LIAN: *Block Cipher* dengan pendekatan *Pseudo-random number*

Suhailie[1], **Josep Andre Ginting**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: 13517045@std.stei.itb.ac.id, 13517108@std.stei.itb.ac.id

Abstract. Jurnal ini memperkenalkan sebuah *block cipher* baru yaitu LIAN *block cipher*. *Block cipher* ini menggunakan implementasi *pseudo-random number* untuk menghasilkan kunci putaran enkripsi yang berbeda pada setiap putaran. LIAN *block cipher* menggunakan blok sebesar 128 bit, dengan pemanfaatan jaringan *feistel* sebanyak 16 kali putaran, dan menggunakan transformasi substitusi, transposisi dan operasi modulo. Panjang kunci yang digunakan pada algoritma LIAN *block cipher* ini minimal sepanjang 128 bit. Hasil eksperimen dan analisis menunjukkan keberhasilan LIAN untuk menerapkan prinsip *confusion* dan *diffusion Shannon* dengan baik.
Keywords: enkripsi dan dekripsi, *block cipher*, *pseudo-random number*, jaringan *feistel*, transformasi, *confusion*, *diffusion*, .

1. Pendahuluan

Informasi telah menjadi salah satu kebutuhan pokok bagi manusia dalam zaman sekarang. Dengan perkembangan jaman yang sangat cepat, teknologi berubah seiring berjalannya waktu dan informasi menjadi sumber utama dalam setiap bidang untuk melakukan perkembangan di bidang tersebut. Dalam era yang canggih dan kompetitif ini, keamanan informasi menjadi salah satu hal yang penting bagi setiap organisasi atau perusahaan. Hal tersebut diperlukan untuk menghindari kecurangan dan menjaga privasi.

Kriptografi menjadi peranan penting dalam menjaga kerahasiaan sebuah informasi. Kriptografi merupakan ilmu dan seni yang mempelajari teknik - teknik untuk menjaga kerahasiaan informasi [8]. Kriptografi terdiri atas 2 tahap yaitu tahap enkripsi sebagai proses perubahan teks informasi menjadi sebuah kode yang disebut *ciphertext*, dan tahap dekripsi sebagai proses perubahan kode kembali menjadi teks informasi semula. Terdapat banyak teknik dalam kriptografi, salah satu teknik tersebut adalah *block cipher*. *Block cipher* merupakan teknik mengenkripsi pesan per blok. Pesan dibagi menjadi blok-blok dalam bentuk bit, kemudian akan dienkripsi dan menghasilkan *ciphertext* per blok.

Beberapa contoh *block cipher* yang telah dibuat adalah *Data Encryption Standard* (DES) dan *Advanced Encryption Standard* (AES). DES merupakan sebuah standar yang menggunakan algoritma *Data Encryption Standard* (DEA) dalam melakukan enkripsi pesan [3]. DES beroperasi pada blok 64 bit, dengan urutan langkah yaitu: permutasi awal (IP), putaran jaringan *feistel* sebanyak 16 kali, dan inversi permutasi awal (IP^{-1}). Dalam putaran jaringan *feistel*, kunci internal yang dipakai dalam putaran dibangkitkan menggunakan kunci eksternal yang diberikan oleh pengguna.

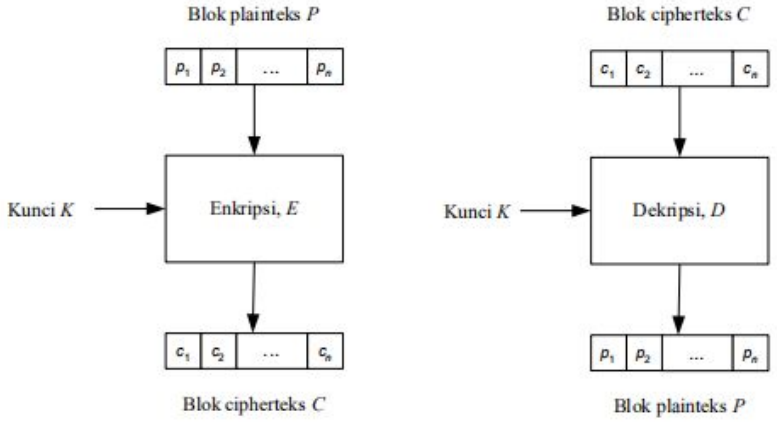
AES merupakan standar yang menggunakan algoritma Rijndael [4]. Standar ini diciptakan untuk menutupi kekurangan dari DES, karena DES dapat dipecahkan melalui teknik *brute-force*. Algoritma Rijndael menggunakan panjang kunci 128 bit sampai dengan 256 bit dengan *step* 32 bit (128, 160, 192, …., 256). Dalam AES, panjang kunci yang ditetapkan adalah 128 bit, 192 bit dan 256 bit. AES menggunakan teknik enkripsi dalam sejumlah putaran seperti DES. Namun, setiap perputaran pada AES menggunakan kunci internal yang berbeda (*round key*).

