Algoritma Block Cipher Souffle

Muhammad Treza Nolandra - 13515080
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung
40132, Indonesia
13515080@std.stei.itb.ac.id

I Kadek Yuda Budipratama Giri - 13516115
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung
40132, Indonesia
13516115@std.stei.itb.ac.id

***Abstrak*—Makalah ini memuat rancangan algoritma blok *cipher* baru, yaitu *souffle cipher*. Algoritma souffle berasal dari kata *shuffle*, yang berarti mengocok. Algoritma ini memanfaatkan banyak pengacakan dan *pseudorandom number generator*. Algoritma ini memanfaatkan prinsip shannon, yaitu *confusion* dan *diffusion*. *Confusion* terjadi ketika karakter pada teks disubstitusi dengan nilai lain yang ada pada *substitution box*. *Diffusion* terjadi ketika bit dan byte pada teks ditukar posisinya. Selain itu, algoritma ini memanfaatkan jaringan feistel. Fungsi enkripsi dalam feistel adalah *vigenere cipher*. Algoritma ini aman untuk dipakai karena tahan serangan *brute force*, serangan analisis frekuensi, dan sebagainya.**

Keywords—confusion, diffusion, feistel, vigenere

I. PENDAHULUAN

Kata kriptografi berasal dari bahasa Yunani, "kryptós" yang berarti tersembunyi dan "gráphein" yang berarti tulisan [9]. Kriptografi telah digunakan oleh Julius Caesar sejak zaman Romawi Kuno. Teknik ini dijuluki *Caesar cipher* untuk mengirim pesan secara rahasia, meskipun teknik yang digunakannya tidak cocok dipakai untuk saat ini. Kriptografi digunakan agar isi pesan, yaitu plainteks, tidak dapat diketahui orang lain kecuali pembuat pesan atau target pembaca pesan. Enkripsi dilakukan agar menyamarkan isi pesan sehingga terbentuk *cipher* atau cipherteks. Dekripsi dilakukan agar pesan yang telah samar menjadi dapat terbaca kembali.

Saat ini, terdapat banyak algoritma *cipher* yang bagus dan aman digunakan. Sebagai contohnya, algoritma 3DES, AES, RSA, dan lain-lain. Algoritma tersebut dirancang agar orang lain sulit menebak plainteks dari cipher yang dihasilkan. Algoritma *cipher* akan aman apabila serangan *brute force*, analisis frekuensi, dan sebagainya sulit untuk memecahkan *cipher*.

Algoritma *cipher* blok yang baik akan menerankan prinsip shannon, yaitu *confusion* dan *diffusion*. Selain itu, jaringan feistel dapat digunakan agar mudah melakukan dekripsi. Algoritma blok *cipher* souffle yang kami rancang memanfaatkan prinsip shannon dan jaringan feistel. Selain itu, algoritma ini memproses pesan dalam *bit* maupun *byte*. Algoritma souffle berasal dari kata *shuffle*, yang berarti mengocok. Algoritma ini melibatkan banyak pengacakan atau penggunaan *pseudorandom number generator*.

II. DASAR TEORI

A. *Confusion* dan *Diffusion*

Claude E. Shannon pada tahun 1945 memperkenalkan dua prinsip dalam teori informasi, yaitu *Confusion* dan *Diffusion*. Kedua prinsip ini mempengaruhi keberhasilan suatu algoritma *cipher block*.

Confusion adalah prinsip dimana hubungan antara cipherteks dan *key* harus dibuat serumit mungkin [1]. Prinsip ini bertujuan untuk membuat hubungan antara cipherteks dan *key* menjadi tersamarkan sehingga kriptanalis sulit menganalisisnya dengan metode statistik [2].

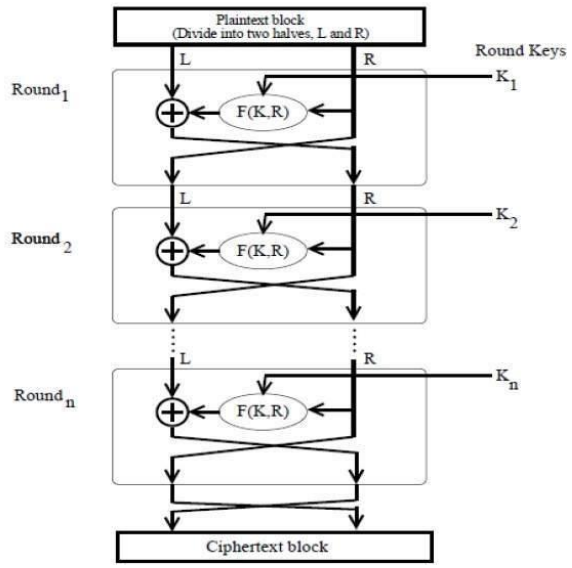
Diffusion adalah prinsip dimana setiap bagian blok/bit dari ciphertext dipengaruhi oleh berbagai blok/bit lain dari ciphertext [3]. Prinsip ini digunakan dengan untuk memperumit hubungan statistik antara cipherteks dan plainteks, sehingga kriptanalis akan kesulitan dalam memetakan hubungan antara cipherteks dan plainteks [2]. Dengan prinsip *diffusion*, diharapkan apabila kita mengganti satu bagian dari plainteks, cipherteks yang dihasilkan akan berbeda secara signifikan dibandingkan cipherteks hasil enkripsi plainteks awal.

