RYPYTH: Algoritma Blok Cipher dengan Pythagoras

Radiyya Dwisaputra-13515023[1] Rahmad Yesa Surya-13515088[2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha Nomor 10, Bandung
13515023@std.stei.itb.ac.id[1] 13515088@std.stei.itb.ac.id[2]

Abstrak — Pada saat ini, algoritma kriptografi modern menawarkan enkripsi yang lebih aman dibandingkan algoritma kriptografi klasik. Hal ini dibuktikan pada beberapa algoritma kriptografi klasik yang sudah dapat dipecahkan. Salah satu contoh algoritma kriptografi modern adalah *cipher* blok yang melakukan enkripsi/dekripsi dalam blok-blok pesan dengan ukuran tertentu. *Cipher* blok juga memiliki kelebihan karena pada umumnya mengimplementasikan jaringan Feistel sehingga proses enkripsi/dekripsi dilakukan dengan proses yang hampir sama. Pada makalah ini, diusulkan sebuah algoritma *cipher* blok baru bernama RYPYTH yang mengimplementasikan jaringan Feistel. RYPYTH didesain untuk memenuhi operasi-operasi dasar substitusi, transposisi, dan pembangkitan *round key* menggunakan deret Fibonacci dan Teorema Pythagoras. Hasil enkripsi RYPYTH membuktikan bahwa RYPYTH memiliki sifat *confusion* dan *diffusion* yang tinggi.

Kata kunci — Fibonacci, Pythagoras, enkripsi, *cipher* blok

I. PENDAHULUAN

Tindakan untuk merahasiakan pesan yang dikirimkan melalui media komunikasi telah dipraktikkan sejak lama, bahkan sebelum komputer ditemukan. Praktik untuk membuat pesan menjadi rahasia dilakukan dengan mengubah teks asli pesan atau disebut dengan *plaintext* menjadi teks yang tidak memiliki makna sama sekali ketika dibaca atau disebut dengan *ciphertext*. Pesan yang sudah dienkripsi menjadi *ciphertext* tidak akan diketahui dengan mudah oleh pihak yang tidak memiliki sebuah alat bantu untuk mendekripsi pesan tersebut, yakni kunci. Kunci hanya diketahui oleh pengirim dan penerima pesan saja, yang berguna untuk mengenkripsi pesan menjadi *ciphertext* bagi pengirim, dan untuk mendekripsi pesan menjadi *plaintext* bagi penerima. Prinsip dasar inilah yang menjadi landasan dari kriptografi.

Dalam sejarah kriptografi, teknik untuk merahasiakan pesan telah dilakukan sejak era sebelum Masehi. Pada tahun 400 - 200 SM di India, *Mlecchita vikalpa* atau seni untuk memahami penulisan *cipher* dan penulisan pesan dengan cara yang janggal telah didokumentasikan sebagai cara untuk berkomunikasi. Di kemudian hari teknik ini diketahui sebagai bentuk dari *cipher* substitusi yang sederhana[1]. Penggunaan *cipher*, yang juga *cipher* substitusi, terdapat pada Caesar *cipher*, yang mana setiap huruf pada *plaintext* disubstitusi dengan sebuah huruf yang memiliki jarak tertentu di alfabet. Dua contoh ini merupakan bagian dari kriptografi klasik.