Algoritma LIAN sendiri menggunakan pendekatan/gagasan *pseudo-random number* dan juga transpos matriks. *Pseudo-random number* digunakan untuk membangkitkan kunci putaran secara acak, sedangkan transpos matriks digunakan untuk transposisi/pergeseran bit pada fungsi internal.

2. Studi Pustaka

2.1. *Block Cipher*

Block Cipher merupakan teknik enkripsi pesan yang beroperasi pada level bit. *Block Cipher* merupakan algoritma kriptografi kunci simetri yang mengenkripsi satu blok plainteks dengan jumlah bit tertentu dan menghasilkan blok cipherteks dengan jumlah bit yang sama. Ukuran blok dapat berupa 64-bit, 128-bit, atau yang lainnya.



Gambar 1. Skema enkripsi dan dekripsi pada *block cipher*.

Block Cipher dapat beroperasi dengan beberapa mode berbeda, diantaranya:

2.1.1. Electronic Code Block (ECB)

Setiap blok plainteks Pi dienkripsikan secara individual dan independen dari blok lainnya menjadi blok cipherteks Ci.

2.1.2. Cipher Block Chaining (CBC)

Setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya, tetapi juga pada seluruh blok plainteks sebelumnya. Hasil enkripsi blok sebelumnya diumpan-balikkan ke dalam enkripsi blok saat ini.

2.1.3. Cipher Feedback (CFB)

Mengatasi kekurangan pada mode CBC apabila diterapkan pada pengiriman data yang belum mencapai ukuran satu blok. Data di enkripsi dalam unit yang lebih kecil daripada ukuran blok. Unit data yang dienkripsi panjangnya bisa 1 bit, 2 bit, 4 bit, 8 bit, dan lain-lain.

2.1.4. Output Feedback (OFB)

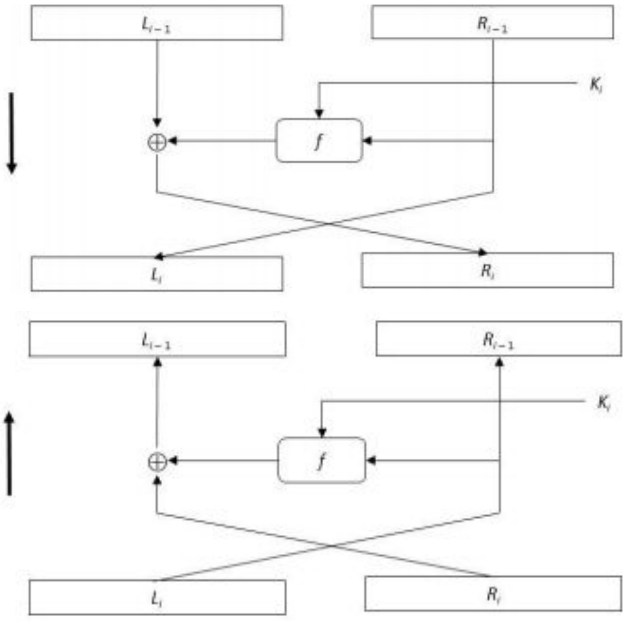
Mode OFB mirip dengan mode CFB, kecuali n-bit hasil enkripsi antrian disalin menjadi elemen posisi paling kanan di antrian.

2.1.5. Counter Mode

Mode *counter* tidak melakukan perantaian (*chaining*) seperti pada mode CBC. *Counter* adalah sebuah nilai berupa blok bit yang ukurannya sama dengan ukuran blok plainteks. Nilai *counter* harus berbeda dari setiap blok yang dienkripsi.

2.2. *Jaringan Feistel*

Jaringan *Feistel* banyak dipakai pada algoritma kriptografi karena model ini bersifat *reversible* untuk proses enkripsi dan dekripsi. Sifat *reversible* ini membuat kita tidak perlu membuat algoritma baru untuk mendekripsi cipherteks menjadi plainteks. Sifat *reversible* tidak tergantung pada fungsi *f* sehingga fungsi *f* dapat dibuat serumit mungkin.