B. Jaringan Feistel

Jaringan Feistel adalah sebuah metode untuk mengubah fungsi apapun menjadi fungsi permutasi [4]. Fungsi yang digunakan merupakan fungsi yang memetakan fungsi *string* input menjadi *string* output. Jaringan Feistel menggunakan parameter fungsi F sebagai fungsi yang memetakan *string* dan sebuah angka d sebagai *round* (jumlah pengulangan proses dalam Jaringan Feistel).

Misal digunakan fungsi F dalam jaringan Feistel dengan parameter d sebagai jumlah *round*. Pada *round* ke-*i*, operasi ini dilakukan dalam Jaringan Feistel [5]:

1. Input dipecah menjadi dua, misal namanya L_{i-1} dan R_{i-1}. Input menjadi $L_{i-1} \,||\, R_{i-1}$
2. Input R_{i-1} dieksekusi dengan fungsi F, menghasilkan $f_i(R_{i-1})$
3. Hasil fungsi tadi dioperasikan dengan operator XOR, menghasilkan $L_{i-1} \oplus f_i(R_{i-1})$
4. Sisi kiri dan kanan dari input ditukar, sehingga menghasilkan keluaran $L_{i-1} \oplus f_i(R_{i-1}) \,||\, L_{i-1}$



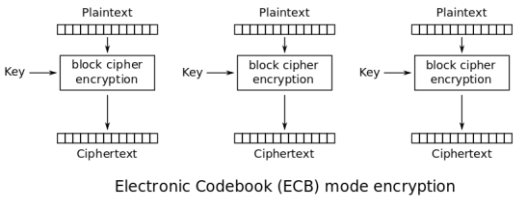
Gambar 1
Ilustrasi Feistel Network
(Sumber: https://www.tutorialspoint.com/cryptography/images/feistel_structure.jpg)

C. Algoritma Blok Cipher Modern

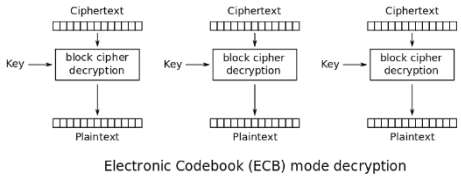
Perbedaan mendasar antara algoritma kriptografi modern dan algoritma kriptografi klasik adalah algoritma kriptografi modern melakukan proses pada mode bit, dimana plainteks, *key*, dan cipherteks diproses sebagai rangkaian bit. Operasi yang paling banyak digunakan adalah XOR (*exclusive OR*) dan tetap menggunakan prinsip seperti substitusi dan transposisi yang lebih rumit. Algoritma kriptografi modern mendorong penggunaan komputer digital dalam pengiriman pesan.

Blok Cipher (*cipher block*) adalah suatu algoritma kriptografi dimana terdapat pembagian plainteks/cipher menjadi sejumlah blok dengan panjang sama pada proses enkripsi/dekripsi. Enkripsi dan dekripsi dilakukan per blok dengan *key* yang disediakan. Metode blok cipher dilakukan dalam beberapa mode operasi, yaitu [6]:

1. *Electronic Code Book* (ECD), dimana tiap blok plainteks akan dienkripsi menjadi blok cipherteks secara individual.

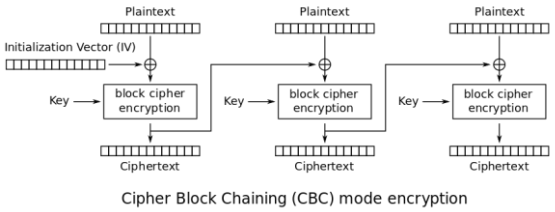


Gambar 2
Enkripsi blok cipher mode ECB
(Sumber: https://en.wikipedia.org/wiki/File:ECB_encryption.svg)

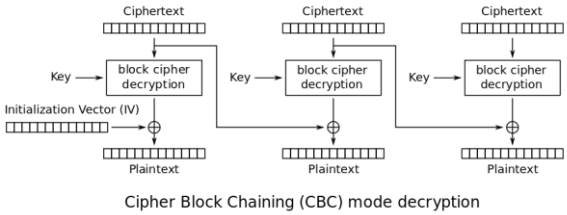


Gambar 3
Dekripsi blok cipher mode ECB
(Sumber: https://en.wikipedia.org/wiki/File:ECB_decryption.svg)

2. *Cipher Block Chain* (CBC), dimana hasil dari tiap blok cipherteks bergantung kepada blok cipherteks lain.

