STRAIT Cipher: Block Cipher Dengan Implementasi Jaringan Feistel

Annisa Ayu Pramesti[1], **Muhammad Fauzan Rafi Sidiq Widjonarto**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: 13518085@std.stei.itb.ac.id, 13518147@std.stei.itb.ac.id

Abstraksi. STRAIT Cipher (Scatter and Twist Cipher) adalah algoritma *block cipher* yang mengimplementasikan jaringan Feistel dan memiliki fitur yang menerapkan transposisi, permutasi, manipulasi bit, serta substitusi yang kompleks. Cipher ini dapat digunakan di berbagai mode operasi seperti ECB, CBC, dan CTR. Cipher ini menerapkan konsep *confusion* dan *diffusion* dari Shannon dengan baik. **Kata kunci**: *block cipher*, jaringan Feistel, *confusion, diffusion*

1. Pendahuluan

1.1. Latar Belakang

Kriptografi merupakan bidang yang tidak terpisahkan di era digital ini, berkaitan dengan keamanan data siber. Pengiriman dan penerimaan suatu data tentunya harus aman dari setiap gangguan dan pemodifikasian oleh pihak ketiga yang ingin mengubah atau membaca isi data. Salah satu implementasi dari kriptografi modern antara lain adalah dengan pendekatan blok, dimana data atau pesan dibagi menjadi blok dengan ukuran tertentu untuk selanjutnya diproses secara kriptografi. Cipher blok kerap digunakan di era digital ini, contoh algoritmanya adalah DES [1] dan AES [2], dengan ukuran blok dan algoritma yang berbeda-beda pula [1][2]. Kedua cipher tersebut juga mengimplementasikan substitusi dengan sebuah *mapping* khusus (S-Box), yang didefinisikan secara terpisah.

1.2. Pendekatan dalam Merancang STRAIT Cipher

STRAIT Cipher merupakan akronim dari *Scatter and Twist*, mengacu ke algoritma implementasi cipher tersebut. STRAIT Cipher dirancang menggunakan konsep manipulasi dari setiap blok yang sedang diproses, dengan mengubah urutan, memutar, mensubstitusi, serta merotasi setiap bit di satu blok. Jumlah putaran feistel juga didapat dari proses manipulasi bit dan putaran dari kunci yang diberikan.

2. Studi Pustaka

Ada beberapa teori yang dipakai dalam mendesain STRAIT Cipher yaitu sebagai berikut.

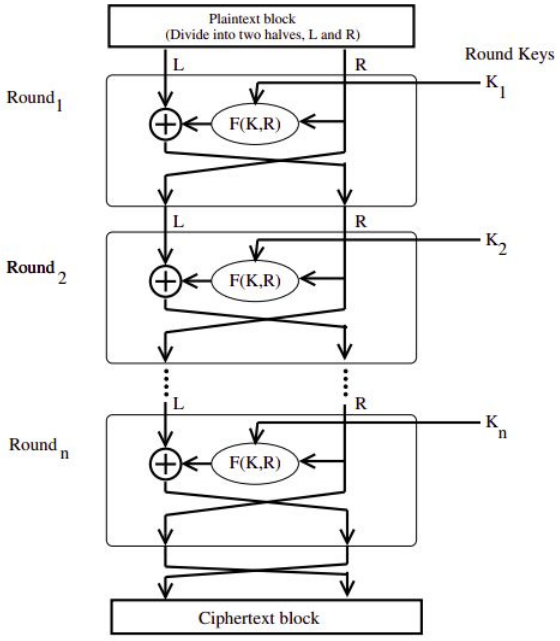
2.1. Block Cipher

Block cipher adalah algoritma yang mengenkripsi satu blok bit *plaintext* x ke blok bit *ciphertext* y. Transformasi *plain text* tersebut dilakukan menggunakan kunci rahasia K, dengan $E_K(x) = y$. Setiap kunci mendefinisikan pemetaan dari blok *plaintext* ke blok *ciphertext*. Untuk ukuran blok adalah b bit, jumlah nilai input yang berbeda ke block cipher adalah 2^b. Hal ini berarti bahwa sebuah block cipher dapat dipandang sebagai cipher dari alfabet ukuran 2^b. Walau demikian, hanya sebagian kecil dari

semua kemungkinan permutasi pada ukuran alfabet 2^b yang akan sesuai dengan kunci tertentu di block cipher.

2.2. Feistel Network

Feistel network adalah sistem kriptografi yang menggunakan algoritma yang sama untuk proses enkripsi dan dekripsinya. Gambar di bawah menunjukkan bahwa *feistel network* terdiri dari n iterasi. Setiap iterasi terdapat fase substitusi dan fase permutasi. Input block untuk setiap iterasi dibagi menjadi dua bagian dengan nama L untuk setengah bagian kiri dan R untuk setengah bagian kanan.



