SwazzyCipher - *Block Cipher* Baru

Rahmat Rafid Akbar - 13520090
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520090@std.stei.itb.ac.id

M. Akyas David Al Aleey - 13520011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520011@std.stei.itb.ac.id

Stefanus Jeremy Aslan - 13519175
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519175@std.stei.itb.ac.id

***Abstract*—Algoritma *cipher* penting untuk terus dikembangkan untuk menjaga keamanan pesan, terutama di era kriptografi modern dengan banyak teknologi yang membuka berbagai kesempatan untuk penyerangan privasi. Salah satu model cipher terus dikembangkan yaitu *block cipher.* Untuk menghasilkan algoritma *block cipher* yang berkualitas, perlu diperhatikan penerapan konsep-konsep panutan *block cipher*. Pada makalah ini, dikembangkan sebuah algoritma *block cipher* bernama SwazzyCipher yang memperhatikan baik konsep-konsep panutan *cipher*, maupun waktu komputasi yang ideal.**

Keywords—Algoritma cipher; kriptografi modern; block cipher; konsep panutan block cipher

I. Pendahuluan

Pesan merupakan objek yang sering kali bersifat rahasia. Oleh sebab itu, penting untuk menjaga kerahasiaan pesan sehingga hanya dapat diakses oleh pihak-pihak yang berwenang. Salah satu ilmu yang menjaga keamanan pesan dari pengaksesan tidak berwenang yaitu kriptografi, ilmu dan seni untuk menjaga keamanan pesan[1]. Di dalam kriptografi, dikenal teknik bernama *cipher*, teknik yang menyandikan pesan yang disebut sebagai *plaintext* dengan mengubahnya menjadi pesan yang tidak dapat dipahami yaitu *ciphertext*, hingga pesan tersebut dikembalikan menjadi *plaintext* menggunakan kunci rahasia.

Kriptografi berkembang seiringan dengan kemajuan teknologi penyimpanan dan pertukaran data. *Block cipher* adalah satu model algoritma *cipher* yang sering digunakan untuk pesan yang disimpan dalam komputer digital. Pada kesempatan ini, penulis merancang dan mengimplementasi *block cipher* baru untuk menghasilkan algoritma yang aman.

II. Studi Pustaka

A. Block Cipher

Kriptografi mengalami modernisasi ketika ditemukannya komputer digital, sedemikian sehingga era kriptografi setelah penemuan tersebut dikenal sebagai kriptografi modern[2]. Dalam komputer digital, data dan informasi disimpan dalam bentuk biner. Oleh sebab itu, algoritma kriptografi modern beroperasi dalam mode bit atau byte. *Cipher* berbasis bit umumnya dikategorikan menjadi dua, yaitu: *stream cipher* dan *block cipher*[2]. Adapun *block cipher* adalah *cipher* yang terlebih dahulu membagi plainteks ke dalam beberapa blok bit atau blok byte dengan ukuran tertentu sebelum dienkripsi.

Dalam mendesain *block cipher*, terdapat beberapa konsep yang umum untuk dipertimbangkan[3]. Konsep-konsep tersebut adalah:

1) Prinsip Confusion dan Diffusion

Prinsip *confusion* dan *diffusion* adalah dua prinsip kriptografi yang dikemukakan oleh Claude Shannon untuk mempersulit kriptanalisis berbasis statistik. Prinsip *confusion* adalah prinsip penyembunyian hubungan statistik yang ada di antara plainteks, ciphertext, dan kunci, sedangkan prinsip *diffusion* adalah prinsip penyebaran pengaruh satu bit plainteks atau kunci ke sebanyak mungkin bit-bit cipherteks[3].

2) Substitusi dan Permutasi

Substitusi dan permutasi merupakan dua operasi yang umum dilakukan dalam enkripsi *block cipher*. Operasi substitusi merupakan operasi penukaran yang menerima input berupa sejumlah bit atau byte berukuran tertentu dan mengeluarkan output berupa sejumlah bit atau byte dengan ukuran lebih besar atau sama dengan jumlah bit atau byte input. Sementara itu, operasi permutasi merupakan operasi pengacakan susunan bit dalam sebuah blok bit sehingga menghasilkan susunan bit baru.

