STUDI & IMPLEMENTASI ALGORITMA TRIPLE DES

Anugrah Adeputra – NIM : 13505093

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15093@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi dan implementasi salah satu algoritma *block cipher,* yaitu *Triple Data Encryption Standard* (*Triple DES*) untuk menyandikan data atau pesan. *Triple Data Encryption Standard* (*Triple DES*) merupakan sebuah algoritma kriptografi simetri yang beroperasi dalam bentuk blok. *Triple DES* mendukung beberapa pilihan panjang kunci. Implementasi *Triple DES* dalam makalah ini meliputi empat mode operasi yaitu mode operasi *Electronic Code Book* (*ECB*), *Cipher Block Chaining* (*CBC*), *Cipher Feedback* (*CFB*), dan *Output Feedback* (*OFB*).

Sebuah perangkat lunak bernama *Triple DES Encryptor* dibangun untuk implementasi algoritma kriptografi *Triple DES ini* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB*. Perangkat lunak *Triple DES Encryptor* tersebut dikembangkan dengan menggunakan *tool* pengembangan *NetBeans IDE* 6.0 dan terutama sekal memanfaatkan JCE (Java Cryptography Extension), dalam lingkungan pengembangan sistem operasi *Windows,* namun implementasinya masih sebatas console agar mudah dilakukan pengubahan. Perangkat lunak *Triple DES Encryptor* ditujukan untuk mendukung penyandian segala jenis arsip, namun implementasinya hingga tahap pengujian baru berhasil mencapai pemrosesan untuk teks saja.

Hasil uji menunjukkan bahwa algoritma *Triple DES* merupakan salah satu solusi yang baik untuk mengatasi masalah keamanan dan kerahasiaan data. Selain itu, implementasi *Triple DES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB* memiliki keuntungan dan kelemahannya masing-masing.

Kata kunci: *Triple Data Encryption Standard*, *Electronic Code Book*, *Cipher Block Chaining*, *Cipher Feedback*, *Output Feedback*, *Triple DES Encryptor*, Enkripsi, Dekripsi.

1. Pendahuluan

Di Era Globalisasi seperti sekarang ini, arus informasi merupakan suatu hal yang memegang peranan penting. Bahkan ada yang mengatakan bahwasanya jika ada yang mampu menguasai jaringan informasi, maka dia akan mampu menguasai dunia. Sehubungan dengan hal tersebut, banyak juga pihak-pihak yang berusaha mencuri atau mengakses informasi yang pihak tersebut tidak memiliki hak untuk melakukan akses terhadap informasi itu. Kriptografi selama ini memegang peranan penting dalam mengatasi masalah tersebut. Untuk memenuhi hal tersebut, dalam kriptografi, terdapat proses untuk menyandikan (enkripsi dan dekripsi) data atau informasi yang akan dikirimkan. Enkripsi dilakukan pada saat pengiriman informasi dengan cara mengubah atau menyandikan informasi dengan suatu mekanisme tertentu sedangkan dekripsi dilakukan pada saat penerimaan informasi dengan cara mengubah informasi yang telah disandikan menjadi informasi asalnya. Proses Dekripsi hanya dapat dilakukan oleh penerima dengan menggunakan kunci rahasia yang sebelumnya telah disepakati bersama.

Algoritma penyandian data yang telah dijadikan standard sejak tahun 1977 adalah *Data Encryption Standard* (*DES*) setelah disetujui oleh *National Bureau of Standard(NBS)* dan setelah dinilai kekuatannya oleh *National Security Agency(NSA)*. Algoritma *DES* dikembangkan di IBM di bawah kepemimpinan W.L. Tuchman pada tahun 1972. Kekuatan *DES* saat itu terletak pada panjang kuncinya yaitu 56-bit. Akibat perkembangan teknologi yang begitu pesat, *DES*, dalam beberapa hal, terbukti kurang

dalam hal jaminan aspek keamanan. Perangkat keras khusus yang bertujuan untuk menentukan kunci 56-bit *DES* hanya dalam waktu beberapa jam sudah dapat dibangun. Dan pada tahun 1998, Electronic Frontier Foundation menggunakan suatu komputer yang dikembangkan secara khusus yang bernama DES Cracker, dalam waktu kurang dari tiga hari telah mampu untuk memecahkan DES. Beberapa pertimbangan tersebut telah manandakan bahwa diperlukan sebuah standard algoritma baru dan kunci yang lebih panjang. Setelah itu, dibuatlah beberapa pengembangan dari DES dengan cara memperbesar ruang kunci. Varian pengembangan DES yang paling dikenal adalah DES Berganda, yakni pemanfaatan DES berkali-kali untuk proses enkripsi dan dekripsinya. Double DES mempunyai kelemahan yaitu ia dapat diserang dengan algoritma yang dikenal sebagai *meet-in-the-middle-attack*, yang pertama kali ditemukan oleh Diffie dan Hellman. Sebagai bentuk pencegahan terhadap serangan tersebut, maka digunakanlah tiga kali langkah DES. Bentuk tersebut dinamakan sebagai Triple DES.