Gambar 1. Alur proses feistel network

Pada awal iterasi R tidak mengalami perubahan sedangkan L dioperasikan melalui sebuah fungsi yang menerima R dan kunci enkripsi (K) sebagai parameter. Fungsi yang dilakukan untuk mengoperasikan L disebut sebagai Fungsi Feistel. Fase permutasi pada akhir setiap iterasi dilakukan dengan menukar L dan R yang dimodifikasi. Dengan begitu, L pada iterasi berikutnya adalah R dari iterasi saat ini dan sebaliknya. Pada Feistel Network, hubungan antara output iterasi ke-i dan output iterasi sebelumnya, yaitu (i-1), diberikan oleh:

$$\begin{aligned} LE_i &= RE_{i-1} \\ RE_i &= LE_{i-1} \oplus F(RE_{i-1}, K_i) \end{aligned} \quad (1)$$

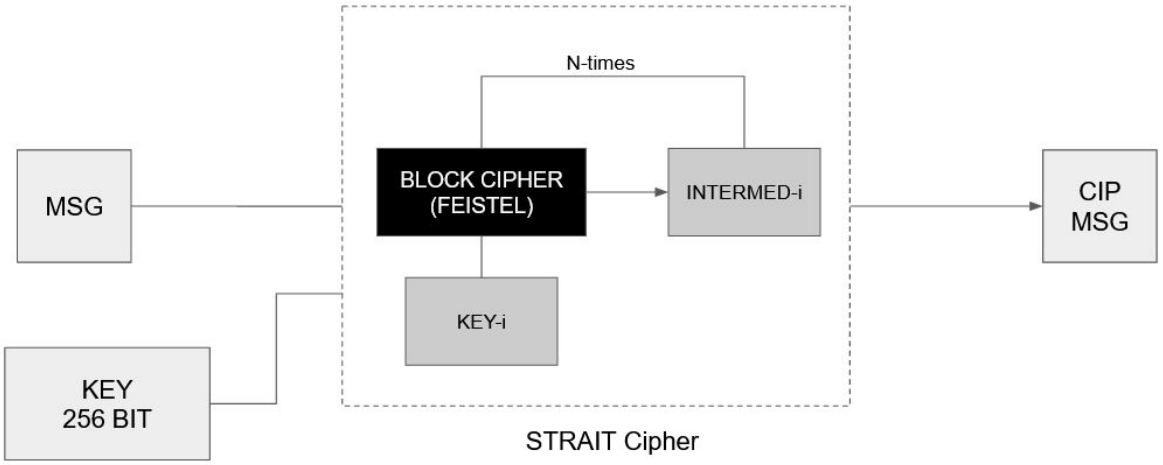
di mana $\oplus$ menunjukkan operasi bitwise EXCLUSIVE-OR. Simbol F menunjukkan Fungsi Feistel yang "mengacak" RE_{i-1} dari iterasi sebelumnya dengan kunci iterasi K_i. Kunci K_i diturunkan dari kunci enkripsi utama.

2.3. Confusion and Diffusion

Claude Shannon mengidentifikasi 2 properti dari *secure cipher* yaitu *confusion* dan *diffusion*[3]. *Confusion* berarti setiap digit biner atau bit dari *ciphertext* harus bergantung pada beberapa bagian dari kunci[3]. Difusi berarti bahwa jika satu bit *plaintext* atau kunci diubah maka perubahan yang terjadi pada *ciphertext* harus signifikan[3]. Kedua properti ini mengaburkan hubungan antara *ciphertext*, *plaintext*, dan kunci.

3. Cara Kerja Cipher

STRAIT Cipher merupakan cipher yang mengimplementasikan jaringan feistel. Masukan dari cipher adalah pesan dan kunci dengan panjang 256 bit. Pesan akan dipotong-potong menjadi blok dengan ukuran 64 bit. Berikut merupakan alur penggunaan dari STRAIT Cipher:



Gambar 2. Alur penggunaan STRAIT Cipher

Berikut merupakan penjelasan dari langkah-langkah dari STRAIT Cipher:

3.1. Proses Awal

Pertama, algoritma menginisialisasikan jumlah putaran feistel yang akan dilakukan. Hal ini dilakukan dengan melakukan operasi XOR ke paruh pertama kunci dan paruh kedua kunci. Lalu hasil XOR tersebut dilakukan rotasi sebesar 5 bit ke kiri. Hasil tersebut dijadikan angka, sebut sebagai n. Banyak putaran feistel yang dilakukan adalah 10 + n mod 11. Sehingga putaran minimal feistel adalah 10 dan maksimal adalah 20.

Pesan akan dipecah menjadi blok dengan panjang 64 bit. Bila ada blok yang kurang dari 64 bit, maka akan dilakukan *padding* dengan byte 0. Setiap blok disimpan di satu larik penyimpanan.

