Disusun oleh:

Ir. Rinaldi Munir, M.T.

Departemen Teknik Informatika
Institut Teknologi Bandung
2004

9. Tipe dan Mode Algoritma Simetri

9.1 Pendahuluan

Algoritma kriptografi (*cipher*) 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).

Baik *cipher* aliran maupun *cipher* blok, keduanya termasuk ke dalam algoritma kriptografi simetri.

9.2 Aliran

Chiper aliran mengenkripsikan plainteks menjadi chiperteks bit per bit (1 bit setiap kali transformasi).

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

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

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:

$$c_i = (p_i + k_i) \bmod 2 \qquad (9.1)$$

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:

$$p_i = (c_i - k_i) \bmod 2 \qquad (9.2)$$

(perhatikan bahwa untuk sistem biner, persamaan 9.2 identik dengan $p_i = (c_i + k_i) \bmod 2$)

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

$$c_i = p_i \quad k_i \qquad (9.3)$$

dan proses deksripsi menggunakan persamaan

$$p_i = c_i \quad k_i \tag{9.4}$$

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:

$$c_i = p_i \quad k_i$$

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

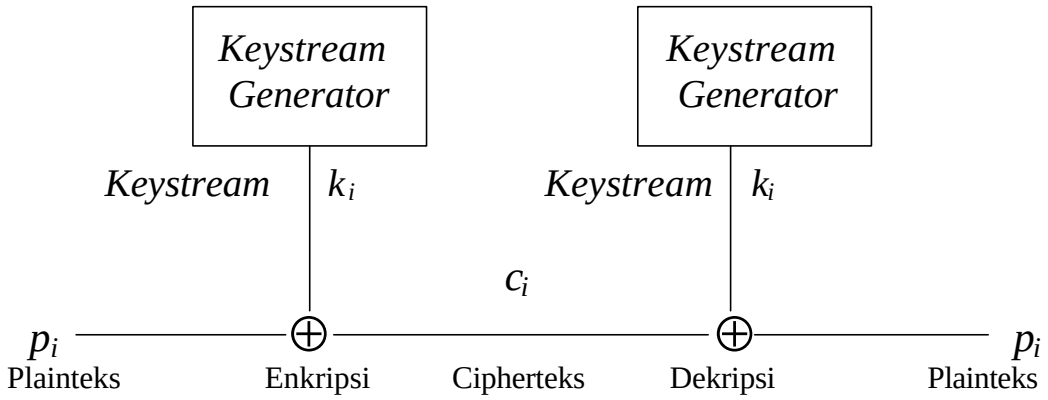
$$p_i = c_i \quad k_i$$

karena

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

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

- Gambar 9.1 memperlihatkan konsep *cipher* aliran.



Gambar 9.1 Konsep *cipher* aliran

Contoh 9.1: 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 keamanan 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.*

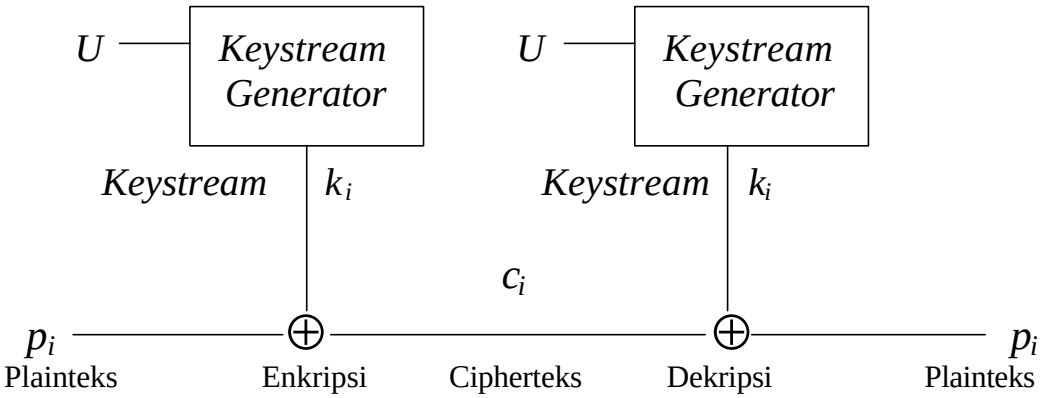
- Tingkat keamanan *cipher* aliran terletak antara algoritma XOR sederhana dengan *one-time pad*. Semakin acak keluaran yang dihasilkan oleh pembangkit aliran-bit-kunci, semakin sulit kriptanalis memecahkan cipherteks.