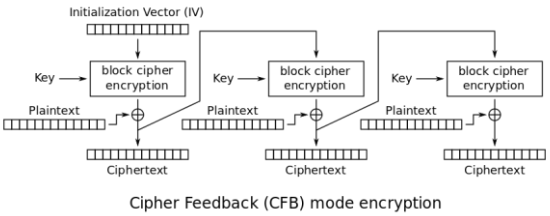


Gambar 4
Enkripsi blok cipher mode CBC
(Sumber: https://en.wikipedia.org/wiki/File:CBC_encryption.svg)

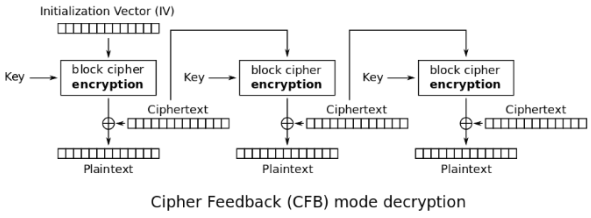


Gambar 5
Dekripsi blok cipher mode CBC
(Sumber: https://en.wikipedia.org/wiki/File:CBC_decryption.svg)

3. *Cipher Feedback* (CFB), dimana sistem blok cipher ini memberikan *feedback* berupa hasil cipher untuk blok cipher selanjutnya. Awalnya, sebuah *initial vector* IV diproses dengan fungsi enkripsi, lalu hasilnya dioperasikan XOR dengan blok plainteks menghasilkan blok cipherteks yang akan digunakan sebagai *feedback* fungsi enkripsi untuk melakukan operasi untuk menghasilkan blok cipherteks berikutnya. Proses dekripsi cipherteks menggunakan proses yang sama, hanya mengganti blok plainteks dengan blok cipherteks.

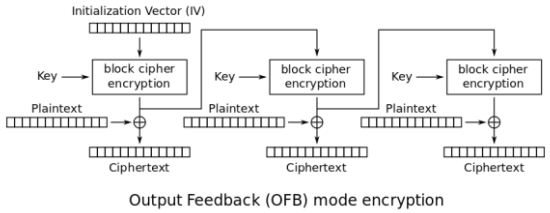


Gambar 6
Enkripsi blok cipher mode CFB
(Sumber: https://en.wikipedia.org/wiki/File:CFB_encryption.svg)

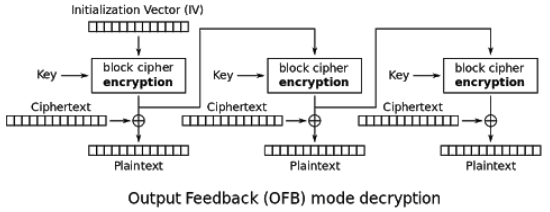


Gambar 7
Dekripsi blok cipher mode CFB
(Sumber: https://en.wikipedia.org/wiki/File:CFB_decryption.svg)

4. *Output Feedback* (OFB), sama seperti mode CFB, tetapi *feedback* yang diberikan bukan blok cipherteks sebelumnya, tetapi hasil enkripsi *feedback* dari blok sebelumnya. Blok pertama akan mengirimkan hasil fungsi enkripsi dengan *feedback* IV, lalu proses berikutnya akan mengirimkan hasil dari fungsi enkripsi dengan *feedback* dari blok sebelumnya.

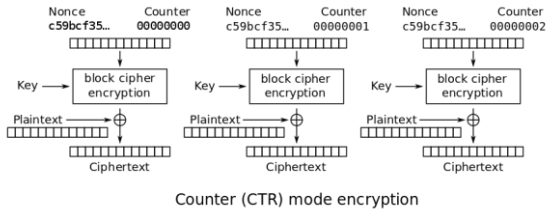


Gambar 8
Enkripsi blok cipher mode OFB
(Sumber: https://en.wikipedia.org/wiki/File:OFB_encryption.svg)

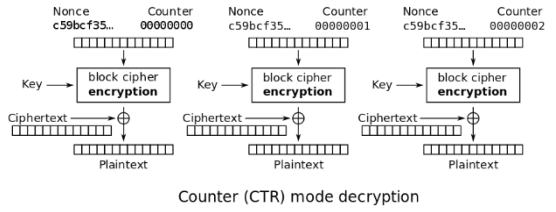


Gambar 9
Dekripsi blok cipher mode OFB
(Sumber: https://en.wikipedia.org/wiki/File:OFB_decryption.svg)

5. *Counter*, sistem blok cipher dengan penyisipan angka *counter* pada *initial vector* IV setiap enkripsi blok.



