Pandora Block Cipher

Farras Mohammad Hibban Faddila (13518017)[1],**Naufal Dean Anugrah (13518123)**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132

E-mail: 13518017@std.stei.itb.ac.id, 13518123@std.stei.itb.ac.id

Abstract. Algoritma Pandora adalah suatu algoritma *block cipher* yang memiliki ukuran blok 64 bit. *Master key* yang digunakan pada algoritma ini berukuran 128 bit. Algoritma Pandora menggunakan struktur jaringan feistel dengan jumlah putaran sebanyak 12 putaran. Komponen utama algoritma Pandora adalah *key generation* dan *round function* yang menerapkan berbagai metode transformasi yang kompleks. Algoritma ini juga menerapkan prinsip *confusion* dan *diffusion* sehingga memiliki tingkat keamanan yang tinggi. Oleh karena itu, algoritma ini cocok untuk dipakai dalam proses enkripsi serta dekripsi pesan yang akan dikirimkan melalui jalur yang tidak aman.
Keywords: *kriptografi*, *cipher, block cipher, mode enkripsi*, *jaringan feistel, algoritma Pandora*

1. Pendahuluan

Sekarang merupakan era informasi. Informasi yang tersedia sangat banyak jumlahnya, baik yang tersedia pada *local storage* maupun *cloud storage* dan metode pengirimannya juga beragam. Ketika sedang mengirimkan informasi, diperlukan sebuah jaminan keamanan agar informasi yang dikirim tidak dapat disalahgunakan oleh pihak yang tidak berkepentingan. Oleh karena itu, dibutuhkan sebuah keilmuan yang mengembangkan serta mempelajari sistem pengamanan dalam pengiriman informasi, yaitu kriptografi.

Secara istilah, kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan (Schneier, 1996). Sedangkan secara bahasa, kriptografi berasal dari dua kata dalam bahasa Yunani, yakni *cryptos* yang berarti rahasia, serta *graphein* yang berarti tulisan, sehingga secara literal kriptografi dapat diartikan sebagai tulisan rahasia. Teknik umum pada kriptografi adalah melakukan enkripsi (penyandian) pada pesan yang akan dikirim, dan ketika pesan sampai kepada penerima, dilakukan dekripsi, yaitu pengembalian pesan ke dalam bentuk semula. Untuk melakukan kedua proses ini, dibutuhkan kunci rahasia yang disepakati oleh pengirim dan penerima terlebih dahulu, sehingga pihak ketiga yang menyadap pesan saat pesan sedang berada pada saluran pengiriman tidak dapat mengerti pesan tersebut, kecuali memiliki kunci yang sama.

Kriptografi sudah ada sejak zaman Yunani dan romawi kuno, dan melahirkan cipher terkenal yaitu Caesar Cipher. Seiring waktu, kriptografi berkembang dan berperan penting dalam sejarah dunia, contohnya pada perang dunia ke-2. Seiring berkembangnya teknik-teknik pada kriptografi, berkembang juga ilmu kriptanalisis, yakni ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui kunci yang digunakan.

Di era modern, kriptografi sangat banyak digunakan pada pengiriman informasi digital. Informasi digital direpresentasikan dalam barisan bit/byte, sehingga algoritma enkripsi modern juga beroperasi pada bit/byte. Algoritma enkripsi modern terbagi ke dalam dua jenis, yakni Stream Cipher dan Block Cipher. Beberapa algoritma enkripsi modern adalah RC4, A5 (Stream Cipher), serta DES, AES (Block Cipher).

DES (Data Encryption Standard) merupakan algoritma *block cipher* yang dikembangkan oleh IBM pada tahun 1972. Ukuran tiap blok adalah 64 bit, dan panjang dari keynya adalah 64 bit juga, namun hanya 56 buah bit yang digunakan. Algoritma ini memanfaatkan jaringan feistel dalam algoritmanya. Sedangkan, AES (Advanced Encryption Standard) merupakan algoritma yang dijadikan sebagai standar kriptografi, menggantikan DES, dan memiliki nama asli Rijndael. Pada algoritma ini, kunci dapat memiliki panjang 128 bit, 192 bit, ataupun 256 bit, sehingga lebih fleksibel. Perbedaan AES dengan DES adalah, AES tidak menggunakan jaringan feistel. AES berbasis pada jaringan substitusi-permutasi. Pada AES, plainteks diubah ke dalam bentuk matriks, yang kemudian dioperasikan dengan berbagai fungsi seperti substitusi, *mix-column* (perkalian matriks pada sebuah *field*), dan lain-lain.

Algoritma Pandora yang dirancang merupakan algoritma *block cipher* yang memanfaatkan jaringan feistel. *Round key* yang akan digunakan oleh tiap putaran pada jaringan feistel dibangkitkan dengan cara yang serupa dengan *key scheduling* pada AES, namun dengan sebuah skema jaringan baru. Selain itu, algoritma Pandora ini juga mengimplementasi *round function* yang memanfaatkan beberapa operasi seperti penyisipan (*interleave*), perkalian matriks, dan *shift left*/*rotation*. Penggabungan operasi perkalian matriks dengan operasi penyisipan bertujuan meningkatkan kompleksitas dari algoritma yang dibuat.

