Omnium *Block Cipher*

Tugas 2 IF4020 Kriptografi

Ahmad Alfani Handoyo - 13520023
Putri Nurhaliza - 13520066
Ubaidillah Ariq Prathama - 13520085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail:
13520023@std.stei.itb.ac.id
13520066@std.stei.itb.ac.id
13520085@std.stei.itb.ac.id

***Abstrak*—Makalah ini dibuat dengan tujuan membuat sebuah algoritma block cipher baru yang memiliki keamanan yang baik. Algoritma yang disulkan menggunakan prinsip diffusion dan confusion, menggunakan chaining CBC, jaringan substitusi dan permutasi, menerapkan cipher berulang, ukuran blok 128 bit, ukuran kunci 128 bit, dan 16 kali putaran. Dengan teknik-teknik tersebut algoritma yang dihasilkan dapat menjadi block cipher yang kuat.**

Kata Kunci—block cipher; diffusion; confusion; CBC; kunci; fungsi putaran; enkripsi; dekripsi

I. Pendahuluan

Seiring berkembangnya teknologi komputasi, banyak informasi sensitif yang harus dijaga kerahasiaannya. Untuk itu, kriptografi mengambil peranan penting untuk melindungi informasi-informasi tersebut. Salah satu kualitas kriptografi adalah ditentukan dari kekuatan enkripsi-dekripsi dari algoritma kriptografi terkait. Kekuatan yang dimaksud adalah seberapa sulit sebuah ciphertext dapat dipecahkan kembali menjadi plaintext.

Algoritma enkripsi konvensional sangat mudah dipecahkan dengan metode seperti serangan frekuensi kemunculan huruf. Hal ini menuntut perkembangan algoritma yang lebih kuat seperti penanganan dalam sekelompok bit. Cara ini juga dapat dipecahkan dengan melakukan serangan frekuensi terhadap kemunculan kelompok bit yang sama.

Oleh karena itu, diperlukan algoritma yang dapat membuat relasi antara plaintext dan ciphertext serumit mungkin. Salah satu cara yang sering digunakan untuk meningkatkan kekuatan enkripsi pada block cipher adalah metode confusion dan diffusion dari Shannon. Metode ini tidak dapat diserang dengan metode konvensional karena perubahan 1 bit pada plaintext, ciphertext, ataupun key akan menyebabkan perubahan drastis secara keseluruhan. Untuk menambah kekuatan enkripsi, digunakan jaringan Feistel. Jaringan Feistel menggunakan prinsip XOR dan membagi plaintext ke dalam 2 blok yang seimbang. Kedua blok tersebut saling mempengaruhi hasil yang satu dengan yang lain dalam setiap bit yang dioperasikan yang menyebabkan jaringan ini mempunyai tingkat kerumitan yang tinggi.

II. Dasar Teori

A. *Block Cipher*

Block cipher adalah salah satu jenis algoritma kriptografi yang digunakan untuk mengamankan pesan atau data dengan cara mengubah pesan tersebut menjadi bentuk yang tidak terbaca atau sulit dibaca oleh pihak yang tidak berhak. Block cipher membagi pesan menjadi blok-blok yang sama besar, kemudian setiap blok dienkripsi menggunakan kunci kriptografi yang dibangkitkan dari sebuah kunci eksternal.

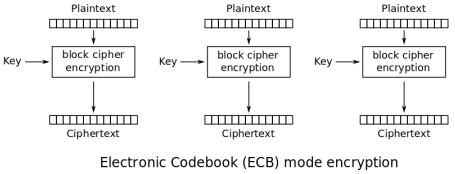
Dekripsi pada block cipher adalah kebalikan dari enkripsi, di mana pesan yang telah dienkripsi dengan block cipher dapat didekripsi kembali ke bentuk pesan asli dengan menggunakan kunci yang sama. Dekripsi ini memungkinkan hanya pihak yang memiliki kunci yang benar untuk membaca pesan yang telah dienkripsi.

Dalam penggunaan block cipher, terdapat beberapa faktor yang perlu diperhatikan agar pesan yang dienkripsi tetap aman, yaitu panjang blok, panjang kunci, dan ketahanan terhadap serangan. Panjang blok harus cukup besar agar tidak mudah diretas, sedangkan panjang kunci harus cukup panjang agar tidak dapat ditebak. Selain itu, block cipher harus tahan terhadap serangan seperti serangan brute force, differential cryptanalysis, dan linear cryptanalysis untuk memastikan keamanan pesan yang dienkripsi.