Gambar 10
Enkripsi blok cipher mode *Counter*
(Sumber:https://en.wikipedia.org/wiki/File:CTR_encryption_2.svg)



Gambar 11
Dekripsi blok cipher mode *Counter*
(Sumber:https://en.wikipedia.org/wiki/File:CTR_decryption_2.svg)

D. *Vigenerè Cipher*

Vigenerè Cipher adalah algoritma kriptografi klasik yang ditemukan oleh Blaise de Vigenère. Metode ini mirip dengan *shift cipher*, tetapi dengan kunci yang dapat berbeda setiap huruf [7]. Kunci berupa kumpulan huruf dengan panjang sembarang. Enkripsi dilakukan dengan menggeser huruf plainteks sebanyak urutan huruf kunci pada alfabet ('a' = 1, 'b'=2, dst). Dekripsi dilakukan dengan menggeser ke belakang huruf cipherteks sebanyak urutan kunci pada alfabet. Cara lain untuk melakukan enkripsi adalah dengan melihat bujursangkar vigenere

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Gambar 12
Bujur Sangkar Vigenerè
(Sumber: https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/FIG-VIG-Table.jpg)

Versi pengembangan dari *Vigenerè Cipher* adalah *Extended Vigenerè Cipher*, dimana algoritma ini dapat memproses seluruh karakter ASCII.

III. RANCANGAN ALGORITMA

Algoritma ini memanfaatkan putaran feistel sehingga untuk mendekripsi menjadi lebih sederhana. Terdapat 10 putaran feistel dalam *souffle cipher* (dimulai dari putaran ke-0 hingga putaran ke-9). Sebelum dan sesudah putaran feistel, terdapat proses substitusi dan permutasi. Pada putaran feistel, fungsi yang digunakan adalah fungsi enkripsi *vigenere cipher*. Kunci yang digunakan untuk algoritma ini sebanyak 2 buah. Masing-masing kunci digunakan untuk hal yang berbeda-beda tergantung putaran yang sedang berlangsung pada putaran ganjil atau genap.

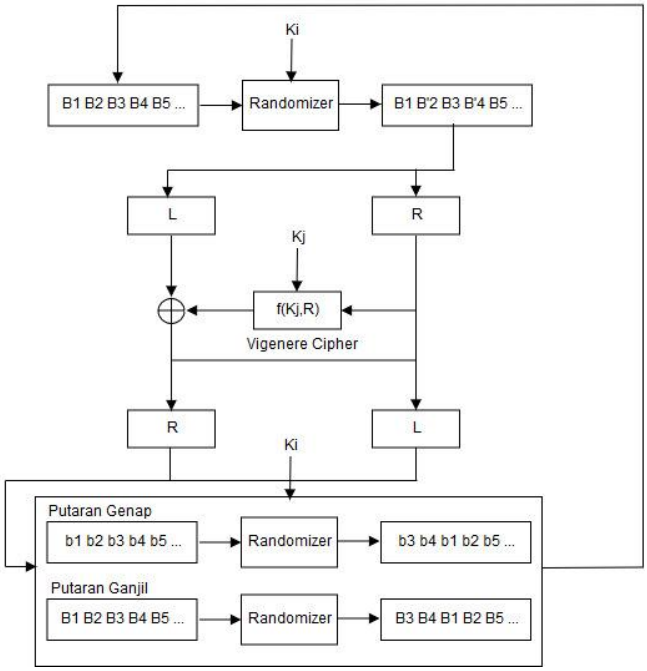
Proses substitusi dilakukan dengan memanfaatkan *Substitution Box*. Proses ini menerapkan prinsip *confusion* Shannon. S-Box (singkatan dari *substitution box*) dibuat berdasarkan kunci yang digunakan dimana dari indeks (0,0) hingga (15,15) diisi dengan nilai secara acak tak berulang. Nilai tersebut berada pada rentang 00 hingga ff. Byte yang disubstitusi pada plainteks dipilih secara acak dengan menggunakan seed kunci. Misalkan dari sekumpulan byte terpilih secara acak byte "ae" untuk disubstitusikan. Byte tersebut disubstitusi dengan nilai pada S-Box yang berada pada indeks (10,14). Indeks tersebut diperoleh dari nilai desimal dari bilangan heksadesimal "a" dan "e". Dalam satu substitusi, jumlah byte yang dipilih secara acak untuk disubstitusikan (bisa byte yang sama) sebanyak setengah ukuran plainteks.

Proses permutasi dilakukan dengan penukaran posisi karakter pada plainteks. Proses ini menerapkan proses *diffusion* Shannon. Posisi-posisi yang ditukar ditentukan melalui *seed pseudo-random generator* dengan kunci yang digunakan. Terdapat 2 jenis permutasi yang digunakan, yaitu permutasi *bit* dan *byte*. Permutasi bit dilakukan dengan cara

menukar posisi-posisi *bit*. Permutasi *byte* dilakukan dengan cara menukar posisi-posisi *byte*. Pada putaran feistel ganjil, permutasi yang dilakukan adalah permutasi *byte*. Permutasi bit dilakukan pada putaran genap.

