Ruadan

Algoritma *block cipher* berdasarkan barisan Fibonacci dengan *key-dependent* S-box

Dewita Sonya Tarabunga
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
dewitast20@gmail.com

Felix Limanta
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
felixlimanta@gmail.com

***Abstrak* – Salah satu sistem yang sering digunakan pada bidang kriptografi untuk mengamankan pesan dari penyadap adalah kriptografi kunci simetris. Sistem ini menggunakan kunci yang sama pada pihak pengirim dan penerima masing-masing untuk melakukan enkripsi dan dekripsi. Makalah ini akan membahas suatu algoritma *block cipher* kunci simetris baru yang menggunakan jaringan Feistel sebagai dasarnya bernama Ruadan. Algoritma ini menggunakan beberapa landasan matematis seperti barisan Fibonacci dan transformasi Pseudo-Hadamard.**

***Kata kunci* – kriptografi, *block cipher*, kunci simetris, Feistel, Fibonacci, S-box, Pseudo-Hadamard.**

I. PENDAHULUAN

Perkembangan kriptografi terjadi dengan sangat pesat pada zaman-zaman perang dunia. Hal ini dikarenakan pihak yang berperang memerlukan suatu cara untuk saling berkomunikasi tanpa mengkhawatirkan bahwa pesan yang dikirim tidak akan disadap dan jatuh ke tangan musuh. Pada zaman modern, di mana semua data terdapat pada jaringan, kriptografi semakin diperlukan untuk memastikan data-data yang dikirim melalui jaringan tidak akan disadap oleh pihak ketiga yang mungkin akan menyalahgunakannya. Karena itu, diperlukan penelitian untuk menemukan dan mengembangkan algoritma kriptografi yang kuat untuk mengamankan pesan.

Salah satu sistem yang sudah digunakan secara luas pada kriptografi adalah kriptografi kunci simetris, yang menggunakan kunci yang sama pada pihak pengirim untuk melakukan enkripsi dan pada pihak penerima untuk melakukan dekripsi. Beberapa algoritma populer yang menggunakan sistem ini adalah DES dan AES.

Makalah ini mendiskusikan sebuah algoritma *block cipher* baru bernama Ruadan. Nama Ruadan diambil dari nama dewa kerahasiaan Irlandia. Dengan memanfaatkan Jaringan Feistel, S-box dependen kunci enkripsi untuk substitusi, dan transformasi Pseudo-Hadamard, algoritma ini diharapkan dapat memberikan kontribusi pada ilmu kriptografi.

II. TINJAUAN PEKERJAAN TERKAIT

A. *Twofish [5]*

Twofish adalah *block cipher* yang menggunakan jaringan Feistel dan merupakan algoritma hasil pengembangan algoritma yang sudah ada sebelumnya, yaitu Blowfish. Twofish adalah salah satu algoritma yang menggunakan *key-dependent* S-box, sama seperti algoritma Ruadan yang kami ajukan.

Algoritma yang merupakan finalis pada kontes AES ini menggunakan transformasi Pseudo-Hadamard dalam fungsi putarannya. Transformasi ini sendiri diadaptasi dari algoritma SAFER. Transformasi ini berfungsi untuk melakukan pengacauan nilai bit pada suatu bilangan dan bersifat dapat dibalikkan. Penjelasan lebih lanjut tentang transformasi Pseudo-Hadamard akan dijelaskan pada bab Dasar Teori. Ruadan menggunakan transformasi ini juga sebagai landasan untuk melakukan pengacauan bit.

B. *Advanced Encryption Standard (AES) [6]*

Algoritma AES ditemukan oleh Joan Daemen dan Vincent Rijmen dan bernama algoritma Rijndael pada awalnya. Namun, karena algoritma ini memenangkan kontes AES yang diselenggarakan oleh NIST, algoritma ini kini lebih dikenal dengan nama AES. Algoritma ini merupakan *block cipher* yang beroperasi pada blok dengan panjang 128-bit atau 256-bit.

Terdapat 4 tahap pada setiap putaran pada algoritma AES; salah satu diantaranya adalah penggunaan S-box. S-box pada algoritma AES merupakan S-box statis yang sudah ditentukan di awal oleh para penemunya. Isi tabel dari S-box Rijndael tidak datang dengan sendirinya, melainkan melalui perhitungan matematika yang akan dijelaskan lebih lanjut pada bab Dasar Teori. Ruadan akan menggunakan S-box Rijndael sebagai basis dari *key-dependent* S-box yang akan dibangkitkan.

III. DASAR TEORI

Algoritma Ruadan menggunakan beberapa konsep sebagai landasan gagasannya. Berikut akan dibahas beberapa konsep tersebut.

A. Block cipher

Block cipher adalah sandi yang menerima masukan *plaintext* dalam blok-blok, yang nantinya dienkripsi sesuai

dengan transformasi yang didefinisikan pada *block cipher* tersebut. Transformasi *block cipher* secara umum dikendalikan oleh suatu kunci, sehingga *block cipher* juga dapat didefinisikan sebagai pemetaan suatu *plaintext* pada *message space* ke suatu *ciphertext* (mungkin terdapat pada *message space* yang sama), berdasarkan suatu kunci.

B. Confusion *dan* Diffusion

Dua prinsip dasar pada desain *block cipher* adalah *confusion* dan *diffusion*.

Confusion menyembunyikan hubungan antara *plaintext* dan *ciphertext*. *Confusion* dapat dihasilkan melalu transformasi substitutsi.

Diffusion berarti penyebaran pengaruh suatu bit *plaintext* atau kunci untuk menyembunyikan karakteristik statistik *plaintext*. Dengan kata lain, perbedaan satu bit masukan *plaintext* atau kunci menyebabkan perubahan yang signifikan. Cara sederhana untuk menghasilkan *diffusion* adalah permutasi *plaintext* pada level tertentu (*byte*/bit).

Operasi-operasi sederhana, seperti substitusi dan permutasi, bila dilakukan berkali-kali pada suatu blok *plaintext* dapat menghasilkan *confusion* dan *diffusion* yang baik. Hal inilah yang mendasari desain *iterated block cipher* pada suatu *block cipher*.

C. Struktur Block Cipher

Secara umum, terdapat 2 jenis struktur *block cipher*: SPN (*Substitution Permutation Network*) dan jaringan Feistel. Contoh *block cipher* berbasis SPN adalah AES, sedangkan contoh *block cipher* berbasis jaringan Feistel adalah DES, Twofish, Blowfish, dan Camellia.

Pada umumnya, SPN memiliki 4 jenis transformasi: transformasi substitusi, transformasi permutasi, transformasi penggabungan linear, dan transformasi penambahan kunci.

Jaringan Feistel. adalah salah satu jenis struktur *block cipher*. Pada jaringan Feistel seimbang, *plaintext* dibagi menjadi dua bagian sama panjang yang akan diproses secara berbeda, misalya L dan R. Nilai L dan R untuk putaran ke-i didefinisikan sebagai

$$L_{i+1} = R_i \quad (1)$$

$$R_{i+1} = L_i \oplus F(R_i, K_i) \quad (2)$$

Ciphertext didefinisikan sebagai (R_{n+1}, L_{n+1}), di mana n adalah jumlah putaran untuk *block cipher* tersebut. $F(R_i, K_i)$ adalah *round function* yang didefinisikan oleh suatu *block cipher*.