(Keystream Generator)

- Pembangkit aliran-bit-kunci dapat membangkitkan bit-bit kunci (*keystream*) 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 aliran-bit kunci diimplementasikan sebagai prosedur algoritmik, sehingga bit-bit kunci (*keystream*) 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 9.2). Pembangkit harus menghasilkan bit-bit kunci yang kuat secara kriptografi.



Gambar 9.2 *Cipher* aliran dengan pembangkit aliran-bit-kunci yang bergantung pada kunci *U*.

- Karena pengirim dan penerima harus menghasilkan bit-bit 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-bit kunci yang panjang.

Contoh 9.2: 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 kunci diperoleh dengan meng-*XOR*-kan bit pertama dengan bit terakhir dari empat bit sebelumnya, sehingga menghasilkan:

111101011001000

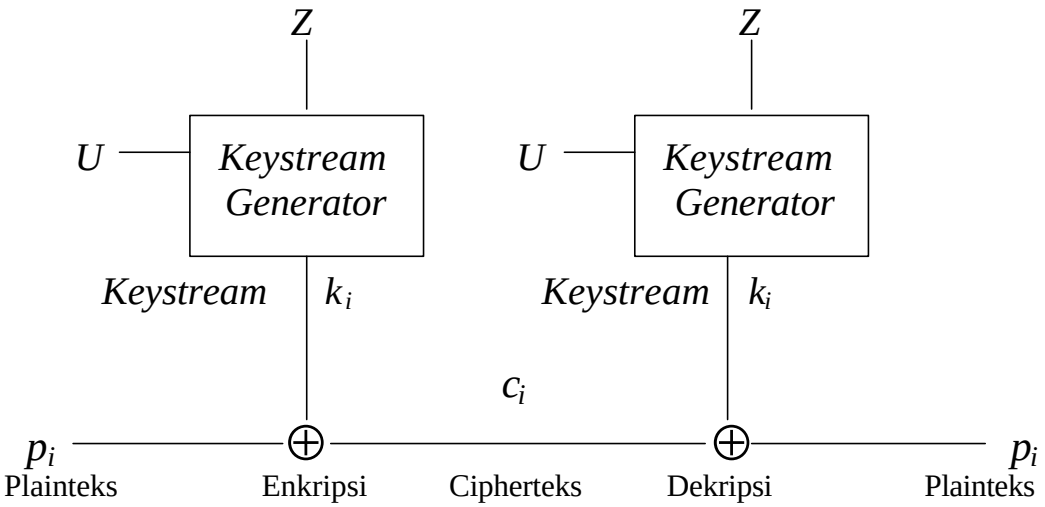
dan akan berulang setiap 15 bit.

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

- Karena *U* adalah besaran yang konstan, maka aliran bit-bit kunci yang dihasilkan pada setiap lelaran tidak berubah jika bergantung hanya pada *U*.

- Ini berarti pembangkit aliran-bit-kunci tidak boleh mulai dengan kondisi awal yang sama supaya tidak menghasilkan kembali bit-bit kunci yang sama pada setiap lelaran.

  Oleh karena itu, beberapa pembangkit aliran-bit-kunci menggunakan **umpan** (*seed*), disimbolkan dengan *Z*, atau kadang-kadang disebut juga **vektor inisialisasi** (IV), agar diperoleh kondisi awal yang berbeda pada setiap lelaran (lihat Gambar 9.3).



Gambar 9.3 *Cipher* aliran dengan pembangkit bit-aliran-kunci yang bergantung pada kunci *U* dan umpan *Z*.

- Dengan demikian, bit-bit 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-bit kunci yang berbeda pula.

- Menggunakan pasangan *Z* dan *U* yang sama dua kali dapat menyebabkan bit-bit kunci (*keystream*) yang sama pada setiap kali. Penggunaan *keystream* yang sama dua kali

memudahkan jenis penyerangan *ciphertext attack* (aka diijelaskan kemudian).

- Karena bit-bit kunci hanya bergantung pada Z dan U, maka bit-bit kunci ini tidak terpengaruh oleh kesalahan transmisi di dalam cipherteks. Jadi, kesalahan 1-bit pada transmisi cipherteks hanya menghasilkan kesalahan 1-bit pada plainteks hasil dekripsi.

