Tugas Makalah I (Pengganti UTS)
IF5054 Kriptografi, Sem. I Tahun 2006/2007

Buatlah makalah yang berisi *technical report* yang berkaitan dengan salah satu dari topik kriptografi di bawah ini:

1. Jenis-jenis serangan (*attack*) pada kriptografi
2. Algoritma kriptografi klasik (misal *Caesar cipher*, Vigenere *cipher*, Playfair *cipher*, dll)
3. Kriptanalisis (analisis frekuensi, *differential analysis*, dll)
4. Algoritma kriptografi modern (*stream cipher* dan *block cipher*)
5. Beberapa algoritma *cipher* blok (misal *DES*, *TDES*, *GOST*, *RC5*, *AES*, dll)
6. Steganografi dan *watermarking*

Makalah dapat berupa:

- Mengulas secara tuntas algoritma kriptografi kunci-simetri tertentu, termasuk perbandinganya dengan algoritma yang sejenis. Sukur-sukur ada program tes yang menguji performansi an keamanannya.
- Mengulas sistem keamanan data dan informasi pada suatu *platform*/*tools*/aplikasi, dsb
- Mengulas aplikasi sistem kriptografi kunci-simetri di bidang tertentu
- Rancangan algoritma kriptografi kunci-simetri yang diusulkan sendiri, lengkap dengan konsep, implementasi, dan pengujiannya.
- Dll

Contoh-contoh judul makalah:

1. Studi dan perbandingan algoritma simetri Camellia dengan *DES*
2. Studi mengenai *Identity-based Encryption*
3. Keamanan pada jaringan VoIP
4. Studi mengenai *spread spectrum steganography* dan aplikasinya
5. *Windows 2000 Encryption File System*
6. dll

Sebelum membuat makalah, anda diharuskan menyusun proposal (format bebas) makalah yang akan anda buat. Proposal setidaknya berisi abstraksi, latar belakang, rumusan masalah, batasan masalah, dll, termasuk daftar pustaka.

Proposal diserahkan kepada dosen IF5054 untuk diperiksa dan disetujui. Penyerahan proposal paling lambat 2 minggu sebelum UTS. Makalah dikumpulkan tepat pada saat UTS Kriptografi (sesuai jadwal).

Makalah ditulis dengan ketentuan berikut:

1. *Font* = *Times New Roman*, Ukuran *font* = 10
2. Lebar spasi = 1
3. Format 2 kolom (lihat contoh)
4. Jumlah halaman minimal = 15 halaman

Makalah tidak boleh sama dengan makalah yang sudah dibuat pada tahun-tahun sebelumnya, selain itu belum pernah diberikan di dalam kuliah.

STUDI DAN IMPLEMENTASI *ADVANCED ENCRYPTION STANDARD* DENGAN EMPAT MODE OPERASI *BLOCK CIPHER*

Chan Lung – NIM : 13501039

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if11039@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi dan implementasi *Advanced Encryption Standard* (*AES*) untuk menyandikan data yang disimpan dalam media penyimpanan. *Advanced Encryption Standard* (*AES*) merupakan sebuah algoritma kriptografi simetri yang beroperasi dalam bentuk blok 128-bit. *AES* mendukung panjang kunci 128-bit, 192-bit, dan 256-bit. Implementasi *AES* 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 *AESEncryptor* dibangun untuk implementasi algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB*. Perangkat lunak *AESEncryptor* dikembangkan dengan menggunakan *tool* pengembangan *Borland Delphi 7.0* dalam lingkungan pengembangan sistem operasi *Windows*. Perangkat lunak *AESEncryptor* mendukung penyandian sembarang arsip berukuran sembarang.

Perangkat lunak *AESEncryptor* tersebut kemudian digunakan untuk membandingkan tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB*. Tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB* diuji dengan melakukan beberapa proses manipulasi terhadap arsip hasil enkripsi seperti pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks. Kemudian, dilakukan proses dekripsi terhadap arsip hasil enkripsi *AESEncryptor* yang telah dimanipulasi tersebut untuk dibandingkan plainteksnya dengan plainteks arsip asal. Hasil uji menunjukkan bahwa algoritma *AES* merupakan salah satu solusi yang baik untuk mengatasi masalah keamanan dan kerahasiaan data. *AES* juga dapat diimplementasikan secara efisien sebagai perangkat lunak dengan implementasi menggunakan tabel. Selain itu, implementasi *AES* dengan mode operasi *ECB*, *CBC*, *CFB*, dan *OFB* memiliki keuntungan dan kelemahannya masing-masing.

***Kata kunci**: Advanced Encryption Standard, electronic code book, cipher block chaining, cipher feedback, output feedback, AESEncryptor*, enkripsi, dekripsi.