Gambar 2. Jaringan *Feistel* pada enkripsi dan dekripsi putaran ke-*i* .

2.3. Prinsip Confusion dan Diffusion Shannon

Diperkenalkan oleh Claude Shannon dalam makalah klasiknya tahun 1949 yaitu *Communication theory of secrecy systems.* Prinsip *confusion* dan *diffusion* diterapkan untuk membuat serangan statistik pada cipher menjadi rumit.

2.3.1. Confusion

Prinsip ini menyembunyikan hubungan apapun yang ada antara plainteks, cipherteks, dan kunci. Prinsip *confusion* membuat kriptanalis frustasi untuk mencari pola-pola statistik yang muncul pada cipherteks. *Confusion* dapat direalisasikan dengan menggunakan algoritma substitusi yang kompleks.

2.3.2. Diffusion

Prinsip ini menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Sebagai contoh, pengubahan kecil pada plainteks sebanyak satu atau dua bit menghasilkan perubahan pada cipherteks yang tidak dapat diprediksi.

2.4. *Iterated Cipher*

Fungsi transformasi sederhana yang mengubah plainteks menjadi cipherteks diulang sejumlah beberapa kali. Pada setiap putaran digunakan kunci putaran (*round key*) yang dikombinasikan dengan plainteks.

$$C_i = f(C_{i-1},\ K_i) \tag{1}$$

i = 1, 2, …, r (r adalah jumlah putaran).

K_i = kunci putaran (*round key*) pada putaran ke i.

f = fungsi transformasi

3. LIAN *Block Cipher*

LIAN *Block Cipher* ini memiliki ukuran blok sebesar 128 bit. Dengan panjang kunci minimal sepanjang ukuran blok yaitu minimal sepanjang 128 bit. LIAN *Block Cipher* ini juga menerapkan prinsip jaringan Feistel dalam algoritmanya dan dapat beroperasi dalam 3 mode operasi yang berbeda yaitu *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), dan *Counter Mode.* Sebagai jaringan *Feistel,* LIAN menerapkan 16 putaran (*round*) ketika beroperasi, dengan fungsi internal yang meliputi substitusi, transposisi/pergeseran dan juga operasi modulo.

Penerapan jaringan *Feistel* dengan input sebesar 128 bit, maka akan dibagi menjadi sub blok kiri dan sub blok kanan dengan masing-masing ukuran 64 bit, sehingga fungsi internal nantinya akan memproses bit dan kunci sepanjang 64 bit. Jaringan *Feistel* akan berjalan sebanyak 16 putaran untuk setiap blok *cipher* yang akan di enkripsi/dekripsi. Dimana setiap putaran akan melakukan operasi-operasi berikut :

3.1. Pembangkitan kunci putaran

Kunci putaran dibangkitkan dengan cara melakukan pengacakan indeks posisi dari kunci eksternal. Proses pengacakan untuk menentukan index dari kunci eksternal yang akan digunakan sebagai kunci putaran didapatkan dengan cara membangkitkan *pseudo-random number* dari kunci eksternal dan juga nomor putaran itu sendiri.

Orde dari kunci eksternal diperoleh dengan cara sebagai berikut. Misalnya kunci eksternalnya adalah ‘KRIPTOGRAFI’, maka orde dari kunci tersebut adalah:

$$orde('KRIPTOGRAFI') = orde('K') + orde('R') + ... + orde('I') \tag{2}$$

Seed yang digunakan pada *pseudo-random number* untuk membangkitkan kunci putaran adalah hasil penjumlahan antara nomor iterasi dengan orde dari kunci eksternal.

$$Seed = orde(K) + n \tag{3}$$

K = kunci eksternal

n = nomor iterasi/putaran

Dari hasil *seed* yang diperoleh, maka akan dilakukan pengacakan bilangan integer sepanjang indeks dari kunci eksternal. Pengacakan dilakukan hingga ditemukan 8 buah karakter hasil, yang jika diubah ke bentuk biner akan menjadi biner berukuran 64 bit.