Walaupun sebuah pesan telah dienkripsi menjadi *ciphertext*, bukan berarti pihak yang tidak memiliki kunci (penyerang) tidak dapat mengetahui isi dari pesan tersebut. Kejadian tersebut mungkin terjadi, namun dengan derajat kesulitan yang berbeda-beda. Derajat kesulitan untuk memecahkan *ciphertext* tanpa kunci bergantung pada teknik enkripsi pesan yang digunakan oleh pengirim pesan. Teknik enkripsi pada kriptografi klasik, seperti pada Caesar *cipher,* dengan seiring berkembangnya ilmu kriptografi, dapat dipecahkan dengan relatif mudah. Caesar *cipher* dapat dipecahkan dengan *exhaustive key search* karena jumlah kuncinya sangat sedikit, yakni hanya 26[2]. Kemudian ada contoh lain yakni Vigenere *cipher*, yang ternyata juga sudah dapat dipecahkan oleh Babbage dan Kasiski[3]. Sebagian besar teknik enkripsi pada kriptografi klasik dapat dengan mudah dipecahkan. Hal ini menjadi motivasi lahirnya teknik-teknik enkripsi yang jauh lebih baik pada kriptografi modern. Prinsip untuk mengenkripsi pesan pada algoritma kriptografi modern pada dasarnya memiliki kemiripan dengan algoritma kriptografi klasik, yakni menggunakan substitusi dan transposisi, namun dengan derajat kerumitan yang lebih tinggi sehingga sulit dipecahkan. Algoritma kriptografi modern beroperasi pada level bit dan seringkali menggunakan operasi biner XOR[4], salah satunya adalah *cipher* blok.

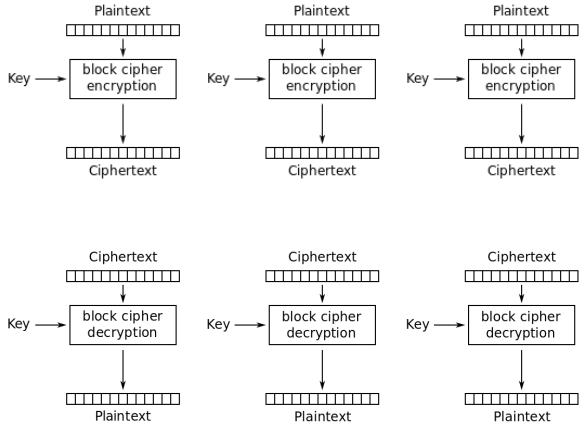
Pada makalah ini, diusulkan sebuah algoritma *cipher* blok bernama RYPYTH yang menggunakan fitur substitusi dan transposisi berdasarkan deret Fibonacci dan teorema Pythagoras. Algoritma ini menerapkan prinsip Feistel, sehingga mengandalkan operasi biner XOR dalam proses enkripsi maupun dekripsi. RYPYTH didesain untuk sulit dipecahkan dengan menerapkan prinsip *confusion* dan *diffusion* Shannon. Selain itu, RYPYTH juga menerapkan prinsip substitusi, transposisi, *iterated cipher*, dan pembangkitan *round-key*.

II. DASAR TEORI

A. *Cipher* Blok

Cipher blok merupakan teknik enkripsi pesan yang beroperasi dengan satuan blok dan level bit. Pesan yang merupakan kumpulan karakter akan diubah terlebih dahulu menjadi kumpulan *bit*. Sebagai contoh pesan ABC akan diubah menjadi 010000010100001001000011. Kumpulan *bit* tersebut akan dibagi menjadi blok-blok dengan ukuran tertentu, yang ditentukan oleh pengirim pesan di awal. Ukuran blok dapat berupa 64-bit, 128-bit, atau 256-bit.

Blok-blok tersebut kemudian akan dienkripsi dengan kunci yang telah ditentukan oleh pengirim pesan. Pada dasarnya, panjang kunci sama dengan panjang blok, sehingga sebuah blok akan dienkripsi dengan kunci yang sama dengan blok yang lain. Hal yang sama juga akan dilakukan ketika proses dekripsi.



Gambar 1. Proses enkripsi/ dekripsi *cipher* blok

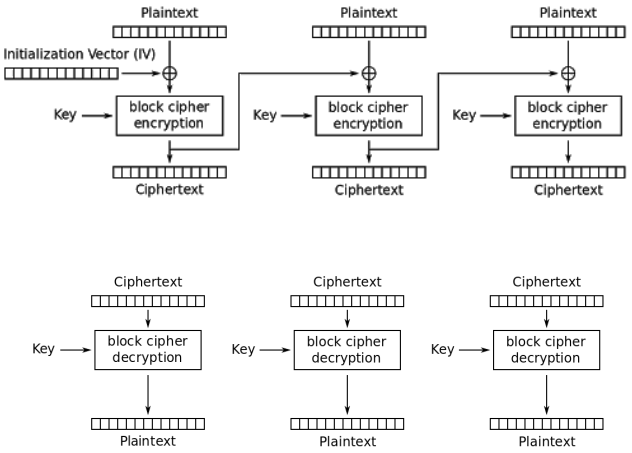
Cipher blok dapat beroperasi dengan lima mode berbeda, yakni:

