekspansi, lalu substitusi dengan kotak-S.

Setelah itu, dilakukan permutasi dengan matriks permutasi P, dan setelah itu di-XOR-kan dengan L. Dengan begitu, hasil XOR akan disubstitusi menjadi R, dan nilai L berubah menjadi nilai R yang sebelumnya. Tahapan ini akan diulang sebesar 16 putaran.



Jaringan *Feistel* pada enkripsi putaran ke-*i*

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

Fig. 10. Enkripsi Fungsi Feistel

4. Permutasi Akhir

Serupa dengan Algoritma DES, permutasi akhir dilakukan dengan melakukan permutasi untuk gabungan L dan R dengan matriks IP^{-1}, algoritma loremCipher melakukan permutasinya akhirnya dengan melakukan permutasi L dengan matriks IP dan R dengan matriks IP^{-1}. Setelah dilakukan permutasi untuk kedua hal tersebut, hasilnya akan digabungkan. Gabungan tersebut merupakan nilai akhirnya, yang masih dalam bentuk biner, lalu diubah menjadi heksadesimal berbasis 4.

C. *Rancangan Algoritma Dekripsi*

Algoritma dekripsi untuk Algoritma loremCipher merupakan sebaliknya dari algoritma enkripsi. Tahapannya adalah sebagai berikut:

1. Mengubah Cipherteks

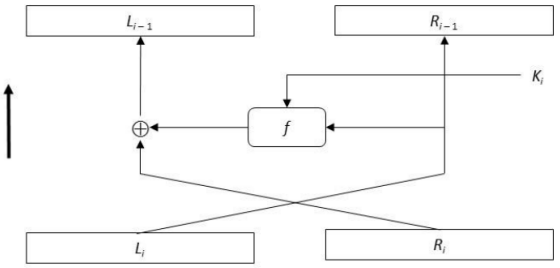
   Cipherteks yang mulanya merupakan heksadesimal basis 4 akan diubah menjadi biner basis 8.

2. Permutasi Awal

   Tahapan ini dilakukan sama seperti permutasi awal untuk algoritma enkripsi.

3. Fungsi Feistel

   Pada tahapan ini, ada perbedaan sedikit dengan fungsi feistel pada algoritma enkripsi. Perbedaannya adalah L dan R pada algoritma enkripsi masing-masing akan diberlakukan sebagai blok kiri dan blok kanan, sedangkan untuk melakukan dekripsi, L dan R masing-masing akan diberlakukan sebagai blok kanan dan blok kiri. Hal ini dilakukan karena blok kiri dan blok kanan selalu ditukar untuk setiap putaran, seperti yang ada pada gambar di bawah.



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

Fig. 11. Dekripsi Fungsi Feistel

4. Permutasi Akhir

Tahapan ini dilakukan sama seperti permutasi akhir untuk algoritma dekripsi, dengan perbedaanya adalah konversi gabungan akhir tersebut menjadi plainteks. Gabungan akhir masih dalam bentuk biner dan dengan ekspektasi hasil akhirnya berupa *string*, maka gabungan tersebut diubah menjadi *string*.

IV. Eksperimen dan Pembahasan Hasil

Algoritma loremCipher yang sudah dirancang sebelumnya diimplementasikan menggunakan python 3 untuk dianalisis ketahanan dan keamanannya. Pada eksperimen ini, algoritma *block cipher* baru akan dihitung waktu eksekusi yang dibutuhkan untuk mengenkripsi dan mendekripsi file dengan panjang file yang beragam, dianalisis kekuatan efek longsoran dari algoritmanya, serta dihitung banyak ruang kuncinya.

A. *Waktu Enkripsi dan Dekripsi pada Ukuran Beragam*

Pada eksperimen bagian ini, 3 jenis file dengan ukuran masing-masing 2.560 bit, 51.200 bit, dan 128.000 bit dienkripsi dan didekripsi menggunakan algoritma loremCipher. Waktu eksekusi dari masing-masing percobaan dicatat untuk dianalisis lebih lanjut. Hasil dari pencatatan waktu untuk masing-masing ukuran file dapat dilihat pada tabel berikut.