B. *Mode Operasi Block Cipher*

Ada beberapa jenis mode operasi block cipher yang digunakan untuk mengamankan pesan, yaitu:

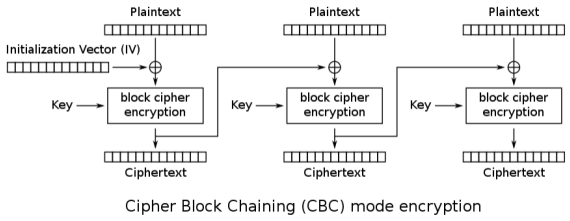
1. Electronic Codebook (ECB)



Gambar 1. Enkripsi ECB

Mode ECB adalah mode operasi block cipher yang paling sederhana. Pada mode ini, setiap blok pesan dienkripsi secara independen menggunakan kunci yang sama. Mode ECB tidak mampu menangani pesan yang sama atau bagian dari pesan yang sama dengan cara yang berbeda-beda, sehingga tidak dianjurkan untuk digunakan pada pengamanan data yang sensitif.

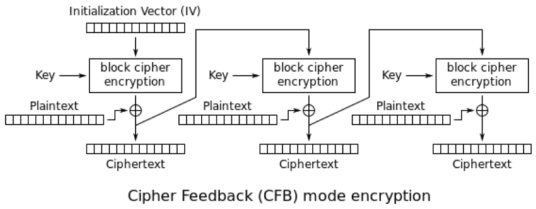
2. Cipher Block Chaining (CBC)



Gambar 2. Enkripsi CBC

Mode CBC adalah mode operasi block cipher yang memperkuat keamanan dengan menggunakan blok sebelumnya sebagai input pada enkripsi blok selanjutnya. Pada mode ini, blok pertama dienkripsi menggunakan IV (Initialization Vector) sebagai input pada enkripsi, sedangkan blok-blok selanjutnya dienkripsi dengan input XOR antara blok sebelumnya dan kunci. Mode CBC lebih aman daripada mode ECB, karena mampu menangani pesan yang sama dengan cara yang berbeda-beda.

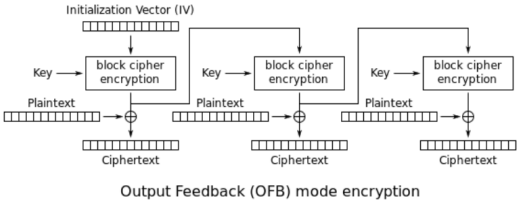
3. Cipher Feedback (CFB)



Gambar 3. Enkripsi CFB

Mode CFB adalah mode operasi block cipher yang menggunakan feedback loop dari keluaran enkripsi blok sebelumnya sebagai input pada enkripsi blok selanjutnya. Pada mode ini, keluaran enkripsi blok sebelumnya dienkripsi kembali dan XOR dengan plaintext pada blok selanjutnya. Mode CFB sangat berguna pada pengiriman pesan secara streaming, di mana setiap blok dapat dienkripsi dan dikirimkan secara terpisah.

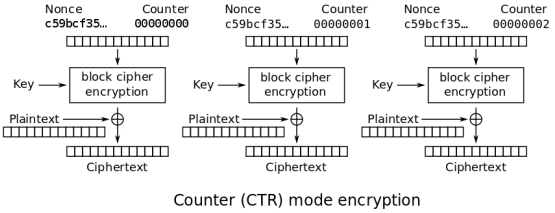
4. Output Feedback (OFB)



Gambar 4. Enkripsi OFB

Mode OFB adalah mode operasi block cipher yang menggunakan feedback loop dari keluaran enkripsi sebagai input pada enkripsi blok selanjutnya. Pada mode ini, keluaran enkripsi dienkripsi kembali dan digunakan sebagai kunci pada enkripsi blok berikutnya. Mode OFB juga berguna pada pengiriman pesan secara streaming, di mana setiap blok dapat dienkripsi dan dikirimkan secara terpisah.

5. Counter (CTR)



Gambar 5. Enkripsi CTR

Mode CTR adalah mode operasi block cipher yang menggunakan counter sebagai input pada enkripsi blok. Pada mode ini, counter dienkripsi menggunakan kunci dan digunakan sebagai input pada XOR antara counter yang telah dienkripsi dan plaintext pada blok selanjutnya. Mode CTR sangat berguna pada pengiriman pesan secara streaming, di mana setiap blok dapat dienkripsi dan dikirimkan secara terpisah.

C. Perancangan Desain Block Cipher

Konsep dasar dalam desain block cipher meliputi:

1. Confusion dan Diffusion

   Konsep *confusion* dan *diffusion* merupakan dasar dari desain block cipher. *Confusion* mengacu pada kebingungan dan kesulitan dalam merekonstruksi plaintext dari ciphertext tanpa mengetahui kunci rahasia. Sedangkan *diffusion* mengacu pada penyebaran informasi dari plaintext ke ciphertext, sehingga setiap bit pada plaintext mempengaruhi banyak bit pada ciphertext. Kombinasi dari kedua konsep ini memastikan bahwa setiap bit pada plaintext memiliki pengaruh yang signifikan pada ciphertext.