3.2. Desain Jaringan Feistel

Jaringan feistel yang diimplementasikan sama dengan jaringan feistel standar. Setiap blok pesan akan dibagi dua bagian (kanan dan kiri, masing-masing memiliki panjang 32 bit) yang akan diputar dengan jaringan feistel biasa. Jaringan feistel akan diputar sebanyak n-kali yang dibangkitkan di proses awal (menggunakan kunci). Untuk iterasi ke-i, kunci untuk fungsi putar adalah kunci awal yang dirotasikan ke kiri sebesar i bit.

3.3. Substitusi S-Box

STRAIT Cipher hanya mempunyai satu S-Box saja, yang dibuat untuk *mapping* setiap byte dari pesan ke satu byte yang lain. Berikut merupakan S-Box dari STRAIT Cipher:

x,y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	30	56	0f	04	3b	40	12	b2	97	21	dd	7f	da	20	d2	2a
1	d4	e0	11	83	40	fa	c7	06	08	44	82	54	10	a2	ae	5f
2	95	c8	9e	b1	3a	00	31	98	f5	bf	0e	92	57	d5	04	94
3	31	f4	6c	21	96	76	a5	f8	f6	35	f5	86	39	1e	f9	f0
4	2f	7a	66	ae	51	04	72	9c	b1	6a	c7	b0	52	cb	33	7b
5	00	c3	97	03	3c	d6	16	cd	fa	2e	e2	bc	dd	91	c0	1d
6	15	3a	4f	4a	17	19	e5	09	6b	1b	3a	31	fb	0e	37	c8
7	b0	29	2f	72	7d	c3	ba	f3	2b	e2	61	0e	42	94	2d	40
8	e3	cd	18	94	e8	13	1a	2e	68	75	22	5c	44	9c	6a	77
9	a0	d8	be	f9	88	b1	51	5a	69	44	65	7d	1f	80	df	25
a	25	64	cb	76	7c	2d	99	0d	00	55	df	e5	f3	4e	d0	c5
b	74	db	5b	d7	60	79	0e	da	4f	8d	9b	c3	7c	58	8d	5c
c	ce	e8	d1	36	66	48	33	49	5f	7c	6d	5a	aa	36	12	fa
d	35	16	9c	37	23	12	bc	ce	9b	34	12	09	11	ab	bb	dc
e	48	e5	bf	c0	66	47	57	8a	b0	5c	7a	fe	25	14	49	77
f	07	67	2c	51	af	fa	7f	8a	dd	0c	cc	1e	cd	66	33	ff

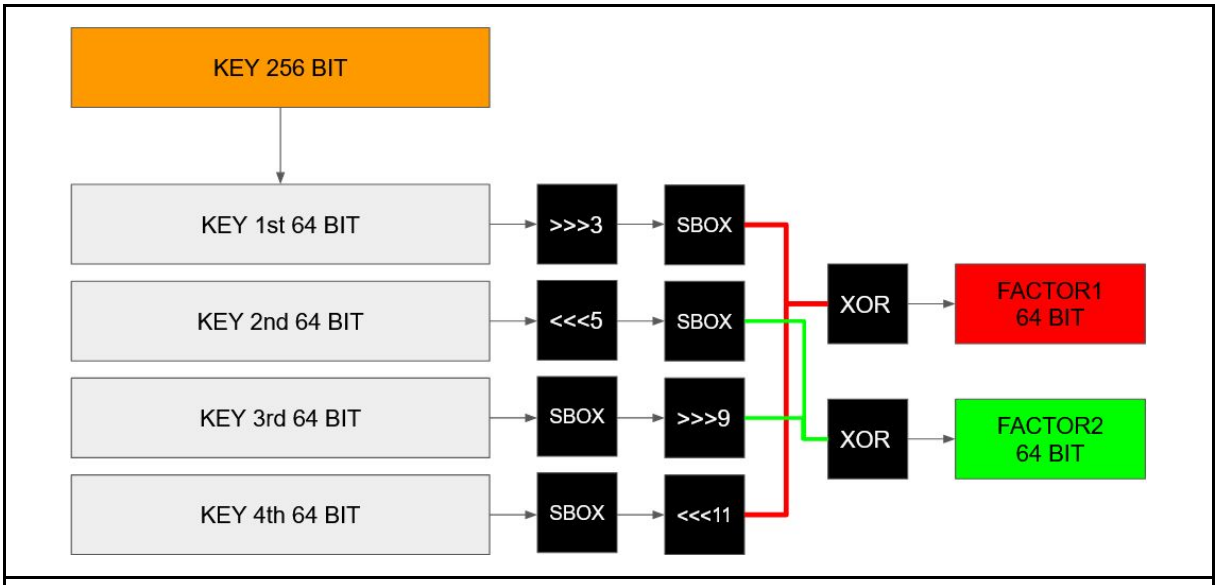
Gambar 3. S-Box dari STRAIT Cipher

Cara substitusi menggunakan S-Box ini adalah dengan mengambil 4 bit pertama (dari *most significant bit*) dari byte di indeks baris dan 4 bit terakhir (ke *least significant bit*) di indeks kolom untuk mendapatkan byte substitusi. Misalkan byte 0xab yang diproses dengan S-Box ini akan disubstitusikan menjadi 0xe5.
Pembangkitan dari setiap sel di S-Box dilakukan secara acak, dan beberapa dari sel memiliki nilai yang sama untuk menerapkan konsep *confusion* dari Shannon, sehingga pesan yang disubstitusikan menggunakan S-Box ini akan sulit untuk direkayasa-balik (*reverse engineered*)