1. *Electronic Codebook* (ECB)
   ECB merupakan mode enkripsi yang paling sederhana daripada *cipher* blok. Pada mode ini, pesan dibagi menjadi blok-blok dengan ukuran panjang blok tertentu. Kemudian, setiap blok dienkripsi secara independen dengan kunci yang telah ditentukan. Proses enkripsi/dekripsi pada ECB dapat dilihat pada Gambar 1.

   Mode ECB membuat setiap blok independen dengan yang lain, artinya tidak ada ketergantungan antara satu blok dengan yang lain. Sebagai contoh, blok *plaintext* yang sama akan dienkripsi menjadi blok *ciphertext* yang juga sama. Hal inilah yang memunculkan kelemahan dari mode ECB, yakni kurangnya *diffusion* yang ada pada *ciphertext*.

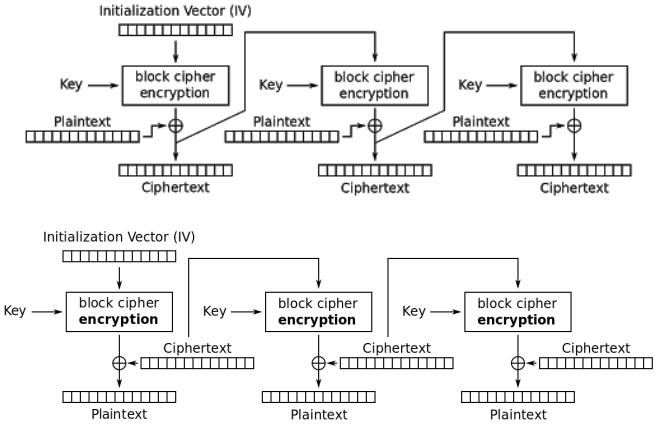
2. *Cipher Block Chaining* (CBC)
   Mode CBC ditemukan oleh Ehrsam, Meyer, dan Tuchman pada 1976[5]. Pada mode ini, pesan dibagi menjadi blok-blok dengan ukuran panjang blok tertentu. Kemudian, dijalankan operasi biner XOR terhadap *plaintext* dengan *ciphertext* blok sebelumnya sebelum *plaintext* tersebut dienkripsi. *Plaintext* blok pertama akan di-XOR dengan sebuah *initialization vector*.

   Mode CBC membuat setiap blok bergantung dengan blok sebelumnya. Hal ini mengakibat sifat *diffusion* yang lebih baik dibandingkan dengan mode ECB. Kelemahan dari mode CBC adalah operasi enkripsi/dekripsi perlu dijalankan secara serial, sehingga memerlukan waktu yang lebih lama dibandingkan jika proses enkripsi/dekripsi dijalankan secara paralel.



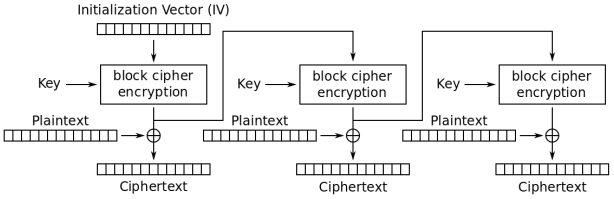
Gambar 2. Proses enkripsi/dekripsi *cipher* blok dengan mode CBC

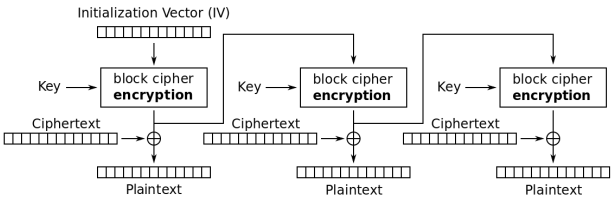
3. *Cipher Feedback* (CFB)
   CFB beroperasi hampir sama dengan CBC dan memiliki sifat *diffusion* yang baik juga karena menciptakan ketergantungan antara blok-blok pesan yang akan dienkripsi. Hal yang membuat CFB berbeda dengan CBC ialah perubahan yang terjadi pada antrian tidak selalu menggunakan cipher melainkan hanya menggantikan 1 byte terakhir saja pada antrian.