Penggunaan operasi substitusi dalam implementasi algoritma *block cipher* menghasilkan efek *confusion*, sedangkan penggunaan operasi permutasi dalam implementasi algoritma *block cipher* menghasilkan efek *diffusion.*

a) Kotak Substitusi

Kotak substitusi atau *substitution-box,* biasa disingkat menjadi kotak-S dan *S-box*, merupakan metode substitusi yang umum digunakan dalam algoritma *cipher.* Metode substitusi ini dilakukan dengan cara menggunakan kotak substitusi untuk melakukan operasi substitusi *look-up table* pada blok bit atau block byte. Adapun kotak substitusi itu sendiri merupakan tabel yang digunakan sebagai acuan untuk memetakan masukan blok bit atau blok byte input ke keluaran blok bit atau blok byte. Oleh sebab itu, operasi substitusi kotak-S disebut sebagai operasi *look-up*.

3) Cipher Berulang

Cipher berulang adalah *cipher* yang melakukan enkripsi dengan mengeksekusi fungsi putaran berulang kali untuk

mengubah blok *plaintext* menjadi blok *ciphertext*. Perulangan eksekusi fungsi putaran ditujukan untuk menghasilkan efek *diffusion* sehingga ciphertext sulit dipecahkan.

4) Pembangkitan Kunci Putar

Setiap putaran di dalam iterated cipher menggunakan kunci internal yang dinamakan kunci putaran. Kunci putaran dibangkitkan dari kunci eksternal yang diberikan oleh pengguna melalui proses yang dinamakan *key expansion* atau *key scheduling*. Di dalam key expansion dilakukan komputasi yang kompleks untuk menghasilkan sejumlah kunci putaran yang berbeda-beda. Dalam pembangkitan kunci putar, panjang kunci putar tidak selalu sama dengan panjang kunci eksternal.

5) Ukuran Blok dan Kunci

Ukuran blok dan kunci mempengaruhi tingkat keamanan yang dihasilkan enkripsi. Semakin besar ukuran blok, semakin besar efek *confusion* yang diberikan oleh algoritma. Di lain sisi, ukuran kunci yang semakin besar memberikan efek *diffusion* yang juga semakin besar. Akan tetapi, ukuran blok dan kunci yang terlalu besar dapat menyebabkan waktu komputasi enkripsi dan dekripsi yang lebih lama sehingga dalam mendesain algoritma, perlu dipertimbangkan ukuran blok dan ukuran kunci yang ideal.

B. *Avalanche Effect*

Efek longsor atau *Avalanche Effect* merujuk pada tingkat perubahan *ciphertext* ketika masukan diubah secara minimal. Semakin besar perbedaan *ciphertext* yang dihasilkan enkripsi ketika masukan diubah secara minimal, semakin besar efek longsor algoritma *cipher*. Oleh sebab itu, efek longsor dapat digunakan untuk mengukur efek *diffusion* dalam penilaian algoritma *cipher*.

C. *Key Space*

Ruang kunci atau *key space* merujuk pada banyaknya kunci unik yang valid digunakan dalam algoritma *cipher*. Oleh sebab itu, semakin besar *key space,* semakin besar efek *confusion* dalam penilaian algoritma *cipher.*

III. Rancangan Block Cipher

A. *Enkripsi*

Enkripsi *block cipher* menerima masukan *plaintext* dan kunci eksternal, serta mengimplementasi operasi-operasi antara lain: substitusi, permutasi, pembangkitan kunci putar dan cipher berulang.

1) Partisi Plaintext

Pada awal mula algoritma enkripsi, *plaintext* yang berisi string dipartisi menjadi beberapa blok berupa binary string dengan panjang 128 bit.

Sebuah karakter memiliki nilai ASCII binary dengan panjang 8 bit (ada 256 karakter ASCII). Sehingga, untuk membentuk sebuah blok berukuran 128 bit dibutuhkan 16 buah karakter.