3.4. Desain Fungsi Putar

Masukan dari fungsi putar adalah potongan blok pesan yang akan diolah secara kriptografi. Karena panjang blok adalah 64 bit, maka potongan blok pesan memiliki panjang 32 bit. Fungsi ini menerima dua argumen, yaitu potongan pesan dan kunci ke-i untuk iterasi ke-i. Dari kunci tersebut, dibangkitkan dua komponen tambahan untuk mengolah yaitu *factor1* dan *factor2*.

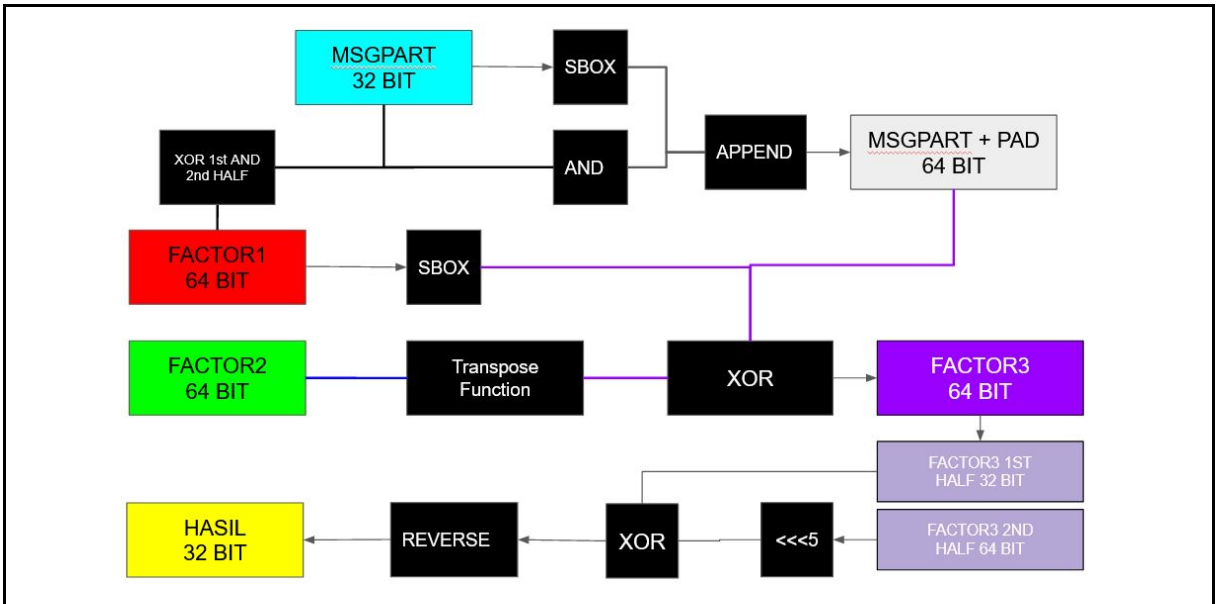
Pembangkitan dari kedua bagian tersebut dilakukan dengan membagi kunci yang panjangnya 256 bit menjadi empat kuartil: seperempat pertama, kedua, ketiga, dan keempat. Dua kuartil pertama akan dilakukan operasi rotasi terlebih dahulu, lalu dilakukan substitusi menggunakan S-Box. Untuk dua kuartil terakhir, dilakukan substitusi dahulu lalu dilakukan operasi rotasi. Rotasi dilakukan ke kanan 3 bit, ke kiri 5 bit, ke kanan 7 bit, dan ke kiri 11 bit secara berturut-turut untuk kuartil pertama, kedua, ketiga, dan keempat. Pemilihan jumlah bit tersebut dipilih agar tidak ada dua bit yang tetap sama urutannya dari depan, sehingga menerapkan prinsip *diffusion* dari Shannon.