Gambar 3. Proses enkripsi/dekripsi *cipher* blok dengan mode CFB

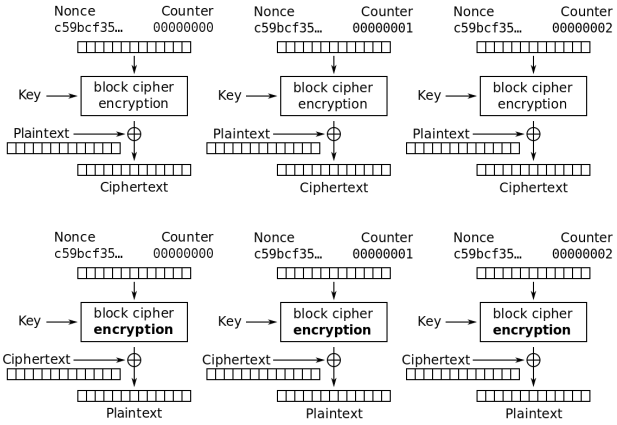
4. *Output Feedback* (OFB)
   OFB mirip dengan CFB yang menggunakan feedback dalam chaining nya tapi pada OFB feedback yang diberikan sedikit berbeda yaitu antrian berikutnya merupakan penambahan 1 byte terdepan dari hasil enkripsi atau dekripsi dari antrian sebelumnya.





Gambar 4. Proses enkripsi/dekripsi *cipher* blok dengan mode OFB

5. *Counter* (CTR)
   Counter sedikit berbeda dibandingkan CFB dan OFB. Counter sendiri lebih mirip dengan ECB hanya saja terdapat sebuah counter. Counter ini bernilai berbeda dari saat tiap block dilakukan enkripsi atau dekripsi. Pada dasarnya counter dapat dijumlahkan 1 untuk memastikan nilai counter selalu berbeda namun dapat juga dilakukan hal lain.



Gambar 5. Proses enkripsi/dekripsi *cipher* blok dengan mode CTR

B. Jaringan Feistel

Jaringan Feistel merupakan *iterated cipher* yang memiliki fungsi internal, disebut sebagai *round function*[6]. Jaringan Feistel ditemukan oleh kriptografer asal Jerman bernama Horst Feistel. Jaringan ini umum digunakan pada *cipher* blok karena memberikan keuntungan, yakni operasi enkripsi dan dekripsi sangat persis bahkan sama pada beberapa kasus. Hal ini dimungkinkan dengan melakukan pembalikan *key schedule*. Sehingga, *cipher* blok dengan jaringan Feistel dapat diimplementasikan dengan lebih mudah.

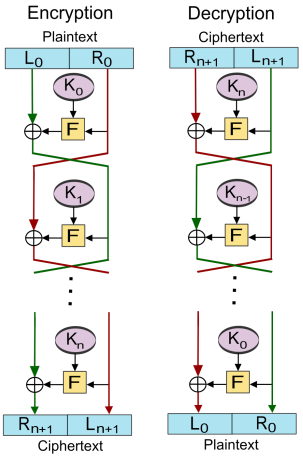
Berikut adalah langkah-langkah operasi enkripsi pada jaringan Feistel.

1. Bagi blok *plaintext* menjadi dua bagian (L_0, R_0)
2. Untuk setiap *round*, lakukan operasi seperti berikut

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

   dimana K_i adalah kunci pada *round* tersebut.
3. Proses tersebut diulangi hingga sejumlah *round* yang telah ditentukan.