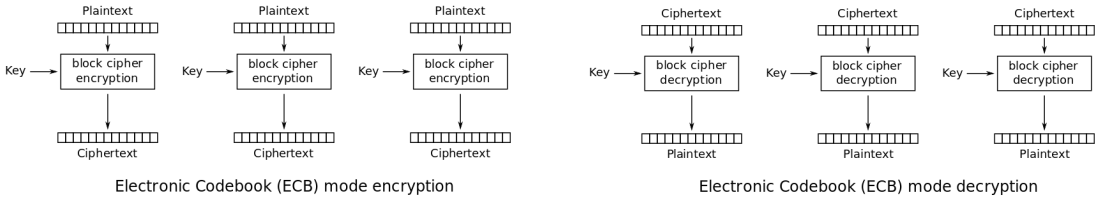
2. Dasar Teori

2.1. Block Cipher

Block Cipher merupakan algoritma enkripsi yang beroperasi pada pesan yang telah dibagi kedalam bagian-bagian yang disebut dengan blok. Pada sebuah plainteks, setiap blok memiliki panjang yang sama. Setiap *block* P_i dienkripsi dengan algoritma E dan kunci k, dan menghasilkan sebuah blok cipherteks $C_i = E_k(P_i)$. Blok-blok ini dapat dienkripsi secara independen, ataupun saling berkaitan, sesuai mode enkripsinya sebagai berikut ini.

2.1.1 Mode ECB

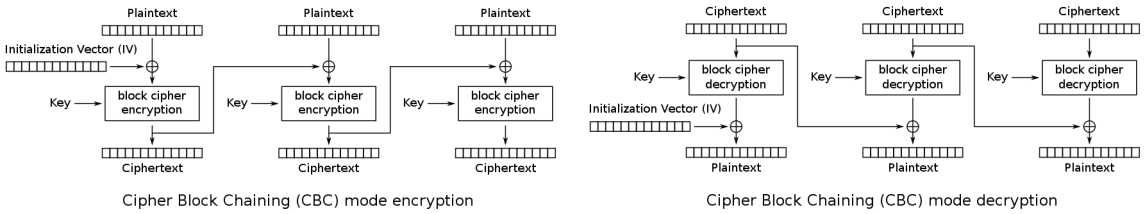
Mode ECB (*Electronic Code Book*) adalah mode *block cipher*, yang enkripsi tiap blok pada plainteks saling independen dan tidak berhubungan. Secara teoritis dimungkinkan untuk membuat sebuah *lookup table* yang memuat daftar seluruh blok plainteks beserta hasil enkripsinya, dan untuk melakukan enkripsi serta dekripsi, cukup dilihat kata yang berkoresponden dengan tiap blok pada *lookup table* tersebut.



Gambar 1 Skema enkripsi (kiri) dan dekripsi (kanan) pada *block cipher* mode ECB

2.1.2 Mode CBC

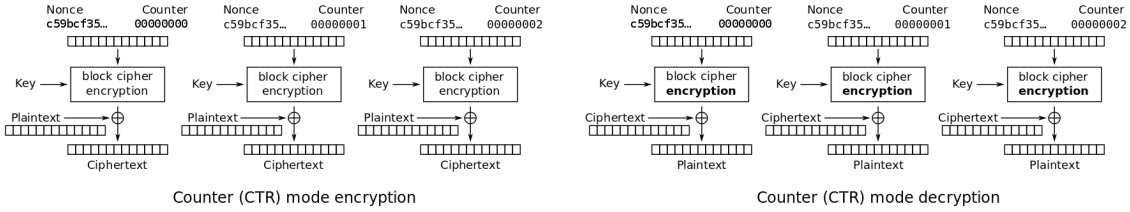
Mode CBC (*Cipher Block Chaining*) merupakan mode pada block cipher yang bertujuan untuk menghasilkan keterkaitan antara blok-blok plainteks pada cipherteks. Secara umum, $C_i = E_k(P_i \oplus C_{i-1})$, yakni hasil enkripsi pada blok sebelumnya menjadi *feedback* pada proses enkripsi blok berikutnya. Akibatnya, CBC memiliki sifat, yakni perubahan pada sebuah blok pada plainteks akan mengakibatkan perubahan pada seluruh cipherteks mulai dari blok tersebut dan blok-blok setelahnya.



Gambar 2 Skema enkripsi (kiri) dan dekripsi (kanan) pada *block cipher* mode CBC

2.1.3 Mode CTR

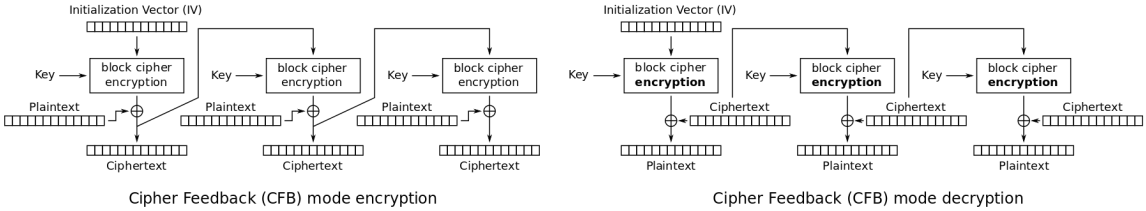
Mode CTR (Counter) adalah mode yang tidak melakukan *chaining* seperti pada CBC. Terdapat sebuah variabel bernama Counter, yang digunakan pada proses enkripsi blok, dan awalnya bernilai 0. Setiap akan melakukan enkripsi pada blok berikutnya, nilai counter di-*increment*.



Gambar 3 Skema enkripsi (kiri) dan dekripsi (kanan) pada *block cipher* mode CTR

2.1.4 Mode CFB

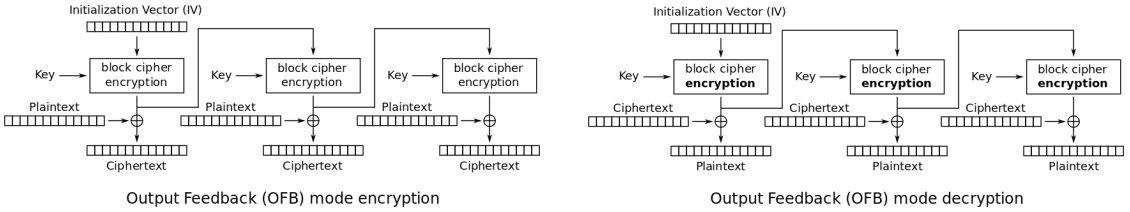
Mode CFB (*Cipher Feedback*) serupa dengan CBC (terdapat *chaining* antar blok), namun terdapat perbedaan utama, yaitu pada CFB dapat dilakukan enkripsi data pada ukuran yang lebih kecil daripada ukuran blok, sehingga mengatasi kasus ketika data yang dikirimkan tidak mencapai satu blok.