3.2. Fungsi internal f

Pada fungsi internal f, akan dilakukan beberapa operasi, operasi-operasi tersebut yaitu:

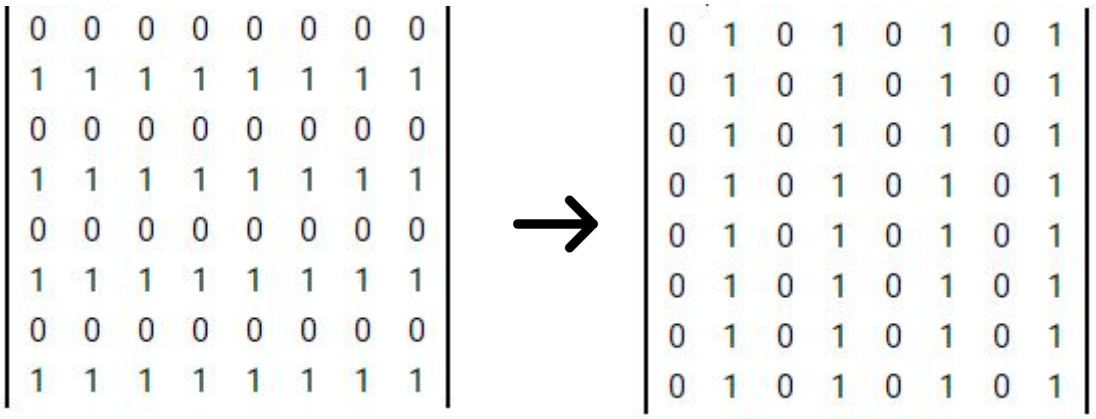
3.2.1. Substitusi bit pada indeks yang bukan perpangkatan 2

Substitusi bit ini dilakukan dengan menukar bit-bit pada indeks yang bukan perpangkatan 2. Penukaran yang dilakukan adalah menukar bit 0 menjadi 1, begitu juga sebaliknya menukar bit 1 menjadi 0 pada indeks yang bukan perpangkatan 2. Karena panjang bit yang masuk ke fungsi internal adalah sepanjang 64 bit, maka indeks yang tidak dilakukan substitusi adalah hanya posisi 0, 1, 2, 4, 8, 16, dan 32.

3.2.2. Transposisi bit

Proses transposisi bit dilakukan dengan mengubah sub blok input sepanjang 64 bit ke dalam bentuk matriks berukuran $8 \, x \, 8$. Kemudian menerapkan transpos matriks ke matriks tersebut. Matriks hasil transpos matriks tersebut kemudian diubah kembali ke bentuk bit sub blok.

Misalnya, 64 bit nya = 0000000011111111000000001111111100000000111111110000000011111111. Hasil transpos nya = 0101010101010101010101010101010101010101010101010101010101010101.



Gambar 3. Ilustrasi proses transpos matriks.

3.2.3. Substitusi S-BOX

S-BOX yang digunakan pada substitusi ini adalah Rijndael S-BOX. Sebelum dilakukan substitusi, sub blok input sepanjang 64 bit setiap 8 bitnya akan diubah ke dalam heksadesimal, sehingga akan terdapat 8 bilangan heksadesimal yang akan disubstitusi dengan S-BOX.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Gambar 4. Rijndael S-BOX.

Hasil substitusi menggunakan rijndael S-BOX tersebut adalah 8 buah bilangan heksadesimal. Sebelum diteruskan ke operasi berikutnya, 8 bilangan heksadesimal tersebut diubah kembali kedalam bentuk biner sepanjang 64 bit.

3.2.4. Operasi modulo

Pada awalnya, kunci putaran sepanjang 64 bit akan diubah kedalam bentuk integer. Input yang merupakan hasil operasi sebelumnya yaitu biner sepanjang 64 bit akan dibagi menjadi potongan-potongan biner berukuran 8 bit, sehingga akan dihasilkan sebanyak 8 buah potongan biner. Masing-masing potongan akan diubah kedalam bentuk integer. Setiap potongan bilangan integer input tersebut akan dikalikan dengan integer kunci putaran, kemudian hasilnya akan diterapkan operasi modulo 255. Semua hasil operasi modulo tersebut kemudian diubah kembali kedalam bentuk biner kemudian digabungkan kembali menjadi biner berukuran 64 bit.