TABLE I. Tabel Hubungan Waktu Enkripsi dan Dekripsi dengan Ukuran File

Panjang File (bit)	**Performansi**		
	Waktu Enkripsi (sekon)	***Waktu Dekripsi (sekon)***	***Total Waktu (sekon)***
2.560 (= 320 char)	0.00198	0.00200	0.00398
51.200 (= 6.400 char)	0.00609	0.00292	0.00901
128.000 (= 16.000 char)	0.00901	0.00304	0.01204

Berdasarkan hasil eksperimen, algoritma loremCipher memiliki waktu eksekusi yang cukup baik. Algoritma ini dapat mengenkripsi pesan dengan 16.000 karakter dalam waktu kurang dari 0.01 detik, serta mendekripsi kembali pesan tersebut dengan waktu kurang dari 0.005 detik.

B. Analisis Efek Longsoran (Avalanche Effect)

Pada eksperimen bagian ini, tingkat perbedaan antara hasil enkripsi akan dibandingkan untuk input *plaintext* yang mirip menggunakan kunci yang sama. Pada eksperimen, *plaintext* awal akan diubah satu karakter secara acak, lalu dibadingkan hasil enkripsinya. Hasil eksperimen dapat dilihat pada tabel berikut.

TABLE II. TABEL ANALISIS EFEK LONGORAN

Plain Text	Kunci	Hasil Enkripsi
“abcdefghijklmnop”	“iniadalahkunci:)”	“8295B7A06A288AC0DE4A247CDB4F216D”
“aacdefghijklmnop”		“4E1F85963A796C675C6831195F62330E”
“aacdefghijklmnnp”		“9195617236246C77DE1C7CBBD01279A9”

*perubahan pada plaintext ditandai dengan highlight

Berdasarkan hasil eksperimen, perubahan satu karakter pada *plaintext* menyebabkan perubahan karakter pada hasil enkripsi yang cukup acak yang sulit diprediksi. Selanjutnya juga dilakukan eksperimen untuk *plaintext* yang sama, namun kunci yang sedikit berbeda. Hasil eksperimen dapat dilihat pada tabel dibawah ini.

TABLE III. TABEL ANALISIS EFEK LONGORAN

Plain Text	Kunci	Hasil Enkripsi
“abcdefghijklmnop”	“iniadalahkunci:)”	“8295B7A06A288AC0DE4A247CDB4F216D”
	“iniadalahkknci:)”	“F1E7B9EBA5A28398F7D86D8FFAD76C9E”
	“iniaaalahkknci:)”	“FAAC7620D98FF3AD16C3B0DE1FCAB2CF”

*perubahan pada plaintext ditandai dengan highlight

Berdasarkan hasil eksperimen, perubahan satu buah karakter pada kunci juga membuat perubahan yang cukup acak pada hasil enkripsi. Hal ini membuat proses kriptanalisis menggunakan statistik sangat sulit untuk dilakukan.

C. Analisis Ruang Kunci (Key Space)

Ruang kunci (*Key Space*) algoritma *loremCipher* sebanyak 2^{128} atau lebih dari 340 x 10^{36}. Jika algoritma ini diserang menggunakan serangan *brute force* dengan kecepatan 1 juta serangan per detik, maka akan dibutuhkan waktu kurang setidaknya 340 x 10^{30} detik atau 107 x 10^{23} tahun untuk mencoba semua kemungkinan kunci.

V. KESIMPULAN DAN SARAN

Algoritma *block cipher* baru, loremCipher, dapat mengenkripsi blok berukuran 128 bit menggunakan panjang kunci 128 bit. Algoritma ini dirancang menggunakan prinsip *confusion* dan *diffusion* Shannon serta menggunakan konsep jaringan Feistel.

Berdasarkan hasil analisis, algoritma ini memiliki tingkat keamanan yang baik. Algoritma ini dapat melakukan enkripsi dan dekripsi dengan kecepatan yang baik sehingga dapat digunakan dengan mudah. Algoritma ini juga memiliki efek longsoran yang sangat kuat sehingga mempersulit kriptanalisis dalam memecahkan algoritma ini. Selain itu, algoritma ini memiliki ruang kunci yang besar sehingga sulit untuk diserang secara *brute force*.

REFERENCES

[1] The DES Algorithm Illustrated, diakses pada 4 Maret 2023.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/13-Block-Cipher-Bagian1-2023.pdf diakses pada 5 Maret 2023.

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/14-Prancangan-block-cipher-2023.pdf, diakses pada 4 Maret 2023.

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/15-Beberapa-block-cipher-bagian1-2023.pdf, diakses pada 4 Maret 2023.

[5] William Stallings, Cryptography and Network Security: Principles and Practice, 7th ed.. England, 2017, pp.87.