Kelebihan jaringan Feistel adalah *round function* yang digunakan tidak harus *invertible*. Hal ini disebabkan pada proses dekripsi, *round function* yang digunakan pada saat enkripsi digunakan kembali pada saat dekripsi, hanya saja dengan urutan key yang dibalik. Struktur feistel network dapat dilihat pada Figur 1.

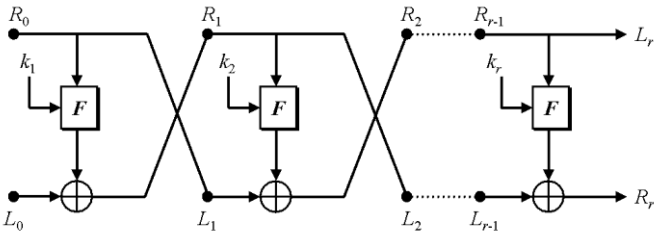

Fig. 1. Struktur jaringan Feistel
(Sumber: https://i.stack.imgur.com/IGGiW.gif)

D. Substitution Box

Substitution box atau disingkat S-box adalah suatu komponen kriptografi yang sering digunakan untuk melakukan substitusi, baik substitusi bit maupun substitusi byte. S-box sering digunakan untuk memenuhi properti *confussion* dari Shannon yang mengharuskan sebuh *block cipher* untuk meminimalisasi hubungan antara kunci dan *ciphertext.* S-box umumnya berbentuk matriks yang berfungsi seperti *lookup table.*

Salah satu algoritma terkenal yang menggunakan S-box dalam algoritma enkripsinya adalah AES atau Rijndael. Ruadan sendiri adalah algoritma yang akan bergantung pada S-box yang bersifat *key-dependent* yang akan digunakan baik pada penjadwalan kunci maupun pada fungsi putaran pada jaringan Feistel. Suatu *key-dependent* S-box memerlukan satu S-box basis dalam perhitungannya. Dalam hal ini, Ruadan menggunakan Rijndael S-box sebagai basisnya.

	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

Fig. 2. S-box Rijndael
(Sumber: https://captanu.wordpress.com/tag/aes)

E. Transformasi Pseudo-Hadamard

Transformasi Pseudo-Hadamard dipakai dalam algoritma-algoritma enkripsi seperti SAFER dan Twofish [5]. Transformasi ini digunakan untuk melakukan difusi kriptografis, yaitu dengan mengacaukan susunan bit. Transformasi ini dapat digunakan karena sifatnya yang *invertible* atau dapat dibalikkan sehingga bit asli sebelum pengacauan dapat diperoleh kembali.

Masukan rangkaian bit dengan panjang genap dipecah menjadi dua, yaitu a dan b. Transformasi dihitung dengan persamaan berikut:

$$a' = (a + b)\ mod\ 2^n \quad (3)$$
$$b' = (a + 2b)\ mod\ 2^n \quad (4)$$

Transformasi ini dinamanakan Pseudo-Hadamard karena transformasi ini merupakan manipulasi dari transformasi Hadamard.

F. *Barisan Fibonacci Modular*

Barisan Fibonacci secara general adalah suatu fungsi yang didefinsikan secara rekursif dengan definisi sebagai berikut.

$$f_0 = 0 \quad (5)$$
$$f_1 = 1 \quad (6)$$
$$f_k = f_{k-1} + f_{k-2},\ k \geq 2 \quad (7)$$

Dapat dilihat dengan mudah bahwa barisan Fibonacci merupakan barisan naik yang nilainya akan terus bertambah menuju tak hingga. Untuk membatasi penambahan nilai barisan ini, maka yang digunakan dalam algoritma Ruadan ini adalah barisan Fibonacci modular. Barisan Fibonacci modular adalah barisan Fibonacci yang hanya diambil nilai modularnya terhadap suatu bilangan yang ditetapkan sebagai modulonya. Secara formal, barisan Fibonacci modular dapat didefinisikan sebagai berikut:

$$f_0 = 0 \quad (8)$$
$$f_1 = 1 \quad (9)$$
$$f_k = (f_{k-1} + f_{k-2})\ mod\ n,\ k \geq 2 \quad (10)$$

Dapat dilihat bahwa terdapat sedikit perubahan pada bagian rekursi, yaitu nilai hasil dari penambahan dua nilai fungsi sebelumnya dimodulo dengan n.

Barisan Fibonacci modular merupakan barisan yang periodik, yaitu terdapat k yang bergantung pada n sehingga $f_i = f_{i+k}$ untuk semua i. Hasil ini ditemukan oleh Lagrange dan angka yang melambangkan nilai periode barisan Fibonacci modular itu disebut periode Pisano, dimana Pisano sendiri adalah penemu dari barisan Fibonacci. Untuk algoritma Ruadan, modulo yang digunakan disesuaikan dengan panjang blok yaitu 32, dan periode Pisano untuk bilangan pangkat dari 2 adalah

$$\pi(n) = \frac{3n}{2} \quad (11)$$

Periode ini cenderung lebih sedikit dibanding nilai periode Pisano untuk angka-angka sekitarnya.

IV. RANCANGAN *BLOCK CIPHER*

Algoritma Ruadan menggunakan struktur jaringan Feistel dengan besar blok pesan 256 bit atau 32 *byte*. Kunci yang digunakan Ruadan berukuran 128 bit atau 16 *byte*. Terdapat tiga hal yang harus ditentukan untuk menggunakan jarigan Feistel, yaitu jumlah putaran, algoritma penjadwalan kunci dan fungsi putaran. Selain itu, algoritma Ruadan meupakan algoritma yang menggunakan *key-dependent* S-box, maka di bawah juga akan dijelaskan algoritma pembangkitan S-box.

A. *Kalkulasi Jumlah Putaran*

Jumlah putaran pada algoritma Ruadan bersifat dinamis, dimana jumlah putaran akan bergantung dengan kunci yang digunakan untuk melakukan enkripsi. Range yang akan digunakan adalah 8 – 16. Maka, jumlah putaran dirumuskan dengan $8 + n$, dimana nilai n adalah nilai yang bergantung pada kunci. Adapun nilai n merupakan hasil *hash code* dari kunci sebagai dalam modulus 8:

$$\sum_{i=0}^{7} s_i(31)^{i+1}\ mod\ 8 \quad (12)$$

Fungsi (12) didasarkan pada fungsi *hash code* String dari bahasa Java.

B. *Pembangkitan S-box*

Pada algoritma Ruadan, S-box digunakan pada penjadwalan kunci dan fungsi putaran. Berbeda dengan kebanyakan algoritma lain seperti AES atau DES dimana S-box bersifat statis, Ruadan menggunakan S-box yang dinamis, yang akan dibangkitkan berdasarkan kunci yang digunakan.

S-box dibangkitkan menggunakan cara yang diusulkan Stoianov pada [7]. Pada cara tersebut, setiap byte pada S-box basis di-XOR dengan sebuah byte dari kunci. S-box yang digunakan sebagai basis adalah S-box Rjindael yang digunakan pada algoritma AES, sedangkan byte yang dipilih untuk meng-XOR S-box hasil adalah byte ke-n, di mana n adalah nilai yang dihasilkan dari (12).