Beberapa mode operasi yang dapat diterapkan pada algoritma kriptografi penyandi blok *Triple DES* di antaranya adalah *Electronic Code Book* (*ECB*), *Cipher Block Chaining* (*CBC*), *Cipher Feedback* (*CFB*), dan *Output Feedback* (*OFB*). Implementasi *AES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB* tentu saja memiliki kelebihan dan kekurangan tertentu dalam aspek tingkat keamanan data.

2. Tipe dan Mode Algoritma Simetri

Algoritma kriptografi (*cipher*) simetri dapat dikelompokkan menjadi dua kategori, yaitu:

1. *Cipher* aliran (*ctream 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.

2.1 *Cipher* Blok

Pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama, biasanya 64 bit(adakalanya lebih). Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci (yang ukurannya sama dengan blok plainteks). Algoritma enkripsi menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks. Dekripsi dilakukan dengan cara yang serupa seperti enkripsi.
Misalkan blok plainteks (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_1, p_2, ..., p_m)$$

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

$$C = (c_1, c_2, ..., c_m)$$

yang dalam hal ini c_i adalah bit 0 atau bit 1 untuk $i = 1, 2, \ldots, m$.

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

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

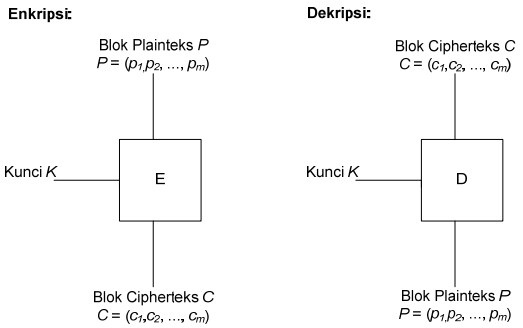
Enkripsi dengan kunci K dinyatakan dengan persamaan

$$E_k(P) = C,$$

sedangkan dekripsi dengan kunci K dinyatakan dengan persamaan

$$D_k(C) = P$$

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



Gambar 1 Skema Enkripsi dan Dekripsi dengan *Cipher* Blok

2.2 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*)

Sebenarnya masih terdapat beberapa pengembangan mode lainnya, di antaranya CTS,CTR,CFB8,OFB8, namun tidak diterapkan pada implementasi ini.

2.2.1 *Electronic Code Book* (*ECB*)

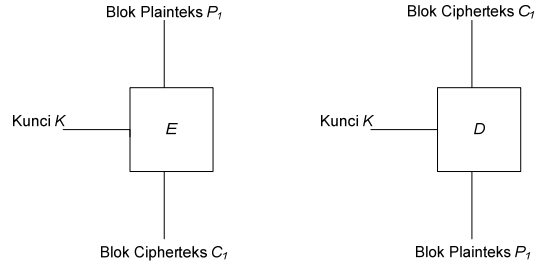
Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen menjadi blok cipherteks C_i. Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

$$C_i = E_k(P_i)$$

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*. Skema enkripsi dan dekripsi dengan mode *ECB* dapat dilihat pada Gambar 2.



Gambar 2 Skema Enkripsi dan Dekripsi dengan Mode *ECB*

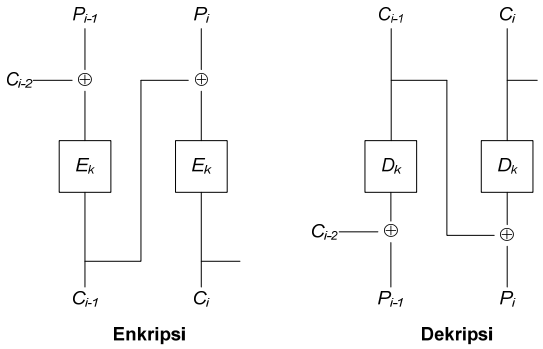
Ada kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan. Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada blok-blok lainnya. Satu cara untuk mengatasi hal ini adalah dengan *padding*, yaitu menambahkan blok terakhir dengan pola bit yang teratur agar panjangnya sama dengan ukuran blok yang ditetapkan.