Gambar 4 Skema enkripsi (kiri) dan dekripsi (kanan) pada *block cipher* mode CFB

2.1.5 Mode OFB

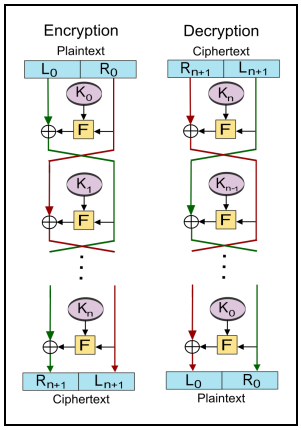
Mode OFB (*Output Feedback*) adalah mode enkripsi yang serupa dengan mode CFB. Perbedaannya terletak pada input blok selanjutnya. Pada mode CFB inputnya adalah cipherteks dari blok sebelumnya, sedangkan pada OFB inputnya adalah keluaran enkripsi dari blok sebelumnya.



Gambar 5 Skema enkripsi (kiri) dan dekripsi (kanan) pada *block cipher* mode OFB

2.2. Jaringan Feistel

Jaringan Feistel merupakan struktur yang banyak digunakan pada perancangan *block cipher*. Beberapa *block cipher* yang menggunakan jaringan feistel adalah DES, LOKI, GOST, Blowfish, dan Twofish. Pada jaringan feistel, blok dibagi ke dalam dua bagian L dan R, dan terdapat beberapa putaran (*round*). Tiap iterasi memanfaatkan sebuah kunci K (*round key*), yang bisa berbeda-beda untuk tiap *round,* dan sebuah fungsi f (*round function*). Struktur jaringan feistel bersifat *reversible* dan tidak bergantung pada fungsi f, sehingga fungsi f tidak perlu bersifat *reversible* dan dapat dibuat serumit mungkin.



Gambar 6 *Jaringan Feistel*

2.3. Prinsip Confusion dan Diffusion

Prinsip confusion dan diffusion dicetuskan oleh Claude Shannon pada tahun 1949 dan sampai sekarang menjadi panduan dalam proses pembuatan algoritma kriptografi. Prinsip confusion membuat proses analisis pola statistik menjadi sulit dilakukan karena hubungan yang tersembunyi antara plainteks, cipherteks, dan kunci. Prinsip diffusion menyebarkan perubahan kecil pada plainteks atau kunci pada sebanyak mungkin cipherteks.

3. Rancangan Algoritma *Block Cipher*

3.1. Overview

Algoritma Pandora merupakan algoritma enkripsi *block cipher* dengan struktur jaringan feistel. Plainteks terbagi ke dalam blok-blok yang berukuran 64 bit. *Master key* berukuran 128 bit. Jumlah *round* pada algoritma Pandora adalah sebanyak 12 *round*.

3.2. Konstanta

3.2.1 Rijndael S-Box (AES S-Box)

Rijndael S-Box merupakan *S-Box* yang digunakan pada algoritma AES. *S-Box* ini digunakan dalam proses substitusi pada pembangkitan *round key* dari algoritma Pandora ini, tepatnya pada fungsi *SubWord*. Box ini mengembalikan hasil substitusi dari sebuah pesan berukuran 8 bit, dengan cara melihat isi dari kotak yang terletak pada baris dan kolom yang posisinya ditunjukkan oleh 4 bit pertama dan 4 bit terakhir dari pesan tersebut. Sebagai contoh, $S_{AES}(01011010) = S_{AES}(0x5a) = 0xbe$, karena kotak pada baris 50 dan kolom $0a$ memuat bilangan be.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 7 *Rijndael S-Box*

3.2.2 S-Box

Selain *Rijndael S-Box*, algoritma Pandora juga memiliki 2 buah *S-Box* lain yang digunakan pada proses

enkripsi pada tiap *round* (putaran). Lebih tepatnya, *S-Box* ini akan digunakan pada bagian *S-Box Compression Network* pada *Round Function* yang akan dijelaskan kemudian.

S0	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	67	210	181	215	80	81	231	96	196	153	121	87	167	124	179	37
10	199	91	138	233	24	122	173	63	147	50	200	36	155	162	176	90
20	4	40	174	220	170	175	188	208	2	156	59	240	221	39	58	79
30	57	73	76	171	243	184	211	148	217	146	195	190	84	126	94	0
40	5	108	83	10	28	56	218	26	109	224	15	17	141	209	232	163
50	117	62	123	206	32	46	151	247	189	30	226	178	49	61	186	242
60	172	3	185	35	51	107	157	42	14	154	197	214	112	191	74	253
70	70	66	106	71	254	169	99	230	202	255	158	43	133	204	183	21
80	45	92	118	164	6	241	159	1	207	102	60	27	38	44	182	134
90	93	250	125	236	166	145	25	228	52	180	95	72	129	127	68	222
a0	136	223	187	47	149	237	235	168	139	161	101	203	111	82	213	131
b0	177	85	53	31	219	135	75	64	88	142	33	160	229	103	105	165
c0	16	140	119	251	86	18	41	137	22	23	143	239	205	115	77	227
d0	144	198	120	12	13	248	8	7	249	98	212	89	110	114	11	55
e0	244	238	100	65	234	116	194	245	19	132	97	225	48	130	216	34
f0	9	192	54	150	193	69	104	201	78	128	152	29	246	20	252	113