C. *Ekspansi dan Penjadwalan Kunci*

Algoritma ekspansi kunci yang digunakan dibuat berdasarka algoritma penjadwalan kunci Rijndael dengan beberapa modifikasi.

Mula-mula, 16 *byte* kunci dipisah menjadi 4 *word*: w_0, w_1, w_2, w_3, masing-masing 4 byte. Kemudian, tergantung *word* keberapa yang sedang diproses, setiap *word* berikutnya dibangkitkan dengan proses sebagai berikut.

Untuk *word* berikutnya, setiap *k* kelipatan 4:

- $w_k = \sim(w_{k-4} \oplus w_{k-1})$
- Geser w_k sebanyak $f_{k/4}$ bit ke kiri secara sirkuler, di mana f_i adalah barisan Fibonacci modular ke-*i*.
- Terapkan transformasi pseudo-Hadabard pada blok 2 byte:
    - $a = w_{k_0}, w_{k_1}$
    - $b = w_{k_2}, w_{k_3}$
- Substitusi w_k menggunakan S-box yang telah dibangkitkan.

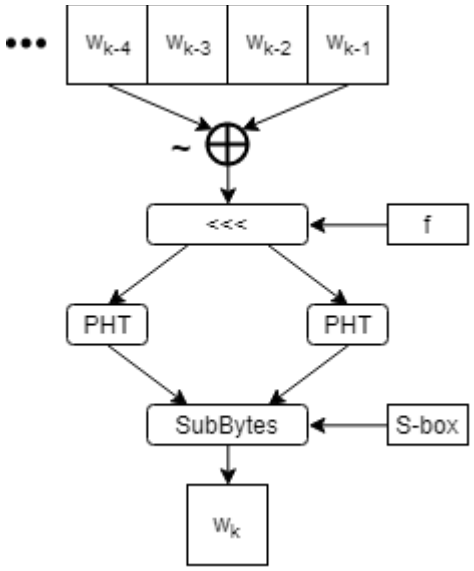

Fig. 3. Ilustrasi ekspansi kunci Ruadan (untuk k ≡ 0 mod 4)
(Sumber: pustaka penulis)

Jika k bukan kelipatan 4:

- $w_i = \sim(w_{k-4} \oplus w_{k-1})$
- Substitusi w_4 menggunakan S-box yang telah dibangkitkan.

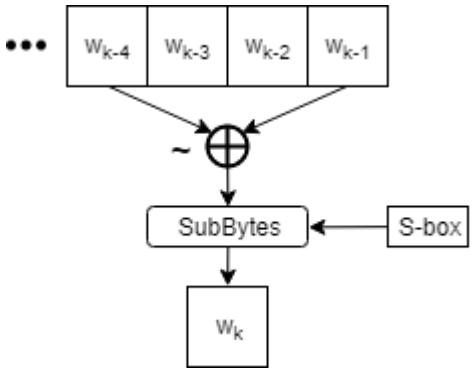

Fig. 4. Ilustrasi ekspansi kunci Ruadan (untuk k ≢ 0 mod 4)
(Sumber: pustaka penulis)

Ini diulang selagi $i < 4n + 4$ agar setiap putaran memiliki 4 *word* kunci. Kunci untuk putaran 1 adalah 4 *word* $w_4 - w_7$. Setiap putaran akan mengambil 4 *word* selanjutnya

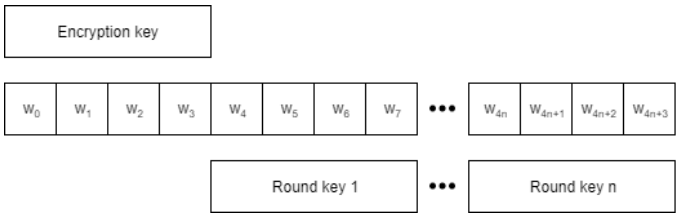

Fig. 5. Ilustrasi penjadwalan kunci pada Ruadan
(Sumber: pustaka penulis)

D. Fungsi Putaran

Salah satu komponen terpenting dalam *block cipher* berbasis jaringan Feistel adalah fungsi putaran (ditandai 'F' pada Figur 1). Algoritma Ruadan merupakan *block cipher* yang bekerja pada blok ukuran 32 byte. Karena pada jaringan Feistel blok dibagi menjadi bagian kiri dan kanan, fungsi putaran menerima masukan kunci sebesar 16 byte dan data untuk dienkripsi sebesar 16 byte.

Terdapat 5 proses yang terjadi di dalam fungsi putaran Ruadan yang akan dijelaskan sebagai berikut.

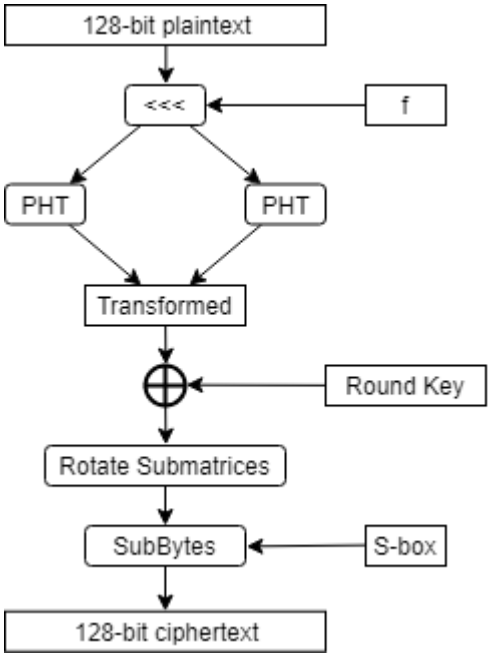

Fig. 6. Ilustrasi fungsi putaran pada Ruadan
(Sumber: pustaka penulis)

1) Penggeseran bit

Untuk setiap 4 *byte* (1 *word*) w_i pada blok, dilakukan penggeseran sirkuler ke kiri sebanyak f_i bit.

Nilai i diambil dari kali keberapa pergeseran bit pada *word* ini dilakukan pada suatu blok 128-bit. Misal, jika *word* yang hendak digeser merupakan *word* ke-0 dan proses enkripsi sedang berada pada putaran ke-4, maka i bernilai $4 * 4 + 0 = 16$. Nilai f_i adalah nilai ke-i pada barisan Fibonacci modular dengan modulus 32.

2) Transformasi Pseudo-Hadabard

Blok kemudian dibagi menjadi 2 bagian, masing-masing 2 *word*. Transformasi Pseudo-Hadabard diterapkan pada dua blok tersebut.

3) Penambahan Kunci

Blok di-XOR dengan kunci internal yang telah dibangkitkan untuk blok tersebut.

4) Rotasi Submatriks

Blok 16 byte direpresentasikan sebagai matriks 4×4. Setiap submatriks berukuran 2×2 diputar 90° searah jarum jam.

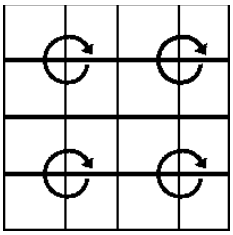

Fig. 7. Ilustrasi rotasi submatriks pada fungsi putaran Ruadan
(Sumber: pustaka penulis)

5) Substitusi S-box

Setiap byte pada blok disubstitusi menurut S-box yang telah dibangkitkan.

V. Eksperimen dan Pembahasan Hasil

Pada bagian ini, akan dilakukan dan diuraikan hasil beberapa eksperimen: enkripsi berkas biner untuk menguji *confusion* dan penggantian kunci untuk menguji *diffusion*.