Padding yang digunakan dalam pengimplementasian algoritma Triple DES ini adalah PKCS di mana konsep dasar dari algoritma padding ini adalah mengisikan bit yang kurang dari sebuah blok dengan bilangan yang merupakan jumlah sisa blok yang kosong, misalnya sisa blok kosongnya 3, maka sisa 3 bit akan diisi oleh 3 buah bit 3, dst.

2.2.2 *Cipher Block Chaining* (*CBC*)

Mode ini menerapkan mekanisme umpan balik (*feedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya diumpanbalikkan 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, kemudia meng-*XOR*-kan hasilnya dengan blok cipherteks sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan maju (*feedforward*) pada akhir proses dekripsi. Skema enkripsi dan dekripsi dengan mode *CBC* dapat dilihat pada Gambar 3.



Gambar 3 Enkripsi dan Dekripsi dengan Mode *CBC*

Secara matematis, enkripsi dengan mode *CBC* dinyatakan sebagai

$$C_i = E_k(P_1 \oplus C_{i-1})$$

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.

Pada mode *CBC*, blok plainteks yang sama menghasilkan blok cipherteks yang berbeda hanya jika blok-blok plainteks sebelumnya berbeda.

2.2.3 *Cipher-Feedback* (*CFB*)

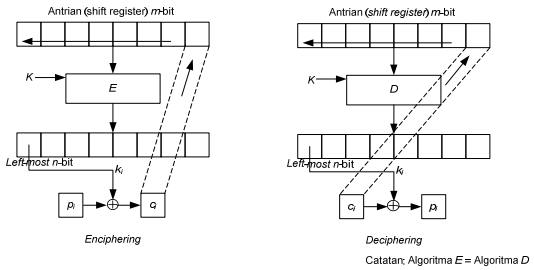
Pada mode *CFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *CFB*-nya disebut *CFB* 8-bit. Secara umum *CFB n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Mode *CFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.

Tinjau mode *CFB n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks menjadi *n*-bit pertama dari cipherteks. Salinan (*copy*) *n*-bit dari cipherteks ini dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.
3. *m*-*n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

Sedangkan, algoritma dekripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks. Salinan (*copy*) *n*-bit dari cipherteks dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m*-*n* lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.
3. *m*-*n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.



Gambar 4 Mode *CFB* n-bit

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. Mode *CFB n*-bit yang bekerja pada blok berukuran *m*-bit dapat dilihat pada Gambar 4.

Secara formal, mode *CFB n*-bit dapat dinyatakan sebagai:

Proses Enkripsi:

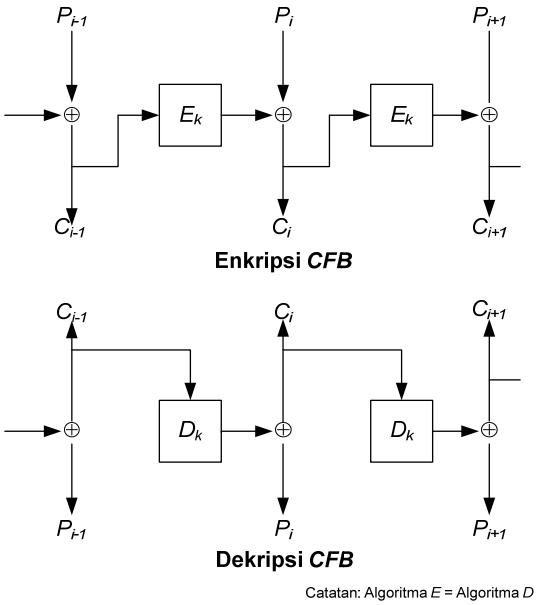
$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

Proses Dekripsi:

$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

yang dalam hal ini:

X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte



Gambar 5 Enkripsi dan Dekripsi Mode *CFB* *n*-bit untuk blok *n*-bit

Jika $m = n$, maka mode *CFB* *n*-bit adalah seperti pada Gambar 5. *CFB* menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya. Skema enkripsi dan dekripsi dengan mode *CFB* dapat dilihat pada Gambar 5.
Dari Gambar 5 dapat dilihat bahwa:

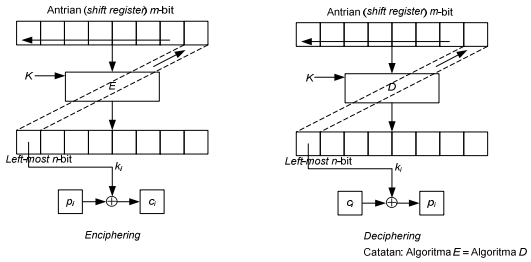
$$C_i = P_i \oplus E_k(C_{i-1})$$
$$P_i = C_i \oplus D_k(C_{i-1})$$

IV pada *CFB* tidak perlu dirahasiakan. *IV* harus unik untuk setiap pesan, sebab *IV* yang sama untuk setiap pesan yang berbeda akan menghasilkan *keystream* k_i yang sama.

