PENYANDIAN DALAM KRIPTOGRAFI

Hendro – NIM : 13505103

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15103@students.if.itb.ac.id

Abstrak

Makalah ini membahas mengenai penyandian yang digunakan dalam kriptografi. Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan (*message*) [*Schneier*, 1996]. Dalam kriptografi, sistem penyandian dapat dikelompokan menjadi 2 kelompok, yaitu sistem *cipher* klasik dan sistem *cipher* modern. Sistem *cipher* klasik adalah sistem algoritma kriptografi yang digunakan pada zaman dahulu yang masih berbasis karakter sehingga algoritmanya tidak terlalu sulit untuk dipecahkan. Sistem *cipher* modern adalah sistem algoritma yang sampai saat ini masih digunakan karena berbasis bit sehingga cukup sulit untuk dipecahkan dan tingkat keamanannya lebih tinggi dibandingkan dengan sistem *cipher* klasik.

Sebelum komputer ada, kriptografi dilakukan dengan sistem cipher klasik (algoritma kriptografi yang bersejarah) yang berbasis karakter. Algoritma yang digunakan termasuk ke dalam sistem kriptografi simetri dan digunakan jauh sebelum sistem kriptografi kunci publik ditemukan. Terdapat sejumlah algoritma yang tercatat dalam sejarah kriptografi (sehingga dinamakan algoritma kriptografi klasik), namun sekarang algoritma tersebut sudah usang karena ia sangat mudah dipecahkan. Algoritma kriptografi klasik terbagi 2, yaitu *cipher* Substitusi (*Substitution ciphers*) dan *cipher* Transposisi (*Transposition ciphers*).

Algoritma kriptografi modern umumnya beroperasi dalam mode bit ketimbang mode karakter (seperti yang dilakukan pada *cipher* substitusi atau *cipher* transposisi dari algoritma kriptografi klasik). Operasi dalam mode bit berarti semua data dan informasi (baik kunci, plainteks, maupun cipherteks) dinyatakan dalam rangkaian (*string*) bit biner, 0 dan 1. Algoritma enkripsi dan dekripsi memproses semua data dan informasi dalam bentuk rangkaian bit. Rangkaian bit yang menyatakan plainteks dienkripsi menjadi cipherteks dalam bentuk rangkaian bit, demikian sebaliknya. Perkembangan algoritma kriptografi modern berbasis bit didorong oleh penggunaan komputer digital yang merepresentasikan data dalam bentuk biner. Algoritma kriptografi modern dapat dikelompokkan menjadi dua kategori, yaitu *cipher* Aliran (*Stream cipher*) dan *cipher* Blok (*Block cipher*).

***Kata kunci**:* Kriptografi, *cipher*, *block cipher*, *stream cipher* , *Substitution ciphers*, *Transposition ciphers*, enkripsi, dekripsi.

1. Pendahuluan

Sekarang ini, penyimpana dan pengiriman data sangat mudah dilakukan. Kita dapat menyimpan dan mengirimkan data-data secara cepat dan praktis. Data dapat disimpan di dalam media elektronik seperti komputer, handphone, telepon, dan lain-lain sehingga kita tidak perlu mengingatnya satu per satu. Dalam proses ini, harus diperhatikan keamanan dan keutuhan data. Data haruslah terjaga kerahasiaannya dan utuh sampai pada saat penerimaan di tujuan.

Untuk itu, diperlukan suatu proses penyandian terhadap data. Proses ini disebut enkripsi (*encryption*) dan dekripsi *(decryption*). Enkripsi adalah proses pengubahan data asli (*plaintext*) menjadi data rahasia (*ciphertext*) pada saat pengiriman (*sending*) sehingga kerahasiaan data terjaga. Sedangkan dekripsi adalah proses pengubahan data rahasia (*ciphertext*) menjadi data asli (*plaintext*) pada saat penerimaan (*receiving*) sehingga data sesuai dengan data asli yang dikirimkan. Dengan proses ini, maka selama proses pengiriman data, data bersifat rahasia sehingga keamanannya dapat terjaga.

Untuk melakukan enkripsi maupun dekripsi, diperlukan suatu algoritma. Algoritma ini disebut algoritma kriptografi. Jadi setiap orang yang mengetahui algoritma kriptografi dari suatu data dapat mengetahui isi dari data tersebut Sehingga biasanya algoritma kriptografi dibuat serumit mungkin. Semakin rumit algoritmanya, semakin aman data tersebut karena semakin sulit untuk mengubah data rahasia menjadi data asli.

2. Algoritma Kriptografi

Menurut sejarahnya, algoritma kriptografi dapat dikategorikan menjadi 2 kelompok, yaitu sistem *cipher* klasik dan sistem *cipher* modern.

2.1.Sistem Cipher Klasik (Algoritma Kriptografi Bersejarah)

Sebelum komputer ada, kriptografi dilakukan dengan algoritma berbasis karakter.

Algoritma yang digunakan termasuk ke dalam sistem kriptografi simetri dan digunakan jauh sebelum sistem kriptografi kunci publik ditemukan.

Terdapat sejumlah algoritma yang tercatat dalam sejarah kriptografi (sehingga dinamakan algoritma kriptografi klasik), namun sekarang algoritma tersebut sudah usang karena ia sangat mudah dipecahkan.

Algoritma kriptografi klasik:
1. Cipher Substitusi (*Substitution Ciphers*)
2. Cipher Transposisi (*Transposition Ciphers*)

2.1.1. Cipher Substitusi

Ini adalah algoritma kriptografi yang mula-mula digunakan oleh kaisar Romawi, Julius Caesar (sehingga dinamakan juga *caesar Cipher*), untuk menyandikan pesan yang ia kirim kepada para gubernurnya.

Caranya adalah dengan mengganti (menyulih atau mensubstitusi) setiap karakter dengan karakter lain dalam susunan abjad (alfabet).

Misalnya, tiap huruf disubstitusi dengan huruf ketiga berikutnya dari susunan akjad. Dalam hal ini kuncinya adalah jumlah pergeseran huruf (yaitu $k = 3$).

Tabel substitusi:

p_i :	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		
c_i :	**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**		

Contoh 1. Pesan

HELLO WORLD

disamarkan (enskripsi) menjadi

KHOOR ZRUOG

Penerima pesan mendekripsi Cipherteks dengan menggunakan tabel substitusi, sehingga Cipherteks

KHOOR ZRUOG

dapat dikembalikan menjadi plainteks semula:

HELLO WORLD

Dengan mengkodekan setiap huruf abjad dengan *integer* sebagai berikut: $A = 0$, $B = 1$, …, $Z = 25$, maka secara matematis *caesar Cipher* menyandikan plainteks p_i menjadi c_i dengan aturan:

$$c_i = E(p_i) = (p_i + 3) \bmod 26 \qquad (1)$$

dan dekripsi Cipherteks c_i menjadi p_i dengan aturan:

$$p_i = D(c_i) = (c_i - 3) \bmod 26 \qquad (2)$$

Karena hanya ada 26 huruf abjad, maka pergeseran huruf yang mungkin dilakukan adalah dari 0 sampai 25. Secara umum, untuk pergeseran huruf sejauh k (dalam hal ini k adalah kunci enkripsi dan deksripsi), fungsi enkripsi adalah

$$c_i = E(p_i) = (p_i + k) \bmod 26 \qquad (3)$$

dan fungsi dekripsi adalah

$$p_i = D(c_i) = (c_i - k) \bmod 26 \qquad (4)$$

Catatan :

1. Pergeseran 0 sama dengan pergeseran 26 (susunan huruf tidak berubah)
2. Pergeseran lain untuk k > 25 dapat juga dilakukan namun hasilnya akan kongruen dengan bilangan bulat dalam modulo 26. Misalnya k = 37 kongruen dengan 11 dalam modulo 26, atau $37 \equiv 11 \pmod{26}$.
3. Karena ada operasi penjumlahan dalam persamaan (3) dan (4), maka *caesar Cipher* kadang-kadang dinamakan juga *additive Cipher*.

Kriptanalisis Terhadap *Caesar Cipher*

Caesar Cipher mudah dipecahkan dengan metode *exhaustive key search* karena jumlah kuncinya sangat sedikit (hanya ada 26 kunci).