Salah satu pengujian *diffusion* yang tidak dilakukan adalah pengubahan 1 bit *plaintext* dan mengamati perubahan *ciphertext* karena perubahan akan bergantung pada mode enkripsi, bukan algoritma. Perilaku perubahan untuk setiap mode adalah

- ECB: hanya blok yang berisi bit yang diubah yang berubah
- CBC: seluruh *ciphertext* berubah
- CFB: *byte* pada bit yang diubah akan berubah, lalu semua blok setelahnya berubah.
- OFB: *byte* pada bit yang diubah akan berubah, lalu semua blok sebelumnya berubah.
- *counter*: hanya *byte* pada bit yang diubah yang berubah.

Kedua eksperimen tersebut akan dijabarkan sebagai berikut.

A. Enkripsi Berkas Biner

Pada eksperimen ini, suatu berkas biner akan dienkripsi menggunakan Ruadan, lalu ditinjau frekuensi kemunculan setiap *byte*. Eksperimen ini ditujukan untuk mengamati persebaran *byte* pada *ciphertext* hasil enkripsi.

Eksperimen dilakukan dengan menerapkan algoritma Ruadan menggunakan 5 teknik *block cipher*: ECB, CBC, CFB, OFB, dan *counter*. Data *plaintext* yang digunakan adalah gambar .jpg dari berkas eksternal. Kunci yang digunakan adalah kunci 16-byte “informatikaitebe”.

Fig. 8. Gambar yang digunakan sebagai *plaintext* untuk pengujian
(Sumber: https://4.bp.blogspot.com/-mLOwpEsNL4Y/UCu0wcVsPBI/AAAAAAAAA6s/7ECKTpxXr3o/s1600/lena.bmp)

```
FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00
01 00 01 00 00 FF DB 00 84 00 03 02 02 0A 0A
0A 08 08 0A 0A 08 08 08 0A 08 08 08 0A 0A 08
0A 0A 0A 08 08 0A 08 08 08 08 08 08 08 08 08
08 08 08 08 08 08 08 08 08 0A 08 08 08 08
0A 0A 09 08 08 0B 0D 0A 08 0D 08 08 0A 08 01
03 04 04 06 05 06 0A 06 06 0A 0F 0D 0B 0D 10
0F 0F 10 10 0F 0D 0D 0F 0D 0F 10 0F 0D 0D 0D
0F 10 0F 0F 0D 0D 0D 0D
```

Fig. 9. Representasi *hexadecimal* 128 *byte* pertama dari *plaintext*

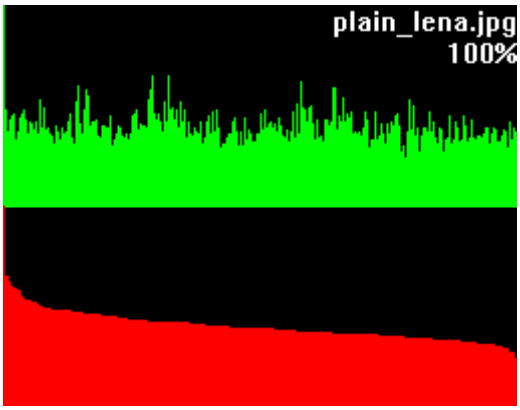

Fig. 10. Histogram persebaran frekuensi *byte* dari *plaintext*

Pada histogram di atas, dapat dilihat persebaran *byte* tidak merata. Bagian hijau menunjukkan histogram persebaran berdasarkan nilai *byte* tersebut (00 sampai FF), sedangkan bagian merah menunjukkan persebaran yang diurutkan besar ke kecil.

Untuk setiap mode enkripsi berikut, akan ditampilkan 128 *byte* pertama dari berkas hasil enkripsi serta histogram persebaran frekuensi dari berkas hasil tersebut. Hasil enkripsi gambar tidak langsung ditampilkan karena proses enkripsi merusak *encoding* JPEG gambar.

1) Electronic Code Book *(ECB)*

```
67 54 EA 2D 62 DA B0 CD 05 10 C5 D5 3D 89 A8
56 69 91 4A AC 86 70 D7 08 30 89 67 A9 2A 77
F1 B2 BC 32 24 FF 60 E8 00 5F 79 06 68 D4 68
F4 70 81 1B 48 2C ED 1D 26 95 3A 9F 16 18 3C
8E A3 DD F0 E6 70 5B D9 37 C8 FD 90 0F 16 B0
64 E9 F0 2C 03 05 5E C6 27 BF 76 A6 E8 F9 3C
E2 24 2A 77 0C B0 61 BF F4 2A 99 31 0B 03 11
46 7E 4F DE 03 97 C8 AE C5 54 0D 8D D5 4D E6
85 3A CB EB 86 0F 0A C5
```

Fig. 11. Representasi *hexadecimal* 128 *byte* pertama dari *ciphertext*, enkripsi ECB

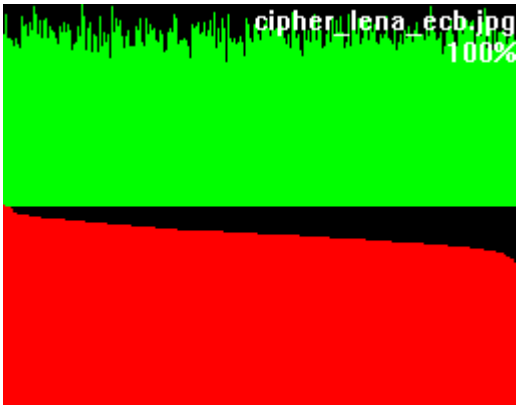

Fig. 12. Histogram persebaran frekuensi *byte* dari *ciphertext*, enkripsi ECB

2) Cipher Block Chaining *(CBC)*

Initialization vector yang digunakan adalah *string* 32 *byte* "jurusanteknikinformatikahidupitb".

```
5F 51 19 F6 7A FE 4F DB EF F2 EC FD 1C 62 79
D2 1C 99 D3 2A CC 6A 91 A5 EF C5 9C 59 F8 8F
FA 2C 15 5B 10 3A 04 22 28 1B C4 4E 79 31 1F
D2 B9 4A 0A B0 D1 4D 3C 3E 35 74 E1 80 A7 2E
85 B1 57 1D 08 61 85 AA A3 52 66 30 C2 2F CB
6A 4B 8C F8 07 8D C2 F5 91 65 CA E3 5C 78 55
D9 4E 8C CC 25 B6 86 83 A0 69 2C 88 E3 39 B7
95 2F 6F 0E 3F ED B1 10 F8 43 F6 DA 96 DC CB
A7 60 C8 12 CE 49 BF 1E
```

Fig. 13. Representasi *hexadecimal* 128 *byte* pertama dari *ciphertext*, enkripsi CBC