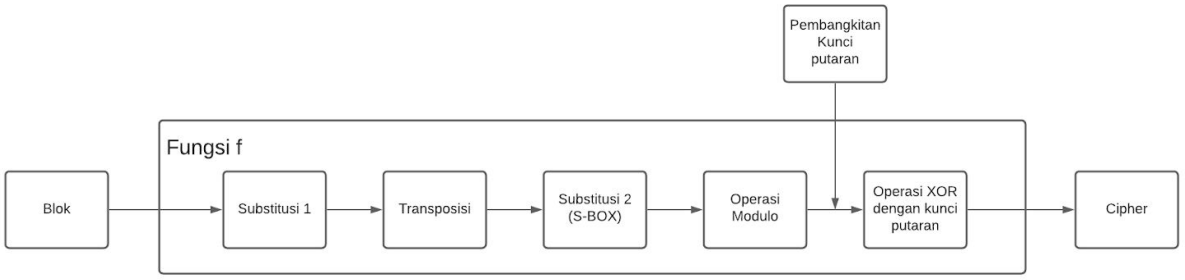
$$I_i = I_i * I \; mod \; 255 \tag{4}$$

I_i = potongan integer 8 bit (i = {1, 2, 3, … n/8)
I = integer kunci

3.2.5. Operasi XOR dengan kunci putaran

Operasi ini hanya operasi XOR biasa antara biner hasil operasi sebelumnya yang berukuran 64 bit dengan kunci putaran berukuran 64 bit.

Berikut merupakan alur proses dalam satu putaran pada jaringan *Feistel* pada algoritma *block cipher* LIAN.



Gambar 5. Proses-proses yang dilakukan pada satu putaran jaringan *feistel.*

4. Eksperimen dan Analisis Hasil

4.1. Eksperimen

Kode sumber program dari algoritma *block cipher* LIAN dapat diakses pada: https://github.com/josepandre99/uts-kripto. Eksperimen pada *block cipher* baru ini dilakukan dengan menggunakan mode operasi *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), dan *Counter Mode.* Pada eksperimen akan dilakukan pengukuran waktu eksekusi pada saat enkripsi maupun dekripsi pada masing-masing mode operasi. Selain pengukuran waktu eksekusi, akan diukur juga perbedaan ukuran file keluaran dari hasil enkripsi pada masing-masing mode operasi. Eksperimen akan dilakukan pada beberapa file masukan yang berbeda yaitu pada file teks yang berukuran 190 bytes, dan pada file gambar (png) berukuran 43,932 bytes (42,9 KB).

Dalam pengujian, sudah dipastikan hasil enkripsi dan dekripsi sudah benar. Hasil pengukuran waktu eksekusi baik pada proses enkripsi maupun dekripsi, dan hasil pengukuran file hasil enkripsi akan digambarkan pada gambar berikut.

Tabel 1. Waktu Eksekusi Enkripsi/Dekripsi dengan beberapa mode

Waktu Eksekusi (dalam second)			
Mode	Operasi	File Teks berukuran 190 B	File Gambar (PNG) berukuran 42.9 KB
ECB	Enkripsi	0.0312521457672119	7.390273332595825
	Dekripsi	0.05855917930603027	10.42755651473999
CBC	Enkripsi	0.039038896560668945	8.216981887817383
	Dekripsi	0.03513455390930176	8.620718717575073
Counter	Enkripsi	0.041968584060668945	8.479292154312134
	Dekripsi	0.03611135482788086	7.600072860717773

Selain menguji waktu eksekusi saat melakukan enkripsi maupun dekripsi, juga dilakukan pengujian hasil file enkripsi. Akan dibandingkan ukuran awal file sebelum di enkripsi dengan ukuran file hasil enkripsi.

Tabel 2. Ukuran File Setelah Enkripsi

Mode	Ukuran File Teks (txt)		Ukuran File Gambar (PNG)	
	Sebelum Enkripsi	Setelah Enkripsi	Sebelum Enkripsi	Setelah Enkripsi
ECB	190 bytes	283 bytes	43.932 bytes	43,936 bytes
CBC	190 bytes	289 bytes	43.932 bytes	43,936 bytes
Counter	190 bytes	292 bytes	43.932 bytes	43,936 bytes

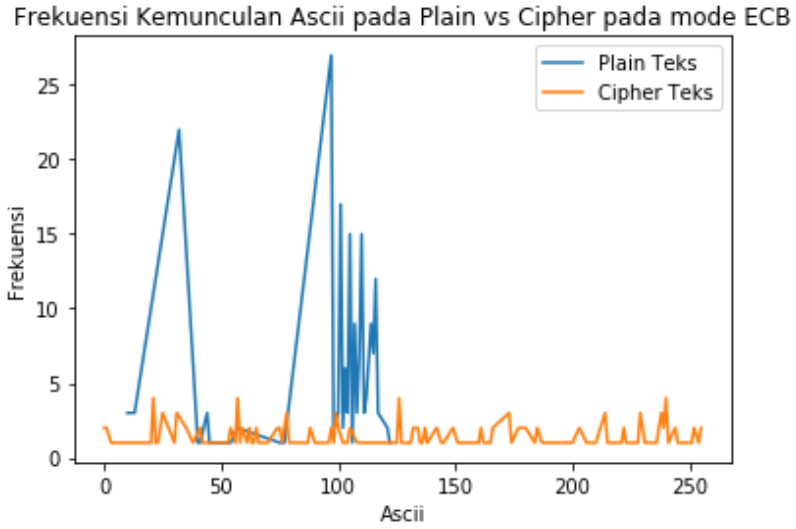
Berdasarkan tabel perbandingan ukuran file sebelum maupun sesudah enkripsi, dapat ditarik kesimpulan, bahwa ukuran file hasil enkripsi lebih besar dibandingkan ukuran file original sebelum melakukan enkripsi. Hal tersebut terjadi karena adanya penambahan border pada file yang dienkripsi agar panjang bit sesuai dengan yang digunakan oleh algoritma LIAN sewaktu melakukan proses enkripsi.