Contoh 2.
Misalkan kriptanalis menemukan potongan Cipherteks (disebut juga *cryptogram*) `XMZVH`. Diandaikan kriptanalis mengetahui bahwa plainteks disusun dalam Bahasa Inggris dan algoritma kriptografi yang digunakan adalah *caesar Cipher*. Untuk memperoleh plainteks, lakukan dekripsi mulai dari kunci yang terbesar, 25, sampai kunci yang terkecil, 1. Periksa apakah dekripsi menghasilkan pesan yang mempunyai makna (lihat Tabel 1).

Tabel 1. Contoh *exhaustive key search* terhadap Cipherteks `XMZVH`

Kunci(k) *Ciphering*	'Pesan' hasil dekripsi	Kunci(k) *Ciphering*	'Pesan' hasil dekripsi
0	**XMZVH**	13	**KZMIU**
25	**YNAWI**	12	**LANJV**
24	**ZOBXJ**	11	**MBOKW**
23	**APCYK**	10	**NCPLX**
22	**BQDZL**	9	**ODQMY**
21	**CREAM**	8	**PERNZ**
20	**DSFBN**	7	**QFSOA**
19	**ETGCO**	6	**RGTPB**
18	**FUHDP**	5	**SHUQC**
17	**GVIEQ**	4	**TIVRD**
16	**HWJFR**	3	**UJWSE**
15	**IXKGS**	2	**VKXTF**
14	**JYLHT**	1	**WLYUG**

Dari Tabel 1, kata dalam Bahasa Inggris yang potensial menjadi plainteks adalah `CREAM` dengan menggunakan k = 21. Kunci ini digunakan untuk mendekripsikan Cipherteks lainnya.

Kadang-kadang satu kunci yang potensial menghasilkan pesan yang bermakna tidak selalu satu buah. Untuk itu, kita membutuhkan informasi lainnya, misalnya konteks pesan tersebut atau mencoba mendekripsi potongan Cipherteks lain untuk memperoleh kunci yang benar.

Contoh 3.
Misalkan potongan Cipherteks **`HSPPW`** menghasilkan dua kemungkinan kunci yang potensial, yaitu k = 4 menghasilkan pesan `DOLLS` dan k = 11 menghasilkan `WHEEL`. Lakukan deksripsi terhadap potongan Cipherteks lain tetapi hanya menggunakan k = 4 dan k = 11 (tidak perlu *exhaustive key search*) agar dapat disimpulkan kunci yang benar.

Cara lain yang digunakan untuk memecahkan Cipherteks adalah dengan statistik, yaitu dengan menggunakan tabel kemunculan karakter, yang membantu mengidentifikasi karakter plainteks yang berkoresponden dengan karakter di dalam Cipherteks (akan dijelaskan kemudian).

Jenis-jenis Cipher Subsbtitusi

a. Cipher abjad-tunggal (*monoalphabetic Cipher* atau *Cipher* substitusi sederhana - *simple substitution Cipher*)

Satu karakter di plainteks diganti dengan satu karakter yang bersesuaian. Jadi, fungsi *Ciphering*-nya adalah fungsi satu-ke-satu.

Jika plainteks terdiri dari huruf-huruf abjad, maka jumlah kemungkinan susunan huruf-huruf Cipherteks yang dapat dibuat adalah sebanyak

26! = 403.291.461.126.605.635.584.000.000

Caesar Cipher adalah kasus khusus dari *Cipher* abjad tunggal di mana susunan huruf Cipherteks diperoleh dengan menggeser huruf-huruf alfabet sejauh 3 karakter.

`ROT13` adalah program enkripsi sederhana yang ditemukan pada sistem UNIX. `ROT13` menggunakan *Cipher* abjad-tunggal dengan pergeseran k = 13 (jadi, huruf `A` diganti dengan **`N`**, `B` diganti dengan **`O`**, dan seterusnya).

Enkripsi arsip dua kali dengan `ROT13` menghasilkan arsip semula:

```
P = ROT13(ROT13(P))
```

b. Cipher substitusi homofonik
(*Homophonic substitution Cipher*)

Seperti *Cipher* abjad-tunggal, kecuali bahwa setiap karakter di dalam plainteks dapat dipetakan ke dalam salah satu dari karakter Cipherteks yang mungkin. Misalnya huruf `A` dapat berkoresponden dengan **`7`**, **`9`**, atau **`16`**, huruf `B` dapat berkoresponden dengan **`5`**, **`10`**, atau **`23`** dan seterusnya.

Fungsi *Ciphering*-nya memetakan satu-ke-banyak (*one-to-many*).

Cipher substitusi homofonik digunakan pertama kali pada tahun 1401 oleh wanita bangsawan Mantua.

Cipher substitusi homofonik lebih sulit dipecahkan daripada *Cipher* abjad-tunggal. Namun, dengan *known-plaintext attack*, Cipher ini dapat dipecahkan, sedangkan dengan *Ciphertext-only attack* lebih sulit.

***c. Cipher* abjad-majemuk**
(*Polyalpabetic substitution Cipher*)

Merupakan *Cipher* substitusi-ganda (*multiple-substitution Cipher*) yang melibatkan penggunaan kunci berbeda.

Cipher abjad-majemuk dibuat dari sejumlah *Cipher* abjad-tunggal, masing-masing dengan kunci yang berbeda.

Kebanyakan *Cipher* abjad-majemuk adalah *Cipher* substitusi periodik yang didasarkan pada periode m.
Misalkan plainteks P adalah

$$P = p_1 p_2 \ldots p_m p_{m+1} \ldots p_{2m} \ldots$$

maka Cipherteks hasil enkripsi adalah

$$E_k(P) = f_1(p_1)\, f_2(p_2) \ldots f_m(p_m)\, f_{m+1}(p_{m+1}) \ldots f_{2m}(p_{2m}) \ldots$$

yang dalam hal ini p_i adalah huruf-huruf di dalam plainteks.

Untuk m = 1, *Cipher*-nya ekivalen dengan *Cipher* abjad-tunggal.

Contoh *Cipher* substitusi periodik adalah Cipher Vigenere yang ditemukan oleh kriptologi Perancis, Blaise de Vigenere pada abad 16. Misalkan K adalah deretan kunci

$$K = k_1\, k_2 \ldots k_m$$

yang dalam hal ini k_i untuk $1 \le i \le m$ menyatakan jumlah pergeseran pada huruf ke-i. Maka, karakter Cipherteks $y_i(p)$ adalah

$$y_i(p) = (p + k_i) \bmod n \qquad (5)$$

Misalkan periode m = 20, maka 20 karakter pertama dienkripsi dengan persamaan (5), dimana setiap karakter ke-i menggunakan kunci k_i. Untuk 20 karakter berikutnya, kembali menggunakan pola enkripsi yang sama.

Cipher abjad-majemuk ditemukan pertama kali oleh Leon Battista pada tahun 1568. Metode ini digunakan oleh tentara AS selama Perang Sipil Amerika.

Meskipun *Cipher* abjad-majemuk dapat dipecahkan dengan mudah (dengan bantuan komputer), namun anehnya banyak program keamanan komputer (*computer security*) yang menggunakan *Cipher* jenis ini.

d. Cipher substitusi poligram
(*Polygram substitution Cipher*)

Blok karakter disubstitusi dengan blok Cipherteks. Misalnya `ABA` diganti dengan **`RTQ`**, `ABB` diganti dengan **`SLL`**, dan lain-lain.

Playfair Cipher, ditemukan pada tahun 1854, termasuk ke dalam *Cipher* substitusi poligram

dan digunakan oleh negara Inggris selama Perang Dunia I.

2.1.2 Cipher Transposisi

Pada *Cipher* transposisi, plainteks tetap sama, tetapi urutannya diubah. Dengan kata lain, algoritma ini melakukan *transpose* terhadap rangkaian karakter di dalam teks.

Nama lain untuk metode ini adalah **permutasi**, karena *transpose* setiap karakter di dalam teks sama dengan mempermutasikan karakter-karakter tersebut.

Contoh 4.
Misalkan plainteks adalah

TEKNIK INFORMATIKA ITB

Untuk meng-enkripsi pesan, plainteks ditulis secara horizontal dengan lebar kolom tetap, misal selebar 5 karakter (kunci $k = 5$):

TEKNI
KINFO
RMATI
KAITB

maka Cipherteksnya dibaca secara vertikal menjadi

TKRKEIMAKNAINFTTIOIB

Untuk mendekripsi pesan, kita membagi panjang Cipherteks dengan kunci. Pada contoh ini, kita membagi 20 dengan 4 untuk mendapatkan 5.