Apabila jumlah karakter dari *plaintext* bukan merupakan kelipatan 16, akan dilakukan penambahan *padding* berupa karakter `*space*` agar *plaintext* menjadi kelipatan 16 dan dapat dilakukan proses enkripsi.

Kemudian, setiap blok yang berisi 16 karakter ASCII akan dipetakan menjadi binary string dengan panjang 128 karakter entry berupa `0 atau `1.

2) Pembangkitan Kunci Putar

Pembangkitan kunci putar dilakukan dengan membangkitkan substring dari kunci eksternal. Putaran enkripsi akan dilakukan sebanyak 16 kali, sehingga diperlukan 16 buah kunci putar berbeda.

Pertama-tama, kunci eksternal akan dihash terlebih dahulu dengan fungsi sha256 yang menghasilkan 64 karakter. Hasil hash dipartisi menjadi 16 buah bagian. Setiap bagiannya akan memiliki panjang 4 bit.

Untuk mendapatkan semua kunci putar, mula-mula diinisiasi sebuah nilai x = 0. Lalu dihitung nilai x akhir untuk setiap iterasi (terdapat 16 iterasi) yang nantinya akan dipakai untuk iterasi selanjutnya.

Pada sebuah iterasi (i) dilakukan penjumlahan nilai ASCII setiap karakter dari sebuah partisi kunci eksternal dalam modulus 256 (2^8) dan disimpan dengan nama **partition_sum**. Nilai x diupdate menjadi penjumlahan antara partition_sum dengan nilai x pada iterasi sebelumnya.

$$(partition\_sum \ += \ ASCII(e), \ for\ e\ in\ partition_i)\ mod\ 256$$

Nilai x yang telah perbarui ini akan dikonversi menjadi binary string dengan panjang 8 bit dan disimpan dengan nama **subkey**. Semenjak nilai karakter ASCII hanya 256 atau 2^8, nilai dari subkey akan direpetisi sebanyak 16 kali sehingga subkey memiliki panjang 128 bit.

```
def subkey_generator(external_key):
    # Hash the external key for security, the result has 64 characters
    hashed_key = hash_string(external_key)
    subkey_len = len(hashed_key) // 16  # 64/16 = 4
    subkeys = []
    x = 0
    for i in range(0, len(hashed_key)//subkey_len):
        partition_sum = 0
        for j in range(0, subkey_len):
            partition_sum = (partition_sum + ord(hashed_key[i*subkey_len+j])) % 256
        x = (x + partition_sum) % 256
        subkey = ''.join(format(x, 'b').zfill(8))
        subkeys.append(subkey*16)
    return subkeys
```

3) Cipher Putar

Cipher putar dilakukan dengan mengeksekusi fungsi putar sebanyak 16 kali iterasi untuk setiap blok bit dari hasil partisi *plaintext* dengan memanfaatkan *subkey* yang dibangkitkan dari kunci eksternal. Operasi-operasi yang dilakukan dalam fungsi putar (F_i) adalah:

a) Pertukaran 64bit Blok Bagian Kiri dan Kanan

Fungsi ini digunakan untuk menukarkan potongan awal blok sepanjang 64 bit dengan potongan akhir blok sepanjang 64 bit.

b) XOR

Setelah kedua bagian blok tersebut ditukarkan, kemudian dilakukan operasi XOR antara blok dengan sub-kunci putar yang telah dibangkitkan. Sub-kunci yang digunakan sesuai dengan urutan iterasi dari fungsi putar, yakni terurut dari sub-kunci indeks ke 0 hingga indeks ke 15. Setelah dilakukan operasi XOR, blok bit kemudian disubstraksi.

c) Substraksi

Proses substraksi diawali dengan melakukan partisi pada blok menjadi sub-blok dengan panjang 4 bit. Kemudian, untuk setiap sub-blok tersebut dikonversi menjadi bilangan desimal dan dilakukan pengurangan sejumlah nilai iterasi atau jumlah putaran saat ini dalam modulus 16 (2^4).