2. Substitusi dan Permutasi

Konsep ini mengacu pada operasi-substitusi dan permutasi pada bit-bit dari blok plaintext. Pada operasi substitusi, setiap bit plaintext diganti dengan bit yang berbeda menggunakan tabel pengganti (S-box). Pada operasi permutasi, urutan bit-bit pada blok plaintext diubah menggunakan tabel permutasi (P-box). Kombinasi dari kedua operasi ini menghasilkan proses transformasi yang kompleks pada plaintext.

3. Cipher Berulang

Cipher berulang adalah jenis enkripsi yang menggunakan kunci yang sama untuk setiap blok plaintext. Jenis enkripsi ini rentan terhadap serangan yang disebut serangan plaintext yang dipilih, karena setiap blok ciphertext yang sama dapat dihasilkan dari blok plaintext yang berbeda.

4. Pembangkitan Kunci Putaran

Pembangkitan kunci putaran adalah proses untuk menghasilkan kunci rahasia dari kunci utama untuk setiap putaran enkripsi. Konsep dasar ini menentukan bagaimana kunci rahasia dihasilkan dan bagaimana kunci rahasia tersebut digunakan dalam proses enkripsi dan dekripsi. Pembangkitan kunci putaran harus dirancang sedemikian rupa sehingga sulit bagi penyerang untuk merekonstruksi kunci rahasia dari plaintext dan ciphertext.

5. Jaringan Feistel

Jaringan Feistel adalah struktur enkripsi yang terdiri dari beberapa putaran yang sama dengan struktur yang mirip. Konsep dasar ini memastikan bahwa blok-blok ciphertext tidak dapat dikaitkan dengan blok-blok plaintext tertentu. Jaringan Feistel juga menggunakan konsep substitusi dan permutasi untuk memperkuat keamanan algoritma enkripsi.

6. Ukuran Blok dan Kunci

Ukuran blok dan kunci adalah faktor penting dalam desain block cipher. Ukuran blok mempengaruhi jumlah bit pada blok plaintext dan ciphertext, sedangkan ukuran kunci mempengaruhi kekuatan kriptografi dari algoritma enkripsi. Ukuran blok dan kunci harus dipilih sedemikian rupa sehingga sulit bagi penyerang untuk merekonstruksi kunci rahasia dari plaintext dan ciphertext.

III. Rancangan Block Cipher

Omnium *block cipher* adalah cipher berulang yang menggunakan metode *chaining* pada CBC (*Cipher block chaining*). Mode CBC digunakan untuk membantu mengimplementasikan prinsip *diffusion*. Cipher menggunakan sebuah kunci dan *Initialization Vector* (IV) sepanjang 128 bit. Enkripsi dan dekripsi dilakukan per blok dimana enkripsi tiap blok individual bergantung pada semua blok-blok sebelumnya. Setiap blok pesan mempunyai panjang 128 bit juga. Enkripsi, dekripsi, dan pembangkitan kunci putaran menggunakan kombinasi dari fungsi substitusi, permutasi, dan *shift* sebagai berikut.

A. Substitusi

Proses substitusi dilakukan menggunakan matriks substitusi (S-Box) berukuran 16x16 yang menampung 256 bit. Operasi substitusi memproses input block cipher yang terdiri atas 16 bytes (128 bit). Fungsi ini disebut sebagai *S*, yang berarti substitusi.

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

Gambar 6. S-box untuk fungsi *S*

Adapun suatu invers dari *S* yaitu S^{-1}. Fungsi invers ini dibutuhkan terutama untuk melakukan dekripsi, yaitu melakukan kebalikan dari substitusi yang dilakukan pada enkripsi.

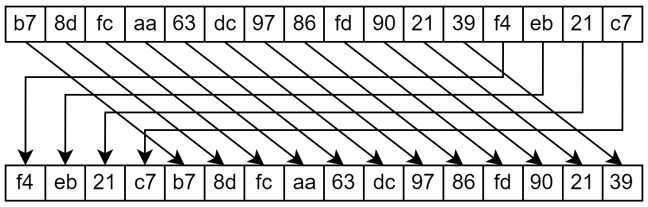
	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	a3	ec	e7	eb	22	ab	af	35	80	51	a7	db	2e	77	fb	e6
10	3a	12	39	ed	2a	c9	67	20	fd	74	f2	ff	9c	f4	e2	30
20	07	2d	93	d6	86	8f	27	3c	84	f5	45	21	e1	78	81	b5
30	54	37	d3	33	b8	96	55	9a	57	b2	10	42	4b	d7	02	e5
40	59	13	dc	ba	bb	ae	ca	f0	c2	8b	76	03	d9	43	df	14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

Gambar 7. S-box invers untuk fungsi S^{-1}

Misal, dilakukan substitusi menggunakan *S* pada byte ‘0x75’ dan ‘0xba’. Maka hasil dari substitusi kedua byte tersebut melalui S-box yaitu ‘0xb5’ dan ‘0x43’ secara terurut. Kedua hasil substitusi pada *S* ini dapat dikembalikan ke input asalnya dengan menggunakan S^{-1}. Melihat dari tabel S-box invers maka ‘0xb5’ menjadi ‘0x75’ dan ‘0x43’ menjadi ‘0xba’.