Algoritma dekripsi identik dengan algoritma enkripsi. Jadi, untuk contoh ini, kita menulis Cipherteks dalam baris-baris selebar 5 karakter menjadi:

TKRK
EIMA
KNAI
NFTT
IOIB

Dengan membaca setiap kolom kita memperoleh pesan semula:

TEKNIK INFORMATIKA ITB

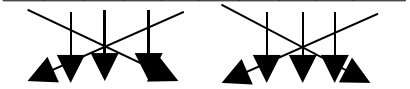
Variasi dari metode transposisi lainnya ditunjukkan pada Contoh 5 dan Contoh 6.

Contoh 5. Misalkan plainteks adalah

GANESHAITB

Plainteks diblok atas 5 karakter. Kemudian, pada tiap blok, karakter pertama dan karakter terakhir dipertukarkan

1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
G	A	N	E	S	H	A	I	T	B



1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
S	A	N	E	G	B	A	I	T	H

maka Cipherteksnya adalah

SANEGBAITH

Dekripsi dilakukan dengan cara yang sama, yaitu Cipherteks diblok atas delapan karakter. Kemudian, pada tiap blok, karakter pertama dan karakter terakhir dipertukarkan.

Contoh 6. Misalkan plainteks adalah

KRIPTOGRAFI

Plainteks disusun menjadi 3 baris ($k = 3$) seperti di bawah ini:

K T A
R P O R F
I G I

maka Cipherteksnya adalah

KTARPORFIGI

Kriptografi dengan alat *scytale* yang digunakan oleh tentara Sparta pada zaman Yunani termasuk ke dalam *Cipher* transposisi.

Lebih Jauh Dengan *Cipher* Abjad-tunggal

Seperti sudah disebutkan sebelum ini, metode *Cipher* abjad-tunggal mengganti setiap huruf di

dalam abjad dengan sebuah huruf lain dalam abjad yang sama.

Jumlah kunci di dalam *Cipher* abjad-tunggal sama dengan jumlah cara menyusun 26 huruf abjad tersebut, yaitu sebanyak

26! = 403.291.461.126.605.635.584.000.000

Ini berarti terdapat 26! buah kunci untuk menyusun huruf-huruf alfabet ke dalam tabel substitusi.

Contohnya, susunan guruf-huruf untuk Cipherteks diperoleh dengan menyusun huruf-huruf abjad secara acak seperti tabel substitusi berikut:

Tabel substitusi:

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

Satu cara untuk membangkitkan kunci adalah dengan sebuah kalimat yang mudah diingat.
Misal kuncinya adalah

cryptography is fun

Dari kunci tersebut, buang perulangan huruf sehingga menjadi

cryptogahisfunbdejklmqvwxz

lalu sambung dengan huruf-huruf lain yang tidak terdapat di dalam kalimat tersebut sehingga menjadi

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

Dengan demikian, tabel substitusi yang diperoleh adalah

Tabel substitusi:

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

Menerka Plainteks dari Cipherteks

Kadang-kadang kriptanalis melakukan terkaan untuk mengurangi jumlah kunci yang mungkin ada.

Terkaan juga dilakukan kriptanalis untuk memperoleh sebanyak mungkin plainteks dari potongan Cipherteks yang disadap. Plainteks yang diperoleh dari hasil terkaan ini biasanya digunakan dalam *known-plaintext attack*.

Asumsi yang digunakan: kriptanalis mengetahui bahwa pesan ditulis dalam Bahasa Inggris dan algoritma kriptografi yang digunakan adalah Cipher abjad-tunggal.

Contoh kasus 1:
Kriptanalis mempunyai potongan Cipherteks

G WR W RWL

Karena hanya ada dua kata yang panjangnya satu huruf dalam Bahasa Inggris (yaitu I dan A), maka **G** mungkin menyatakan huruf A dan **W** menyatakan huruf I, atau sebaliknya.

Kemungkinan **G** adalah huruf A dapat dieliminasi, maka diapstikan **G** = I, sehingga dengan cepat kriptanalis menyimpulkan bahwa potongan Cipherteks tersebut adalah

I AM A MA*

Dengan pengetahuan Bahasa Inggris, karakter terakhir (*) hampir dipastikan adalah huruf N, sehingga kalimatnya menjadi

I AM A MAN

Hasil ini mengurangi jumlah kunci dari 26! menjadi 22!

Contoh kasus 2:
Kriptanalis mempunyai potongan Cipherteks

HKC

Tidak banyak informasi yang dapat disimpulkan dari *cryptogram* di atas. Namun kriptanalis dapat mengurangi beberapa kemungkinan kunci, karena –sebagai contoh– tidak mungkin Z diganti dengan **H**, Q dengan **K**, dan K dengan **C**.

Namun, jumlah kemungkinan kunci yang tersisa tetap masih besar. Jika pesan yang dikirim memang hanya tiga huruf, maka *exhaustive key search* akan menghasilkan kata dengan tiga huruf berbeda yang potensial sebagai plainteks.

Contoh kasus 3:
Kriptanalis mempunyai potongan Cipherteks

HATTPT

Dalam hal ini, kriptanalis dapat membatasi jumlah kemungkinan huruf plainteks yang dipetakan menjadi T.

Kriptanalis mungkin mendeduksi bahwa salah satu dari **T** atau **P** merepresentasikan huruf vokal. Kemungkinan plainteksnya adalah CHEESE, MISSES, dan CANNON.

Contoh kasus 4:
Kriptanalis mempunyai potongan Cipherteks

HATTPT

dan diketahui informasi bahwa pesan tersebut adalah nama negara. Dengan cepat kriptanalis menyimpulkan bahwa polygram tesrebut adalah GREECE.

Dalam hal ini, kriptanalis dapat membatasi jumlah kemungkinan huruf plainteks yang dipetakan menjadi T.

Kriptanalis mungkin mendeduksi bahwa salah satu dari **T** atau **P** merepresentasikan huruf vokal. Kemungkinan plainteksnya adalah CHEESE, MISSES, dan CANNON.

Metode Statistik dalam Kriptanalisis

Metode yang paling umum digunakan dalam memecahkan Cipherteks adalah menggunakan statistik.

Dalam hal ini, kriptanalis menggunakan tabel frekuensi kemunculan huruf-huruf dalam teks bahasa Inggris. Tabel 2 memperlihatkan frekuensi kemunculan huruf-huruf abjad yang diambil dari sampel yang mencapai 300.000 karakter di dalam sejumlah novel dan suratkabar.

Tabel 2. Frekunsi kemunculan (relatif) huruf-huruf dalam teks Bahasa Inggris

Huruf	%	Huruf	%
A	8,2	N	6,7
B	1,5	O	7,5
C	2,8	P	1,9
D	4,2	Q	0,1
E	12,7	R	6,0
F	2,2	S	6,3
G	2,0	T	9,0
H	6,1	U	2,8
I	7,0	V	1,0
J	0,1	W	2,4
K	0,8	X	2,0
L	4,0	Y	0,1
M	2,4	Z	0,1

Tabel 2 di atas pada mulanya dipublikasikan di dalam *Cipher-Systems: The Protection of Communications* dan dikompilasi oleh H. J. Beker dan F.C. Piper

Terdapat sejumlah tabel frekuensi sejenis yang dipublikasikan oleh pengarang lain, namun secara umum persentase kemunculan tersebut konsisten pada sejumlah tabel.

Bila Cipher abjad-tunggal digunakan untuk meng-engkripsi pesan, maka kemunculan huruf-huruf di dalam plainteks tercermin pada tabel 2 di atas. Misalnya bila di dalam Cipher abjad-tunggal huruf **R** menggantikan huruf E, maka frekuensi **R** di dalam Cipherteks sama dengan frekuensi E di dalam plainteksnya.

2.2. Algoritma Kriptografi Modern

Algoritma kriptografi modern umumnya beroperasi dalam mode bit ketimbang mode karakter (seperti yang dilakukan pada *cipher* substitusi atau *cipher* transposisi dari algoritma kriptografi klasik).

Operasi dalam mode bit berarti semua data dan informasi (baik kunci, plainteks, maupun cipherteks) dinyatakan dalam rangkaian (*string*) bit biner, 0 dan 1. Algoritma enkripsi dan dekripsi memproses semua data dan informasi dalam bentuk rangkaian bit. Rangkaian bit yang menyatakan plainteks dienkripsi menjadi cipherteks dalam bentuk rangkaian bit, demikian sebaliknya.