$$x = i \ mod \ 16$$

Jika i = 0, maka pengurangan dilakukan untuk :

$$x = 0 \ mod \ 16$$

Hasil substraksi tersebut selanjutnya dikonversi kembali menjadi binary string sepanjang 4 bit. Keseluruhan hasil substraksi dari sub-blok tersebut tidak lupa digabung kembali menjadi binary string sepanjang 128 bit.

d) Substitusi

Operasi substitusi direalisasikan menggunakan sebuah kotak S (S-box) yang mengacu kepada Rijndael S-box yang diterapkan pada algoritma *Advanced Encryption Standard* (AES).

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

Gambar 3.1. Rijndael S-Box

Proses substitusi tersebut dilakukan dengan mempartisi blok menjadi 16 buah sub-blok dengan panjang 8 bit masukan. Untuk setiap sub-blok, akan dilakukan penentuan entry pada S-box yang akan digunakan untuk proses substitusi. Penentuan dilakukan dengan mencari nilai indeks baris dan indeks kolom terlebih dahulu. Indeks baris dari S-Box diambil dari 4 bit pertama dari sub-blok tersebut dan untuk indeks kolom dari S-Box diambil dari 4 bit terakhir dari sub-blok tersebut yang masing-masing kemudian dikonversi kedalam bilangan desimal.

Setelah didapat indeks baris dan kolom, didapat nilai substitusi untuk 8 bit keluaran yang merupakan entry matriks dengan indeks baris dan kolom yang sesuai.

Contoh penerapan substitusi dijelaskan di bawah ini.

Diketahui 8 bit masukan yakni 10010111, maka 4 bit pertama yakni 1001 atau setara dengan bilangan desimal senilai 9 dan 4 bit terakhir yakni 0111 atau setara dengan bilangan desimal senilai 7. Sehingga, diperoleh indeks barisnya yaitu 9 dan indeks kolomnya yaitu 7. Didapat elemen hasil substitusinya adalah bilangan hexadesimal senilai 156. Elemen tersebut kemudian dikonversi kembali menjadi binary string sepanjang 8 bit.

Setelah semua sub-blok selesai disubstitusi, hasil substitusi sub-blok tersebut digabung kembali menjadi 128 bit di akhir.

e) Permutasi (Shifting)

Operasi permutasi dilakukan dengan melakukan pergeseran dari sebuah blok dengan panjang 128 bit. Pergeseran yang dimaksud adalah melakukan *block shifting* dengan cara membentuk blok tersebut sedemikian rupa sehingga blok sepanjang 128 bit seolah-olah menjadi matriks berukuran 8 x 16. Dalam arti lain, blok sepanjang 128 bit tersebut dipartisi secara terurut menjadi 8 buah baris matriks dengan panjang kolom 16 bit. Kemudian, dilakukan *shifting* ke kiri untuk setiap baris dari matriks tersebut sebanyak i dengan i adalah nilai indeks baris dari matriks tersebut.

	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	...	$a_{0,15}$
Geser 1 ←	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	...	$a_{1,15}$
Geser 2 ←	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	...	$a_{2,15}$
	...	...	...	...	...	...
Geser 15 ←	$a_{15,0}$	$a_{15,1}$	$a_{15,2}$	$a_{15,3}$	...	$a_{15,15}$

Gambar 3.2. Ilustrasi *shifting left* pada blok

Shifting ini dilakukan secara siklik, yakni secara siklik entry paling kiri yang bergeser diisi kembali di paling kanan baris. Hasil sub-blok dari matriks yang sudah digeser tersebut kemudian digabung kembali menjadi blok sepanjang 128 bit.

B. Dekripsi

Dekripsi *block cipher* menerima masukan *ciphertext* dan kunci eksternal. Adapun operasi-operasi yang dilakukan adalah:

1) Partisi Ciphertext

Sama halnya dengan algoritma enkripsi, pada algoritma dekripsi *ciphertext* yang berupa string akan dipartisi menjadi beberapa blok dengan ukuran 128 bit.