S1	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	252	85	154	31	30	225	151	220	229	190	174	22	81	177	157	47
10	42	88	114	254	27	24	198	224	99	103	131	178	158	80	52	7
20	135	111	29	128	75	20	73	146	168	175	185	192	115	169	79	230
30	67	159	237	41	150	211	241	201	78	98	125	145	209	148	84	195
40	44	17	112	130	1	71	35	123	25	249	68	69	39	137	38	218
50	28	77	156	186	21	121	140	34	246	233	50	94	26	207	2	113
60	139	11	60	6	19	76	239	129	171	3	247	63	240	90	243	179
70	97	18	143	193	173	110	101	203	182	235	92	33	242	91	160	100
80	66	108	187	149	250	161	212	210	109	215	23	153	70	16	221	53
90	232	15	253	95	89	36	136	144	119	37	51	126	138	167	48	217
a0	59	184	133	245	213	152	236	82	134	194	86	147	83	223	46	56
b0	104	124	58	165	116	166	61	244	234	197	183	40	96	117	43	55
c0	32	170	216	102	4	181	72	204	10	65	188	164	14	251	226	214
d0	238	231	54	57	0	141	122	176	248	172	180	219	155	196	118	87
e0	208	93	74	163	162	206	62	105	227	9	120	5	127	45	8	222
f0	189	107	49	64	200	132	13	106	202	12	205	255	191	228	142	199

Gambar 8 *S-Box 0* (kiri) dan *S-Box 1* (kanan)

3.2.3 P-Box

P-Box merupakan box yang melakukan permutasi pada urutan byte pesan.

4	6	5	3	13	11	12	14	7	8	2	10	0	15	9	1

Gambar 9 *P-Box*

3.2.4 Round

Banyaknya *round* (putaran) yang dilakukan pada enkripsi sebuah blok adalah 12 kali.

3.3. Key

Terdapat kunci awal (*external key*) K yang disepakati bersama oleh pengirim dan penerima. Selanjutnya, dengan menggunakan fungsi hash MD5, K akan diekspansi menjadi sebuah kunci K_{expand} yang berukuran 128 bit. Kunci ini nantinya akan digunakan sebagai masukan pada algoritma *key scheduling* untuk menghasilkan *round key* pada jaringan feistel sejumlah banyak *round*.

3.4. Round Key Generation

Round Function pada jaringan feistel membutuhkan sebuah kunci berukuran 64 bit. Ketika memasuki *round function*, kunci tersebut dibagi menjadi dua bagian, yaitu K_1 dan K_2 yang masing-masing berukuran 32 bit. K_1 digunakan pada algoritma *Interleave key*, dan K_2 digunakan pada *matrix expansion*.

Untuk melakukan pembangkitan kunci pada tiap putaran di jaringan feistel, dibuat sebuah jaringan untuk *key scheduling*. Jaringan ini menyerupai jaringan pada algoritma *key scheduling* pada AES. Berikut ini merupakan beberapa fungsi serta konstanta yang dibutuhkan pada algoritma pembangkitan kunci ini. Sebagai terminologi, kita sebut *subkey* sebagai bagian dari *key* yang terdiri atas 32 bit.

3.4.1 SubWord

Fungsi ini melakukan substitusi pada byte-byte dari subkey. Substitusi dilakukan dengan menggunakan AES Substitution Box (S_{AES}).

$$SubWord(S) = SubWord(S_0S_1S_2S_3) = S_{AES}(S_0)\, S_{AES}(S_1)\, S_{AES}(S_2)\, S_{AES}(S_3)$$

3.4.2 RotWord

Fungsi ini melakukan rotasi *cyclic shift* pada byte-byte dari subkey, yakni byte pertama dipindahkan menjadi byte terakhir. Fungsinya ditunjukkan sebagai berikut.

$$RotWord(S_0S_1S_2S_3) = S_1S_2S_3S_0$$

3.4.3 Round Constant

Round constant merupakan konstanta yang digunakan dalam penjadwalan kunci. Konstanta yang digunakan merupakan konstanta yang digunakan dalam penjadwalan kunci AES, yang dibangkitkan sebagai berikut, dan disimpan pada sebuah larik RC .

$$RC_1 = 1$$

$$RC_i = 2RC_{i-1} \text{ , jika } i > 1 \text{ dan } RC_{i-1} < 0x80$$

$$RC_i = 2RC_{i-1} \oplus 0x11B \text{ , jika } i > 1 \text{ dan } RC_{i-1} \geq 0x80$$

Berikutnya adalah deskripsi dari algoritma penjadwalan kunci, serta kunci yang dihasilkan untuk tiap putaran pada jaringan feistel. Awalnya, jaringan *key scheduling* ini menerima *external key* K yang berukuran 128 bit. Kemudian, *external key* tersebut dibagi ke dalam empat bagian (*subkey*) yang masing-masing berukuran 32 bit, yaitu k00, k01, k02, k03. Keempat *subkey* ini akan dioperasikan untuk menghasilkan empat *subkey* berikutnya, dan masing-masing kelompok *subkey* akan digunakan sebagai *round key*. Notasi serta hubungan mereka adalah sebagai berikut.