Fungsi kunci pertama dan kunci kedua berbeda-beda tergantung pada genap-ganjilnya putaran. Pada putaran genap, kunci pertama digunakan untuk proses permutasi dan substitusi, sedangkan kunci kedua digunakan untuk *vigenere cipher* pada putaran feistel. Sebaliknya pada putaran ganjil, kunci pertama untuk *vigenere cipher* pada putaran feistel, sedangkan kunci kedua untuk proses permutasi dan substitusi.

Gambaran cara kerja algoritma souffle dapat dilihat pada gambar 13. Ki dan Kj adalah kunci 1 dan kunci 2 pada saat putaran genap. Sebaliknya, Ki dan Kj adalah kunci 2 dan kunci 1 pada saat putaran ganjil.



Gambar 13
Rancangan Algoritma Souffle

IV. EKSPERIMEN

Pengujian algoritma *souffle cipher* dilakukan untuk mengetahui kinerja algoritma *cipher* tersebut. Algoritma diprogram dengan bahasa pemrograman Python. Plainteks yang digunakan adalah "Lorem ipsum dolor sit amet.". Pengujian melibatkan proses enkripsi dan dekripsi. Kunci yang digunakan adalah kunci berukuran 8 byte. Kunci 1 yang digunakan adalah "Kriptografi", sedangkan kunci 2 yang digunakan adalah "Menyenangkan". Plainteks dipotong menjadi blok-blok berukuran 8 byte. Hasil pengujian dapat dilihat pada tabel 1 hingga tabel 5. Mode OFB merupakan mode operasi dengan waktu eksekusi tercepat. Hasil waktu eksekusi masing-masing enkripsi dan dekripsi dengan *souffle cipher* dapat dilihat pada gambar 14.

Tabel 1 Enkripsi dengan Mode ECB

Plainteks (string)	Lorem ipsum dolor sit amet.
Plainteks (hex)	4c6f72656d20697073756d20646f6c6f7220736 97420616d65742e
Cipher (hex)	b4a58f63fc270fb4cf923bd84c6c572056c575f8 ea5345988e7b97

Tabel 2 Enkripsi dengan Mode CBC

Plainteks (string)	Lorem ipsum dolor sit amet.
Plainteks (hex)	4c6f72656d20697073756d20646f6c6f7220736 97420616d65742e
Cipher (hex)	92d9cb4b1a4f51b7c16e4e11087d1c8f3217d6c 1d494997e5605bc

Tabel 3 Enkripsi dengan Mode CFB

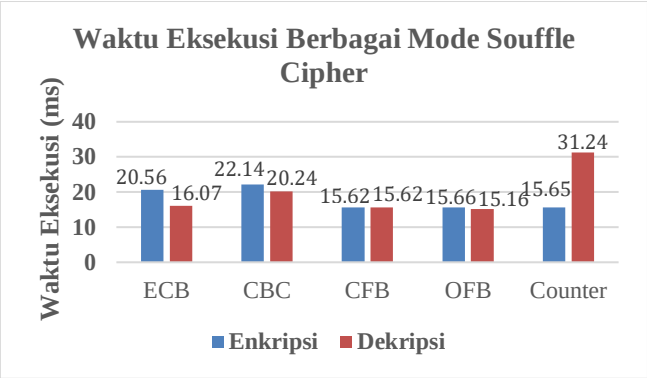
Plainteks (string)	Lorem ipsum dolor sit amet.
Plainteks (hex)	4c6f72656d20697073756d20646f6c6f7220736 97420616d65742e
Cipher (hex)	78e1303f14bc310ba79f6554e890e76717f6357 2cfb895b4f4e224

Tabel 4 Enkripsi dengan Mode OFB

Plainteks (string)	Lorem ipsum dolor sit amet.
Plainteks (hex)	4c6f72656d20697073756d20646f6c6f7220736 97420616d65742e
Cipher (hex)	78e1303f14bc310b5399e5efcb2d5460c25d70e 9bd733235bbb7fd

Tabel 5 Enkripsi dengan Mode Counter

Plainteks (string)	Lorem ipsum dolor sit amet.
Plainteks (hex)	4c6f72656d20697073756d20646f6c6f7220736 97420616d65742e
Cipher (hex)	f8670aecccd53c8be4b96a7dc96de08cce937c5a 90230a0addb314