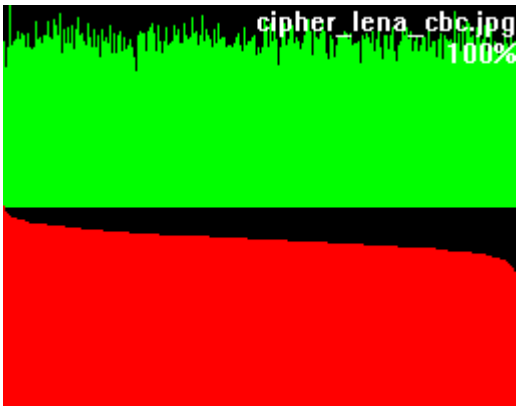

Fig. 14. Histogram persebaran frekuensi *byte* dari *ciphertext*, enkripsi CBC

3) Cipher Feedback *(CFB)*

```
87 2D 23 08 27 35 F4 CD 30 AC 09 74 72 55 77
B1 24 AD 14 25 21 97 DE CA 6D 17 35 3C 9C E4
3D 6C 20 AA 14 E2 14 CE D1 1A 77 9C F9 D1 9C
58 FB 2F 91 08 82 36 25 55 C1 93 04 61 87 8E
CD 92 01 72 63 A9 40 08 C9 2D F3 7B 84 21 7C
F5 40 C5 65 A1 04 0C 7F FD EB CE FA 3E 3D 82
5B B2 3A A8 8A 16 B0 C6 35 ED 06 90 13 32 F6
44 7C 8D 82 FC D5 57 01 FA E8 A3 FD 7E FF F5
E3 81 F1 A7 E6 B9 4D 68
```

Fig. 15. Representasi *hexadecimal* 128 *byte* pertama dari *ciphertext*, enkripsi CFB

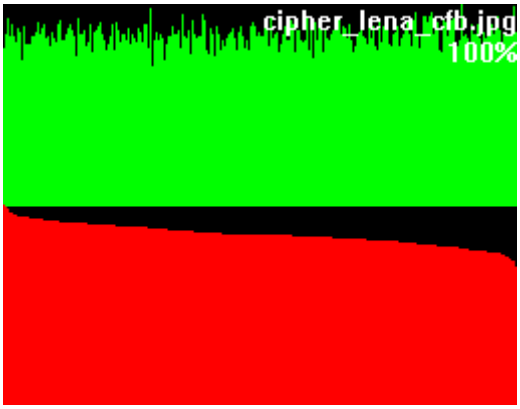

Fig. 16. Histogram persebaran frekuensi *byte* dari *ciphertext*, enkripsi CFB

4) Output Feedback *(OFB)*

```
87 2D 23 08 27 35 F4 CD 30 AC 09 74 72 55 77
B1 24 AD 14 25 21 97 DE CA 6D 17 35 3C 9C E4
3D 6C B5 50 F5 19 29 9A 60 C2 99 49 89 68 F1
CD 4E 53 9F 26 5F 34 EF 2B 84 7D E2 20 72 92
EC 50 02 53 F1 F5 8B 21 8F 1B C9 59 2B 2E 3C
F8 AB F8 6D 83 13 0D 4E B9 C5 A2 00 E6 CB 6A
53 4A 10 8E 60 EB 54 2A B2 61 B1 9B 72 C0 24
7E 9F C0 4A D1 AB E0 06 AF 2A 3E 25 BE 7B DB
9D 42 38 A1 91 4E 65 AA
```

Fig. 17. Representasi *hexadecimal* 128 *byte* pertama dari *ciphertext*, enkripsi OFB

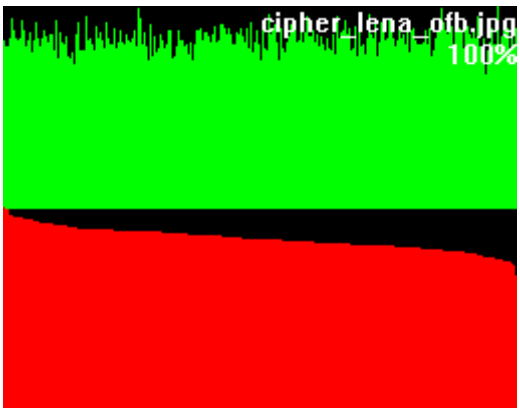

Fig. 18. Histogram persebaran frekuensi *byte* dari *ciphertext*, enkripsi OFB

5) Counter *(CTR)*

Counter diinisalisasi dengan nilai 0.

```
87 2D 23 08 27 35 F4 CD 30 AC 09 74 72 55 77
B1 24 AD 14 25 21 97 DE CA 6D 17 35 3C 9C E4
3D 6C 13 9D 3A 50 A0 B5 32 2E 4D 0E 36 ED 23
6B DA 62 61 05 30 C5 A6 F1 35 51 5B F0 B5 F5
F5 24 C4 FE FB 92 F3 F0 C0 CC FC 84 43 9E 03
60 C0 D1 74 71 CF BC 8A 4E D5 22 09 9B C1 A2
F4 76 84 D5 5A 8E 9A 69 FC CF 0A 09 40 C0 1F
72 98 F2 CB 52 2C F0 55 B9 69 31 1E 81 20 96
52 16 2D 0A 0B 2B BB 6C
```

Fig. 19. Representasi *hexadecimal* 128 *byte* pertama dari *ciphertext*, enkripsi *counter*

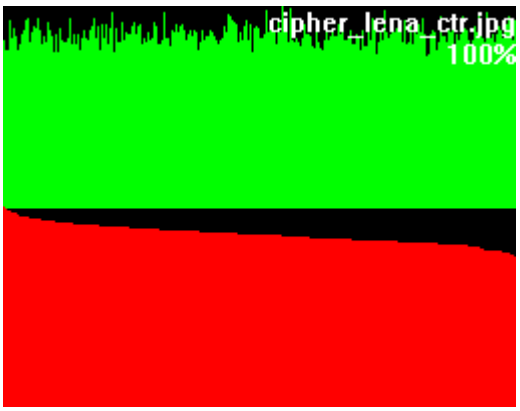

Fig. 20. Histogram persebaran frekuensi *byte* dari *ciphertext*, enkripsi *counter*

Dibandingkan dengan persebaran *byte* gambar *plaintext*, persebaran *byte* gambar *ciphertext* jauh lebih merata untuk setiap mode enkripsi.

B. Penggantian Sedikit Kunci

Pada eksperimen ini, suatu berkas teks akan dienkripsi menggunakan Ruadan. Setelah itu, satu bit dari kunci akan diubah, lalu berkas teks tersebut dienkripsi kembali. Eksperimen ini ditujukan untuk mengamati perbedaan *ciphertext* hasil kedua kunci yang hanya berbeda 1 bit.