4.2. Analisis Hasil

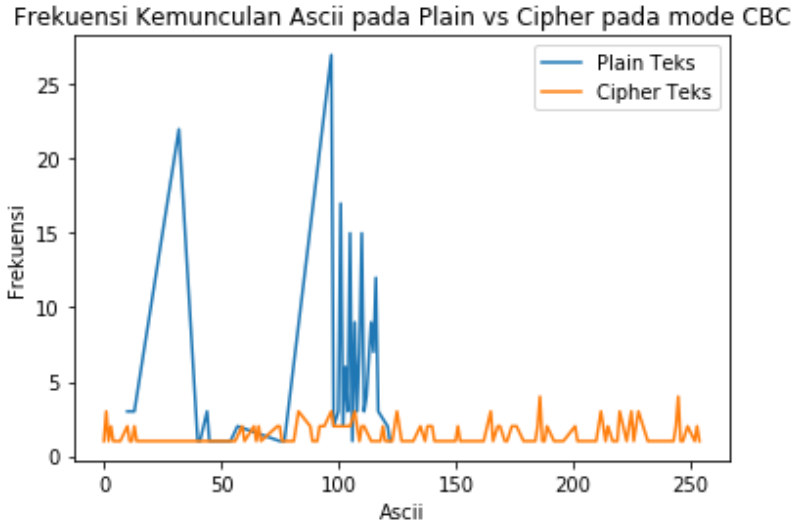
Analisis yang dilakukan adalah analisis penerapan *confusion* dan *diffusion* pada algoritma LIAN.

4.2.1. Analisis Confusion

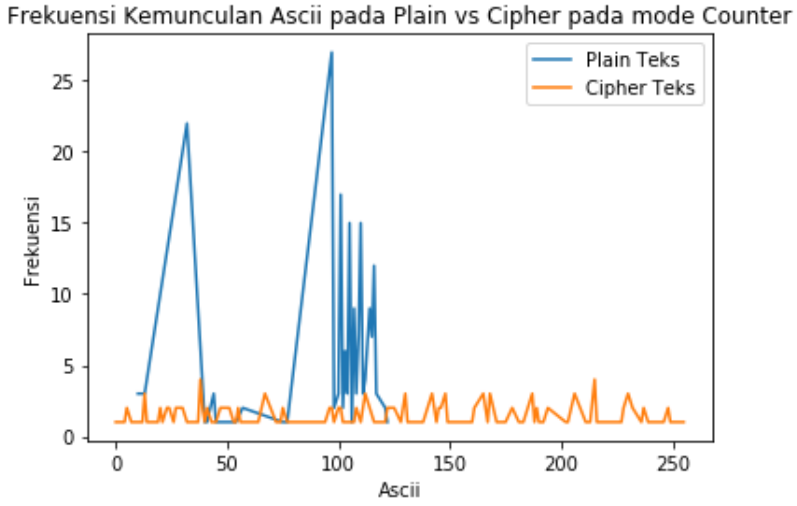
Analisis *confusion* dilakukan dengan menganalisa frekuensi kemunculan setiap karakter (Ascii) sebelum dan sesudah melakukan enkripsi. Analisis hanya dilakukan pada file teks (txt) berukuran 190 bytes (terdiri dari 190 karakter ascii). Frekuensi kemunculan karakter (Ascii) sebelum dan sesudah enkripsi adalah sebagai berikut.



Gambar 6. Frekuensi kemunculan karakter (Ascii) sebelum dan sesudah melakukan enkripsi dengan mode ECB.



Gambar 7. Frekuensi kemunculan karakter (Ascii) sebelum dan sesudah melakukan enkripsi dengan mode CBC.



Gambar 8. Frekuensi kemunculan karakter (Ascii) sebelum dan sesudah melakukan enkripsi dengan mode *Counter*.