Gambar 14
Diagram Waktu Eksekusi Berbagai Mode Souffle Cipher

V. ANALISIS KEAMANAN

A. Analisis Serangan *Brute Force*

Salah satu cara untuk memecahkan *souffle cipher* tanpa melalui serangan kunci adalah dengan melakukan serangan *brute force* untuk menebak kunci yang digunakan. Waktu yang dibutuhkan untuk menebak kunci yang digunakan kurang lebih sama untuk menebak *password* yang digunakan untuk mengamankan akun. Semakin banyak panjang kunci, semakin sulit melakukan serangan *brute force*. Kunci dengan 8 karakter memiliki 6.634.204.312.890.620 kemungkinan kombinasi, sedangkan kunci dengan 12 karakter memiliki 540.360.087.662.637.000.000.000 kemungkinan kombinasi [10]. Menurut [9], password dengan 8 karakter dapat dipecahkan kurang lebih dalam waktu 5 jam. Namun, password dengan 12 karakter dapat dipecahkan dalam kurang lebih 2 abad.

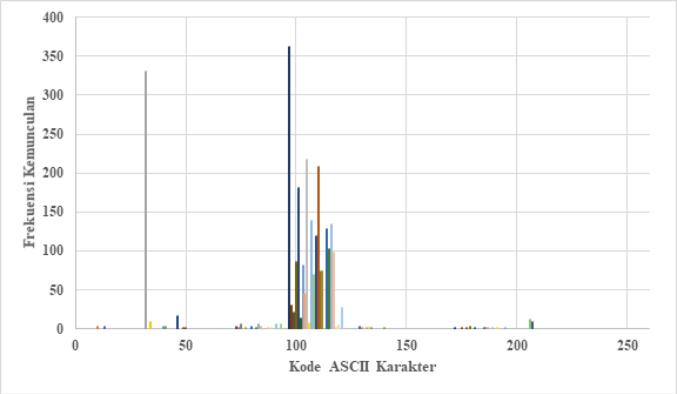
Disebabkan oleh penggunaan 2 kunci, *brute force* menjadi lebih sulit menebak plainteks. Jika tiap kunci berukuran 8 byte, berarti *brute force* bagaikan harus menebak kunci berukuran 16 byte. Berarti, waktu yang dibutuhkan untuk menebak kunci akan lebih besar dari 2 abad. Selain itu, kunci berpasangan dan tidak boleh ditukar. Jika kunci 1 adalah "Kriptografi" dan kunci 2 adalah "Menyenangkan", kunci 1 tidak dapat dibalik menjadi "Menyenangkan" dan kunci 2 menjadi "Kriptografi". Jika *brute force* menebak kunci 1 adalah "Kriptografi", kunci 2 haruslah "Menyenangkan" agar dapat didekripsi.

B. Analisis Frekuensi Kemunculan Karakter

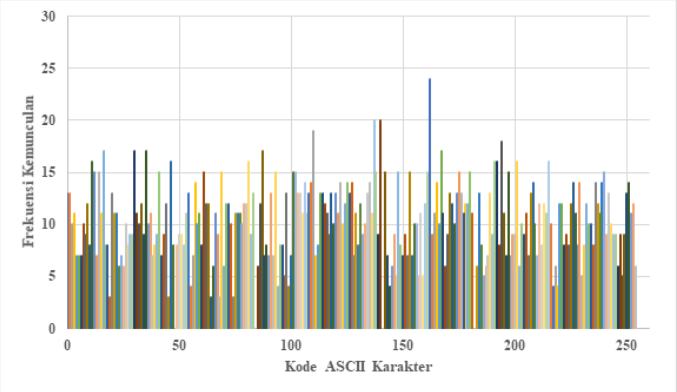
Analisis frekuensi kemunculan karakter dilakukan untuk melihat bagaimana frekuensi karakter yang dihasilkan jika plainteks dienkripsi dengan *souffle cipher*. Jika enkripsi dilakukan dengan *vigenere cipher*, *caesar cipher*, atau *cipher* sejenis, frekuensi kemunculan karakter akan sama. Oleh karena itu, cipher dapat mudah ditebak plainteksnya walaupun tanpa kunci dari frekuensi kemunculan. Misalkan pada cipher huruf "f" adalah huruf dengan frekuensi tertinggi. Disebabkan oleh huruf yang sering muncul pada bahasa Inggris adalah huruf "e", maka kemungkinan huruf "e" dienkripsi menjadi huruf "f".

Analisis dilakukan dengan menggunakan teks berbahasa Indonesia yang dapat dilihat pada laman *https://id.wikipedia.org/wiki/Kriptografi*. Teks yang diambil adalah 4 paragraf awal yang ada pada laman tersebut. Teks tersebut memiliki frekuensi kemunculan karakter-karakter yang dapat dilihat pada gambar 15. Proses enkripsi dilakukan dengan mode CBC. Dengan menggunakan kunci 1 "Kriptografi" dan kunci 2 "Menyenangkan", diperoleh cipher dengan frekuensi kemunculan karakter yang dapat dilihat pada gambar 16. Pesan dibagi menjadi blok berukuran 8.