Sebelum dilakukan partisi blok, akan digenapkan terlebih dahulu panjang dari ciphertext menjadi kelipatan 16. Alasannya sama dengan algoritma enkripsi, yakni dibutuhkan 16 karakter untuk membentuk blok dengan 128 bit. Hal ini

dilakukan dengan menambahkan padding berupa karakter `*space*`.

Kemudian, setiap 16 karakter tersebut dipetakan menjadi 128 bit binary string untuk setiap bloknya dengan panjang 128 bit karakter entry berupa `0 atau `1.

2) Pembangkitan Kunci Putar

Sama halnya dengan pembangkitan kunci putar pada algoritma enkripsi, pembangkitan kunci putar dilakukan dengan membangkitkan substring dari kunci eksternal dengan menjumlahkan nilai ASCII dari karakter setiap partisi dari eksternal key. Partisi didapatkan dengan melakukan hash sha256 pada eksternal key yang menghasilkan 64 karakter. Terdapat 16 partisi yang memiliki panjang 4 bit.

3) Cipher Putar

Cipher putar dilakukan dengan mengeksekusi fungsi putar sebanyak 16 kali untuk setiap blok bit dari hasil partisi *ciphertext* dengan memanfaatkan *subkey* yang dibangkitkan dari kunci eksternal. Operasi-operasi fungsi putar pada algoritma dekripsi *ciphertext* cukup identik dengan operasi fungsi putar pada algoritma enkripsi. Hanya saja, algoritma ini dibalik pemakaiannya seperti pada algoritma kriptografi *block-cipher* pada umumnya. Operasi-operasi yang dilakukan dalam fungsi putar (F_i) adalah:

a) Permutasi Balik (Reshifting)

Operasi permutasi balik ditujukan untuk mengembalikan hasil permutasi pada algoritma enkripsi ke bentuk semula dengan melakukan pergeseran pada blok ke arah kanan. Pergeseran yang dimaksud adalah block shifting yakni, pergeseran dilakukan secara siklik pada setiap baris dari matriks berukuran 8 x 16 yang merepresentasikan binary string dari blok bit dengan panjang 128 bit. Pergeseran dilakukan sebanyak i dengan i adalah nilai indeks dari baris matriks tersebut.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	...	$a_{0,15}$	
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	...	$a_{1,15}$	→ Geser 1
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	...	$a_{2,15}$	→ Geser 2
...	...	...	...	...	...	
$a_{15,0}$	$a_{15,1}$	$a_{15,2}$	$a_{15,3}$	...	$a_{15,15}$	→ Geser 15

Gambar 3.3. Ilustrasi *shifting right* pada blok

b) Substitusi Balik

Setelah blok cipher dipermutasi balik, dilakukan operasi substitusi balik yang ditujukan untuk mengembalikan hasil substitusi ke bentuk semula. Operasi ini direalisasikan dengan menggunakan sebuah S-box yang sama dengan S-box yang digunakan pada algoritma enkripsi sebelumnya, yakni Rijndael S-box.

Proses substitusi balik dilakukan dengan mempartisi blok cipher menjadi 16 buah sub-blok dengan panjang 8 bit masukan. Untuk setiap sub-blok, binary string dari sub-blok tersebut dikonversikan menjadi nilai desimal (dengan rentang 0-256) yang nantinya akan dicari di dalam S-box. Entry-entry dari matriks S-box ini unik, sehingga nilai desimal tersebut dapat dicari posisinya pada S-box sehingga didapat nilai indeks baris dan kolom dari sebuah entry yang sesuai.