Berdasarkan grafik pada gambar diatas, sebelum melakukan enkripsi, frekuensi kemunculan karakter ascii cukup tinggi di beberapa karakter saja. Sedangkan frekuensi kemunculan karakter-karakter setelah dilakukan enkripsi menjadi lebih merata (antara 0-5 karakter ascii saja). Dengan hasil tersebut, algoritma LIAN berhasil menerapkan prinsip *confusion* berjalan dengan baik.

4.2.2. Analisis Diffusion

Pada analisis *diffusion,* dilakukan perbandingan hasil enkripsi apabila dilakukan perubahan pada satu huruf, yaitu mengubah huruf pertama dari huruf ‘K’ menjadi ‘k’. Analisis dilakukan pada file teks (txt) berukuran 190 bytes.

Table 5. Tabel Analisis *Diffusion* dengan mode ECB, CBC, dan *Counter*

File Masukan :
Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Menez, 1996).

Hasil Enkripsi :

Mode ECB :

ôÎRiv¸ð ~îlCç Ùd 6=J)Wnü9Ëï,3A#rJÿd:ËôaÖ¥À9
56F?9øÜú}Ä>`*ð¦±~pKàð³O~Nhåcì4áA_×Zb#8´¦þÞc£ÃX@.
^üN>¹êðKXÖä)t´®³ÿ~Òå¡19B&cè;±¢Ýiíßå Èî¹ÖØaïñ»fõ¡ûÞL;N

Mode CBC :

õS¡Î×B³«ÔL®o^S»Û¨üK×ºµÌ*òbùÜi$wý8°BÖ6XSs\XJ@Y/öô¦Ê¬h¹½bº©K¥}JÔõ)¨áÂ^ômQº
¥âx#äõ¹÷Ääk{½;õlÜl; C\äºßk
°y©èýhá"i}rk
A@tùÔ¶Ënaoñó
®wÉ[n<MÉá+aë¥þÒ¸
¼Àa ÜÐ
}Ù(¢v7

Mode Counter :

¥¦+
¯K¨P7§Î»ùï\Â¥-i8ýÓ¶½Z¨ØÎ
øäèz² e&n¡¼þkb0 ú¥3/&&MeT× I,öòC«ìß)p}aÿl@æ'C¡Õµ¾Jñ&äÊfH^×piøKæ(ãèæ7y`6¦;}ÂR»·`aûx
¨ôz¿»uËÛ)"tdl0À5²Qd'
ÎCí½3×p×%/

File Masukan setelah huruf pertama diubah:
kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Menez, 1996).

Hasil Enkripsi setelah huruf pertama diubah:

Mode ECB :

«la>¤Í9Y
@$ç Ùd 6=J)Wnü9Ëï,3A#rJÿd:ËôaÖ¥À9
56F?9øÜú}Ä>`*ð¦±~pKàð³O~Nhåcì4áA_×Zb#8´¦þÞc£ÃX@.
^üN>¹êðKXÖä)t´®³ÿ~Òå¡19B&cè;±¢Ýiíßå Èî¹ÖØaïñ»fõ¡ûÞL;N

Mode CBC :

ç]ÑîÅ
|EN_ÓH`:1[¾³±¦ú«ÞÝn!û2M6å ëowÐuêWÌ ôÝ©uÏTÝì#´°·Li]o-Oø4~«êþ0®¥Ðäô°
«ñ¥ªÔ>M?¨ÂVUWPÛjÙSO)ù!B^mV>NÃÊÂà(NVº+?v¿_WqÌ µOí3³p
ËÁwÓJýAôö|} [½6pê[¤

Mode Counter :

ª¥¦+
¯K¨P7§Î»ùï\Â¥-í8ýÓ¶½Z¨ØÎ
øäèz² e&n¡¼þkb0 ú¥3/&&MeT× I,öòC«ìß)p}aÿl@æ'C¡Õµ¾Jñ&äÊfH^×piøKæ(ãèæ7y`6¦;}ÂR»·`aûx
¨ôz¿»uËÛ)"tdl0À5²Qd'
ÎCí½3×p×%/

Setelah dilakukan pengubahan sebesar satu byte yaitu mengubah huruf pertama dari ‘K’ menjadi ‘k’, terdapat perubahan yang merubah keseluruhan huruf setelahnya pada enkripsi dengan menggunakan mode ECB dan CBC, sedangkan pada mode *Counter* perubahan hanya terjadi pada huruf pertama yaitu huruf yang diganti karena memang pada algoritma mode *Counter* blok yang masuk ke algoritma LIAN adalah *counter*-nya bukan blok plainteks. Berdasarkan hal tersebut, menunjukkan bahwa algoritma LIAN telah menerapkan prinsip *diffusion* dengan baik.