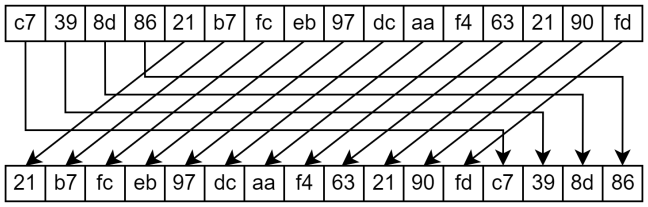
B. Shift

Proses shift menjadi salah satu teknik permutasi dengan melakukan pergeseran bit pada blok 128 bit ke kanan sejauh 32 bit atau 4 byte secara siklik. Fungsi ini disebut sebagai *RS*, yang berarti right shift.



Gambar 8. Contoh *shift* dengan fungsi *RS*

Adapun suatu invers dari *RS* yaitu fungsi RS^{-1} yang kebalikan dari *RS*, melakukan pergeseran bit pada blok 128 bit ke kiri sejauh 32 bit atau 4 byte secara siklik.



Gambar 9. Contoh *shift* dengan fungsi RS^{-1}

C. Permutasi

Proses permutasi dilakukan menggunakan tabel permutasi (P-Box) berupa array yang menampung 128 bit. Operasi permutasi memproses input block cipher yang terdiri atas 16 bytes (128 bit). Fungsi ini disebut sebagai *P*, yang berarti permutasi. Permutasi diterapkan karena membantu menyembunyikan hubungan statistik yang ada di antara plainteks dan cipherteks, serta kunci. Hal ini karena permutasi dilakukan pada kelompok data yang lebih kecil berupa bit. Sehingga, permutasi membantu mengimplementasikan fungsi *confusion*.

Block cipher yang diterima *P* diubah menjadi representasi biner. Kemudian, posisi setiap bit dipindahkan posisinya berdasarkan P-Box. Misal, elemen pertama P-Box adalah ‘0x7b’, maka artinya elemen ke-‘0x7b’ atau ke-123 di antara bit-bit tersebut dipindahkan ke posisi pertama. Di akhir operasi, hasil permutasi bit-bit tersebut dikembalikan ke representasi byte hexadecimal. Pada algoritma ini, digunakan P-Box sebagai berikut.

```
[
 7b, 74, 72, 3f, 1e, 5a, 45, 00,
 23, 19, 13, 2e, 25, 52, 43, 4c,
 35, 16, 27, 6e, 42, 7d, 12, 67,
 10, 54, 49, 2c, 06, 1f, 0a, 47,
 6d, 6b, 2f, 60, 30, 4a, 2a, 36,
 33, 3c, 1c, 5c, 44, 57, 07, 7c,
 5b, 4b, 1a, 03, 1d, 70, 7f, 73,
 1b, 7a, 50, 05, 04, 7e, 37, 3a,
 0c, 59, 34, 39, 71, 6c, 58, 0f,
 02, 38, 41, 53, 08, 6f, 63, 56,
 29, 46, 26, 3b, 22, 68, 11, 76,
 62, 65, 64, 31, 77, 78, 3e, 01,
 2d, 79, 32, 0d, 48, 3d, 55, 2b,
 75, 5d, 20, 51, 4d, 0b, 5e, 6a,
 09, 15, 4e, 21, 61, 28, 14, 17,
 0e, 69, 18, 24, 4f, 66, 40, 5f
]
```