Langkah yang sama dilakukan untuk operasi dekripsi, dengan menukar posisi bagian kiri dengan bagian kanan blok *ciphertext*.



Gambar 6. Proses enkripsi/dekripsi pada jaringan Feistel

C. Prinsip *Confusion* dan *Diffusion* Shannon

Dalam makalah yang ia tulis, Claude Elwood Shannon menyebutkan bahwa

1. *Confusion* adalah membuat hubungan antara kunci dengan *ciphertext* sekompleks mungkin;
2. *Diffusion* adalah sifat yang mana redundansi pada statistik dari sebuah *plaintext* tidak tergambarkan pada statistik *ciphertext*[7].

Confusion dapat diperoleh dengan melakukan substitusi, misalnya menggunakan S-BOX seperti yang digunakan pada algoritma enkripsi AES. Kemudian, *diffusion* dapat diperoleh dengan melakukan transposisi atau pergeseran bit yang menyusun blok.

D. Deret Fibonacci dan Teorema Pythagoras

Deret Fibonacci, atau yang dinotasikan dengan F_n merupakan sebuah deret yang mana sebuah bilangan merupakan hasil penjumlahan dari dua bilangan sebelumnya, diawali dari 0 dan 1[8].

$$F_0 = 0, F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

Selanjutnya, Teorema Pythagoras menyatakan bahwa kuadrat dari hipotenusa merupakan jumlah dari kuadrat dari kedua sisi yang lain. Teorema ini dapat ditulis menjadi Persamaan Pythagoras seperti berikut[9].

$$a^2 + b^2 = c^2$$

III. RANCANGAN ALGORITMA

RYPYTH dirancang sebagai *cipher* blok yang menerapkan prinsip jaringan Feistel dan dapat beroperasi kedalam lima mode yang berbeda, yakni ECB, CBC, CFB, OFB, dan CTR. Sebagai jaringan Feistel, RYPYTH menerapkan 16 *round* ketika beroperasi, dengan fungsi internal yang meliputi substitusi dan transposisi. Ukuran panjang blok yang digunakan pada RYPYTH adalah 128 bit.

Dengan demikian, pada jaringan Feistel, ukuran sub blok kiri dan sub blok kanan adalah masing-masing 64 bit. Hal ini berarti bahwa fungsi internal akan memproses bit dan kunci dengan panjang 64 bit.

Fungsi internal RYPYTH didesain untuk memiliki *round* sebanyak 8. Setiap satu *round*, akan dijalankan operasi sebanyak 6 operasi sebagai berikut.

A. Substitusi Bit Pada Indeks Bilangan Non-Fibonacci
Substitusi bit pada bagian ini dilakukan dengan melakukan invers, sehingga bit yang bernilai 1 akan disubstitusi dengan nilai 0, dan sebaliknya. Substitusi tersebut dilakukan pada bit yang berada pada indeks bilangan non-Fibonacci. Seperti yang diketahui, bilangan-bilangan berikut adalah bilangan Fibonacci: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]. Dengan demikian, bit pada indeks bilangan-bilangan tersebut tidak akan disubstitusi.

B. Substitusi Menggunakan S-BOX
S-BOX yang digunakan untuk substitusi adalah Rijndael S-BOX, yang juga digunakan untuk substitusi pada algoritma enkripsi AES. Sebelum dilakukan substitusi, sub blok yang berukuran panjang 64 bit akan dikonversi terlebih dahulu menjadi bilangan heksadesimal. Panjang bit yang direpresentasikan oleh sebuah bilangan heksadesimal adalah 8 bit, sehingga akan terdapat 8 bilangan heksadesimal. Selanjutnya, bilangan heksadesimal tersebut disubstitusi menggunakan S-BOX.

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

Gambar 7. Rijndael S-BOX

C. Transposisi
Transposisi pada RYPYTH dilakukan dengan mekanisme seperti ini.

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & 3 \\ 5 & 2 \end{bmatrix}$$

Dalam susunan 4 bilangan, bilangan ke-1 dan bilangan ke-4 akan bertukar posisi, sedangkan bilangan ke-2 dan ke-3 tetap.