Perkembangan algoritma kriptografi modern berbasis bit didorong oleh penggunaan komputer digital yang merepresentasikan data dalam bentuk biner.

Algoritma kriptografi yang beroperasi dalam mode bit dapat dikelompokkan menjadi dua kategori:

1. *Cipher* aliran (*stream cipher*)
   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit.

2. *Cipher* blok (*block cipher*)
   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya. Misalnya panjang blok adalah 64 bit, maka itu berarti algoritma enkripsi memperlakukan 8 karakter setiap kali penyandian (1 karakter = 8 bit dalam pengkodean ASCII).

Rangkaian bit

Rangkaian bit yang dipecah menjadi blok-blok bit dapat ditulis dalam sejumlah cara bergantung pada panjang blok.

Contoh: Plainteks 100111010110 dibagi menjadi blok bit yang panjangnya 4 menjadi

1001 1101 0110

Setiap blok menyatakan bilangan bulat dari 0 sampai 15, yaitu

9 13 6

Bila plainteks dibagi menjadi blok-blok yang berukuran 3 bit, maka rangkaian bit di atas menjadi:

100 111 010 110

Setiap blok menyatakan bilangan bulat dari 0 sampai 7, yaitu

4 7 2 6

Bila panjang rangkaian bit tidak habis dibagi dengan ukuran blok yang ditetapkan, maka blok yang terakhir ditambah dengan bit-bit semu yang disebut *padding bits*.
Misalnya rangkaian bit di atas dibagi menjadi blok 5-bit menjadi

10011 10101 **000**10

Blok yang terakhir telah ditambahkan 3 bit 0 di bagian awal (dicetak tebal) agar ukurannya menjadi 5 bit. *Padding bits* dapat mengakibatkan ukuran plainteks hasil dekripsi lebih besar daripada ukuran plainteks semula.

Cara lain untuk menyatakan rangkaian bit adalah dengan notasi heksadesimal (HEX). Rangkaian bit dibagi menjadi blok yang berukuran 4 bit dengan representasi dalam HEX adalah:

0000 = 0	0001 = 1	0010 = 2	0011 = 3
0100 = 4	0101 = 5	0011 = 6	0111 = 7
1000 = 8	1011 = 9	1010 = A	1011 = B
1100 = C	1101 = D	1101 = E	1111 = F

Misalnya, plainteks 100111010110 dibagi menjadi blok bit yang panjangnya 4 menjadi

1001 1101 0110

yang dalam notasi HEX adalah

9 D 6

Operator XOR

Operator biner yang sering digunakan dalam *cipher* yang yang beroperasi dalam mode bit adalah XOR atau *exclusive-or*.

Notasi matematis untuk opeartor XOR adalah □ (dalam Bahas C, operator XOR dilambangkan dengan ^).

Operator XOR diperasikan pada dua bit dengan aturan sebagai berikut:

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Operator XOR identik dengan penjumlahan modulo 2.
Misalkan *a*, *b*, dan *c* adalah peubah Boolean. Hukum-hukum yang terkait dengan operator XOR:

(i) $a \oplus a = 0$
(ii) $a \oplus b = b \oplus a$ (Hukum komutatif)
(iii) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$(Hukum asosiatif)

Jika dua rangkaian dioperasikan dengan XOR, maka operasinya dilakukan dengan meng-XOR-kan setiap bit yang berkoresponden dari kedua ramngkaian bit tersebut.

Contoh: 10011 ⊕ 11001 = 01010
yang dalam hal ini, hasilnya diperoleh sebagai berikut:

```
1   0   0   1   1
1   1   0   0   1   ⊕
-----------------------
0   1   0   1   0
```

Algoritma enkripsi sederhana yang menggunakan XOR adalah dengan meng-XOR-kan plainteks (P) dengan kunci (K) menghasilkan cipherteks:

$$C = P \oplus K \qquad (6.1)$$

Karena meng-XOR-kan nilai yang sama dua kali menghasilkan nilai semula, maka proses dekripsi menggunakan persamaan:

$$P = C \oplus K \qquad (6.2)$$

Contoh:

plainteks	01100101	(karakter 'e')
kunci	00110101	(karakter '5')
cipherteks	01010000	(karakter 'P')
kunci	00110101	(karakter '5')
plainteks	01100101	(karakter 'e')

Program komersil yang berbasis DOS atau Macintosh menggunakan algoritma XOR sederhana ini. Sayangnya, algoritma XOR sederhana tidak aman karena cipherteksnya mudah dipecahkan.

Cara memecahkannya adalah sebagai berikut (asumsi: panjang kunci adalah sejumlah kecil *byte*):

a. Cari panjang kunci dengan prosedur *counting coincidence* sbb: XOR-kan cipherteks terhadap dirinya sendiri setelah digeser sejumlah *byte*, dan hitung jumlah *byte* yang sama. Jika pergeseran itu kelipatan dari panjang kunci (yang tidak diketahui), maka 6% dari *byte* akan sama. Jika tidak, maka 0.4% akan sama. Angka persentase ini disebut *index of coincidence*. Pergeseran terkecil mengindikasikan panjang kunci yang dicari.

b. Geser cipherteks sejauh panjang kunci dan XOR-kan dengan dirinya sendiri. Operasi ini menghasilkan plainteks yang ter-XOR dengan plainteks yang digeser sejauh panjang kunci tersebut.

2.2.1. Cipher Aliran (*Stream Cipher*)

Cipher aliran merupakan salah satu tipe algoritma kriptografi kriptografi simetri.

Cipher aliran mengenkripsikan plainteks menjadi Cipherteks bit per bit (1 bit setiap kali transformasi).

Catatan: Variasi *Cipher* aliran lainnya adalah mengenkripsikan plainteks menjadi Cipherteks karakter per karakter atau kata per kata, misalnya pada *Vigenere Cipher* dan *one-time pad Cipher*.

Cipher aliran pertama kali diperkenalkan oleh Vernam melalui algoritmanya yang dikenal dengan nama **Vernam *Cipher***.

Vernam *cipher* diadopsi dari *one-time pad cipher*, yang dalam hal ini karakter diganti dengan bit (0 atau 1). Cipherteks diperoleh dengan melakukan penjumlahan modulo 2 satu bit plainteks dengan satu bit kunci:

$$c_i = (p_i + k_i) \bmod 2 \qquad (6.3)$$

yang dalam hal ini,

p_i : bit plainteks
k_i : bit kunci
c_i : bit cipherteks

Plainteks diperoleh dengan melakukan penjumlahan modulo 2 satu bit cipherteks dengan satu bit kunci:

$$p_i = (c_i - k_i) \bmod 2 \qquad (6.4)$$

Dengan kata lain, Vernam *cipher* adalah versi lain dari *one-time pad cipher*

Oleh karena operasi penjumlahan modulo 2 identik dengan operasi bit dengan operator XOR, maka persaman (6.3) dapat ditulis sebagai

$$c_i = p_i \oplus k_i \qquad (6.5)$$

dan proses deksripsi menggunakan persamaan

$$p_i = c_i \oplus k_i \qquad (6.6)$$

Pada *cipher* aliran, bit hanya mempunyai dua buah nilai, sehingga proses enkripsi hanya menyebabkan dua keadaan pada bit tersebut: berubah atau tidak berubah. Dua keadaan tersebut ditentukan oleh kunci enkripsi yang disebut **aliran-bit-kunci** (*keystream*).

Aliran-bit-kunci dibangkitkan dari sebuah pembangkit yang dinamakan pembangkit aliran-bit-kunci (*keystream generator*). Aliran-bit-kunci (sering dinamakan *running key*) di-XOR-kan dengan aliran bit-bit plainteks, $p_1, p_2, \ldots, p_i$, untuk menghasilkan aliran bit-bit cipherteks:

$$c_i = p_i \oplus k_i$$

Di sisi penerima, bit-bit cipherteks di-XOR-kan dengan aliran-bit-kunci yang sama untuk menghasilkan bit-bit plainteks:

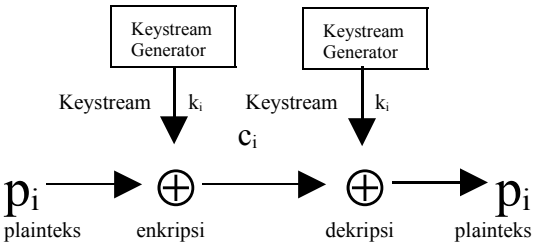
$$p_i = c_i \oplus k_i$$

karena

$$c_i \oplus k_i = (p_i \oplus k_i) \oplus k_i = p_i \oplus (k_i \oplus k_i)$$
$$= p_i \oplus 0 = p_i$$

Catatlah bahwa proses enkripsi dua kali berturut-turut menghasilkan kembali plainteks semula

Gambar 6.1 memperlihatkan konsep *cipher* aliran.