Gambar 4. Mekanisme pembangkitan factor1 dan factor2

Setelah membangkitkan factor1 dan factor2, akan dilakukan langkah-langkah sebagai berikut:

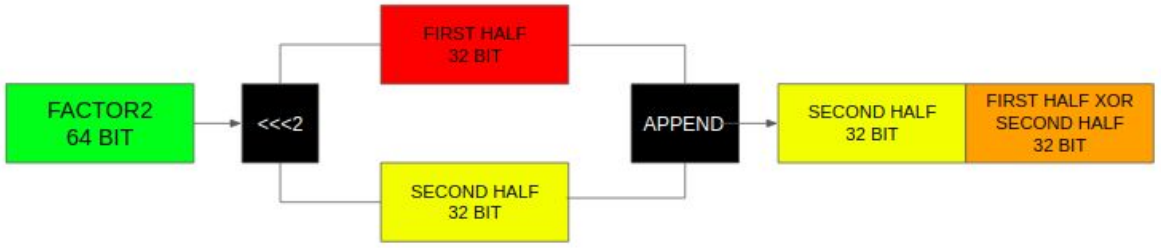
- Mensubstitusikan potongan pesan dengan S-Box, lalu melakukan *append* dengan hasil operasi AND dari potongan pesan dan hasil XOR paruh dari factor1. Sebut hasil ini sebagai MSGPART+PAD dengan panjang 64 bit (dua buah 32 bit yang di-*append*)
- Melakukan substitusi S-Box ke factor1. Sebut hasilnya sebagai SFAC1
- Melakukan fungsi transposisi ke factor2. Sebut hasilnya sebagai TFAC2
- Melakukan operasi XOR MSGPART+PAD, SFAC1, dan TFAC2. Hasil XOR ini disbut sebagai factor3 dengan panjang 64 bit.
- Factor3 dibagi menjadi dua paruh, dan pada paruh kedua dilakukan rotasi 5 bit kearah kiri. Lalu dilakukan operasi XOR ke kedua paruh.
- Hasil XOR kedua paruh factor3 dibalikkan urutan bitnya (*reverse*). Hasil dari pembalikkan ini merupakan hasil dari fungsi putar



Gambar 5. Alur fungsi putar di jaringan feistel

3.5. Fungsi transpose

Fungsi transpose yang diimplementasikan memiliki tujuan sebagai faktor transposisi dari STRAIT cipher. Fungsi ini melakukan left shift sebanyak 2 kali dari factor2, selanjutnya disebut dengan shifted_factor2 lalu memecahnya menjadi 2 bagian. Keluaran dari fungsi ini adalah hasil *append* dari bagian kedua dan bagian pertama yang sudah di-xor dengan bagian kedua dari shifted_factor2.



Gambar 6. Alur fungsi transpose

4. Eksperimen dan Analisis

4.1. Implementasi

Implementasi dari STRAIT Cipher dibuat di bahasa Python versi 3. Program menerima key sepanjang 256 bit, pesan, mode (ECB, CBC, atau CTR) dan *Initialization vector* (IV) untuk mode CBC dan CTR dengan panjang 64 bit. Parameter tersebut akan dimasukkan ke kelas STRAIT. Berikut merupakan spesifikasi dari kelas STRAIT dengan sintaks bahasa Python versi 3:

```
class Mode(enum.Enum):
    # Kelas enumerasi Mode untuk membuat aspek diskrit di masukan mode
    ECB = 1
    CBC = 2
    CTR = 3

class STRAIT:
    def __init__(self, key : str, mode : Mode):
        # KONSTRUKTOR: Menginisialisasi objek dengan key dan mode

    def init_message_buffers(self, msg):
        # Memotong-motong pesan menjadi array of blocks dengan ukuran 64 bit

    def generate_feistel_iter(self):
        # Membangkitkan jumlah putaran feistel dari key

    def encrypt(self, msg, IV : str = None):
        # Melakukan proses enkripsi pesan. Bila mode yang diinisialisasikan
adalah CBC atau CTR, maka dibutuhkan masukan IV

    def decrypt(self, msg, IV : str = None):
        # Melakukan proses dekripsi pesan. Bila mode yang diinisialisasikan
adalah CBC atau CTR, maka dibutuhkan masukan IV

    def sbox(self, inp : bytes):
        # Melakukan substitusi bytes masukan dengan S-Box
```

```
    def generate_factors_from_key(self):
        # Fungsi pembangkitan factor1 dan factor2 dari key ke-i

    def generate_factor3(self, msg_block : bytes):
        # Fungsi pembangkitan factor3 dari potongan msg, factor1, dan factor2

    def transpose_fac2(self):
        # Fungsi transpose untuk transposisi factor

    def round_func(self, msg_part : bytes):
        # Fungsi putaran di putaran feistel, menggunakan fungsi transpose_fac2,
generate_factors_from_key dan generate_factor3

    def feistel_network_enc(self, msg_block : bytes):
        # Fungsi putaran feistel untuk enkripsi

    def feistel_network_dec(self, msg_block : bytes):
        # Fungsi putaran feistel untuk dekripsi
```

S-Box diimplementasikan dengan *hardcode* di kode cipher. S-Box dijadikan variabel global di dalam kode implementasi. Pada fungsi encrypt dan decrypt, mekanisme dari enkripsi dan dekripsi diimplementasikan menurut masukan Mode oleh pengguna. Di setiap fungsi, dilakukan asersi bahwa setiap masukan memiliki ukuran yang sesuai dengan algoritma (kunci memiliki panjang 256 bit, IV memiliki panjang 64 bit, dan lain-lain).