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

  1. *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:

$$\begin{aligned} P \oplus C &= P \oplus (P \oplus K) \\ &= (P \oplus P) \oplus K \\ &= 0 \oplus K \\ &= K \end{aligned}$$

Contoh 9.3:

P	01100101		(karakter ‘e’)
K	00110101	$\oplus$	(karakter ‘5’)
C	01010000		(karakter ‘P’)
P	01100101	$\oplus$	(karakter ‘e’)
K	00110101		(karakter ‘5’)

2. *Ciphertext-only attack*
   Serangan ini terjadi jika *keystream* yang sama digunakan dua kali terhadap potongan plainteks yang berbeda. Serangan semacam ini disebut juga *keystream reuse attack*.

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

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus K) \oplus (P_2 \oplus K) \\ &= (P_1 \oplus P_2) \oplus (K \oplus K) \\ &= (P_1 \oplus P_2) \oplus 0 \\ &= (P_1 \oplus P_2) \end{aligned}$$

   Jika salah satu dari P_1 atau P_2 diketahui atau dapat diterka, maka *XOR*-kan salah satu plainteks tersebut dengan cipherteksnya untuk memperoleh bit-bit kunci K yang berkoresponden:

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

   Selanjutnya P_2 dapat diungkap dengan kunci K ini.

   Jika P_1 atau P_2 tidak diketahui, dua buah plainteks yang ter-*XOR* satu sama lain ini dapat diketahui dengan menggunakan nilai statistik dari pesan. Misalnya dalam teks Bahasa Inggris, dua buah spasi ter-*XOR*, atau satu spasi dengan huruf ‘e’ yang paling sering muncul, dsb. Kriptanalis cukup cerdas untuk mendeduksi kedua plainteks tersebut.

Contoh 9.4:

P_1	01100101		(karakter ‘e’)
K	00110101	$\oplus$	(karakter ‘5’)
C_1	01010000		(karakter ‘P’)
P_2	01000010		(karakter ‘B’)
K	00110101	$\oplus$	(karakter ‘5’)
C_2	01110111		(karakter ‘w’)

$P_1 \oplus P_2$ = 01100101 $\oplus$ 01000010 = 00100111
$C_1 \oplus C_2$= 01010000 $\oplus$ 01110111 = 00100111

Perhatikan bahwa $P_1 \oplus P_2 = C_1 \oplus C_2$

Jika P_1 atau P_2 telah diketahui, maka *XOR*-kan plainteks tersebut dengan cipherteks yang berkoresponden untuk memperoleh K (seperti pada Contoh 9.3).

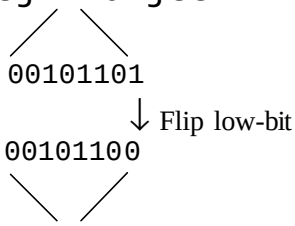
- Pesan moral dari dua serangan di atas adalah: pengguna *cipher* aliran harus mempunyai bit-bit kunci yang tidak dapat diprediksi sehingga mengetahui sebagian dari bit-bit kunci kunci tidak memungkinkan kriptanalis dapat mendeduksi bagian sisanya.

3. Flip-bit attack

Serangan ini tidak bertujuan menemukan kunci atau mengungkap plainteks dari cipherteks, tetapi mengubah bit cipherteks tertentu sehingga hasil dekripsinya berubah. Pengubahan dilakukan dengan membalikkan (*flip*) bit tertentu (0 menjadi 1, atau 1 menjadi 0).

Contoh 9.5:

P : `QT-TRNSFR US $00010,00 FRM ACCNT 123-67 TO`
C: `uhtr07hjLmkyR3j7 kdhj38lkkldkYtr#)oknTkRgh`



C: `uhtr07hjLmkyR3j7 kdhj38lkkldkYtr#)oknTkRgh`
P : `QT-TRNSFR US $10010,00 FRM ACCNT 123-67 TO`

Pengubahan 1 bit U dari cipherteks sehingga menjadi T.
Hasil dekripsi: $10,00 menjadi $ 10010,00

Pengubah pesan tidak perlu mengetahui kunci, ia hanya perlu mengetahui posisi pesan yang diminati saja.

Serangan semacam ini memanfaatkan karakteristik *cipher* aliran yang sudah disebutkan di atas, bahwa kesalahan 1-bit pada cipherteks hanya menghasilkan kesalahan 1-bit pada plainteks hasil dekripsi.

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

  1. Mengenkripsikan data pada saluran yang menghubungkan antara dua buah komputer.
  2. Mengenkripsikan suara 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*).