1. Pendahuluan

Pengiriman data dan penyimpanan data melalui media elektronik memerlukan suatu proses yang dapat menjamin keamanan dan keutuhan dari data yang dikirimkan tersebut. Data tersebut harus tetap rahasia selama pengiriman dan harus tetap utuh pada saat penerimaan di tujuan. Untuk memenuhi hal tersebut, dilakukan proses penyandian (enkripsi dan dekripsi) terhadap data yang akan dikirimkan. Enkripsi dilakukan pada saat pengiriman dengan cara mengubah data asli menjadi data rahasia sedangkan dekripsi dilakukan pada saat penerimaan dengan cara mengubah data rahasia menjadi data asli. Jadi data yang dikirimkan selama proses pengiriman adalah data rahasia, sehingga data asli tidak dapat diketahui oleh pihak yang tidak berkepentingan. Data asli hanya dapat diketahui oleh penerima dengan menggunakan kunci rahasia.

Algoritma penyandian data yang telah dijadikan standard sejak tahun 1977 adalah *Data Encryption Standard* (*DES*). Kekuatan *DES* ini terletak pada panjang kuncinya yaitu 56-bit. Perkembangan kecepatan perangkat keras dan meluasnya penggunaan jaringan komputer terdistribusi mengakibatkan penggunaan *DES*, dalam beberapa hal, terbukti sudah tidak aman

dan tidak mencukupi lagi terutama dalam hal yang pengiriman data melalui jaringan internet. Perangkat keras khusus yang bertujuan untuk menentukan kunci 56-bit *DES* hanya dalam waktu beberapa jam sudah dapat dibangun. Beberapa pertimbangan tersebut telah manandakan bahwa diperlukan sebuah standard algoritma baru dan kunci yang lebih panjang.
Pada tahun 1997, *the U.S. National Institue of Standards and Technology* (*NIST*) mengumumkan bahwa sudah saatnya untuk pembuatan standard algoritma penyandian baru yang kelak diberi nama *Advanced Encryption Standard* (*AES*). Algoritma *AES* ini dibuat dengan tujuan untuk menggantikan algoritma *DES* yang telah lama digunakan dalam menyandikan data elektronik. Setelah melalui beberapa tahap seleksi, algoritma *Rijndael* ditetapkan sebagai algoritma kriptografi *AES* pada tahun 2000. Algoritma *AES* merupakan algoritma kriptografi simetrik yang beroperasi dalam mode penyandi blok (*block cipher*) yang memproses blok data 128-bit dengan panjang kunci 128-bit (*AES*-128), 192-bit (*AES*-192), atau 256-bit (*AES*-256).

Beberapa mode operasi yang dapat diterapkan pada algoritma kriptografi penyandi blok *AES* 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 [3]. 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

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

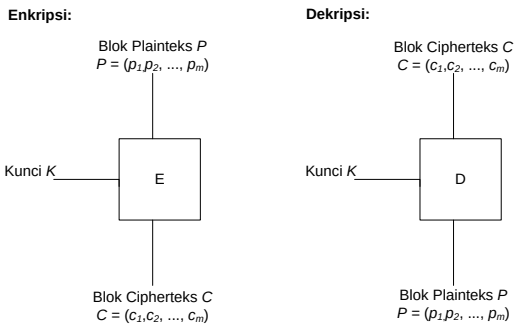
Enkripsi dengan kunci K dinyatakan dengan persamaan

$$E_k(P) = C,$$

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*)

2.2.1 *Electronic Code Book* (*ECB*)

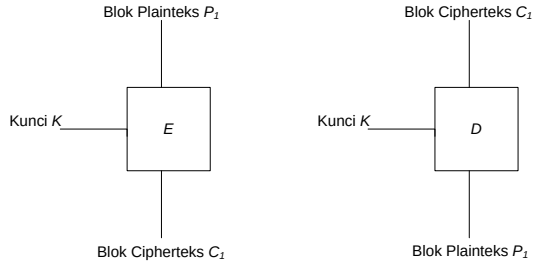
Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen menjadi blok cipherteks C_i. Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

$$C_i = E_k(P_i)$$

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.



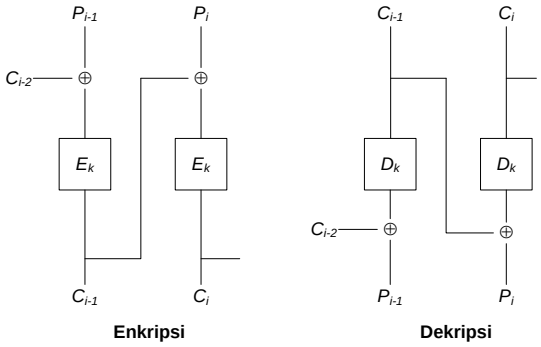
Gambar 2 Skema Enkripsi dan Dekripsi dengan Mode *ECB*

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.