Eksperimen dilakukan dengan menerapkan algoritma Ruadan menggunakan 5 teknik *block cipher*: ECB, CBC, CFB, OFB, dan *counter*. Data *plaintext* yang digunakan adalah teks dari berkas eksternal. Kunci yang semula digunakan adalah kunci 16-byte "inf**o**rmatikaitebe". Kunci yang telah diubah adalah "inf**n**rmatikaitebe".

```
69 6E 66 6F 72 6D 61 74 69 6B 61 69 74 65 62 65
```

Fig. 21. Representasi *hexadecimal* kunci semula. Bagian yang akan diganti ditandakan merah.

```
69 6E 66 6E 72 6D 61 74 69 6B 61 69 74 65 62 65
```

Fig. 22. Representasi *hexadecimal* kunci yang diganti. Bagian yang diganti ditandakan merah

1) Electronic Code Book *(ECB)*

```
9F 39 1C A3 36 6E 9B BE 0D E5 6D E1 7F BA 91 DD
95 D7 96 E5 4F 34 07 43 C2 3D A7 F6 60 90 CD 0F
67 08 F9 E1 5C 71 E0 5D 4E 7A 8F 4C 18 A9 92 AE
C9 AC 14 85 C4 27 61 FE C9 A3 6F 0D 13 21 E6 73
DF E6 AA E4 18 E7 F7 08 A6 A5 D1 DC 42 EF E9 0D
55 FA 1A 5B 7B 8F A8 36 8D 37 C2 F6 81 5E 26 93
12 88 77 77 77 92 95 30 28 EC CE 50 8E D8 EC 81
86 09 07 A8 12 45 E8 21 65 32 06 3E 47 3D 86 4D
```

Fig. 23. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode ECB semula

```
D9 99 4C 99 D3 76 59 94 9B 5E A7 79 5C 04 28 38
1B CC 22 0F 09 85 29 0D 28 89 F5 45 F3 99 0D 05
72 5B 28 E7 78 E3 81 4F 04 F2 2A B1 F8 EA 27 90
3F E1 69 BE AC 4F 48 EA 64 C0 78 6D 75 AA C0 E1
AD 11 94 93 E5 86 58 C8 92 6C 16 6A 66 85 E1 69
A5 61 04 8A 83 EF A8 92 9A 4E 70 4B 7F 7A E1 CF
BB 00 A3 62 BF 87 D4 CA 81 58 F1 6A 39 D1 10 2B
A4 F2 BF F4 C0 71 12 CF E2 C6 3A 18 3F 22 5D C4
```

Fig. 24. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode ECB yang diganti.

2) Cipher Block Chaining *(CBC)*

Initialization vector yang digunakan adalah *string* 32 *byte* "jurusanteknikinformatikahidupitb".

```
F8 52 6D AE 61 D4 E6 0C 13 0F 60 7D 23 5E 49 11
3C 52 36 C4 55 4A 4D F2 66 52 02 72 C8 2F BE BA
F5 90 3B 07 3C 9A E0 7F 14 19 65 59 C5 D0 28 AD
74 36 A5 15 41 51 03 40 6E 4A D8 6A E8 3B 72 C5
55 5E 12 61 19 C5 DE A4 40 05 B7 08 00 B8 33 5D
05 66 03 1E F3 9C AB A4 71 0A AE B8 40 C4 7F 1B
20 43 D3 88 81 75 D5 64 D6 09 2E 3A D8 A2 5B EB
17 A8 D6 8C 5F A4 3A F6 72 54 4B F6 B6 06 01 FE
```

Fig. 25. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CBC semula

```
A8 F5 65 7E 97 28 61 5B 31 64 EB 93 64 DC 05 5D
C6 20 36 7C 1F 02 5B 61 95 24 D6 32 72 04 DC E3
CF DF F8 64 46 49 E5 F3 68 06 23 B3 D5 3B 79 C1
8C 5E 0C 68 F2 17 87 C7 5D FE 6B F8 DB F5 51 72
D6 C7 BB BB 49 0D 2F 1A EB 60 E5 25 C1 E8 62 0C
6C 7D AB 9F 39 06 69 7E D2 64 38 FB FA DA 19 E1
EF 01 61 71 7D 8D 6A 55 35 06 94 4E AF DF FE C4
B3 84 EE BE EA 8E 95 A5 33 6B 4B 55 83 39 3D 9E
```

Fig. 26. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CBC yang diganti.

3) Cipher Feedback *(CFB)*

```
34 9A AE 8D 4A 05 D7 FB 0A 9F 64 55 17 3A 1B DF
56 8C 67 4C AA 6C BF 23 08 60 1B 1E F5 81 59 17
68 E6 C6 3C B6 65 B9 DC 04 38 57 2C 1C 43 49 CB
B3 B9 EC 24 F8 E9 CA 12 14 C5 77 AC E1 A5 EF 70
0A 59 D9 9C A6 CF 7C A2 3E A5 C1 44 79 BA 35 C5
06 63 A8 0B BB C3 D7 30 1D 76 86 FB 94 0C 4E 15
00 34 14 FA 64 8A DC 41 01 28 59 AC 36 D9 3B A2
81 05 87 4B EB DF C3 AF BA 2A 2D 74 E0 99 46 99
```

Fig. 27. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CFB semula

```
06 10 4B 11 FB 1A 5D 40 F2 1A 98 F2 13 AD AE 83
07 EE D9 69 3D D5 36 F6 CD 21 1B 83 D5 5D 64 76
0E 59 D6 28 D1 94 8D 7F 3A D7 57 F6 7A EF 6F B8
1A CC AD 60 11 A3 12 6E F5 EC A7 52 28 AF 31 60
09 D7 FD 25 BD D1 90 E7 FF BF 39 F7 5D 3A A0 C7
2D CA E8 F3 6F 69 D3 C4 34 F6 C5 0E 33 94 DC 0C
6B 81 06 77 42 AC B9 44 64 6B AA 3E 35 31 BC 81
31 20 EB CB E4 1F BA 30 FA 3A C5 B8 9F 82 F1 8C
```

Fig. 28. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CFB yang diganti. Bagian yang diganti ditandakan merah.

4) Output Feedback *(OFB)*

```
34 9A AE 8D 4A 05 D7 FB 0A 9F 64 55 17 3A 1B DF
56 8C 67 4C AA 6C BF 23 08 60 1B 1E F5 81 59 17
D8 39 8B 74 55 E7 18 EA F0 25 EA 12 90 B4 27 30
F9 49 75 59 8B 4A F8 5B CA 6C 15 F4 81 3B 2A 3A
D9 89 EC 5B F3 7C B1 71 45 4F 5A 9B C3 84 16 A7
38 62 2D D3 AD C4 6C 99 AE 4B 3C 2F 77 FD 16 CD
3B 4B D1 1F 92 B6 0A A2 58 1D F9 B3 33 AA D3 89
60 CC 05 48 4F DF 1F A5 B2 3C 42 C2 F0 22 05 89
```

Fig. 29. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CFB semula

```
06 10 4B 11 FB 1A 5D 40 F2 1A 98 F2 13 AD AE 83
07 EE D9 69 3D D5 36 F6 CD 21 1B 83 D5 5D 64 76
99 62 D3 09 17 C8 0A AB F5 E1 72 AE C4 41 E8 0A
36 CD BE BD BE 24 36 CF 10 51 0F A5 B3 EF 5A 17
5A 7F A8 E5 43 33 B0 49 8F 79 31 B6 88 33 37 10
B9 73 8E E7 D7 8C 3F 6B E7 C8 EF 51 BC D0 5C 5B
1A F4 67 C3 5F C1 5F 92 EC 07 EB DD 28 C0 9F 15
E4 81 77 8F 24 BA F6 A8 DD 93 F8 EA 85 C7 63 3A
```

Fig. 30. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode CFB yang diganti. Bagian yang diganti ditandakan merah.