Notasi:

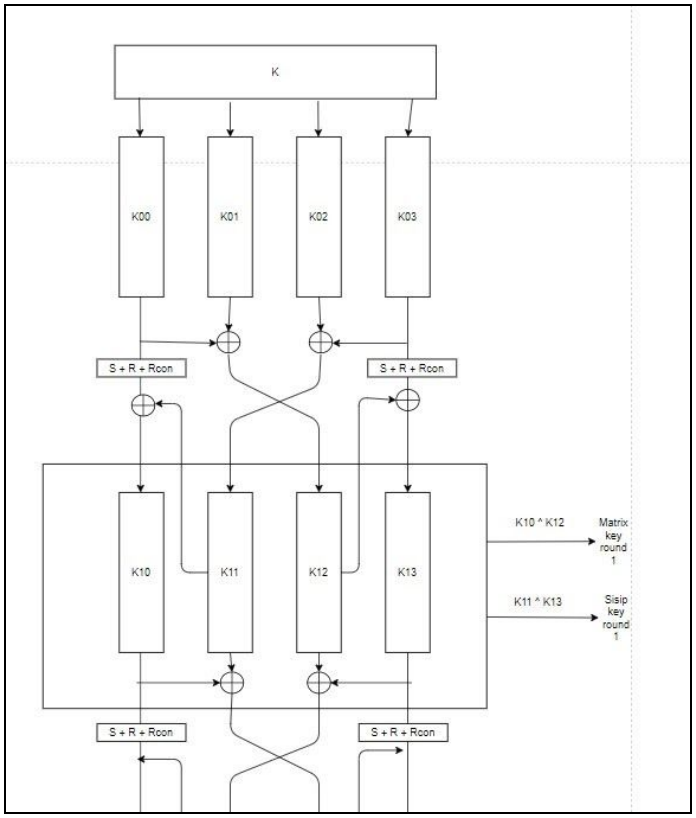
- k_i : Kunci yang dihasilkan untuk putaran ke-i (128 bit)
- $k_{i,0}, k_{i,1}, k_{i,2}, k_{i,3}$: Potongan dari k_i yang masing-masing berukuran 32 bit. Dengan kata lain, berlaku hubungan $k_i = k_{i,0}k_{i,1}k_{i,2}k_{i,3}$.

Algoritma pada tiap putaran untuk pembangkitan *round key*:

- $k_{i,2} := k_{i-1,0} \oplus k_{i-1,1}$

- $k_{i,1} := k_{i-1,2} \oplus k_{i-1,3}$
- $k_{i,0} := k_{i,1} \oplus SubWord(RotWord(k_{i-1,0})) \oplus RConst_{[\frac{i}{4}]}$
- $k_{i,3} := k_{i,2} \oplus SubWord(RotWord(k_{i-1,3})) \oplus RConst_{[\frac{i}{4}]}$

Berikut ini merupakan skema jaringan *key scheduling*:



Gambar 10 *Round key generation* algoritma Pandora

Pada tiap putaran ke-i dihasilkan sebuah kunci $k_i = k_{i,0}k_{i,1}k_{i,2}k_{i,3}$. Nilai dari $k_{i,0} \oplus k_{i,2}$ akan menjadi kunci pada proses *matrix expansion* di f-box (*round function*) (*round key part one*), serta nilai $k_{i,1} \oplus k_{i,3}$ akan menjadi kunci pada proses *interleave key* di f-box (*round key part two*).

3.5. Round Function

Round function adalah fungsi yang digunakan untuk mengenkripsi setengah blok teks pada setiap ronde algoritma Pandora. Bagian ini akan menjelaskan berbagai fungsi transformasi yang digunakan pada *round function* algoritma Pandora.

3.5.1. Interleave key

Fungsi ini melakukan proses penyisipan bit kunci pada input *round function*. Input (32 bit) dan *round key part 1* (32 bit) akan diproses dan menghasilkan keluaran dengan ukuran 64 bit. Aturan penyisipannya adalah setiap bit kunci akan dipasangkan dengan satu bit input pada posisi yang sama. Kemudian, bit kunci akan diletakkan di kanan bit input jika bit kunci tersebut bernilai 1, sebaliknya bit kunci akan diletakkan di kiri bit input.

Sebagai contoh, misal 8 bit pertama kunci dan input secara berturut-turut adalah 11001100 dan 00001111, maka outputnya adalah 01 01 00 00 11 11 01 01. Hal ini dilakukan terhadap setiap pasang bit kunci dan input yang masing-masing berukuran 32 bit sehingga didapatkan output berukuran 64 bit.

3.5.2. Expansion Matrix

Fungsi ini melakukan proses ekspansi input dengan memanfaatkan operasi perkalian matriks dalam modulo 256. Inputnya adalah hasil fungsi *interleave key* (64 bit) dan *round key part 2* (32 bit) yang diproses menghasilkan keluaran berukuran 128 bit. Input akan dikelompokkan menjadi 8 nilai byte yang disusun dalam matriks input berukuran 1x8. Kemudian, dari *round key part 2* akan dibangkitkan matriks ekspansi berisi byte berukuran 8x16.

Notasikan K_2 sebagai *round key part 2* yang berukuran 32 bit tersebut, dan $K_{2,0}K_{2,1}K_{2,2}K_{2,3}$ sebagai byte-byte dari K_2. Notasikan M sebagai matriks ekspansi, dan M_{ij} sebagai elemen matriks pada baris ke-i dan kolom ke-j.

Matriks ekspansi dibangkitkan sesuai aturan berikut:

$$M_{ij} = S_{AES}(K_{2,\left[\frac{i}{4}\right]} << i) \oplus S_{AES}(K_{2,(j \bmod 4)} << (i+1))$$

dengan S_{AES} merupakan Rijndael *S-Box.*

Kemudian, dilakukan *cross product* antara matriks input dengan matriks ekspansi sehingga didapatkan matriks berukuran 1x16. Elemen matriks tersebut merupakan keluaran dari fungsi ini.

3.5.3. P-Box

Fungsi ini akan melakukan proses permutasi menggunakan *P-Box* yang telah didefinisikan pada bagian 3.2.3.

