Bahan Kuliah ke-10

IF5054 Kriptografi

Tipe dan Mode Algoritma Simetri
(Bagian 2)

Disusun oleh:

Ir. Rinaldi Munir, M.T.

Departemen Teknik Informatika
Institut Teknologi Bandung
2004

9.3 *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).

- Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci (yang ukurannya sama dengan ukuran blok plainteks). Algoritma enkripsi menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks.

- Dekripsi dilakkan dengan cara yang serupa seperti enkripsi.

- 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

  $$E_K(P) = C$$

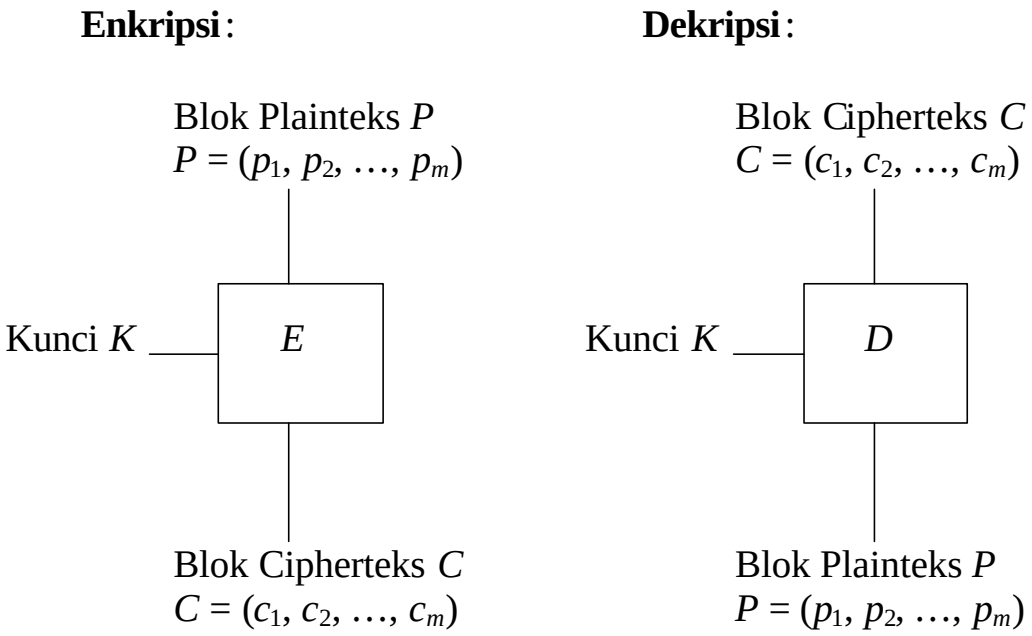
  untuk enkripsi, dan

  $$D_K(C) = P$$

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

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

  Skema enkripsi dan dekripsi dengan *cipher* blok digambarkan pada Gambar 9.4. Fungsi E dan D dispesifikasikan oleh kriptografer.



Gambar 9.4 Skema enkripsi dan dekripsi pada *cipher* blok

9.3.1 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.

9.3.2 *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.

   Prinsip *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.

9.3.3 *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* (*CFB*)
  4. *Output Feedback* (*OFB*)

9.3.4 *Electronic Code Book (ECB)*

- Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen meknajdi blok cipherteks C_i .

- Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

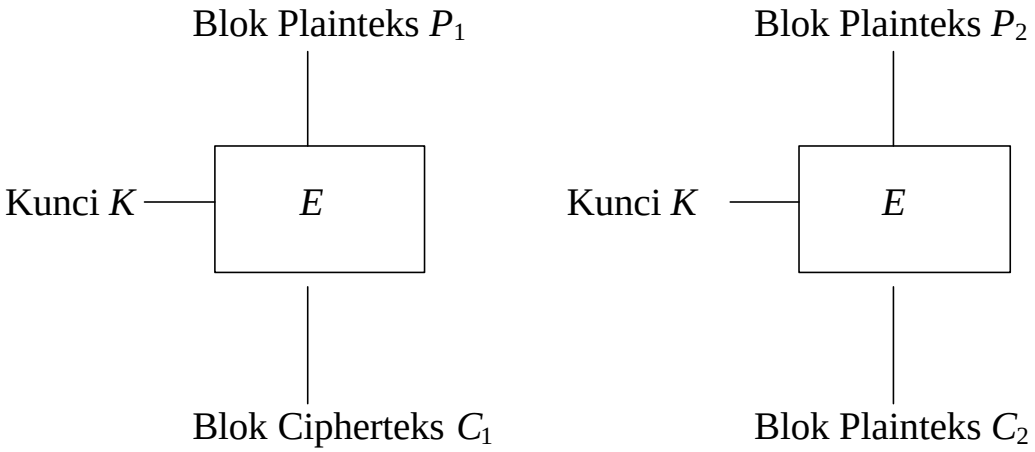
$$C_i = E_K(P_i)$$

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 9.4 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 9.4 Skema enkripsi dan dekripsi dengan mode *ECB*

Contoh 9.6: Misalkan plainteks (dalam biner) adalah

```
1010001000111010 1001
```

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 adalah (panjangnya juga 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

```
1010001000111010 1001
```
(A23A9 dalam notasi HEX)

adalah

```
0010001100010010 0100
```
(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 9.7. 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.*