Gambar 6.1 Konsep *cipher* aliran

Contoh: Misalkan plainteks adalah

1100101

dan aliran-bit-kunci adalah

1000110

maka cipherteks yang dihasilkan adalah

0100011

yang mana diperoleh dengan meng-XOR-kan bit-bit plainteks dengan bit-bit aliran-kunci pada posisi yang berkoresponden.

Keamanan sistem *cipher* aliran bergantung seluruhnya pada pembangkit aliran-bit-kunci. Jika pembangkit mengeluarkan aliran-bit-kunci yang seluruhnya nol, maka cipherteks sama dengan plainteks, dan proses enkripsi menjadi tidak artinya.

Jika pembangkit mengeluarkan aliran-bit-kunci dengan pola 16-bit yang berulang, maka algoritma enkripsinya menjadi sama seperti enkripsi dengan XOR sederhana yang memiliki tingkat kemanan yang tidak berarti.

Jika pembangkit mengeluarkan aliran-bit-kunci yang benar-benar acak (*truly random*), maka algoritma enkripsinya sama dengan *one-time pad* dengan tingkat keamanan yang sempurna. Pada kasus ini, aliran-bit-kunci sama panjangnya dengan panjang plainteks, dan kita mendapatkan *cipher* aliran sebagai *unbreakable cipher*.

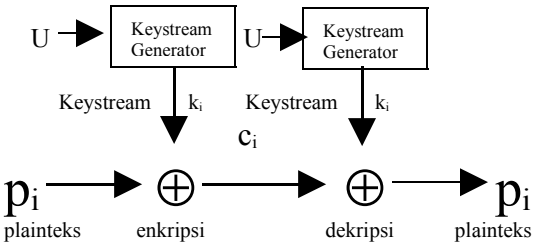
Tingkat keamanan *cipher* aliran terletak antara algoritma XOR sederhana dengan *one-time pad*. Semakin acak keluaran yang dihasilkan oleh pembangkit aliran-bit-kunci, semakin sulit kriptanalis memecahkan cipherteks.

Pembangkit aliran-bit-kunci (*Keystream Generator*)

Pembangkit bit-aliran-kunci dapat membangkitkan bit-aliran-kunci berbasis bit per bit atau dalam bentuk blok-blok bit. Untuk yang terakhir ini, *cipher* blok dapat digunakan untuk untuk memperoleh *cipher* aliran.

Untuk alasan praktis, pembangkit bit-aliran-kunci diimplementasikan sebagai prosedur algoritmik, sehingga bit-aliran-kunci dapat dibangkitkan secara simultan oleh pengirim dan penerima pesan.

Prosedur algoritmik tersebut menerima masukan sebuah kunci *U*. Keluaran dari prosedur merupakan fungsi dari *U* (lihat Gambar 6.2). Pembangkit harus menghasilkan bit-aliran-kunci yang kuat secara kriptografi.



Gambar 6.2 *Cipher* aliran dengan pembangkit bit-aliran-kunci yang bergantung pada kunci *U*.

Karena pengirim dan penerima harus menghasilkan bit-aliran-kunci yang sama , maka keduanya harus memiliki kunci *U* yang sama. Kunci *U* ini harus dijaga kerahasiaanya.

Cipher aliran menggunakan kunci *U* yang relatif pendek untuk membangkitkan bit-aliran-kunci yang panjang.

Contoh: Misalkan *U* adalah kunci empat-bit yang dipilih sembarang, kecuali 0000. Bit-aliran-kunci yang dibangkitkan akan berulang setiap 15 bit. Misalkan

$$U = 1111$$

Barisan bit-bit aliran-kunci diperoleh dengan meng-XOR-kan bit pertama dengan bit terakhir dari empat bit sebelumnya, sehingga menghasilkan:

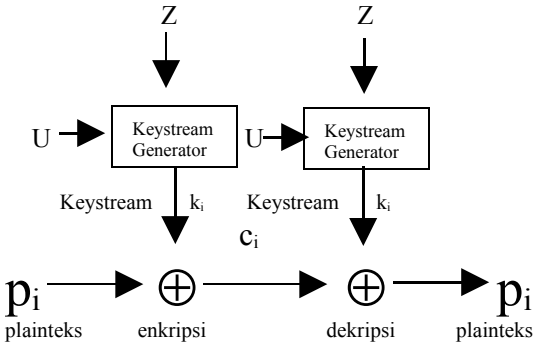
111101011001000

dan akan berulang setiap 15 bit.

Secara umum, jika panjang kunci *U* adalah *n* bit, maka bit-aliran-kunci tidak akan berulang sampai $2^n - 1$ bit.

Karena *U* adalah besaran yang konstan, maka bit-aliran-kunci yang dihasilkan pada setiap lelaran tidak berubah jika bergantung hanya pada *U*.
Ini berarti pembangkit bit-aliran-kunci tidak boleh mulai dengan kondisi awal yang sama supaya tidak menghasilkan kembali bit-aliran-kunci yang sama pada setiap lelaran.

Oleh karena itu, beberapa pembangkit bit-aliran-kunci menggunakan besaran **vektor inisialisasi** atau **umpan** (*seed*), disimbolkan dengan *Z*, agar diperoleh kondisi awal yang berbeda pada setiap lelaran (lihat Gambar 6.3).



Gambar 6.3 *Cipher* aliran dengan pembangkit bit-aliran-kunci yang bergantung pada kunci *U* dan umpan *Z*.

Dengan demikian, bit-aliran-kunci *K* dapat dinyatakan sebagai hasil dari fungsi *g* dengan parameter kunci *U* dan masukan umpan *Z*:

$$K = g_K(Z)$$

sehingga proses enkripsi dan dekripsi didefinisikan sebagai

$$C = P \oplus K = P \oplus g_K(Z)$$
$$P = C \oplus K = C \oplus g_K(Z)$$

Nilai Z yang berbeda-beda pada setiap lelaran menghasilkan bit-aliran-kunci yang berbeda pula.

Merancang pembangkit bit-aliran-kunci yang bagus cukup sulit karena membutuhkan pengujian statistik untuk menjamin bahwa keluaran dari pembangkit tersebut sangat mendekati barisan acak yang sebenarnya.

Serangan Terhadap Cipher Aliran

Serangan yang dapat dilakukan oleh kriptanalis terhadap *cipher* aliran adalah:

a. *Known-plaintext attack*
   Misalkan kriptanalis memiliki potongan plainteks (*P*) dan cipherteks (*C*) yang berkoresponden, maka ia dapat menemukan

bagian bit-aliran-kunci (K) yang berkoresponden dengan meng-XOR-kan bit-bit plainteks dan cipherteks:

$$P \oplus C = P \oplus (P \oplus K) = (P \oplus P) \oplus K$$
$$= 0 \oplus K = K$$

b. *Ciphertext-only attack*
Misalkan kriptanalis memiliki dua potongan cipherteks berbeda (C_1 dan C_2) yang dienkripsi dengan bit-aliran-kunci yang sama. Ia meng-XOR-kan kedua cipherteks tersebut dan memperoleh dua buah plainteks yang ter-XOR satu sama lain:

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K)$$
$$= (P_1 \oplus P_2) \oplus (K \oplus K)$$
$$= (P_1 \oplus P_2) \oplus 0$$
$$= (P_1 \oplus P_2)$$

P_1 dan P_2 dapat diperoleh dengan mudah. Selanjutnya, XOR-kan salah satu plainteks dengan cipherteksnya untuk memperoleh bit-aliran-kunci K yang berkoresponden:

$$P_1 \oplus C_1 = P_1 \oplus (P_1 \oplus K) = K$$

Pesan dari dua serangan di atas adalah: pengguna *cipher* aliran harus mempunyai bit-aliran-kunci yang tidak dapat diprediksi sehingga mengetahui sebagian dari bit-aliran-kunci tidak memungkinkan kriptanalis dapat mendeduksi bagian sisanya.

Aplikasi Cipher Aliran

Cipher aliran cocok untuk mengenkripsikan aliran data yang terus menerus melalui saluran komunikasi, misalnya:

a. Mengenkripsikan data pada saluran yang menghubungkan antara dua buah komputer.
b. Mengenkripiskan suara digital pada jaringan telepon *mobile* GSM.