Nilai indeks baris dan kolom dikonversi menjadi binary string yang masing-masing berukuran 4 bit. Binary string dari indeks indeks merepresentasikan 4 bit pertama dan binary string dari indeks kolom merepresentasikan 4 bit terakhir. Kedua binary string ini digabungkan kembali sehingga membentuk 8 bit keluaran sub-blok. Setelah semua sub-blok selesai disubstitusi balik, hasil substitusi sub-blok tersebut digabung kembali menjadi 128 bit di akhir.

c) Adisi (Substraksi Balik)

Proses substraksi balik diawali dengan melakukan partisi pada blok menjadi sub-blok dengan panjang 4 bit. Kemudian, untuk setiap sub-blok tersebut dikonversi menjadi bilangan desimal dan dilakukan pengurangan. Dikarenakan algoritma dekripsi merupakan invers dari algoritma enkripsi maka pengurangan dilakukan secara terbalik dari nilai iterasi atau jumlah putaran saat ini dalam modulus 16 (2^4).

$$x = 16 - (15 - i) \; mod \; 16,$$

Jika i = 0, maka pengurangan dilakukan untuk :

$$x = 16 - (15 - 0) \; mod \; 16,$$

ini sama halnya dengan menambah sub-blok dengan 15 yang merupakan pengurangan terakhir dari algoritma enkripsi.

Hasil adisi tersebut selanjutnya dikonversi kembali menjadi binary string sepanjang 4 bit. Keseluruhan hasil substraksi dari sub-blok tersebut tidak lupa digabung kembali menjadi binary string sepanjang 128 bit.

d) Re-XOR

Blok hasil adisi (substraksi ulang) kemudian di XOR kan dengan subkey yang telah dibangkitkan dari kunci eksternal. Sama halnya dengan proses adisi yang merupakan kebalikan dari proses substraksi. Penggunaan Subkey dilakukan secara terbalik dari subkey indeks ke 15 hingga indeks ke 0.

e) Pertukaran 64bit Blok Bagian Kiri dan Kanan

Setelah blok dilakukan Re-XOR. Blok tersebut dipertukarkan kembali antara potongan awal blok dengan potongan akhir blok. Potongan ini masing-masing memiliki panjang 64 bit karena dipotong di tengah-tengah blok.

IV. Eksperimen dan Pembahasan Hasil

Kode program ditulis dalam bahasa *python* dan eksperimen implementasi algoritma SwazzyCipher dilakukan pada komputer dengan spesifikasi RAM 12GB, CPU Amd R5 5265, dan hanya membuka 1 aplikasi, yakni VS Code.

Pada eksperimen kali ini, masukan plaintext dan kunci dilakukan dengan menuliskannya ke dalam file 1p.txt dan 1k.txt. Lalu ditampilkan text awal, hasil enkripsi/dekripsi, dan waktu eksekusi yang ditampilkan dalam satuan detik (*seconds*) yang merupakan *pref_counter* dari library *time*.

a) Waktu Eksekusi Enkripsi dan Dekripsi

i) Small (16 char)

Kunci (16 char)	Waktu Eksekusi	
	Enkripsi	Dekripsi
`Crypt0gr4phySusy`	4.19 ms	3.27 ms
Plaintext (16 char)	**Ciphertext**	
`Everyone pretend`	`8lXö™ˆåë™çˆu]`	

ii) Medium (64 char)

Kunci (16 char)	Waktu Eksekusi	
	Enkripsi	Dekripsi
`Crypt0gr4phySusy`	10.58 ms	8.33 ms
Plaintext (64 char)	**Ciphertext**	
`Everyone pretends to like wheat til you mention barley is better`	`8lXö™ˆåë™çˆu]øp+”è5:î}ä> …RåÈzìö™¾§³/9- N!‡ îõ[«|è°ÔŠ‡‡˜%`	

iii) Large (128 char)

Kunci (16 char)	Waktu Eksekusi	
	Enkripsi	Dekripsi
`Crypt0gr4phySusy`	12.92 ms	18.92 ms
Plaintext (16 char)	**Ciphertext**	
`Jason didn’t understand why his parents wouldn’t let him sell his little sister at the garage sale. He hadn't had his cup of tea`	`?9õŸgšg{s• éá³ÿÇ_I ¬rR)½ø@¥X“- 4°Ç ËJU ½]„óÀF(æŠiä.(áÕo©ï +¨X ÃAyPíìÉDï-¾0ëÚí»[, j÷qK°^z $ßdŸk³6j÷çŸR@nÔèo¯ j ‘ýü‰HØm<Ôå® ``	

iv) Very Large (512 char)