3.5.4. Rot Compression Network

Pada bagian ini, hasil dari *P-Box* sejumlah 16 byte akan dikompresi memanfaatkan fungsi berikut (fungsi rot_x adalah akan melakukan proses *circular left shift* sejauh x bit):

```
function sub_f_box(byt1: int, byt2: int, x: int) returns byt_out: int
    -> rot_x((byt1 ^ byt2) + x, x)
```

Output dari *Rot Compression Network* adalah (out[x] dan inp[x] berturut-turut menyatakan byte output dan input pada posisi ke-x):

```
out[0] = sub_f_box(inp[1], inp[3], 0)
out[1] = sub_f_box(inp[0], inp[2], 1)
out[2] = sub_f_box(inp[5], inp[7], 2)
out[3] = sub_f_box(inp[4], inp[6], 3)
out[0] = sub_f_box(inp[9], inp[11], 4)
out[1] = sub_f_box(inp[8], inp[10], 5)
out[2] = sub_f_box(inp[13], inp[15], 6)
out[3] = sub_f_box(inp[12], inp[14], 7)
```

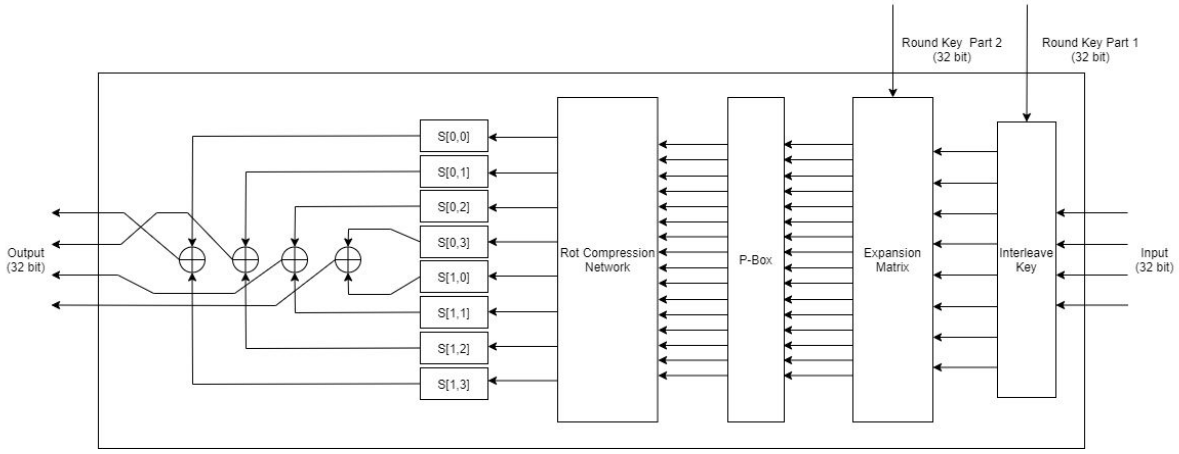
3.5.5. S-Box Compression Network

Pada bagian ini, hasil dari *rot compression network* sejumlah 8 byte akan disubstitusi dengan memanfaatkan kedua *S-Box* yang telah didefinisikan pada bagian 3.2.2. Proses substitusi akan dilakukan menggunakan fungsi berikut (pada Gambar 11 dinotasikan sebagai S[s_box_id, offset]):

```
function s_box(byt: int, s_box_id: int, offset: int) returns byt_out: int
    byt = (byt + offset) mod 256
    -> S_BOX_LIST[s_box_id][(byt >> 4) & 0xf][byt & 0xf]
```

Output dari *S-Box Compression Network* adalah (out[x] dan inp[x] berturut-turut menyatakan byte output dan input pada posisi ke-x):

```
out[0] = s_box(inp[0], 0, 0) ⊕ s_box(inp[7], 1, 0)
out[1] = s_box(inp[1], 0, 1) ⊕ s_box(inp[6], 1, 1)
out[2] = s_box(inp[2], 0, 2) ⊕ s_box(inp[5], 1, 2)
out[3] = s_box(inp[3], 0, 3) ⊕ s_box(inp[4], 1, 3)
```



Gambar 11 *Round function* algoritma Pandora

4. Pengujian dan Analisis Hasil

Program Pandora *block cipher* diimplementasi dengan menggunakan bahasa *python*.

4.1. Pengujian

Berikut ini merupakan plainteks yang akan dienkripsi. Plainteks diambil dari puisi Chairil Anwar yang berjudul "Aku". Kunci yang digunakan adalah **abcdefgh12345678**. Hasil yang ditampilkan dalam bentuk byte, dan unicode, serta dalam dua mode (ECB dan CBC).

<table>
<tr><th colspan="2">Plainteks</th></tr>
<tr><td colspan="2">Kalau sampai waktuku
Aku mau tak seorang kan merayu
Tidak juga kau
Tak perlu sedan itu
Aku ini binatang jalang
Dari kumpulannya terbuang
Biar peluru menembus kulitku
Aku tetap meradang menerjang
Luka dan bisa kubawa berlari
Berlari
Hingga hilang pedih peri
Dan aku akan lebih tidak peduli
Aku mau hidup seribu tahun lagi</td></tr>
<tr><td>Kunci</td><td>abcdefgh12345678</td></tr>
<tr><td colspan="2">Hasil Enkripsi Mode ECB</td></tr>
<tr><td>Byte</td><td>Teks</td></tr>
<tr><td>bc0e2c9fa07f7b546cc371b39d8185ce659d1b1c0f2cf452f85cb68afa2cb1b9f99c73680f4347b9f2fa642e56d96153755955cff633d2b6f5a05dc72f5b424e611cb94420756f34ef45323701f3eba6afb82817d25c52e26bfa4539e276ed7b0454fb232836a877ba936b6d6b85a2f50c7fae3469109f7e720136b3eb7be39ed38deba06777c41f800f4d9d4ea8ccf00b764fe97664315916e082404290ec623f9983176477ff4ef70d731a8850b73b91571fc0714ce5b15678d7ddf172227735a3abefd1cc9cb8c0811cbe259ab67e188fbfa31d25d1eade91901679c62efcb071d80a4a93eb39acb6a69ad913e1978c8b417783f0dbf23c4ed752dc957924c973ba38481a82372dead23f4ad03531d8611b5c39e05409a7004060176fcd143cab2457f80fab01f1f953cd738524304667075211920ea57261a66d0fc2b9f7d2dd022e8c7bfbbe4a7aa16320bc76dd</td><td>¼,Ÿ
{TlÃq³…Îe,ôRø\¶Šú,±¹ùœshCG¹òúd.VÙaSuYUÏö
3Ò¶õ]Ç/[BNa¹D
uo4ïE27óë¦¯‚(Ò\RâkúE9âví{Tû#(6¨wº“kmk…¢õ®
4iŸ~r6³ë{ãžÓë gwÄ€MN¨Ìð
vOévd1Yà,@Bıb?™ƒdwÿN÷
sˆP·;‘WÀqLå±Vx×Ýñr"w5£«ïÑÌœ¸À¾%š¶~¿£%Ñ
êÞ‘yÆ.ü°qØ
J“ë9–¶¦šÙá—Œ‹AwƒðÛò<N×RÜ•y$Ésº8H,7-êÒ?
JÐ51Øa\9àT §</td></tr>
<tr><td colspan="2">Hasil Enkripsi Mode CBC</td></tr>
<tr><td>Byte</td><td>Teks</td></tr>
</table>