Alasannya, jika bit cipherteks yang diterima mengandung kesalahan, maka hal ini hanya menghasilkan satu bit kesalahan pada waktu dekripsi, karena tiap bit plainteks ditentukan hanya oleh satu bit cipherteks. Kondisi ini tidak benar untuk *cipher* blok karena jika satu bit cipherteks yang diterima mengandung kesalahan, maka kesalahan ini akan merambat pada seluruh blok bit plainteks hasil dekripsi (*error propagation*).

2.2.2. Cipher Blok (*Block Cipher*)

Pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama, biasanya 64 bit (tapi adakalanya lebih). Algoritma enkripsi menghasilkan blok cipherteks yang – pada kebanyakan sistem kriptografi simetri – berukuran sama dengan blok plainteks.

Dengan blok *cipher*, blok plainteks yang sama akan dienkripsi menjadi blok cipherteks yang sama bila digunakan kunci yang sama pula. Ini berbeda dengan *cipher* aliran dimana bit-bit plainteks yang sama akan dienkripsi menjadi bit-bit cipherteks yang berbeda setiap kali dienkripsi.

Misalkan blok plainteks (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_1, p_2, \ldots, p_m)$$

yang dalam hal ini p_i adalah 0 atau 1 untuk i = 1, 2, …, m, dan blok cipherteks (C) adalah

$$C = (c_1, c_2, \ldots, c_m)$$

yang dalam hal ini c_i adalah 0 atau 1 untuk i = 1, 2, …, m.

Bila plainteks dibagi menjadi n buah blok, barisan blok-blok plainteks dinyatakan sebagai

$$(P_1, P_2, \ldots, P_n)$$

Untuk setiap blok plainteks P_i, bit-bit penyusunnya dapat dinyatakan sebagai vektor

$$P_i = (p_{i1}, p_{i2}, \ldots, p_{im})$$

Enkripsi dan dekripsi dengan kunci K dinyatakan berturut-turut dengan persamaan

$$E_K(P) = C$$

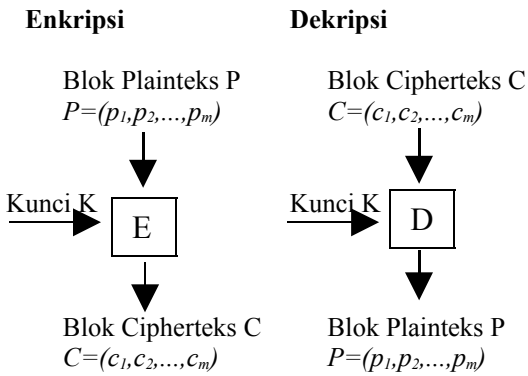
untuk enkripsi, dan

$$D_K(C) = P$$

Fungsi E haruslah fungsi yang berkoresponden satu-ke-satu, sehingga

$$E^{-1} = D$$

Skema enkripsi dan dekripsi dengan *cipher* blok digambarkan pada Gambar 1.



Gambar 1 Skema enkripsi dan dekripsi pada *cipher* blok

Teknik Kriptografi Klasik yang Digunakan pada Cipher Blok

Algoritma blok *cipher* menggabungkan beberapa teknik kriptografi klasik dalam proses enkripsi. Dengan kata lain, *cipher* blok dapat diacu sebagai super-enkripsi.

Teknik kriptografi klasik yang digunakan adalah:

1. Substitusi.
   Teknik ini mengganti satu atau sekumpulan bit pada blok plainteks tanpa mengubah urutannya. Secara matematis, teknik substitusi ini ditulis sebagai

   $$c_i = E(p_i),\ i = 1, 2, \ldots \text{(urutan bit)}$$

   yang dalam hal ini c_i adalah bit cipherteks, p_i adalah bit plainteks, dan f adalah fungsi substitusi.

   Dalam praktek, E dinyatakan sebagai fungsi matematis atau dapat merupakan tabel susbtitusi (*S-box*).

2. Transposisi atau permutasi
   Teknik ini memindahkan posisi bit pada blok plainteks berdasarkan aturan tertentu. Secara matematis, teknik transposisi ini ditulis sebagai

   $$C = PM$$

   yang dalam hal ini C adalah blok cipherteks, P adalah blok plainteks, dan M adalah fungsi transposisi.

Dalam praktek, M dinyatakan sebagai tabel atau matriks permutasi.

Selain kedua teknik di atas, *cipher* blok juga menggunakan dua teknik tambahan sebagai berikut:

3. Ekspansi
   Teknik ini memperbanyak jumlah bit pada blok plainteks berdasarkan aturan tertentu, misalnya dari 32 bit menjadi 48 bit. Dalam praktek, aturan eskpansi dinyatakan dengan tabel.

4. Kompresi
   Teknik ini kebalikan dari ekspansi, di mana jumlah bit pada blok plainteks diciutkan berdasarkan aturan tertentu. Dalam praktek, aturan kompresi dinyatakan dengan tabel.

Prinsip Penyandian Shannon

Pada tahun 1949, Shannon mengemukakan dua prinsip (*properties*) penyandian (*encoding*) data. Kedua prinsip ini dipakai dalam perancangan *cipher* blok yang kuat. Kedua prinsip Shannon tersebut adalah:

1. *Confusion*
   Prinsip ini menyembunyikan hubungan apapun yang ada antara plainteks, cipherteks, dan kunci. Sebagai contoh, pada *cipher* substitusi seperti *caesar cipher*, hubungan antara cipherteks dan plainteks mudah diketahui, karena satu huruf yang sama pada plainteks diganti dengan satu huruf yang sama pada cipherteksnya.

   Prinsip *confusion* akan membuat kriptanalis frustasi untuk mencari pola-pola statistik yang muncul pada cipherteks. Confusion yang bagus membuat hubungan statistik antara plainteks, cipherteks, dan kunci menjadi sangat rumit.

2. *Diffusion*
   Prinsip ini menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Sebagai contoh, pengubahan kecil pada plainteks sebanyak satu atau dua bit menghasilkan perubahan pada cipherteks yang tidak dapat diprediksi.

Prinisp *diffusion* juga menyembunyikan hubungan statistik antara plainteks, cipherteks, dan kunci dan membuat kriptanalisis menjadi sulit.

Untuk mendapatkan keamanan yang bagus, prinsip *confusion* dan *diffusion* diulang berkali-kali pada sebuah blok tunggal dengan kombinasi yang berbeda-beda.

Mode Operasi Cipher Blok

Plainteks dibagi menjadi beberapa blok dengan panjang tetap. Beberapa mode operasi dapat diterapkan untuk melakukan enkripsi terhadap keseluruhan blok plainteks. Empat mode operasi yang lazim diterapkan pada sistem blok *cipher* adalah:

1. *Electronic Code Book* (*ECB*)
2. *Cipher Block Chaining* (*CBC*)
3. *Cipher Feedback* (*CFC*)
4. *Output Feedback* (*OFB*)

Hanya 2 mode operasi saja yang akan dibahas dalam kuliah ini, yaitu *ECB* dan *CBC*.

Electronic Code Book (ECB)

Pada mode ini, setiap blok plainteks dienkripsi secara individual dan independen.

Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

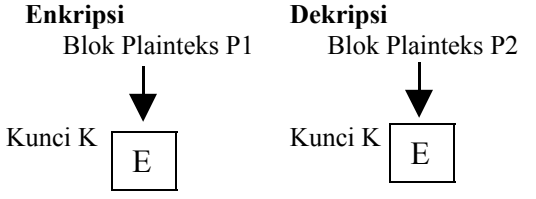
$$C_i = E_K(P_i)$$

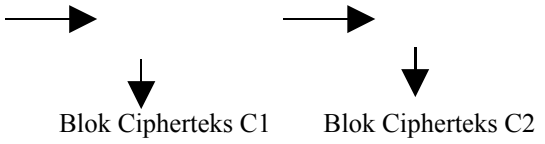
dan dekripsi sebagai

$$P_i = D_K(C_i)$$

yang dalam hal ini, P_i dan C_i masing-masing blok plainteks dan cipherteks ke-i.

Gambar 2 memperlihatkan enkripsi dua buah blok plainteks, P_1 dan P_2 dengan mode *ECB*, yang dalam hal ini E menyatakan fungsi enkripsi yang melakukan enkripsi terhadap blok plainteks dengan menggunakan kunci K.