4.2. Simulasi dan Analisis Mode ECB

Pada simulasi ini, terdapat tiga test case yang digunakan. Test case pertama adalah sebagai berikut.

Kunci	nama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy

Berikut merupakan percobaan pertama untuk mode ECB:

Kunci	`6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69`
Plaintext	`68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00`
Ciphertext	`4a f8 68 af 7d 40 90 af b5 2d e1 5d 00 d0 cc c8 ba 87 f5 94 80 eb 90 e1 14 27 f4 15 ea 52 10 e9 da 3a 3b e4 5c 0a 2d dd`

Pada percobaan kedua, kunci dibedakan 1 bit untuk melihat perubahan pada *ciphertext.*

Kunci	mama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy

Kunci	`6d 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61`

	72 20 73 61 79 61 20 62 6f 62 69
Plaintext	68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	24 6d 1f 76 26 27 77 c0 5f cc 10 94 25 49 2a 09 9a ba c6 7d e5 14 7e 30 16 6e 2d b2 72 12 2b 4a db ca 2c 5f 38 fa a2 92

Perbedaan *ciphertext* test case pertama pada simulasi mode ECB dan *ciphertext* test case kedua hanyalah terlihat cukup signifikan. Hal ini mengimplikasikan bahwa perubahan sekecil apapun pada kunci akan merubah total *ciphertext* yang dihasilkan pada mode ECB.

Pada percobaan ketiga, kunci disamakan dengan percobaan 1 tetapi *plaintext* dibedakan 1 bit untuk melihat perubahan pada *ciphertext*.

Kunci	nama saya alice, pacar saya bobi
Plaintext	guang guanheng is a really handsome boy

Kunci	6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	67 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	e1 28 54 27 34 43 8e 77 b5 2d e1 5d 00 d0 cc c8 ba 87 f5 94 80 eb 90 e1 14 27 f4 15 ea 52 10 e9 da 3a 3b e4 5c 0a 2d dd

Perbedaan *ciphertext* test case pertama pada simulasi mode ECB dan *ciphertext* test case terakhir hanyalah pada 64 bit pertama karena pada mode ECB proses enkripsi dilakukan per *block* dengan ukuran 64 bit.

4.3. Simulasi dan Analisis Mode CBC

Pada simulasi ini, terdapat tiga test case yang digunakan. Test case pertama adalah sebagai berikut.

Kunci	nama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy
IV	12345678

Berikut merupakan percobaan pertama untuk mode CBC:

Kunci	6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72

	65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	36 8e 3d ef bc ef ec 31 3f 6d 48 d4 cc d8 ad 79 03 da 05 05 9e 09 7b c9 cd c6 01 8c 2c 08 8c d6 be eb 6f 30 61 39 ec 6f

Pada percobaan kedua, kunci dibedakan 1 bit untuk melihat perubahan pada *ciphertext.*

Kunci	mama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy

Kunci	6d 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	fd 66 65 94 64 57 67 7e 7d d4 11 ae e2 15 00 47 c9 fa 61 d6 50 d6 0b a8 c9 e4 e4 26 39 ad f8 e0 f7 73 f7 7f b5 74 05 24

Perbedaan *ciphertext* test case pertama pada simulasi mode CBC dan *ciphertext* test case kedua hanyalah terlihat cukup signifikan. Hal ini mengimplikasikan bahwa perubahan sekecil apapun pada kunci akan merubah total *ciphertext* yang dihasilkan pada mode CBC.

Pada percobaan ketiga, kunci disamakan dengan percobaan 1 tetapi *plaintext* dibedakan 1 bit untuk melihat perubahan pada *ciphertext*.

Kunci	nama saya alice, pacar saya bobi
Plaintext	guang guanheng is a really handsome boy

Kunci	6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	67 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	4c 28 49 ab 12 4a ba c1 aa 5c 78 6e cd bf 42 3d 66 c9 58 1d f2 7a 84 b9 1d 62 6d e8 51 5d 89 ce 0d aa c0 e4 3c 03 7f 45

Berbeda dengan mode ECB, perbedaan *ciphertext* test case pertama pada simulasi mode CBC dan *ciphertext* test case terakhir sangat tinggi karena proses enkripsi dilakukan dengan "mengacak" seluruh block pada *plaintext* yang ada.

4.4. Simulasi dan Analisis Mode CTR

Pada simulasi ini, terdapat tiga test case yang digunakan. Test case pertama adalah sebagai berikut.