b538454bfbe88e892f9f2928f58142dc50889f46bc94980f4a89381858667738a36dc5c699fb6e0258d6a9fd5859aed73ee713a016d1ba5a4d98229150c661339a5c0806148f988396c861dfa92c3059b1a204ae1e39420ab18fd73dc1339e94c65517d4656f4fdd0d8b87a18675f4df89a3f015eb6d1ea214142db47686897bfeaeacd0fb2a572ddd54ce86632c85fa43ca59eabe69c685201c8cb71ca643a60889b5d8bfe668d28bd0440f7bc76018329384e349219cdb13d999cfc4af85ec5ae82eb835ef941aedda110c270e3fec5ae491707223f15749d8faef206008ab522caa07facc7faeafd059580afcc4e2360d191a9c3c9d5876b4665a1da36ea28faaca2353ddb2c3b56c4e8d5a0f05c4a00e726b47278153e934a552967f4a2330c3d03f0c785144a052d9bdfc95c5a1f1b593946efc8a4dea1d0f39a83d7b0c98de1c00d7622aed5ea4964d9cae2a3f	µ8EKûèŽ‰/Ÿ)(õBÜPˆŸF¼”˜J‰8Xfw8£mÅÆ™ûnXÖ©ýXY®×>ç ÑºZM˜"‘PÆa3š\˜*f*–Èaß©,0Y±¢®9B±×=Á3ž”ÆUÔeoOÝ‹‡¡†uôß‰£ðëm¢-´v†‰{þ®¬Đû*W-ÝTÎ†c,…úCÊYê¾iÆ…Œ·¦C¦‰µØ¿æhÒ‹ĐD{Ç`2“„ãI!œÛÙ™ÏÄ¯…ìZè.,5ï”íÚ'?ìZä‘pr#ñWIØúï `«R,ªúÌ®¯ĐYXüÄâ6œ<Xv´fZ£n¢ªÊ#SÝ²ÃµlNZÄ rkG'Sé4¥R–J#0ÃĐ?xQDRÙ½ü•Å¡ñµ“”nüŠMê9¨={˜Þ

Berikut ini merupakan *screenshot* dari output program yang menampilkan hasil enkripsi (dalam byte) serta waktu yang dibutuhkan untuk mengeksekusi *block cipher* dengan plainteks di atas, dalam mode EBC.

```
CIPHERTEXT --------------------------------------------------
b'\xbc\x0e,\x9f\xa0\x7f{Tl\xc3q\xb3\x9d\x81\x85\xcee\x9d\x1b\x1c\x0f,\xf4R\xf8\\\xb6
[BNa\x1c\xb9D uo4\xefE27\x01\xf3\xeb\xa6\xaf\xb8(\x17\xd2\\R\xe2k\xfaE9\xe2v\xed{\x0
x8d\xeb\xa0gw\xc4\x1f\x80\x0fM\x9dN\xa8\xcc\xf0\x0bvO\xe9vd1Y\x16\xe0\x82@B\x90\xecb
ab\xef\xd1\xcc\x9c\xb8\xc0\x81\x1c\xbe%\x9a\xb6~\x18\x8f\xbf\xa3\x1d%\xd1\xea\xde\x9
f0\xdb\xf2<N\xd7R\xdc\x95y$\xc9s\xba8H\x1a\x827-\xea\xd2?J\xd051\xd8a\x1b\\9\xe0T\t\
ra\xa6m\x0f\xc2\xb9\xf7\xd2\xdd\x02.\x8c{\xfb\xbe\xff\xdf\x1d\x8b;\x92\xd8N'
--------------------------------------------------
time ellapsed for encryption: 0.4741859436035156 s
ok
```