Gambar 10. P-box untuk fungsi *P*

Adapun suatu invers dari *P* yaitu P^{-1} yang merupakan kebalikan dari P melakukan proses reverse. Fungsi ini menggunakan tabel permutasi yang merepresentasikan invers atau kebalikan dari cara kerja P-Box. Misal, jika pada P-Box posisi pertama (index 0) ditempati oleh ‘0x7b’ (123). maka ‘0x00’ terdapat di posisi ke 123 pada P-Box. Proses permutasi dilakukan pada setiap bit dengan aturan yang sama dan kemudian hasil permutasi bit-bit tersebut dikembalikan ke representasi byte hexadecimal. Pada algoritma ini, digunakan P-Box invers sebagai berikut.

```
[
 07, 5f, 48, 33, 3c, 3b, 1c, 2e,
 4c, 70, 1e, 6d, 40, 63, 78, 47,
 18, 56, 16, 0a, 76, 71, 11, 77,
 7a, 09, 32, 38, 2a, 34, 04, 1d,
 6a, 73, 54, 08, 7b, 0c, 52, 12,
 75, 50, 26, 67, 1b, 60, 0b, 22,
 24, 5b, 62, 28, 42, 10, 27, 3e,
 49, 43, 3f, 53, 29, 65, 5e, 03,
 7e, 4a, 14, 0e, 2c, 06, 51, 1f,
 64, 1a, 25, 31, 0f, 6c, 72, 7c,
 3a, 6b, 0d, 4b, 19, 66, 4f, 2d,
 46, 41, 05, 30, 2b, 69, 6e, 7f,
 23, 74, 58, 4e, 5a, 59, 7d, 17,
 55, 79, 6f, 21, 45, 20, 13, 4d,
 35, 44, 02, 37, 01, 68, 57, 5c,
 5d, 61, 39, 00, 2f, 15, 3d, 36
]
```

Gambar 11. P-box inverse untuk fungsi P^{-1}

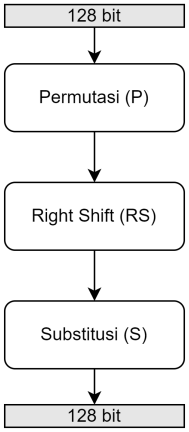
D. Key Schedule

Key schedule digunakan untuk membangkitkan 15 kunci tambahan dengan kunci asli sebagai kunci pertama.

Pembangkitan kunci dilakukan dengan melakukan operasi *right shift* dan substitusi secara berturut-turut pada *key*. 16 kunci ini kemudian akan digunakan pada proses enkripsi maupun dekripsi pada setiap putaran.

E. Enkripsi

Proses enkripsi pada Omnium Block Cipher dilakukan dengan mengubah plainteks menjadi blok plainteks dengan panjang setiap blok adalah 128 bit. *Padding* '0x00' akan ditambahkan jika diperlukan untuk melengkapi 128 bit pada blok terakhir. Proses enkripsi memanfaatkan key dan *Initialization Vector* (IV) dengan panjang 16 bytes. Key yang digunakan pada setiap *enciphering* berbeda karena dibangkitkan dari *key schedule* seperti yang dijelaskan di atas.

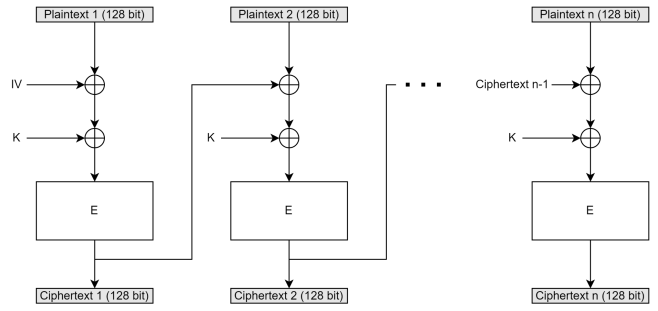


Gambar 12. Fungsi *E*

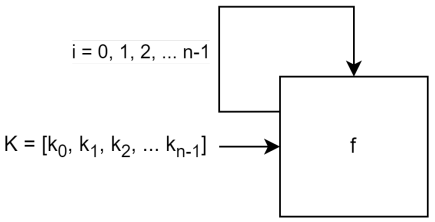
Proses enkripsi, disebut sebagai fungsi *f*, dilakukan dengan cara berikut pada setiap blok. Misalkan suatu blok adalah *b*. Didefinisikan pula sebuah fungsi *E* yang merupakan urutan operasi permutasi, *shift*, dan substitusi. Atau dalam kata lain,

$$E=S(RS(P(b))).$$

1. Jika blok tersebut merupakan blok pertama maka lakukan operasi XOR *b* dengan IV. Untuk blok-blok setelahnya, lakukan operasi XOR *b* dengan tepat 1 blok sebelumnya.
2. Lakukan operasi XOR *b* dengan *key*.
3. Jalankan fungsi E pada *b*.



Gambar 13. Fungsi *f*

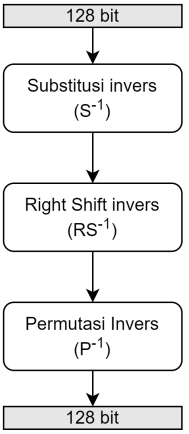


Gambar 14. Putaran enkripsi, dengan putaran ke-*i* menggunakan kunci ke-*i*

Lakukan proses enkripsi tersebut, yaitu fungsi *f*, total sebanyak 16 putaran dengan putaran ke-*i* menggunakan kunci ke-*i*. Pada akhirnya, hasil enkripsi seluruh blok 16 bytes akan digabungkan kembali menjadi sebuah ciphertext yang satu.