Hasil akhir dari tahap III.B adalah 8 bilangan heksadesimal yang telah disubstitusi. Operasi transposisi seperti yang telah dijelaskan akan diterapkan pada 8 bilangan tersebut. Dengan demikian, bilangan ke-1 akan ditukar dengan bilangan ke-4, kemudian bilangan ke-5 akan ditukar dengan bilangan ke-8. Bilangan selain itu tidak berubah posisi.

D. Pertukaran Bit
Hasil dari operasi tahap III.C adalah 8 bilangan heksadesimal yang telah ditransposisi. Bilangan-bilangan tersebut selanjutnya dikonversi terlebih dahulu kedalam bentuk biner, sehingga diperoleh kembali biner sepanjang 64.

Pertukaran bit pada RYPYTH dilakukan pada satuan biner dengan panjang 5. Bit ke-1 akan ditukar dengan bit ke-4 dan bit ke-2 akan ditukar dengan bit ke-5, sedangkan bit-3 tidak berubah. Dengan demikian, biner 11100 akan berubah menjadi 00111.

Namun demikian, dengan pertukaran bit yang bekerja dengan biner sepanjang 5 bit, dari biner sepanjang 64, akan ada sisa 4 bilangan yang tidak dapat menerapkan operasi pertukaran bit. Hal ini disengaja demikian sehingga terdapat bagian yang tetap seperti semula. 4 bilangan tersebut merupakan bilangan yang terletak di tengah, yakni bilangan ke-31, ke-32, ke-33, dan ke-34.

E. Operasi Biner XOR dengan Kunci
Hasil dari tahap III.D adalah biner dengan panjang 64. Biner tersebut kemudian dipecah menjadi biner-biner kecil dengan panjang 8, sehingga akan didapati 8 biner. Kemudian, kunci dengan panjang 64 akan dipecah menjadi lebih kecil dengan masing-masing berpanjang 8.

Misalkan P adalah hasil tahap III.D, K adalah kunci, Pn adalah P bagian ke-n, dan H adalah hasil tahap III.E, maka:

P = P1 P2 P3 P4 P5 P6 P7 P8
K = K1 K2 K3 K4 K5 K6 K7 K8
H = (P1 XOR K8) (P2 XOR K7) (P3 XOR K6)
(P4 XOR K5) (P5 XOR K4) (P6 XOR K3)
(P7 XOR K2) (P8 XOR K1)

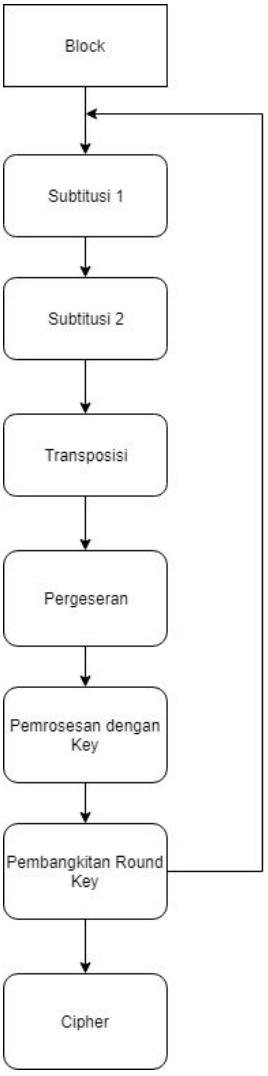
F. Pembangkitan *Round Key*
Pada tahap ini, *round key* dibangkitkan untuk digunakan pada *round* fungsi internal selanjutnya. Pembangkitan ini dilakukan menggunakan Teorema Pythagoras. Hasil dari tahap III.E merupakan biner dengan panjang 64. Biner tersebut kemudian dipecah menjadi biner-biner kecil dengan panjang 8, sehingga akan didapati 8 biner. Kemudian, kunci dengan panjang 64 akan dipecah menjadi lebih kecil dengan masing-masing berpanjang 8.