9.3.5 *Cipher Block Chaining (CBC)*

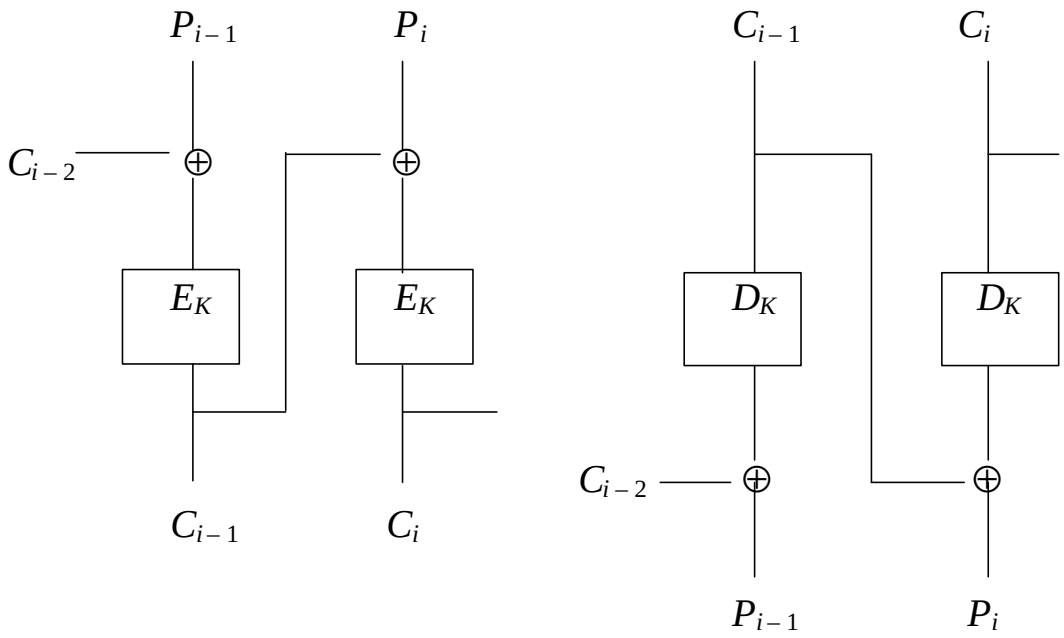
- Mode ini menerapkan mekanisme umpan-balik (*feedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok yang *current*.

- 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 9.5 memperlihatkan skema mode operasi *CBC*.



Enkripsi **Dekripsi**

Gambar 9.5 Skema enkripsi dan dekripsi dengan mode *CBC*

- Secara matematis, enkripsi dengan mode *CBC* dinyatakan sebagai

$$C_i = E_K(P_i \oplus C_{i-1})$$

dan dekripsi sebagai

$$P_i = D_K(C_i) \oplus C_{i-1}$$

Yang dalam hal ini, $C_0 = IV$ (*initialization vector*). *IV* dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program.

Jadi, untuk menghasilkan blok cipherteks pertama (C_1) , *IV* digunakan untuk menggantikan blok cipherteks sebelumnya, C_0.
Sebaliknya pada dekripsi, blok plainteks diperoleh dengan cara meng-*XOR*-kan *IV* dengan hasil dekripsi terhadap blok cipherteks pertama.

- Perhatikan bahwa enkripsi terhadap blok *i* adalah fungsi dari semua plainteks dari blok 0 sampai blok *i* – 1, sehingga blok plianteks yang sama menghasilkan blok cipherteks yang berbeda hanya jika blok-blok plainteksnya sebelumnya berbeda.

- Jika blok-blok plainteks sebelumnya ada yang sama, maka ada kemungkinan cipherteksnya sama. Untuk mencegah hal ini, maka digunakan *IV* yang merupakan data acak sebagai blok pertama. *IV* tidak mempunyai makna, ia hanya diguanakan untuk membuat tiap blok cipherteks menjadi unik.

Contoh 9.8. Tinjau kembali plainteks dari Contoh 9.6:

```
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 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.

C_1 diperoleh sebagai berikut:

$P_1 \oplus C_0$ = 1010 ⊕ 0000 = 1010

Enkripsikan hasil ini dengan fungsi *E* sbb:
1010 ⊕ *K* = 1010 ⊕ 1011 = 0001
Geser (*wrapping*) hasil ini satu bit ke kiri: 0010
Jadi, C_1 = 0010 (atau 2 dalam HEX)

C_2 diperoleh sebagai berikut:

$P_2 \oplus C_1$ = 0010 ⊕ 0010 = 0000
0000 ⊕ *K* = 0000 ⊕ 1011 = 1011
Geser (*wrapping*) hasil ini satu bit ke kiri: 0111
Jadi, C_2 = 0111 (atau 7 dalam HEX)

C_3 diperoleh sebagai berikut:

$P_3 \oplus C_2$ = 0011 ⊕ 0111 = 0100
0100 ⊕ *K* = 0100 ⊕ 1011 = 1111
Geser (*wrapping*) hasil ini satu bit ke kiri: 1111
Jadi, C_2 = 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 sama dengan posisi blok cipherteksnya.

Keuntungan Mode CBB

- Karena blok-blok plainteks yang sama tidak menghasilkan blok-blok cipherteks yang sama, maka kriptanalisis menjadi lebih sulit. Inilah alasan utama penggunaan mode *CBC* digunakan.

Kelemahan Mode CBC

- 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.