F. Dekripsi

Proses dekripsi pada Omnium Block Cipher dilakukan dengan alur yang terbalik dari enkripsi. Operasi dimulai dengan mengubah ciphertext menjadi blok ciphertext dengan panjang setiap blok adalah 128 bit. Proses dekripsi harus menggunakan key dan Initialization Vector (IV) dengan panjang 16 bytes yang digunakan pada proses enkripsi sebelumnya. Sebelum dilakukan dekripsi, blok ciphertext disalin terlebih dahulu karena terdapat operasi yang membutuhkan referensi block cipher awal.

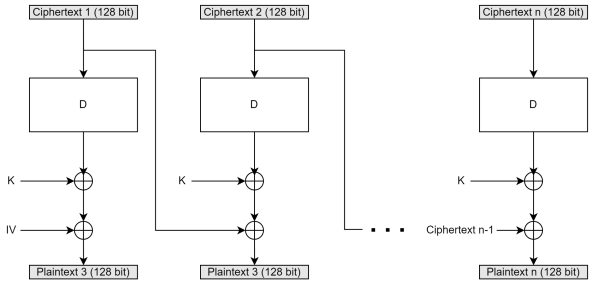


Gambar 15. Fungsi *D*

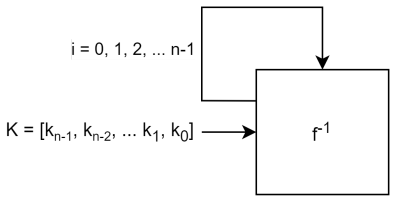
Proses dekripsi, disebut sebagai fungsi f^{-1}, dilakukan dengan cara berikut pada setiap blok. Misalkan suatu blok adalah *b*. Didefinisikan sebuah fungsi D yang merupakan operasi invers substitusi, *right shift*, dan permutasi secara berturut-turut. Atau dalam kata lain,

$$D=P^{-1}(RS^{-1}(S^{-1}(b))).$$

1. Jalankan fungsi D pada *b*.
2. Lakukan operasi XOR *b* dengan *key*.
3. Jika blok tersebut merupakan blok pertama maka lakukan operasi XOR b dengan IV. Untuk blok-blok setelahnya, lakukan operasi XOR b dengan tepat 1 blok sebelumnya.



Gambar 16. Fungsi f^{-1}



Gambar 17. Putaran dekripsi, dengan putaran ke-*i* menggunakan kunci ke-*(16-i-1)*

Lakukan proses dekripsi tersebut, yaitu fungsi f^{-1}, total sebanyak 16 putaran putaran dengan putaran ke-*i* menggunakan kunci ke-(16-*i*-1). Pada akhirnya, hasil dekripsi seluruh blok 16 bytes akan digabungkan kembali menjadi sebuah plaintext yang satu. Dapat diperhatikan bahwa fungsi *D* merupakan fungsi E yang dioperasikan secara terbalik.

IV. Hasil Eksperimen dan Pembahasan

Algoritma yang sudah dibuat dapat digunakan untuk melakukan enkripsi sebuah plaintext menjadi ciphertext dan sebaliknya. Program akan menerima input berupa mode yang akan dilakukan, file input, kunci, dan IV. Program akan mengeluarkan output berupa file plaintext ataupun ciphertext.

A. Analisis Waktu Enkripsi dan Dekripsi

Untuk melihat contoh plaintext yang digunakan dan ciphertext yang dihasilkan dapat dengan menjalankan program dari source code:

https://github.com/Putriliza/Kripto-2

```
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 1
Plaintext (.txt file): small.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  0.029974699020385742
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 1
Plaintext (.txt file): medium.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  1.4687352180480957
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 1
Plaintext (.txt file): large.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  2.4290554523468018
```

Gambar 18. Waktu Enkripsi

```
                                    python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 2
CipherText (.txt file): 05-03-2023 16.58.14.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  0.01603531837463379
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 2
CipherText (.txt file): 05-03-2023 16.58.49.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  1.3252863883972168
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 2
CipherText (.txt file): 05-03-2023 16.59.15
Traceback (most recent call last):
  File "c:\Belajar Python\Kripto-2\cipher.py", line 117, in <module>
    file = open(ciphertext, 'rb')
FileNotFoundError: [Errno 2] No such file or directory: '05-03-2023 16.59.15'
PS C:\Belajar Python\Kripto-2> python -u "c:\Belajar Python\Kripto-2\cipher.py"
Welcome to Omnium block cipher!
Operation:
1. Encrypt
2. Decrypt
Choose operation (1-2): 2
CipherText (.txt file): 05-03-2023 16.59.15.txt
Key (16 byte): PdSgVkYp3s6v8y/B
Initialization Vector (16 byte). Leave blank for default: dRgUkXp2r5u8x/A?
Time taken:  2.8928442001342773
```