Dari gambar dapat dilihat bahwa frekuensi kemunculan karakter ASCII pada plainteks dan cipher berbeda jauh. Oleh karena itu, dapat disimpulkan bahwa *souffle cipher* aman dari serangan analisis frekuensi..



Gambar 15
Diagram Frekuensi Kemunculan Karakter ASCII pada Plainteks



Gambar 16
Diagram Frekuensi Kemunculan Karakter ASCII pada Cipher

C. Analisis Perubahan Sedikit Karakter pada Kunci

Ketika karakter pada kunci dirubah sedikit, semua mode enkripsi menghasilkan *cipher* yang berbeda jauh. Hal ini berarti kunci yang dipakai akan sulit ditebak karena mengubah satu karakter pada kunci akan mengakibatkan *cipher* akan berbeda jauh. Hasil enkripsi dapat dilihat pada tabel 6 hingga tabel 10.

Tabel 6 Enkripsi dengan Mode ECB

Kunci 1	Ciphertext
"Kriptografi"	b4a58f63fc270fb4cf923bd84c6c572056c575f8ea5345988e7b97
"Kriptagrafi"	96111511a7023734b65cb113b2d7367a74621539b8a56b91d9a0fa

Tabel 7 Enkripsi dengan Mode CBC

Kunci 1	Ciphertext
“Kriptografi”	92d9cb4b1a4f51b7c16e4e11087d1c8f3217d6c1d494997e5605bc
“Kriptagrafi”	da9e70b1623e0e1d4c2c4ceced5fefe4b40ecb9ca27dde337ee946

Tabel 8 Enkripsi dengan Mode CFB

Kunci 1	Ciphertext
“Kriptografi”	78e1303f14bc310ba79f6554e890e76717f63572cfb895b4f4e224
“Kriptagrafi”	4bd247b47f0e39f51df2c3c719f19dc538af6edcc772ffdfde1415

Tabel 9 Enkripsi dengan Mode OFB

Kunci 1	Ciphertext
“Kriptografi”	78e1303f14bc310b5399e5efcb2d5460c25d70e9bd733235bbb7fd
“Kriptagrafi”	4bd247b47f0e39f5358c886090c06812611cb579cf8c1aa4668b80

Tabel 10 Enkripsi dengan Mode Counter

Kunci 1	Ciphertext
“Kriptografi”	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
“Kriptagrafi”	223ab2b24eae9ef6bc6c55dcc852e97be0da4b36667eb4293cf14c

D. Analisis Perubahan Sedikit Karakter pada Plainteks

Ketika karakter pada plainteks diubah sedikit, semua mode enkripsi kecuali mode *counter* menghasilkan *cipher* yang berbeda jauh. Hal ini berarti mode *counter* relatif kurang aman. Hasil enkripsi dapat dilihat pada tabel 11 hingga tabel 15.

Tabel 11 Enkripsi dengan Mode ECB

Plainteks	Ciphertext
Lorem ipsum dolor sit amet.	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
Lorom ipsum dolor sit amet.	b4791517ce34bb7ecf923bd84c6c572056c575f8ea5345988e7b97

Tabel 12 Enkripsi dengan Mode CBC

Plainteks	Ciphertext
Lorem ipsum dolor sit amet.	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
Lorom ipsum dolor sit amet.	05f98cce1b87db20a5afba8e58d1ea673a5fd5015f28244b662830

Tabel 13 Enkripsi dengan Mode CFB

Plainteks	Ciphertext
Lorem ipsum dolor sit amet.	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
Lorom ipsum dolor sit amet.	78e1303514bc310bc0b7a19c5ec7905d5df8419f7e241e93e78354

Tabel 14 Enkripsi dengan Mode OFB

Plainteks	Ciphertext
Lorem ipsum dolor sit amet.	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
Lorom ipsum dolor sit amet.	78e1303514bc310b5399e5efcb2d5460c25d70e9bd733235bbb7fd

Tabel 15 Enkripsi dengan Mode Counter

Plainteks	Ciphertext
Lorem ipsum dolor sit amet.	f8670aecccd53c8be4b96a7dc96de08cce937c5a90230a0addb314
Lorom ipsum dolor sit amet.	f8670ae6ccd53c8be4b96a7dc96de08cce937c5a90230a0addb314

E. Analisis Blok yang Berulang

Misalkan n adalah ukuran blok. Pengujian untuk blok berulang dilakukan dengan 2 kasus, yaitu: karakter yang berulang berukuran sama dengan n dan tidak sama dengan n. Ketika karakter yang berulang tidak sama dengan n, hasilnya cipher tidak mengandung karakter berulang. Ketika karakter yang berulang sama dengan n, hasilnya cipher tidak mengandung karakter berulang kecuali dengan mode ECB. Dapat disimpulkan bahwa mode ECB kurang aman.