Kunci	nama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy
IV	12345678

Berikut merupakan percobaan pertama untuk mode CTR:

Kunci	6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	67 94 e5 d6 ad 31 1f a4 c3 63 75 34 d5 d9 43 aa 44 a3 46 de 13 d3 ec a5 1b 93 7f 4b 72 c1 50 94 a6 04 5e a2 05 4c 68 47

Pada percobaan kedua, kunci dibedakan 1 bit untuk melihat perubahan pada *ciphertext.*

Kunci	mama saya alice, pacar saya bobi
Plaintext	huang guanheng is a really handsome boy

Kunci	6d 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	68 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72 65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	cd 38 ab 8d 15 bb b9 07 a3 a0 6b 13 07 31 b3 8d 50 01 4b 83 ea c8 e9 78 64 a8 96 28 c5 6b 11 13 3c 06 c1 29 70 db 17 56

Perbedaan *ciphertext* test case pertama pada simulasi mode CTR dan *ciphertext* test case kedua hanyalah terlihat cukup signifikan. Hal ini mengimplikasikan bahwa perubahan sekecil apapun pada kunci akan merubah total *ciphertext* yang dihasilkan pada mode CTR.

Pada percobaan ketiga, kunci disamakan dengan percobaan 1 tetapi *plaintext* dibedakan 1 bit untuk melihat perubahan pada *ciphertext*.

Kunci	nama saya alice, pacar saya bobi
Plaintext	guang guanheng is a really handsome boy

Kunci	6e 61 6d 61 20 73 61 79 61 20 61 6c 69 63 65 2c 20 70 61 63 61 72 20 73 61 79 61 20 62 6f 62 69
Plaintext	67 75 61 6e 67 20 67 75 61 6e 68 65 6e 67 20 69 73 20 61 20 72

	65 61 6c 6c 79 20 68 61 6e 64 73 6f 6d 65 20 62 6f 79 00
Ciphertext	68 94 e5 d6 ad 31 1f a4 c3 63 75 34 d5 d9 43 aa 44 a3 46 de 13 d3 ec a5 1b 93 7f 4b 72 c1 50 94 a6 04 5e a2 05 4c 68 47

Perbedaan *ciphertext* test case pertama pada simulasi mode CTR dan *ciphertext* test case terakhir terdapat pada bit pertama sesuai dengan perbedaan *plaintext* karena pada mode CTR proses enkripsi dilakukan per *block* dan *plaintext* tidak dioperasikan pada fungsi enkripsi. Perubahan hasil *ciphertext* secara total dapat dilakukan dengan mengubah IV.

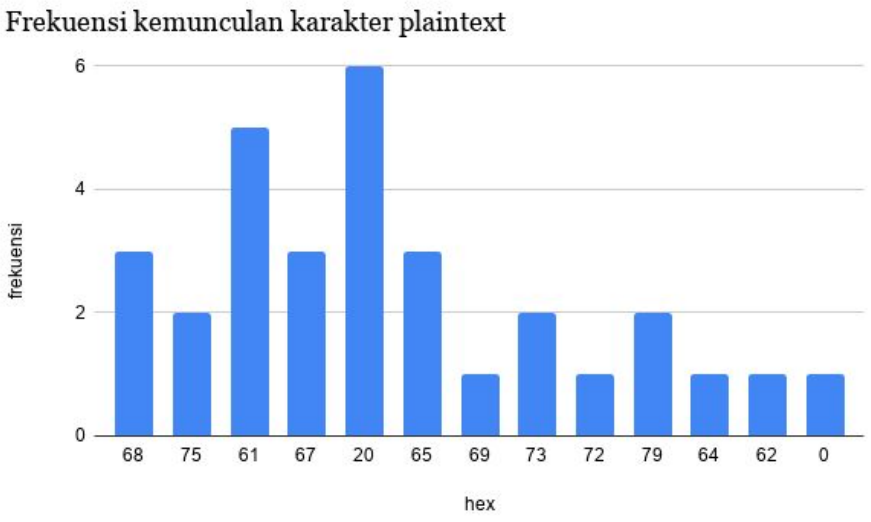
5. Analisis Sekuritas

5.1. Analisis serangan brute force

Algoritma yang diimplementasikan di fungsi putar memiliki operasi AND yang irreversibel, sehingga hal ini menyulitkan serangan *brute force* dengan pendekatan *reverse engineering*. Dan untuk melakukan *brute force* kunci yang panjangnya 256 bit, harus dilakukan pencarian ruang dengan ukuran yang sangat besar, yaitu 2^{256} kemungkinan. Bahkan dengan asumsi sebuah GPU bisa melakukan 1 milyar hash/detik, lalu dikombinasikan dengan 4 milyar GPU yang dimiliki oleh 4 milyar manusia yang ada di 4 milyar Bumi di 4 milyar galaksi, masih dibutuhkan 2^{128} detik (kira-kira 4 milyar kali umur alam semesta) untuk menemukan 1 dari 4 milyar kemungkinan. Oleh karena itu, STRAIT Cipher cukup aman dari serangan *brute force*.