Gambar 19. Waktu Dekripsi

File	Waktu		
	Size	***Enkripsi***	***Dekripsi***
small.txt	56 bytes	0.03s	0.02s
medium.txt	2989 bytes	1.47s	1.33s
large.txt	7308 bytes	2.43s	2.89s

Tabel 1. Waktu Enkripsi dan Dekripsi

Dapat dilihat bahwa waktu enkripsi dan dekripsi kurang lebih sama. Hal ini dikarenakan pada dasarnya enkripsi dan dekripsi pada block cipher merupakan fungsi yang sama, hanya dibalik saja urutannya. Waktu yang dibutuhkan untuk enkripsi dipengaruhi oleh panjang plaintext. Operasi utama dari algoritma ini adalah melakukan enkripsi sebanyak 16 kali pada plaintext yang sudah dibagi menjadi ukuran 16 byte. Oleh karena itu kompleksitasnya adalah $O(N)$, dimana N adalah panjang plaintext. Dapat dilihat bahwa hasil tersebut tercermin pada hasil testing yang juga cenderung linear. Kompleksitas ini cukup baik untuk N yang besar.

B. Analisis Efek Longsoran (Avalanche Effect)

Block cipher yang baik sebaiknya memiliki efek longsoran yang tinggi artinya walaupun hanya mengubah sedikit plaintext ataupun kunci, hasil ciphertext yang dihasilkan harus berbeda jauh. Testing diharapkan langsung dilakukan dalam bentuk file pada source code yang tertera, tidak menggunakan text yang terdapat pada laporan karena mungkin terdapat kesalahan konversi byte.

1. Perbedaan 1 Byte pada Kunci

   Plaintext:

   Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Key 1:

PdSgVkYp3s6v8y/B

Ciphertext 1:

淝Oy靨e虋聶咢愸琵球东訠銁븯屒　碻鸛　蝦难吾
藍唜樢荀　　　軩

Key 2:

PdSgVkYp3s6v8yuB

Ciphertext 2:

ž'ú_çO^È ,*ª„À+p}jº
'bI¸tXÔÊ6@'JvÚ Œ³Åß|h¬YôIæë6\þÚ‡µÜ;Qáug

Terdapat perbedaan sebesar 100% pada ciphertext 1 dan ciphertext 2 yang merupakan hal yang baik.

2. Perbedaan 1 Byte pada Plaintext

Key:

PdSgVkYp3s6v8y/B

Plaintext 1:

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Ciphertext 1:

淝Oy靨e虋聶咢愸琵球东訠銁븯屒　碻鸛　蝦难吾
藍唜樢荀　　　軩

Plaintext 2:

Lorem ipsum dolor sit amet. consectetur adipiscing elit.

Ciphertext 2:

淝Oy靨e虋聶咢愸□?β□槭鱝鵫훠伈翕ൺ奐葱↰□?
臆輳傉墻

Terdapat perbedaan sebesar 75% pada ciphertext 1 dan ciphertext 2 yang merupakan hal yang cukup baik. Kesamaan terjadi pada 8 byte pertama dari ciphertext. Hal ini dikarenakan perubahan plaintext baru terjadi di tengah plaintext. Hal ini dapat menjadi celah keamanan, tetapi tidak terlalu besar karena perubahannya sendiri sudah sebesar 75%.

C. Analisis Ruang Kunci

Key space merupakan nilai yang menyatakan berapa banyaknya jumlah bit yang digunakan untuk mengenkripsi suatu plain text. Ukuran dari kunci haruslah cukup besar agar dapat menghindari serangan brute force search attack. Misalkan panjang dari kunci yang digunakan adalah sepanjang N bits, maka akan ada kombinasi sebanyak 2^N kombinasi nilai yang mungkin untuk kunci tersebut. Apabila nilai N sangat besar, maka akan sangat sulit bagi penyerang untuk mencari semua kemungkinan dari kunci tersebut. Adapun untuk kasus terburuk, jumlah minimum pencarian yang harus dieksekusi oleh penyerang untuk mendapatkan kunci yang sesuai seharusnya senilai dengan $2^{N/2}$.

Keamanan dari suatu algoritma kriptografi ditentukan oleh kemampuannya untuk bertahan dari serangan brute force. Untuk algoritma ini dengan ukuran dari kunci yaitu N = 128, maka akan dibutuhkan waktu sekitar 5.61×10^{36} tahun (dengan asumsi processor membutuhkan waktu satu detik untuk satu juta proses) untuk melakukan pencarian brute force. Hal ini tidak sebanding dengan waktu untuk pengiriman pesan dibandingkan dengan waktu untuk pencarian kata kunci yang sesuai. Sehingga menurut analisis key space algoritma ini merupakan algoritma yang aman dari serangan brute force.