5) Counter *(CTR)*

Counter diinisalisasi dengan nilai 0.

```
34 9A AE 8D 4A 05 D7 FB 0A 9F 64 55 17 3A 1B DF
56 8C 67 4C AA 6C BF 23 08 60 1B 1E F5 81 59 17
7E F4 44 3D DC C8 4A 06 24 62 55 97 42 12 B3 01
07 6A 1A A8 C2 90 49 77 73 BC D2 93 98 4F EC 97
D3 EE 94 8A BC AB 84 AC 2D FF 65 03 A8 AD 0F 55
E4 D3 E9 24 BD 44 65 E4 A4 83 9B 13 E3 A6 2C A8
F5 08 9F B1 29 24 38 A2 63 11 FE 81 B2 29 54 99
33 DA 46 47 74 E0 44 E8 7D 68 57 69 6A 47 DB 4F
```

Fig. 31. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode *counter* semula

```
06 10 4B 11 FB 1A 5D 40 F2 1A 98 F2 13 AD AE 83
07 EE D9 69 3D D5 36 F6 CD 21 1B 83 D5 5D 64 76
C7 0A 4B 28 A7 C7 E5 FB E0 F9 DA 54 B7 C8 A3 CF
7A DF 45 89 86 FE 13 D7 5E A3 D4 7E 0D 5E EF 20
B9 CA BC DC 4F AB B1 7F 61 DA 37 0A 85 19 48 F5
2C 0B 26 A1 67 A4 31 68 89 50 2F 3A 7F 9A CC 5E
2C 53 CD F0 10 18 7A 56 1D 7F 0A A8 59 3D AF C6
02 EF 9A 16 AA 91 EC 95 6F 71 BE 8E E8 96 A0 48
```

Fig. 32. Representasi *hexadecimal* 128 *byte* dari *ciphertext* mode *counter* yang diganti. Bagian yang diganti ditandakan merah.

Pada setiap mode enkripsi, hampir tidak ada *byte* yang sama antara kedua *ciphertext*. Dengan demikian, perubahan 1 bit pada kunci menghasilkan perubahan yang signifikan pada *ciphertext* hasil enkripsi.

VI. Analisis Keamanan

Pada bagian ini, akan dibahas empat tipe serangan yang umum dilakukan pada sebuah *block cipher*: serangan *brute force*, analisis frekuensi, kriptanalisis linear, dan kriptanalisis diferensial. Setelah itu, akan ditinjau apakah Ruadan memenuhi prinsip *confusion* dan *diffusion*.

A. *Serangan* Brute Force

Serangan *brute force* merupakan metode memecahkan enkripsi *ciphertext* dengan menebak dan mencoba kunci satu persatu hingga didapat hasil yang diinginkan. Serangan *brute force* menggunakan mode *exhaustive search*. Dengan metode ini, kunci pasti akan ditemukan, tetapi waktu yang dibutuhkan untuk menemukan kunci akan sangat lama dan meningkat secara eksponensial seiring pertambahan panjang kunci.

Pada Ruadan, kunci yang digunakan memiliki panjang 16 *byte*, atau 128 bit. Kemungkinan kunci yang dapat dibentuk dari 128 bit adalah 2^{128}. Dengan asumsi serangan dijalankan oleh mesin yang mampu menguji 10^6 kunci per detik, waktu yang diperlukan untuk menguji seluruh kemungkinan kunci satu per satu adalah 3.4×10^{32} detik, setara dengan 3.93×10^{27} hari atau 1.075×10^{25} tahun.

Dapat dilihat bahwa biaya yang dibutuhkan untuk memecahkan 128 bit kunci sangat mahal, baik waktu maupun sumber daya lain. Dengan demikian, dapat disimpulkan bahwa penggunaan kunci 128 bit memberikan keamanan yang cukup bagi Ruadan.

1) Cipher Block Chaining *(CBC)*

Penggunaan *initialization vector* pada CBC menambah kompleksitas lebih untuk serangan *brute force*. *Initialization vector* memiliki panjang yang sama dengan panjang blok yang masuk, yaitu 32 *byte* atau 256 bit.

Jika *initialization vector* dirahasiakan, terdapat 2^{256} kemungkinan yang harus dicoba. Bersama dengan kunci, terdapat 2^{384} kemungkinan yang harus dicoba. Dengan asumsi serangan dijalankan oleh mesin yang mampu menguji 10^6 kunci per detik, waktu yang diperlukan untuk menguji seluruh kemungkinan kunci dan seluruh kemungkinan *initialization vector* satu per satu adalah 3.93×10^{97} detik, setara dengan 6.069×10^{82} hari atau 1.242×10^{75} tahun.

B. *Analisis Frekuensi*

Analisis frekuensi merupakan teknik memecahkan enkripsi *ciphertext* dengan memperhatikan frekuensi kemunculan huruf, bigram, atau kelompok huruf yang sering muncul pada *plaintext*, lalu menarik korelasi pemetaan *plaintext* ke *ciphertext* berdasarkan frekuensi tersebut. Sebagai contoh, pada bahasa Inggris, huruf yang sering muncul adalah E, T, A, dan O, sedangkan huruf yang jarang muncul adalah Z, Q, dan X.

Untuk *ciphertext* biner, frekuensi yang dianalisis merupakan frekuensi setiap *byte*. Hasil eksperimen pada Bab V-A memperlihatkan bahwa persebaran *byte* pada *ciphertext* lebih merata dibandingkan pada *plaintext*.

Pada *plaintext*, *byte* yang paling sering muncul adalah 00 sebagai *padding*. Frekuensi *byte* sisanya tergantung variasi warna setiap *pixel* yang ada pada gambar, di mana gambar yang dominan satu warna dan tingkat keterangan akan memiliki persebaran yang lebih tidak merata dibandingkan gambar yang bervariasi.

Pada *ciphertext* hasil enkripsi kelima mode, tidak ada *byte* yang benar-benar menonjol, sehingga pemetaan *byte* pada *plaintext* ke *ciphertext* tidak mungkin dilakukan. Ini disebabkan dua *byte* yang sama akan dienkripsi menjadi dua *byte* yang berbeda, tergantung posisinya dalam teks dan mode enkripsi yang digunakan.

Berdasarkan analisis tersebut, dapat disimpulkan bahwa serangan analisis frekuensi sulit diterapkan pada algoritma Ruadan. Serangan analisis frekuensi cocok untuk diterapkan pada kasus jika pemetaan *plaintext* ke *ciphertext* pasti.

C. *Kriptanalisis Linear*

Kriptanalisis linear merupakan teknik pemecahan enkripsi dengan menemukan suatu persamaan linear yang dapat memodelkan apa yang dilakukan oleh algoritma sandi (memiliki bias tinggi). Persamaan yang dihasilkan mendeskripsikan hubungan antara *ciphertext*, *plaintext*, dan kunci.

Persamaan linear dalam konteks kriptanalisis linear adalah persamaan yang terdiri atas variabel biner dan operator XOR. Suatu persamaan linear dianggap memiliki bias tinggi jika probabilitas kebenarannya jauh melebihi 0.5.

Salah satu langkah dalam kriptanalisis linear untuk melakukan serangan terhadap suatu *block cipher* adalah analisis S-box. S-box dalam *block cipher* menghasilkan sifat nonlinear pada *block cipher* tersebut.

S-box yang digunakan Ruadan dibangkitkan pada saat awal enkripsi. Terdapat 256 kemungkinan S-box yang dibangkitkan, satu untuk masing-masing *byte* kemungkinan peubah.