Gambar 12 Output program

4.2 Analisis Brute Force

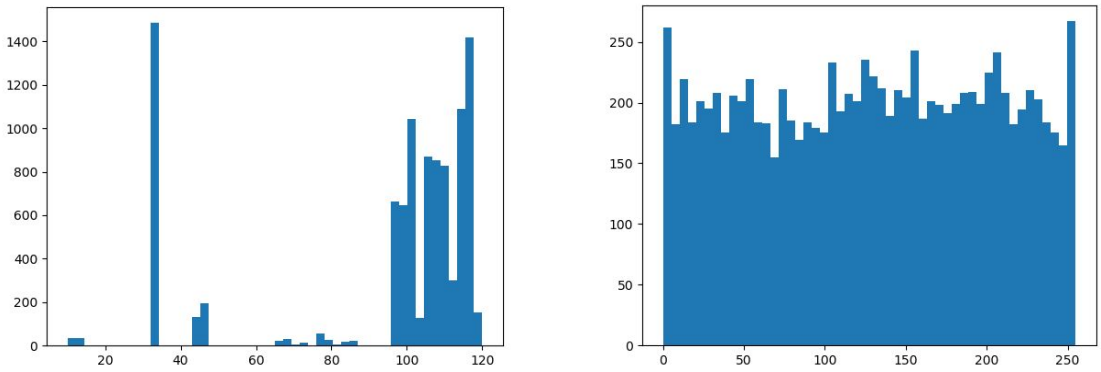
Algoritma Pandora memiliki kunci sepanjang 128 bit. Akibatnya untuk melakukan serangan *brute force* dibutuhkan pengecekan sebanyak:

$$2^{128} = 3,4028 \times 10^{38}$$

Kemudian, jika diasumsikan dalam 1 detik dapat dilakukan pengecekan terhadap 10^8 kemungkinan, maka dibutuhkan waktu $3,4028 \times 10^{30}$ detik atau sekitar $1,0790 \times 10^{23}$ tahun. Oleh karena itu, dapat disimpulkan bahwa algoritma Pandora aman dari ancaman serangan *brute force*.

4.3 Analisis Prinsip Confusion dan Diffusion

Analisis prinsip *confusion* dilakukan dengan melihat frekuensi kemunculan tiap byte pada cipherteks. Untuk keperluan analisis *confusion* akan digunakan teks lorem ipsum yang berukuran 10.066 byte. Berikut adalah histogram yang berisi distribusi kemunculan suatu nilai byte pada plainteks serta cipherteks yang dienkripsi dengan mode CBC menggunakan kunci “kriptografiasikk” dan iv “testiviv”.



Gambar 13 Histogram distribusi kemunculan nilai byte pada plainteks (kiri) dan cipherteks mode CBC (kanan)

Dapat dilihat bahwa pada histogram di atas, cipherteks memiliki distribusi yang merata. Oleh karena itu, dapat disimpulkan bahwa algoritma Pandora sudah memenuhi prinsip *confusion*.

Gambar 14 Citra Lenna

Analisis prinsip *diffusion* dilakukan dengan melakukan uji coba penggantian 1 bit pada plainteks serta 1 bit pada kunci. Data yang menjadi acuan perbandingan adalah, hasil enkripsi citra Lenna (Gambar 14) menggunakan kunci “kriptografiasikk” dan iv “testiviv”. Citra plain tersebut memiliki ukuran 11.524 byte

dan setelah dienkripsi menjadi 11.528 byte karena penambahan padding. Mode operasi yang digunakan dalam analisis ini adalah mode CBC. Berikut adalah tabel ringkasan hasil percobaan.

Posisi bit plainteks yang diubah	Jumlah byte dengan posisi dan nilai yang sama	Persentase perubahan cipherteks
4	46	99.601%
22	36	99.688%
24	51	99.558%
243	96	99.167%
807	147	98.725%

Kunci	Jumlah byte dengan posisi dan nilai yang sama	Persentase perubahan cipherteks
kriptografiasik**l**	42	99.635%
lriptografiasikk	35	99.696%
kriptografi**b**sikk	52	99.548%
kripto**h**rafiasikk	52	99.548%
kr**j**ptografiasikk	41	99.644%

Dapat dilihat bahwa pada tabel di atas, perubahan 1 bit pada plainteks maupun 1 bit kunci menghasilkan perbedaan yang besar pada cipherteksnya. Oleh karena itu, dapat disimpulkan bahwa algoritma Pandora sudah memenuhi prinsip *diffusion*.

5. Kesimpulan dan Saran Pengembangan

5.1. Kesimpulan

Algoritma Pandora sudah berhasil melakukan proses enkripsi dan dekripsi pesan dengan baik. Algoritma ini juga sudah menerapkan prinsip *confusion* dan *diffusion* dari Shannon. Selain itu, penggunaan kunci sepanjang 128 bit membuat serangan *brute force* menjadi sulit untuk dilakukan. Sehingga algoritma ini dapat digunakan untuk proses enkripsi dan dekripsi pesan yang akan dikirimkan melalui jalur yang tidak aman.

5.2. Saran

Performa algoritma Pandora dapat dianalisis dan ditingkatkan sehingga proses enkripsi dan dekripsi dapat dilakukan dalam waktu yang lebih singkat. Selain itu, dapat dilakukan analisis keamanan yang lebih detail terhadap komponen maupun keseluruhan algoritma Pandora.

6. Referensi

[1] NIST. 2001. *Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES)* https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[2] Munir, Rinaldi. 2020. *Slide Kuliah IF4020 Kriptografi: Kuliah Pengantar*.

[3] Munir, Rinaldi. 2020. *Slide Kuliah IF4020 Kriptografi: Kriptografi Modern (Bagian 3: Block Cipher).*

[4] Munir, Rinaldi. 2020. *Slide Kuliah IF 4020 Kriptografi: Review Beberapa Block Cipher dan Stream Cipher (Bagian 4: Advanced Encryption Standard (AES))*.

Acknowledgments

Pertama, penulis berterima kasih kepada Tuhan Yang Maha Esa, karena penulisan jurnal ini berjalan dengan lancar dan selesai tepat waktu. Penulis juga mengucapkan terima kasih kepada Dosen pengampu mata kuliah IF4020 Kriptografi, Bapak Dr. Rinaldi Munir, yang telah memberikan ilmu mengenai kriptografi serta memberikan kesempatan untuk merancang *block cipher* sendiri. Terakhir, penulis berterima kasih kepada seluruh pihak yang telah membantu selama pengerjaan jurnal ini, serta selama proses pembuatan *block cipher* ini.