D. Analisis Known Plaintext Attack

Algoritma yang kami buat menggunakan mode operasi Cipher Block Chaining (CBC). CBC menggunakan input dari plaintext sebelumnya sebagai "vektor inisialisasi" (IV) untuk mengenkripsi blok plaintext berikutnya. CBC memiliki keuntungan dalam memberikan keamanan yang lebih tinggi dibandingkan dengan mode operasi yang sederhana seperti ECB. Namun, CBC juga masih bisa terkena serangan known plaintext attack.

Serangan known plaintext attack adalah jenis serangan kriptografi yang dilakukan oleh penyerang yang memiliki akses ke plaintext dan ciphertext yang telah dienkripsi menggunakan kunci rahasia yang sama. Dalam serangan ini, penyerang mencoba untuk merekonstruksi kunci rahasia yang digunakan dalam enkripsi dengan cara menganalisis perbedaan antara plaintext dan ciphertext.

Untuk melakukan serangan known plaintext attack pada CBC, penyerang harus memiliki beberapa pasang plaintext dan ciphertext yang telah dienkripsi dengan menggunakan kunci rahasia yang sama. Kemudian, penyerang dapat menggunakan plaintext dan ciphertext ini untuk merekonstruksi kunci rahasia dengan cara berikut:

- Memperoleh IV

  Penyerang harus memperoleh IV yang digunakan dalam enkripsi. IV dapat diperoleh dari plaintext dan ciphertext pertama yang dienkripsi.

- Menghitung XOR antara ciphertext dan plaintext

  Penyerang harus menghitung XOR antara ciphertext dan plaintext yang telah dienkripsi. Hasil dari operasi XOR ini akan menghasilkan nilai yang dikenal sebagai "diferensial".

- Mencari perbedaan antara blok plaintext

  Setiap blok plaintext memiliki perbedaan dengan blok plaintext sebelumnya. Penyerang dapat mencari perbedaan ini dengan menghitung XOR antara plaintext pada blok saat ini dan plaintext pada blok sebelumnya.

- Mencari perbedaan antara blok ciphertext

Setiap blok ciphertext memiliki perbedaan dengan blok ciphertext sebelumnya. Penyerang dapat mencari perbedaan ini dengan menghitung XOR antara ciphertext pada blok saat ini dan ciphertext pada blok sebelumnya.

- Mencari S-Box dan P-Box

  Setelah perbedaan plaintext dan ciphertext telah ditemukan, penyerang dapat mencoba untuk memperoleh informasi tambahan tentang S-Box dan P-Box yang digunakan dalam enkripsi.

- Mencari kunci rahasia

  Dengan menggunakan informasi yang diperoleh dari tahap sebelumnya, penyerang dapat mencoba untuk merekonstruksi kunci rahasia yang digunakan dalam enkripsi.

Oleh karena itu, serangan known-plaintext attack untuk algoritma yang kami buat tidak terlalu berbahaya karena sulit untuk dipecahkan.

V. Kesimpulan dan Saran

Berdasarkan hasil pengujian dan analisis, algoritma Omnium Block Cipher yang telah dibuat dapat digunakan untuk melakukan enkripsi dan dekripsi pada sebuah teks. Pendekatan yang digunakan merupakan pendekatan yang sudah ada, tetapi dengan modifikasi yang dilakukan dapat dihasilkan sebuah algoritma yang cukup efektif dan aman, Rancangan algoritma yang sudah dibuat masih dapat dikembangkan lebih lanjut, terutama dalam optimasi kecepatan enkripsi dan dekripsi.

Acknowledgment

Kami ingin menyampaikan rasa terima kasih yang tulus kepada semua pihak yang telah memberikan kontribusi pada penelitian ini. Pertama-tama, puji syukur ke hadirat Tuhan yang Maha Esa yang telah memberikan kesempatan kepada penulis untuk menyelesaikan makalah ini dengan baik. Kami ingin berterima kasih kepada dosen pengampu mata pelajaran IF4020 Kriptografi, Dr. Ir. Rinaldi Munir, MT. yang telah memberikan pengetahuan dan dukungan terhadap topik yang diangkat pada makalah ini. Penulis berharap agar makalah ini dapat mendukung pengembangan block cipher yang lebih aman dan menambah wawasan pembaca mengenai block cipher.

Link ke Program

Program Python bersama dengan beberapa *testcase*: https://github.com/Putriliza/Kripto-2

Referensi

[1] Rinaldi Munir. Cipher. *Block Cipher*. Slide Kuliah IF4020 Kriptografi, 2023.

[2] J. Daemen, V. Rijmen. The Rijndael Block Cipher.The Design of Rijndael, AES - The Advanced Encryption Standard, Springer-Verlag 2002 (238 pp.)

[3] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.

[4] Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A. (1996). "Chapter 7: Block Ciphers". Handbook of Applied Cryptography