Pada RYPYTH, didefinisikan sebuah fungsi PYTH(x,y) sebagai berikut.

$$PYTH(x,y) = floor(\sqrt{x^2 + y^2}) \, mod \, 256$$

Misalkan P adalah hasil tahap III.E, K adalah kunci, Pn adalah P bagian ke-n, dan H adalah hasil tahap III.F, maka:

P = P1 P2 P3 P4 P5 P6 P7 P8
K = K1 K2 K3 K4 K5 K6 K7 K8
H = PYTH(P1,K1) PYTH(P2,K2) PYTH(P3,K3) PYTH(P4,K4) PYTH(P5,K5) PYTH(P6,K6) PYTH(P7,K7) PYTH(P8,K8)



Gambar 7. Urutan tahap-tahap fungsi internal pada RYPYTH

IV. HASIL PENGUJIAN DAN ANALISIS

A. Pengujian

Kode sumber program dapat diakses pada : https://github.com/radiyyadwi/utskripto

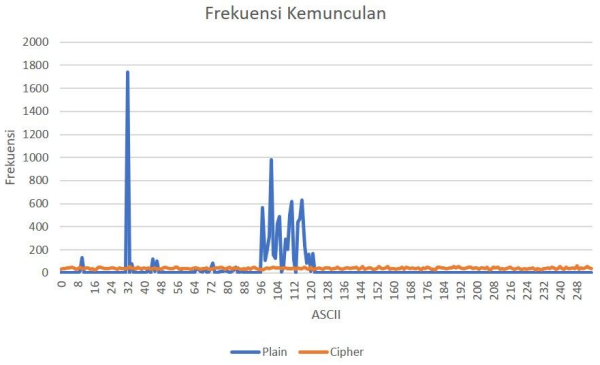
Pengujian pada RYPYTH dilakukan dengan metode-metode berikut : ECB, CBC, CFB, OFB, counter. Dalam pengujian kali ini hal yang dilakukan ialah pengukuran waktu eksekusi baik pada enkripsi atau dekripsi pada setiap metode. Pengukuran dilakukan dengan menggunakan file masukan dengan dua ukuran berbeda yaitu 100 bytes dan 10.161 bytes. Dalam pengujian ini juga dipastikan bahwa hasil enkripsi dan dekripsi selalu benar. Pengukuran waktu eksekusi digambarkan pada gambar berikut (dalam second).

Mode	Operasi	10KB	100B
ECB	enkripsi	11.42306	0.129906
	dekripsi	10.39629	0.105938
CBC	enkripsi	10.20463	0.115935
	dekripsi	9.579835	0.101921
CFB	enkripsi	154.6584	1.552089
	dekripsi	181.4132	1.602083
OFB	enkripsi	173.3762	1.626067
	dekripsi	139.1971	1.628068
counter	enkripsi	8.651096	0.12091
	dekripsi	8.763969	0.105917

Gambar 8. Waktu eksekusi RYPYTH algorithm

B. Analisis

Analisis yang dilakukan pertama ialah analisis *confusion* dan *diffusion* pada RYPYTH. Analisis *confusion* dilakukan dengan menganalisa frekuensi kemunculan ascii sebelum dilakukan enkripsi dibandingkan dengan setelah dilakukan enkripsi. Analisis berikut dilakukan dengan file masukan berukuran 10.161 bytes. Frekuensi kemunculan sebelum dan sesudah dilakukan enkripsi digambarkan pada grafik berikut.



Gambar 9. Grafik frekuensi kemunculan pada *plaintext* dan *ciphertext*

Sesuai hasil frekuensi kemunculan yang telah dilakukan, maka prinsip *confusion* telah dijalankan dengan baik pada RYPYTH. Pada analisis *diffusion* dilakukan sebuah perbandingan hasil enkripsi apabila dilakukan perubahan pada satu huruf. Analisis ini dilakukan dengan menggunakan file masukan berukuran 100 bytes dengan menggunakan metode CFB. Berikut hasil perbandingan eksekusi yang dilakukan.