Gambar 2 Skema enkripsi dan dekripsi dengan mode *ECB*

Contoh 1: Misalkan plainteks (dalam biner) adalah

10100010001110101001

Bagi plainteks menjadi blok-blok yang berukuran 4 bit:

1010 0010 0011 1010 1001

atau dalam notasi HEX adalah A23A9.

Misalkan kunci (K) yang digunakan panjangnya 4 bit

1011

atau dalam notasi HEX adalah B.

Misalkan fungsi enkripsi E yang sederhana (tetapi lemah) adalah dengan meng-XOR-kan blok plainteks P_i dengan K, kemudian geser secara *wrapping* bit-bit dari $P_i \oplus K$ satu posisi ke kiri.

Proses enkripsi untuk setiap blok digambarkan sebagai berikut:

	1010	0010	0011	1010	1001
	1011	1011	1011	1011	1011
Hasil XOR:	0001	1001	1000	0001	0010
Geser 1 bit ke kiri:	0010	0011	0001	0010	0100
Dalam notasi HEX:	2	3	1	2	4

Jadi, hasil enkripsi plainteks

10100010001110101001
(A23A9 dalam notasi HEX)

adalah

00100011000100100100
(23124 dalam notasi HEX)

Catatlah bahwa blok plainteks yang sama selalu dienkripsi menjadi blok cipherteks yang sama (atau identik). Pada contoh 1 di atas, blok 1010 muncul dua kali dan selalu dienkripsi menjadi 0010.

Contoh yang lebih nyata misalkan pesan

KUTU BUKU DI LEMARIKU

dibagi menjadi blok-blok yang terdiri dua huruf (dengan menghilangkan semua spasi) menjadi

KU TU BU KU DI LE MA RI KU

maka blok yang menyatakan "KU" akan dienkripsi menjadi blok cipherteks (dua huruf) yang sama.

Kata "*code book*" di dalam *ECB* muncul dari fakta bahwa karena blok plainteks yang sama selalu dienkripsi menjadi blok cipherteks yang sama, maka secara teoritis dimungkinkan membuat buku kode plainteks dan cipherteks yang berkoresponden.

Namun, semakin besar ukuran blok, semakin besar pula ukuran buku kodenya. Misalkan jika blok berukuran 64 bit, maka buku kode terdiri dari $2^{64} - 1$ buah kode (*entry*), yang berarti terlalu besar untuk disimpan. Lagipula, setiap kunci mempunyai buku kode yang berbeda.

Padding

Ada kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan (misalnya 64 bit atau lainnya). Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada blok-blok lainnya.

Satu cara untuk mengatasi hal ini adalah dengan *padding*, yaitu menambahkan blok terakhir dengan pola bit yang teratur agar panjangnya sama dengan ukuran blok yang ditetapkan. Misalnya ditambahkan bit 0 semua, atau bit 1 semua, atau bit 0 dan bit 1 berselang-seling.

Misalkan ukuran blok adalah 64 bit (8 *byte*) dan blok terakhir terdiri dari 24 bit (3 *byte*). Tambahkan blok terakhir dengan 40 bit (5 byte) agar menjadi 64 bit, misalnya dengan menambahkan 4 buah *byte* 0 dan satu buah *byte* angka 5. Setelah dekripsi, hapus 5 *byte* terakhir dari blok dekripsi terakhir.

Keuntungan Mode ECB

1. Karena tiap blok plainteks dienkripsi secara independen, maka kita tidak perlu mengenkripsi file secara linear. Kita dapat mengenkripsi 5 blok pertama, kemudian blok-blok di akhir, dan kembali ke blok-blok di tengah dan seterusnya.

   Mode *ECB* cocok untuk mengenkripsi arsip (*file*) yang diakses secara acak, misalnya arsip-arsip basisdata. Jika basisdata dienkripsi dengan mode *ECB*, maka sembarang *record* dapat dienkripsi atau didekripsi secara independen dari *record* lainnya (dengan asumsi setiap *record* terdiri dari sejumlah blok diskrit yang sama banyaknya).

   Jika mode *ECB* dikerjakan dengan prosesor paralel (*multiple processor*), maka setiap prosesor dapat melakukan enkripsi atau dekripsi blok plainteks yang berbeda-beda.

2. Jika satu atau lebih bit pada blok cipherteks mengalami kesalahan, maka kesalahan ini hanya mempengaruhi cipherteks yang bersangkutan pada waktu dekripsi. Blok-blok cipherteks lainnya bila didekripsi tidak terpengaruh oleh kesalahan bit cipherteks tersebut.

Kelemahan ECB

1. Karena bagian plainteks sering berulang (sehingga terdapat blok-blok plainteks yang sama), maka hasil enkripsinya menghasilkan blok cipherteks yang sama (lihat Contoh 1).

   Bagian plainteks yang sering berulang misalnya kata-kata seperti (dalam Bahasa Indonesia) *dan*, *yang*, *ini*, *itu*, dan sebagainya.

   Di dalam *e-mail*, pesan sering mengandung bagian yang redundan seperti *string* 0 atau spasi yang panjang, yang bila dienkripsi maka akan menghasilkan pola-pola cipherteks yang mudah dipecahkan dengan serangan yang berbasis statistik (menggunakan frekuensi kemunculan blok cipherteks). Selain itu, *e-mail* mempunyai struktur yang teratur yang menimbulkan pola-pola yang khas dalam cipherteksnya.

   Misalnya kriptanalis mempelajari bahwa blok plainteks 5EB82F (dalam notasi HEX) dienkripsi menjadi blok AC209D, maka setiap kali ia menemukan cipherteks AC209D, ia dapat langsung mendekripsinya menjadi 5EB82F.

Satu cara untuk mengurangi kelemahan ini adalah menggunakan ukuran blok yang besar, misalnya 64 bit, sebab ukuran blok yang besar dapat menghilangkan kemungkinan menghasilkan blok-blok yang identik.

2. Pihak lawan dapat memanipulasi cipherteks untuk "membodohi" atau mengelabui penerima pesan.

   Contoh 2. Misalkan seseorang mengirim pesan

   Uang ditransfer lima satu juta rupiah

   Andaikan bahwa kriptanalis mengetahui bahwa blok plainteks terdiri dari dua huruf (spasi diabaikan sehingga menjadi 16 blok plainteks) dan blok-blok cipherteksnya adalah

   C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}

   Misalkan kriptanalis berhasil mendekripsi keseluruhan blok cipherteks menjadi plainteks semula, sehingga ia dapat mendekripsi C_1 menjadi Ua, C_2 menjadi ng, C_3 menjadi di dan seterusnya. Kriptanalis memanipulasi cipherteks dengan membuang blok cipheteks ke-8 dan 9 sehingga menjadi

   C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}

   Penerima pesan mendekripsi cipherteks yang sudah dimanipulasi dengan kunci yang benar menjadi

   Uang ditransfer satu juta rupiah

   Karena dekripsi menghasilkan pesan yang bermakna, maka penerima menyimpulkan bahwa uang yang dikirim kepadanya sebesar satu juta rupiah.

Kedua kelemahan di atas dapat diatasi dengan mengatur enkripsi tiap blok individual bergantung pada semua blok-blok sebelumnya. Dengan cara ini, blok plainteks yang identik akan menghasilkan blok cipherteks yang berbeda, dan manipulasi cipherteks mungkin menghasilkan pesan hasil dekripsi yang tidak mempunyai makna.

Prinsip inilah yang mendasari mode operasi cipher blok yang kedua, yaitu *Cipher Block Chaining*.

Cipher Block Chaining (CBC)

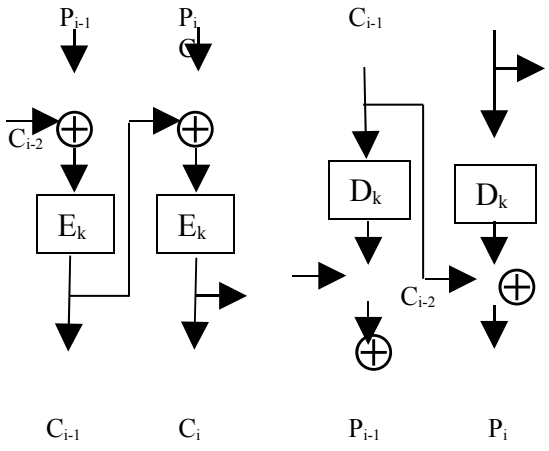
Mode ini menerapkan mekanisme umpan-balik (*feedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok yang *current*.