2.2.4 *Output-Feedback* (*OFB*)

Pada mode *OFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *OFB*-nya disebut *OFB* 8-bit. Secara umum *OFB* *n*-bit mengenkripsi plainteks sebanyak n bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Mode *OFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.
Tinjau mode *OFB* *n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *OFB* adalah sebagai berikut (lihat Gambar 6):

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil enkripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks menjadi *n*-bit pertama dari cipherteks.
3. *m*-*n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.



Gambar 6 Mode *OFB* n-bit

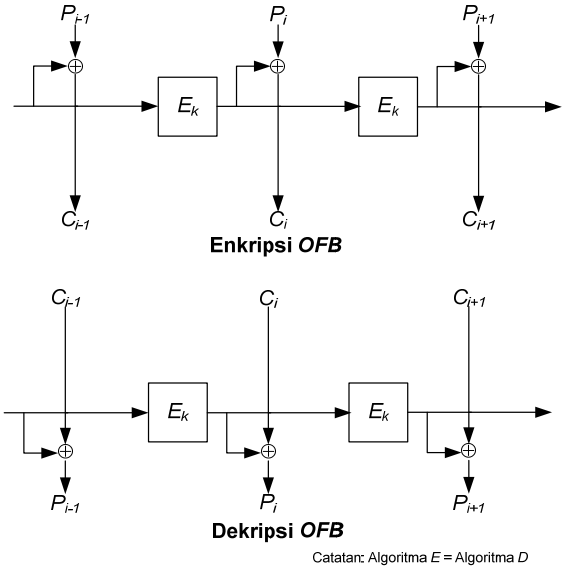
Sedangkan, algoritma dekripsi dengan mode *OFB* adalah sebagai berikut (lihat Gambar 6):

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi dimasukkan ke dalam antrian

(menempati *n* posisi bit paling kanan antrian), dan *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil dekripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks.

3. *m*-*n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. Mode *OFB n*-bit yang bekerja pada blok berukuran *m*-bit dapat dilihat pada Gambar 6.



Gambar 7 Enkripsi dan Dekripsi *OFB n*-bit untuk blok n-bit

Secara formal, mode *OFB n*-bit dapat dinyatakan sebagai:

Proses Enkripsi:

$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$$

Proses Dekripsi:

$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$$

yang dalam hal ini:

X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = *Most Significant Byte*
LSB = *Least Significant Byte*

Jika *m* = *n*, maka mode *OFB n*-bit adalah seperti pada Gambar 6. *OFB* menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya. Skema enkripsi dan dekripsi dengan mode *OFB* dapat dilihat pada Gambar 7.

3. Triple *Data Encryption Standard* (*Triple DES*)

3.1 Bentuk Umum Triple DES

Konsep Triple DES sebenarnya sama dengan DES, namun terdapat beberapa pengembangan, yaitu:

Bentuk umum TDES (mode EEE):

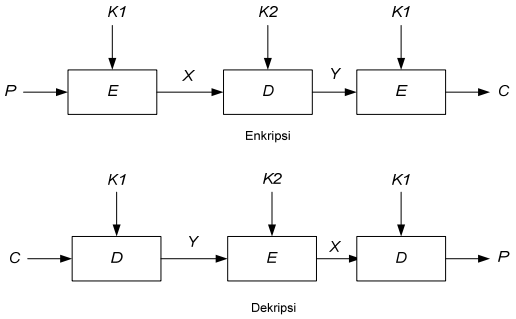
Enkripsi: $C = E_{K3}(E_{K2}(E_{K1}(P)))$

Dekripsi: $P = D_{K1}(D_{K2}(D_{K3}(C)))$

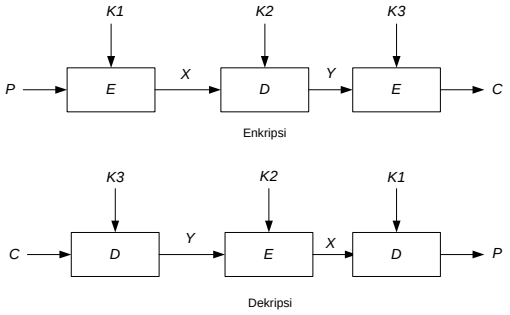
Untuk menyederhanakan TDES, maka langkah di tengah diganti dengan D (mode EDE).