Pengujian dilakukan dengan menggunakan n = 8. Enkripsi pada tabel 16 menggunakan plainteks “abcdefghabcdefghabcdefghabcdefgh”. Enkripsi pada tabel 17 menggunakan plainteks “wxyzwxyzwxyzwxyzwxyz”. Kedua plainteks menggunakan kunci 1 dan kunci 2 yang sama seperti sebelummya, yaitu “Kriptografi” dan “Menyenangkan”.

Tabel 16 Enkripsi dengan Ukuran Teks Berulang Sama Dengan Ukuran Blok

Mode	Ciphertext
ECB	348e425a799c587b348e425a799c587b348e425a799c587b348e425a799c587b
CBC	5c455a82b13e34f9158633731826205 88fe98451361888941df045aa712dc32b

CFB	55ec213e1cfa3f1347ec3b9c572bf3215256 0184f5f8163cc431692881e8eef7
OFB	55ec213e1cfa3f13418eebabca245f67d11f6 0e4ac353430bfa1b037e154e65e
Counter	d56a1bedc4933293f6ae6439c864eb8bddd 16c5781650c0fd9a5590b74cec576

Tabel 17 Enkripsi dengan Ukuran Teks Berulang Tidak Sama Dengan Ukuran Blok

Mode	Ciphertext
ECB	828e5ec56e4825f8828e5ec56e4825f8 e24f7c4d
CBC	f77e592bde531ab33d25afc1305e6d0f 3de1b0a9
CFB	43f63b200ee42101e4f52a2e7895c81e e7af1104
OFB	43f63b200ee421015794f1b5d83a4175 c7057afa
Counter	c37001f3d68d2c81e0b47e27da7af599 cbcb7649

VI. KESIMPULAN DAN SARAN

Algoritma souffle yang dirancang sudah cukup baik untuk mengenkripsi dan mendekripsi pesan. Hal ini disebabkan algoritma ini aman dipakai berdasarkan hasil pengujian dan analisis. Sebaiknya mode cipher yang tidak digunakan adalah mode ECB dan mode *counter*. Mode ECB kurang baik digunakan karena plainteks yang mengandung kata berulang akan menghasilkan *cipher* yang berulang juga. Selain itu, perubahan sedikit karakter pada mode *counter* menyebabkan hasil *cipher* hanya berbeda jumlah karakter yang sama pula. Hal ini mengakibatkan kedua mode tersebut relatif lebih mudah dipecahkan dibanding dengan mode lain. Untuk mode ECB, plainteks dengan kata berulang hanya akan menghasilkan *cipher* yang berulang jika dihasilkan blok dengan isi karakter yang sama. Hal ini dapat diatasi dengan mengubah ukuran blok, sehingga tidak ada blok dengan isi karakter yang sama. Algoritma *souffle* ini dapat memanfaatkan fungsi enkripsi selain *vigenere cipher* (pada jaringan feistel). Saran yang dapat dilakukan ke depannya adalah melakukan analisis fungsi enkripsi apa yang paling baik, optimal, dan aman untuk *souffle cipher*.

REFERENSI

[1] Wade Trappe, Lawrence Washington. *Introduction to Cryptography and Coding Theory*. Upper Saddle River, NJ: Pearson, 2005, pp 38.

[2] "Difference Between Confusion and Diffusion (with comparison chart)". Internet: https://techdifferences.com/difference-between-confusion-and-diffusion.html [12 Maret 2019]

[3] "Shannon's Idea of Confusion and Diffusion". Internet: https://www.cse.ust.hk/faculty/cding/CSIT571/SLIDES/confdiffu.pdf [12 Maret 2019]

[4] "Unbalanced Feistel Networks and Block-Cipher Design". Internet: https://www.schneier.com/academic/paperfiles/paper-unbalanced-feistel.pdf [12 Maret 2019]

[5] "Block Ciphers: Feistel Proof". Internet: https://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/feistelProof.pdf [12 Maret 2019]

[6] "Block Cipher Modes of Operation". Internet: http://www.crypto-it.net/eng/theory/modes-of-block-ciphers.html

[7] Wade Trappe, Lawrence Washington. *Introduction to Cryptography and Coding Theory*. Upper Saddle River, NJ: Pearson, 2005, pp 16 - 18

[8] "Estimating Password Cracking Times". Internet: https://www.betterbuys.com/estimating-password-cracking-times/ [12 Maret 2019]

[9] Rinaldi Munir. *Pengantar Kriptografi*. Slide Kuliah IF4020 Kriptografi, 2018.

[10] Mark Burnett, David Kleiman. *Perfect Passwords*. Syngress, 2006, pp 60.

PERNYATAAN

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

Bandung, 13 Maret 2019

Muhammad Treza N
(13515080)

I Kadek Yuda B. G.
(13516115)