Masukan: *The Project Gutenberg EBook of The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle 15 in o*
Luaran: **&}s-****C;4**!2**13****'** 8<***%*******, ****Σ**w**c**®_*?***7Nz**!** --- ******b**d** **Hy4F**

Masukan:
THe Project Gutenberg EBook of The Adventures of Sherlock Holmes by Sir Arthur Conan Doyle 15 in o

Luaran:
$◆J_◆s-◆6W◆l◆C;3◆◆◆◆◆5◆و◆η&ى
◆◆◆◆◆◆◆◆◆◆j◆8,◆◆3□◆w/g.◆®F◆◆◆◆7◆eH◆!◆◆

◆◆◆<◆a◆

◆d◆◆Y◆-@y4F◆

Setelah dilakukan perhitungan ketika dilakukan perubahan 1 byte yakni huruf h menjadi huruf H terjadi perubahan *ciphertext* yakni sebanyak 55 byte dibandingkan dengan *ciphertext* sebelumnya Hal ini menunjukkan *diffusion* berjalan pada RYPYTH.
Analisis *bruteforce* attack pada rypyth ialah dengan menghitung waktu yang dibutuhkan untuk mendapatkan kunci dengan menggunakan *bruteforce*. Kunci yang kami gunakan memiliki panjang yang bebas, yang pada saat enkripsi atau dekripsi akan dilakukan sebuah pemrosesan agar panjang kunci menjadi 64 bit yang membuat
kunci 1 : “abcdefgh”
kunci 2 : “abcdefhij”
kunci 3 : “abc”
ketiga kunci diatas tidak akan menghasilkan kunci untuk enkripsi dan dekripsi yang sama. Oleh karena itu semakin panjang kunci yang digunakan maka akan semakin sulit untuk *bruteforce* dilakukan.

V. KESIMPULAN

RYPYTH merupakan algoritma blok *cipher* yang aman karena dilakukan pemrosesan kunci sedemikian rupa sehingga perubahan satu huruf pada kunci dapat menyebabkan enkripsi dan dekripsi menggunakan kunci yang jauh berbeda. Algoritma block cipher ini juga menggunakan prinsip *confusion diffusion* hal ini ditunjukkan dengan analisis frekuensi kemunculan ascii dan efek perubahan *ciphertext* saat dilakukan perubahan pada plaintext..Selain itu RYPYTH juga merupakan algoritma blok *cipher* yang unik karena dalam pengimplementasian nya menggunakan pythagoras untuk membangkitkan round key.

REFERENSI

[1] Kahn, David, The Codebreakers. Simon and Schuster, p. 74, December 1996.

[2] Munir, Rinaldi. Algoritma Kriptografi Klasik (Bagian I). Slide Presentasi Kuliah IF4020, p. 12. Diakses pada 12 Maret 2019.

[3] Munir, Rinaldi. Algoritma Kriptografi Klasik (Bagian II). Slide Presentasi Kuliah IF4020, p. 3. Diakses pada 12 Maret 2019.

[4] Munir, Rinaldi. Algoritma Kriptografi Modern. Slide Presentasi Kuliah IF4020, p. 4. Diakses pada 12 Maret 2019.

[5] William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman, Message Verification and Transmission Error Detection by Block Chaining. US Patent 4074066. 1976.

[6] Menezes, Alfred J.; Oorschot, Paul C. van; Vanstone, Scott A. Handbook of Applied Cryptography, 5th ed, p. 251. 2001.

[7] Shannon, Claude. Communication Theory of Secrecy Systems. Bell System Technical Journal, vol. 28, p 656-715. 1949.

[8] Singh, Parmanand. The So-called Fibonacci Numbers in Ancient and Medieval India. Historia Mathematica, p. 229-244. 1985.

[9] Judith D. Sally; Paul Sally. Roots to Research: A Vertical Development of Mathematical Problems. American Mathematical Society Bookstore, p. 63. 2007.

PERNYATAAN

Dengan ini kami menyatakan bahwa makalah yang kami tulis ini adalah tulisan kami sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 13 Maret 2019

Radiyya Dwisaputra
(13515023)

Rahmad Yesa Surya
(13515088)