4.2.3. Analisis Keamanan

Pada algoritma *block cipher* LIAN, panjang kunci minimal adalah 128 bit, maka akan terdapat sebanyak ada sebanyak $2^{128} = 3,4\ x\ 10^{38}$ kemungkinan kunci. Jika komputer tercepat dapat mencoba satu juta kunci setiap detik, maka akan dibutuhkan waktu minimal $5,4\ x\ 10^{24}$ tahun untuk mencoba seluruh kunci. Jika komputer tercepat dapat mencoba satu juga kunci setiap milidetik, maka dibutuhkan waktu minimal $5,4\ x\ 10^{18}$ tahun untuk mencoba seluruh kunci.

Dengan panjang kunci minimal saja, algoritma LIAN sudah sangat aman karena membutuhkan waktu yang sangat lama untuk mencoba keseluruhan kunci yang mungkin, apalagi pada algoritma ini panjang kunci dapat lebih dari 128 bit. Berdasarkan hal tersebut, maka algoritma *block cipher* LIAN dapat dikatakan aman.

5. Kesimpulan dan Saran Pengembangan

5.1. Kesimpulan

- Algoritma LIAN *block cipher* telah berhasil melakukan enkripsi dan juga dekripsi pesan dengan baik dengan menerapkan konsep jaringan *Feistel*.
- Algoritma LIAN juga berhasil menerapkan prinsip *confusion* dan *diffusion* shannon.
- Dari sudut keamanan, algoritma LIAN dapat menjaga kerahasiaan kunci yang digunakan dengan menambah panjang kunci.
- Algoritma LIAN dapat menambah keacakan pembangkitan kunci putaran menggunakan *pseudo-random number*.

5.2. Saran Pengembangan

Implementasi saat ini masih diterapkan pada mode ECB, CBC, dan *Counter* saja, untuk kedepannya mungkin dapat diterapkan ke mode lainnya seperti CFB dan OFB.

6. References

[1] Block Cipher. https://cryptobounce.wordpress.com/2008/06/19/block-cipher/. Diakses pada tanggal 20 Oktober 2020

[2] Munir, Rinaldi. Pengantar Kriptografi (2020). Slide Presentasi Kuliah IF4020. Diakses pada tanggal 22 Oktober 2020

[3] Munir, Rinaldi. Kriptografi Modern (Bagian 1). Slide Presentasi Kuliah IF4020, p. 5-13. Diakses pada tanggal 10 Oktober 2020.

[4] Munir, Rinaldi. Kriptografi Modern (Bagian 3). Slide Presentasi Kuliah IF4020. Diakses pada tanggal 10 Oktober 2020.

[5] Munir, Rinaldi. Review Beberapa Block Cipher dan Stream Cipher (Bagian 1). Slide Presentasi Kuliah IF4020. Diakses pada tanggal 10 Oktober 2020.

[6] Munir, Rinaldi. Review Beberapa Block Cipher dan Stream Cipher (Bagian). Slide Presentasi Kuliah IF4020. Diakses pada tanggal 10 Oktober 2020.

[7] Munir, Rinaldi. Review Beberapa Block Cipher dan Stream Cipher (Bagian 4). Slide Presentasi Kuliah IF4020. Diakses pada tanggal 10 Oktober 2020.

[8] Prof. Dr. Hilal Hadi Salih, Dr. Ahmed Tariq Sadiq M.Sc., Alaa K.Frhan. *Proposal of New Block Cipher Algorithm*.

[9] Transpos Matriks, https://www.yuksinau.id/elemen-matriks/ . Diakses pada tanggal 21 Oktober 2020.

7. *Acknowledgments*

Puji syukur ke hadirat Tuhan Yang Maha Esa yang telah memberikan kesempatan kepada penulis untuk menyelesaikan jurnal ini dengan tepat waktu. Penulis berterima kasih kepada dosen pengampu mata kuliah IF4020 Kriptografi, bapak Dr. Ir. Rinaldi Munir, MT., yang telah memberikan wawasan dan pengetahuan mengenai kriptografi sehingga LIAN *block cipher* dapat dibuat. Penulis juga berterima kasih kepada semua pihak yang telah turut membantu dalam menulis jurnal ini. Dengan jurnal ini, penulis berharap dapat membantu pengembangan wawasan pembaca.