Kunci (16 char)	Waktu Eksekusi	
	Enkripsi	Dekripsi
`Crypt0gr4phySusy`	51.19 ms	57.38 ms
Plaintext (16 char)	**Ciphertext**	
`The hummingbird's wings blurred while it eagerly sipped the sugar water from the feeder. The river stole the gods.The father handed each child a roadmap at the beginning of the 2-day road trip and explained it was so they could find their way home. Joe made the sugar cookies Susan decorated them. She insisted that cleaning out your closet was the key to good driving. Writing a list of random sentences is harder than I initially thought it would be. The shy gangster like into start that day with a pink scarf`	`!ëB±XcÄÄïî¼^Ñ³J&ø .È¿øXÐ ñ#@çª˜)Yâû3 }P î:âªçXCø%£¦C8b ¾KHE[…t2Í“wÊÝN',Aû û3,22“¿Íone.°¨qM4Ê <7 ¥:·ßÝ.°†¢ÄÕE}Pg° ¾ ˆç=oz™C …,vùB´L$ˆ<û ÀÍ2WË«‡ °»ëøâ1Òø¬¹wÂ0í £! È!ûû3zÃT‹ 'N½™ø¡íö™H'sP}èññ. Ìå4d'°“wq´(ßÝ.°Ã ñ.ƒ …è¢Æ¯ßÝ.°Ñ³Rƒ8r² S ŠÖ,«9 ü2:çâ©,ÐB}Ph°Wßà@g ÎP e 7Ê¶ûp 27kgÎ„M#¡ …óéÝà|04ä ¹:óyH”¿¿ “…ìŸ¿â1Zÿ,KÖ Škä=bš Nýq¨káOõXÕ;pûßÝ.°v \®°'‹yO@özÐ6LÜOKÆÓ .ú‹£! La½¿s‘ ¸Ø÷Æ¯Á Õø h ÝÍ\/;XÐŸ¿9ùªçåRÊk² X £iÏÔxZZw ¶ ÿ ì…iÏ-:".wª© ¨ƒ°è 1ã†z™ù/Xõ¤¥‹v\n{- Û ð0ùWebÃéà±ø™lhRƒzÈ ²ÃSîUMù Õ; È“ `	

Dari 4 buah contoh pesan yang dilakukan enkripsi dan dekripsi, terlihat bahwa waktu eksekusi relatif singkat. Korelasi antara waktu eksekusi dan panjang teks pesan juga relatif linier.

b) Analisis Efek Longsoran (Avalanche Effect)

Efek longsoran dapat dikalkulasi dengan mengukur perbandingan antara jumlah bit *ciphertext* yang berbeda dengan panjang *ciphertext* apabila masukan *plaintext* mengalami perubahan secara minimal.

$$Avalanche\ Effect = Changed\ bit\ /\ Ciphertext\ Length,$$

Pada eksperimen pengujian algoritma, dilakukan pengujian efek longsoran dengan membandingkan dua hasil enkripsi dari masukan *plaintext* berukuran 16 bytes dengan dengan 1 buah karakter berbeda. Adapun detail dari pengujian efek longsor dapat dilihat sebagai berikut.

	Masukan 1	Masukan 2
Plaintext	pembunuh**a**nmunir	pembunuh**u**nmunir
Ciphertext	^'CÄšk\—õ‰æÚd^	^'CÄšk¸óõ‰æÚd^
Efek longsor	7 bit : 128 bit = 0.0546875	