Ada dua versi TDES dengan mode EDE:
- Menggunakan 2 kunci
- Menggunakan 3 kunci

Berikut merupakan skema Triple DES dengan 2 kunci:



Dan di bawah ini skema Triple DES dengan 3 kunci:



4. Java Cryptography Extension

The Java™ Cryptography Extension (JCE) menyediakan framework dan implementasi untuk enkripsi, pembangkitan kunci, persetujuan kunci, Algoritma Kode Autentikasi Pesan,dll.

5. Pengujian

Text: "Ini text default untuk ujicoba tugas makalah kriptografi"
Hex setelah bytenyadikonversi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

2 kunci Mode ECB:

Kunci 1: abcdefgh
Kunci 2: ijklmnop

Ciphertext:
4b1d29973e21d14cd63c4141e5215f955
fd69ea77827527f14ac8dfc1288f8f0bbc
10b6f7c80d6b30d96e03223be9a28242f
5ed0a5ca1c60f5c9919b420fd3ac (64 bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

3 kunci Mode ECB:

Kunci 1: abcdefgh
Kunci 2: ijklmnop
Kunci 3: qrstuvwx

Ciphertext:
714fb67d0980f744a7385668978986cc3
1bec68b877c6e4280dd517bd267c6204
534868e444ff6e3104e69d0d0d962c735
d0351946424237154 08d6fe15805a3
(64 bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

2 kunci Mode CBC

Kunci 1: abcdefgh
Kunci 2: ijklmnop

Ciphertext:
5d13d0e8cfb8fd274fc6eccfca87a3ae15c
b2882f5f1c1f353b093c3332974048acd
daf262c000753ce23979d75b1d16f10e0
ddf0ed84e27271cb11f4dd82455 (64 bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

3 kunci Mode CBC:

Kunci 1: abcdefgh
Kunci 2: ijklmnop
Kunci 3: qrstuvwx

Ciphertext:
bdf0ea60d68f8f7528a00723fda508b02a
38f2e86e3f0fe779246558bee204eb7b52
375539cca2a70d0ef04d558a9ccd375bd
0dc686d501421634ae499255226 (64 bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

2 kunci mode CFB:

Kunci 1: abcdefgh
Kunci 2: ijklmnop

Ciphertext:

805bb5984fbdf2c27c82f1d2eeca128b9d
b63b1cc2e4370a0239fdba3a9104a2f95
3c0f6f819701246b4e607edc20c7b0bc7f
72b12e3110c9cb1e67ab7dec635(64
bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

3 kunci mode CFB

Kunci 1: abcdefgh
Kunci 2: ijklmnop
Kunci 3: qrstuvwx

Ciphertext:
e28fd45a7bcb75ac724f5def45631c5552
05b86378d35abba5e5c9c089e7c2cbd1f
fa7ab97f67e9bb4943ff2eb024f12924c3
5ef8410b7adcff845dd21f189ae (64
bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

2 kunci mode OFB

Kunci 1: abcdefgh
Kunci 2: ijklmnop

Ciphertext:
805bb5984fbdf2c2389ea4b8b1549aa70
67cb11b2fb82bd059950e10c39f0e5743
33ef6d96b5b545a8c49bdd9cba079eae3
5a8033139e22af17f2ecf23cc8333 (64
bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

3 kunci mode OFB

Kunci 1: abcdefgh
Kunci 2: ijklmnop
Kunci 3: qrstuvwx

Ciphertext:
e28fd45a7bcb75ac31fdf400c5a2fb38ed
ff26b795058929dbc1a375d35925894f3
58dd89707a671205a41e1406f8b2af761
3c703f0918df58aa1adf8f46eda5 (64
bytes)

Hasil Dekripsi:
496e6920746578742064656661756c74
20756e74756b20756a69636f62612074
75676173206d616b616c6168206b7269
70746f6772616669 (56 bytes)

Kesimpulan:
Untuk Proses Enkripsi dan Dekripsi untuk keempat mode serta penggunaan 2 kunci maupun 3 kunci sudah dapat diimplementasi dengan baik, dapat dilihat dari hasil pengujian, tidak terjadi kesalahan, sehingga bisa dikatakan implementasi yang dilakukan sudah cukup baik.

Pemanfaatan JCE dapat membantu implementasi dan pengembangan dari suatu program aplikasi yang terkait dengan kriptografi.

DAFTAR PUSTAKA

Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html