Caranya, blok plainteks yang *current* di-XOR-kan terlebih dahulu dengan blok cipherteks hasil enkripsi sebelumnya, selanjutnya hasil peng-XOR-an ini masuk ke dalam fungsi enkripsi.

Dengan mode *CBC*, setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya.

Dekripsi dilakukan dengan memasukkan blok cipherteks yang *current* ke fungsi dekripsi, kemudian meng-XOR-kan hasilnya dengan blok cipherteks sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan-maju (*feedforward*) pada akhir proses dekripsi.

Gambar 3 memperlihatkan skema mode operasi *CBC*.



Enkripsi **Dekripsi**

Misalkan kunci (*K*) yang digunakan adalah (panjangnya juga 4 bit)

1011

atau dalam notasi HEX adalah B. Sedangkan IV yang digunakan seluruhnya bit 0 (Jadi, C_0 = 0000)

Misalkan fungsi enkripsi E yang sederhana (tetapi lemah) adalah dengan meng-XOR-kan blok plainteks P_i dengan K, kemudian geser secara *wrapping* bit-bit dari $P_i \oplus K$ satu posisi ke kiri.

1. C_1 diperoleh sebagai berikut:

$$P_1 \oplus C_0 = 1010 \oplus 0000 = 1010$$

Enkripsikan hasil ini dengan fungsi E sbb:

$$1010 \oplus K = 1010 \oplus 1011 = 0001$$

Geser (*wrapping*) hasil ini satu bit ke kiri:
0010
Jadi, $C_1 = 0010$ (atau 2 dalam HEX)

2. C_2 diperoleh sebagai berikut:

$P_2 \oplus C_1 = 0010 \oplus 0010 = 0000$
$0000 \oplus K = 0000 \oplus 1011 = 1011$

Geser (*wrapping*) hasil ini satu bit ke kiri:
0111
Jadi, $C_2 = 0111$ (atau 7 dalam HEX)

3. C_3 diperoleh sebagai berikut:

$P_3 \oplus C_2 = 0011 \oplus 0111 = 0100$
$0100 \oplus K = 0100 \oplus 1011 = 1111$

Geser (*wrapping*) hasil ini satu bit ke kiri:
1111
Jadi, $C_3 = 1111$ (atau F dalam HEX)

Demikian seterusnya, sehingga plainteks dan cipherteks hasilnya adalah:

Pesan (plainteks): A23A9
Cipherteks (mode *ECB*): 23124
Cipherteks (mode *CBC*): 27FBF

Terlihat bahwa dengan menggunakan mode *CBC*, blok plainteks yang sama (A dalam HEX) dienkripsikan menjadi dua blok cipherteks yang berbeda (masing-masing 2 dan B). Bandingkan dengan mode *EBC* yang menghasilkan blok cipherteks yang sama (2 dalam HEX) untuk dua buah blok yang sama (A).

Dengan kata lain, pada mode *CBC*, tidak ada korelasi antara posisi blok plainteks yang sams dengan posisi blok cipherteksnya.

Perambatan Kesalahan

Karena blok cipherteks yang dihasilkan selama proses enkripsi bergantung pada blok-blok cipherteks sebelumnya, maka kesalahan satu bit pada sebuah blok plainteks akan merambat pada blok cipherteks yang berkoresponden dan semua blok cipherteks berikutnya.

Tetapi, hal ini berkebalikan pada proses dekripsi. Kesalahan satu bit pada blok cipherteks hanya mempengaruhi blok plainteks yang berkoresponden dan satu bit pada blok plainteks berikutnya (pada posisi bit yang berkoresponden pula).

Kesalahan bit cipherteks biasanya terjadi karena adanya gangguan (*noise*) saluran komunikasi data selama transmisi atau *malfunction* pada media penyimpanan.

Persoalan Keamanan yang Muncul pada Mode CBC

1. Karena blok cipherteks mempengaruhi blok-blok berikutnya, pihak lawan dapat menambahkan blok cipherteks tambahan pada akhir pesan terenkripsi tanpa terdeteksi. Ini akan menghasilkan blok plainteks tambahan pada waktu dekripsi.

   Pesan moral untuk masalah ini, pengirim pesan seharusnya menstrukturkan plainteksnya sehingga ia mengetahui di mana ujung pesan dan dapat mendeteksi adanya blok tambahan.

2. Pihak lawan dapat mengubah cipherteks, misalnya mengubah sebuah bit pada suatu blok cipherteks. Tetapi hal ini hanya mempengaruhi blok plainteks hasil dekripsinya dan satu bit keslahan pada posisi plainteks berikutnya.

3. Kesimpulan

- Enkripsi adalah proses pengubahan data asli (*plaintext*) menjadi data rahasia (*ciphertext*) pada saat pengiriman (*sending*) sehingga kerahasiaan data terjaga. Sedangkan dekripsi adalah proses pengubahan data rahasia (*ciphertext*) menjadi data asli (*plaintext*) pada saat penerimaan (*receiving*) sehingga data sesuai dengan data asli yang dikirimkan.
- Menurut sejarahnya, algoritma kriptografi dapat dikategorikan menjadi 2 kelompok, yaitu sistem *cipher* klasik dan sistem *cipher* modern.
- Algoritma kriptografi klasik terbagi 2, yaitu *Cipher* Substitusi (*Substitution Ciphers*) dan *Cipher* Transposisi (*Transposition Ciphers*).
- Cipher Substitusi mengganti (menyulih atau mensubstitusi) setiap karakter dengan karakter lain dalam susunan abjad (alfabet).
- Cipher Substitusi dapat dikelompokkan menjadi 4 jenis, yaitu cipher abjad-tunggal, cipher substitusi homofonik, cipher abjad-majemuk, dan cipher substitusi poligram.
- Cipher Tansposisi memiliki algoritma yang melakukan transposisi pada karakter dalam plainteks.
- Algoritma kriptografi modern umumnya beroperasi dalam mode bit ketimbang mode karakter (seperti yang dilakukan pada *cipher* substitusi atau *cipher* transposisi dari algoritma kriptografi klasik).
- Algoritma kriptografi modern terbagi 2 kategori, yaitu Cipher Aliran (*Stream Cipher*) dan Cipher Blok (*Block Cipher*).
- *Chiper* aliran mengenkripsikan plainteks menjadi chiperteks bit per bit (1 bit setiap kali transformasi).
- Serangan pada chiper aliran adalah *Known-plaintext attack* dan *Ciphertext-only attack.*
- Pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama, biasanya 64 bit (tapi adakalanya lebih). Algoritma enkripsi menghasilkan blok cipherteks yang – pada kebanyakan sistem kriptografi simetri – berukuran sama dengan blok plainteks.
- Teknik kriptografi pada cipher blok ada 4, yaitu substitusi,tansposisi/permutasi, ekspansi, dan kompresi.
- Pada tahun 1949, Shannon mengemukakan dua prinsip (*properties*) penyandian (*encoding*) data, yaitu confusion dan diffusion. Kedua prinsip ini dipakai dalam perancangan *cipher* blok yang kuat.
- Empat mode operasi yang lazim diterapkan pada sistem blok *cipher* adalah:
  1. *Electronic Code Book* (*ECB*)
  2. *Cipher Block Chaining* (*CBC*)
  3. *Cipher Feedback* (*CFC*)
  4. *Output Feedback* (*OFB*)

DAFTAR PUSTAKA

[1] http://id.wikipedia.org/wiki/Kriptografi
Tanggal akses: 28 Desember 2006 pukul 10:00.

[2]http://www.geocities.com/amwibowo/resource/komparasi/bab3.html
Tanggal akses: 28 Desember 2006 pukul 10:00.

[3] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.
Tanggal akses: 28 Desember 2006 pukul 10:00.

[4] http://kur2003.if.itb.ac.id/
Tanggal akses: 28 Desember 2006 pukul 10:00.

[5] http://www.informatika.org/~rinaldi
Tanggal akses: 28 Desember 2006 pukul 10:00.

[6] http://hadiwibowo.wordpress.com/

Tanggal akses: 28 Desember 2006 pukul 10:00.

[7]*http://tedi.heriyanto.net/papers/p_kripto.html*
Tanggal akses: 28 Desember 2006 pukul 10:00.

[8]*http://ilmukomputer.com/umum/husni-kriptografi.php*
Tanggal akses: 28 Desember 2006 pukul 10:00.