Misalkan persamaan linear yang digunakan untuk memodelkan S-box adalah

$$aX_1 \oplus bX_2 \oplus cX_3 \oplus dX_4 = eY_1 \oplus fY_2 \oplus gY_3 \oplus hY_4 \qquad (13)$$

di mana abcd dan efgh adalah *bitmask* untuk masukan X dan keluaran Y. Misalkan, jika abcd = 1001 dan efgh = 0110, persamaan linear yang dianalisis adalah

$$X_1 \oplus X_4 = Y_2 \oplus Y_3 \qquad (14)$$

Untuk setiap S-box, terdapat 256 × 256 persamaan linear yang harus dianalisis, sehingga untuk suatu kunci tertentu terdapat $256^3 = 2^{24}$ kombinasi persamaan linear yang harus dianalisis.

Di samping S-box, kunci juga berpengaruh pada aspek-aspek lain, seperti jumlah putaran, yang harus dianalisis. Semua ini menyebabkan kriptanalisis linear sulit dilakukan untuk Ruadan.

D. *Kriptanalisis Diferensial*

Kriptanalisis diferensial menyerang suatu sandi dengan mencari *differential output* yang sering muncul berdasarkan *differential input* tertentu. *Differential input* ΔX didefinisikan sebagai

$$\Delta X_i = X_i' \oplus X_i'' \qquad (15)$$

di mana X_i' dan X_i'' adalah bit ke-i dari dua masukan *plaintext* berbeda, sedangkan ΔX_i adalah selisih dari kedua bit tersebut. *Differential output* ΔY juga didefinisikan serupa, yaitu

$$\Delta Y_i = Y_i' \oplus Y_i'' \qquad (16)$$

di mana Y_i' dan Y_i'' adalah bit ke-i dari dua masukan *ciphertext* berbeda, sedangkan ΔY_i adalah selisih dari kedua bit tersebut.

Secara teoritis, untuk *differential input* sembarang, probabilitas suatu *differential output* adalah $\frac{1}{2^n}$ [5]. Jika demikian, usaha yang diperlukan untuk kriptanalisis diferensial sama dengan usaha yang dilakukan untuk serangan *brute force.* Meskipun demikian, pada kenyataannya terdapat pasangan *differential input* dan *output* tertentu yang memiliki probabilitas kemunculan yang relatif tinggi. Probabilitas ini yang dimanfaatkan untuk kriptanalisis diferensial.

Salah satu aspek yang dapat dianalisis pada kriptanalisis diferensial adalah S-box. Pada S-box hasil pembangkitan, terdapat 256 *differential input* yang perlu dianalisis, kemudian setiap *differential input* tersebut menghasilkan 256 *differential output*.

S-box yang digunakan pada Ruadan didasarkan pada S-box Rijndael. S-box ini memiliki nilai probabilitas diferensial sebagian besar pasangan $\frac{2}{256}$, sedangkan probabilitas diferensial maksimum $\frac{4}{256}$ [9], yang berarti terdapat minimal satu pasang *differential input/output* (ΔX, ΔY) yang memiliki frekuensi kemunculan 4 pada S-box tersebut. Karena terdapat 256 kemungkinan S-box yang digunakan Ruadan, terhadap 256 S-box tersebut dilakukan kalkulasi maksimum probabilitas diferensial dengan persamaan

$$DP^S(\Delta X \rightarrow \Delta Y) = \left(\frac{\#\{x \in X \mid S(x) \oplus S(x \oplus \Delta X) = \Delta Y\}}{2^m}\right) \qquad (17)$$

Dengan nilai maksimum probabilitas diferensial $\frac{4}{256}$ dan perlunya analisis aspek-aspek lain (jumlah putaran, pergeseran, dll.), *differential cryptanalisis* sulit dilakukan untuk Ruadan.

E. Confusion dan Diffusion

Prinsip *confusion* dari Ruadan dapat dilihat dari pembahasan mengenai analisis frekuensi, kriptanalisis linear, dan kriptanalisis diferensial. Analisis frekuensi tidak mungkin dilakukan karena *byte* yang sama dapat dienkripsi menjadi *byte* yang berbeda, sehingga keterkaitan antara *plaintext* dan *ciphertext* tidak dapat diamati. Kriptanalisis linear secara teori mungkin dilakukan, tetapi jumlah persamaan yang harus dianalisis hanya dari satu aspek Ruadan terlalu banyak untuk dikerjakan dalam waktu yang rasional. Untuk kriptanalisis diferernsial, probabilitas diferensial maksimum yang rendah

menyebabkan kriptanalisis sulit dilakukan. Dengan demikian, nilai *confusion* dari Ruadan dapat dikatakan cukup.

Prinsip *diffusion* dari Ruadan dapat dilihat dari eksperimen perubahan kunci, di mana perubahan kunci 1 bit menghasilkan perubahan drastis pada *ciphertext*. Dengan demikian, nilai *diffusion* dari Ruadan dapat dikatakan cukup.

VII. Kesimpulan dan Saran

Peningkatan kemampuan komputasi perangkat keras mengakibatkan tingkat keamanan yang tinggi menjadi suatu hal yang sangat krusial. Ruadan menggunakan struktur Feistel dan S-box yang tergantung kunci untuk meningkatkan tingkat keamanan tersebut. Eksperimen dan analisis keamanan telah menunjukkan bahwa Ruadan memegang prinsip *confusion* dan *diffusion*. Berdasarkan analisis yang dilakukan, algoritma ini memiliki keamanan yang baik sehingga bisa menjadi alternatif dalam enkripsi *block cipher*.

Untuk pengembangan selanjutnya, algoritma ini dapat dimodifikasi untuk mendukung besar blok yang bervariasi, seperti 192 bit. Selain itu, struktur Feistel juga dapat dimodifikasi, misalnya dengan pembagian blok kiri dan kanan yang tidak sama panjang.

Referensi

[1] A. J. Menezes, P. C. van Oorshot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton: CRC Press, 2001.

[2] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 2nd ed. Indianapolis, IN: Wiley, 2015.

[3] W. Stallings, *Cryptography and network security: principles and practice*. Boston: Pearson Prentice Hall, 2017.

[4] R. Munir, *Diktat Kuliash IF5054 Kriptografi*, Bandung: Departemen Teknik Informatika Institut Teknologi Bandung, 2005.

[5] B. Schneier dkk. *Twofish: A 128-Bit Block Cipher*. Submisi AES, https://www.schneier.com/academic/paperfiles/paper-twofish-paper.pdf. 1998. [Diakses 25-Feb-2018].

[6] J. Daemen dan V. Rijmen. *AES Proposal: Rijndael*. Submisi AES, https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf. 1998. [Diakses 25-Feb-2018].

[7] N. Stoianov, *A new approach of generating key-dependent SBOXes in AES*: STO Information Systems and Technology Panel (IST) Symposium, 24–25 September 2001, Koblenz, Jerman. Tersedia: NATO STO, https://www.sto.nato.int/publications. [Diakses 25-Feb-2018].

[8] H. Heys, "A tutorial on linear and differential cryptanalysis": Technical Report CORR 2001-17, Centre for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, Kanada, Mar-2001. Tersedia: https://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf. [Diakses 16-Maret-2018].

[9] E. Rose, "Is the difference distribution table of AES S-box uniform?," *Cryptography Stack Exchange*, 11-Jul-2016. [Daring]. Tersedia: https://crypto.stackexchange.com/questions/37659/is-the-difference-distribution-table-of-aes-s-box-uniform. [Diakses 16-Mar-2018].