Dari hasil pengujian efek longsor, didapatkan nilai perbandingan yang sangat kecil. Ini menunjukkan bahwa algoritma SwazzyCipher masih kurang memuaskan dalam menimbulkan efek *diffusion.*

c) Analisis Ruang Kunci

Pada algoritma *block cipher* yang dikembangkan ini, panjang kunci yang digunakan yakni sebesar 16 *bytes* atau 128 bit. Sehingga, kombinasi ruang kunci yang mungkin untuk dibangkitkan ada sebanyak 2^{128} atau 3.4 x 10^{38} buah kombinasi kunci. Dengan banyaknya kemungkinan kunci tersebut, akan sulit untuk melakukan penyerangan secara *brute force*, baik untuk ukuran komputer secanggih dan secepat apapun. Misalnya terdapat suatu komputer yang dapat membangkitkan 1 miliar kunci setiap detiknya. Maka, dibutuhkan waktu sebanyak 1.08 x 10^{22} tahun untuk mencoba semua kemungkinan kunci.

V. Kesimpulan dan Saran

A. Kesimpulan

Algoritma SwazzyCipher merupakan *block cipher* baru yang dapat melakukan enkripsi dan dekripsi pada suatu pesan. Algoritma ini akan membagi suatu pesan menjadi beberapa blok berukuran 128 bit kemudian melakukan proses enkripsi atau dekripsi dengan menggunakan kunci sepanjang 128 bit. Algoritma ini melakukan proses perulangan enkripsi dan dekripsi sebanyak 16 putaran dengan menggunakan sub-kunci berbeda yang dibangkitkan dari operasi terhadap kunci eksternal. Algoritma ini menerapkan prinsip *diffusion* dan *confusion* dari Shannon.

Berdasarkan hasil analisis yang telah dilakukan, bisa disimpulkan bahwa algoritma ini terhitung sangkil dalam melakukan enkripsi dan dekripsi suatu pesan. Selain itu, algoritma ini juga aman terhadap serangan *brute force* yang mungkin terjadi. Akan tetapi, algoritma ini ternyata masih lemah dalam menimbulkan efek *diffusion*.

B. Saran

Algoritma SwazzyCipher memiliki *key space* yang sangat besar dan sulit untuk dipecahkan dengan mencari korelasi antara kunci dan *ciphertext*. Waktu komputasi enkripsi dan dekripsi algoritma SwazzyCipher juga cukup memuaskan untuk masukan dengan ukuran yang besar. Akan tetapi, algoritma ini didapati kurang dari penilaian efek longsor, sehingga dibutuhkan pengembangan lebih lanjut dalam merealisasikan prinsip *diffusion*.

Repositori Kode Sumber

https://github.com/Kyasaaa/Block-Cipher.git

Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih sebesar-besarnya kepada beberapa pihak yang telah berperan besar dalam penyelesaian makalah ini :

1) Tuhan Yang Maha Esa karena dengan berkat dan rahmat-Nya, makalah ini dapat terselesaikan dengan baik tanpa kurang satu bagian pun.
2) Dr. Ir. Rinaldi, M.T. sebagai dosen pengampu mata kuliah IF4020 Kriptografi yang telah membimbing penulis selama perkuliahan.
3) Teman-teman perkuliahan yang telah memberikan dukungan selama perkuliahan.

Referensi

[1] Rinaldi Munir, Diktat Kuliah IF4020 : Pengantar Kriptografi, Bandung : Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/01-Pengantar-Kriptografi-(2023).pdf (Diakses pada 3 Maret 2023)

[2] Rinaldi Munir, Diktat Kuliah IF4020 : Kriptografi Modern, Bandung : Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/11-Kripto-modern-2023.pdf (Diakses pada 3 Maret 2023)

[3] Rinaldi Munir, Diktat Kuliah IF4020 : Kriptografi Modern (Bagian 4: Prinsip Perancangan Block Cipher), Bandung : Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/14-Prancangan-block-cipher-2023.pdf (Diakses pada 3 Maret 2023)

[4] Situs New S-box calculation for Rijndael-AES based on an artificial neural network (redalyc.org) (Diakses pada 4 Maret 2023)

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, 5 Maret 2023

Mahasiswa 1

Rahmat Rafid Akbar
NIM. 13520090

Mahasiswa 2

M. Akyas David Al Aleey
NIM. 13520011

Mahasiswa 3

Stefanus Jeremy Aslan
NIM. 13519175