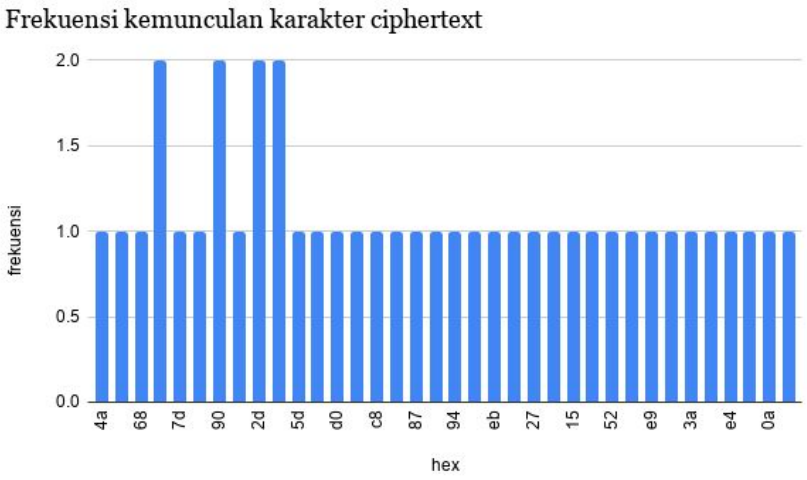
5.2. Serangan analisis frekuensi

Berikut merupakan grafik frekuensi kemunculan karakter pada *plaintext* dari mode ECB :



Gambar 7. Frekuensi kemunculan karakter *plaintext* test case pertama mode ECB

Berikut merupakan grafik frekuensi kemunculan karakter pada *ciphertext* dari mode ECB :



Gambar 8. Frekuensi kemunculan karakter *ciphertext* test case pertama mode ECB

Pada kedua gambar di atas terlihat bahwa frekuensi kemunculan karakter sangat berbeda pada *plaintext* dan *ciphertext.* Oleh karena itu, dapat disimpulkan bahwa STRAIT cukup aman dari serangan analisis frekuensi karena frekuensi karakter pada *ciphertext* tidak berhubungan dengan frekuensi karakter dari *plaintext*.

5.3. *Analisis* Confusion *dan* Diffusion

Pada simulasi dan analisis yang telah dilakukan, setiap perubahan 1 bit kunci akan menghasilkan perubahan yang besar pada *ciphertext* untuk semua mode. Pada mode ECB dan CTR, perubahan 1 bit pada *plaintext* akan mengubah *ciphertext* pada blok yang bersesuaian sebanyak 1 blok / 64 bit untuk ECB, dan sebanyak 1 bit untuk CTR. Pada mode CBC, perubahan 1 bit *plaintext* menyebabkan perubahan besar pada seluruh blok *ciphertext.* Oleh karena itu, mode CBC telah memenuhi properti *confusion* dan *diffusion* sedangkan mode EBC memenuhi kedua properti tersebut hanya pada 1 blok saja.

6. Kesimpulan dan Pengembangan Lanjutan

STRAIT Cipher adalah cipher berbasis jaringan feistel yang memiliki mode ECB, CBC, dan CTR yang mempunyai implementasi yang fokus ke pemutaran dan pengacakan secara intensif. STRAIT CIpher menerima pesan dan kunci ukuran 256 bit, dan beroperasi dengan ukuran blok 64 bit. Putaran feistel, serta kunci setiap putaran digenerasikan hanya dengan manipulasi pengacakan dan pemutaran kunci awal. STRAIT Cipher mengimplementasikan konsep *confusion* dan *diffusion* dari Shannon dengan baik, sehingga bisa dipakai untuk menjaga kerahasiaan dan integritas pesan. Hasil pesan STRAIT Cipher sangat sukar untuk dipecahkan dengan analisis frekuensi ataupun dengan *brute force*. Untuk pengembangan lanjutan, cipher ini dapat dikembangkan dengan mode lain seperti CFB dan OFB.

7. Referensi

[1] National Institute of Standards and Technology, Data Encryption Standard (DES). (U.S.: U.S.Department of Commerce)

[2] National Institute of Standards and Technology, Advanced Encryption Standard (AES). (U.S.: U.S.Department of Commerce)

[3] Claude E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, vol. 28-4, pages 656–715, 1949.

Acknowledgments
Penulis ingin berterima kasih dan bersyukur kepada Allah SWT Tuhan Yang Maha Esa, karena-Nya penulis bisa berkuliah di ITB jurusan Teknik Informatika. Penulis berterima kasih kepada keluarga penulis yang mendukung keberjalanan kuliah. Penulis juga ingin berterima kasih ke Dr. Ir. Rinaldi Munir, M.T. atas bimbingan dan ilmu yang beliau berikan di mata kuliah IF4020 Kriptografi.