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*)

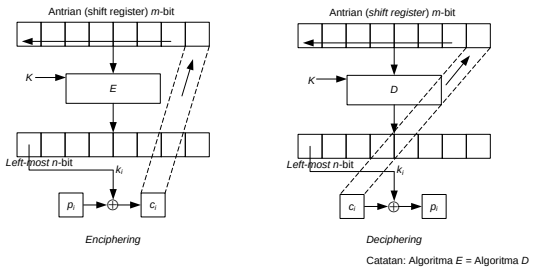
Pada mode *CFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *CFB*-nya disebut *CFB* 8-bit. Secara umum *CFB n*-bit mengenkripsi 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:

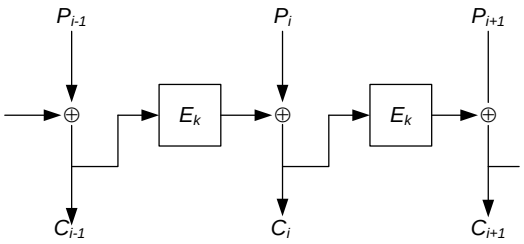
$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \parallel C_i$$

Proses Dekripsi:

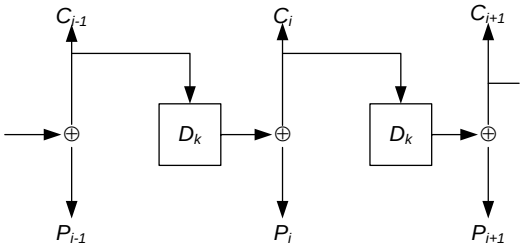
$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

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



Enkripsi *CFB*



Dekripsi *CFB*

Catatan: Algoritma *E* = Algoritma *D*

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:

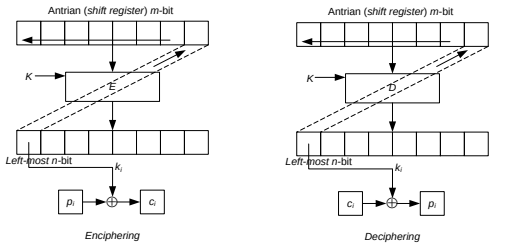
$$C_i = P_i \oplus E_k(C_{i-1})$$
$$P_i = C_i \oplus D_k(C_{i-1})$$

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.



Catatan: Algoritma *E* = Algoritma *D*

Gambar 6 Mode *OFB* n-bit

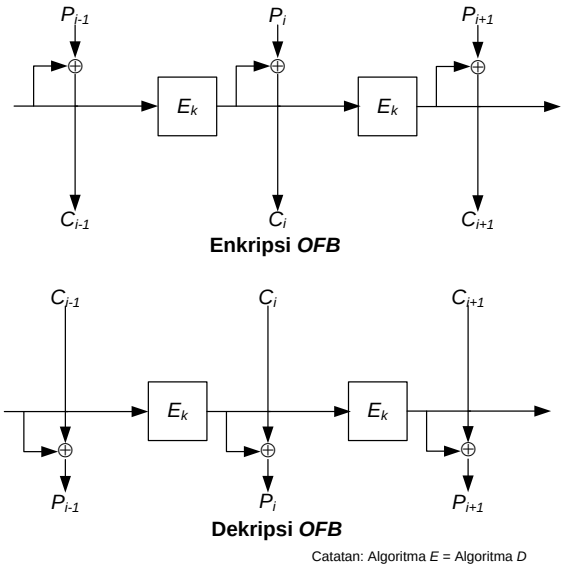
Sedangkan, algoritma dekripsi dengan mode *OFB* adalah sebagai berikut (lihat Gambar 6):

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. *Advanced Encryption Standard* (*AES*)

3.1 Panjang Kunci dan Ukuran Blok *Rijndael*

Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Karena *AES* menetapkan bahwa ukuran blok harus 128 bit, dan panjang kunci harus 128, 192, dan 256 bit, maka dikenal *AES*-128, *AES*-192, *AES*-256. Setiap blok dienkripsi dalam sejumlah putaran tertentu bergantung pada panjang kuncinya.

Tabel 1 Jumlah Putaran Setiap Blok pada *AES*

Varian *AES*	Panjang Kunci (*Nk words*)	Ukuran Blok (*Nb words*)	Jumlah Putaran (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Catatan: 1 *word* = 32 bit

Secara de-fakto, hanya ada dua varian *AES*, yaitu *AES*-128 dan *AES*-256, karena akan sangat jarang pengguna menggunakan kunci yang panjangnya 192 bit.
Karena *AES* mempunyai panjang kunci paling sedikit 128 bit, maka *AES* tahan terhadap serangan *exhaustive key search* dengan teknologi saat ini. Dengan panjang kunci 128-bit, maka terdapat $2^{128} \approx 3{,}4 \times 10^{38}$ kemungkinan kunci. Jika digunakan sebuah mesin dengan semilyar prosesor paralel, masing-masing dapat menghitung sebuah kunci setiap satu *pico* detik,

maka akan dibutuhkan waktu 10^{10} tahun untuk mencoba seluruh kemungkinan kunci.

3.2 Algoritma *Rijndael*

Seperti pada *DES*, *Rijndael* menggunakan substitusi dan permutasi, dan sejumlah putaran. Untuk setiap putarannya, *Rijndael* menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. Tetapi tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware* [1].
Garis besar algoritma *Rijndael* yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

1. *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *ByteSub*: substitusi byte dengan menggunakan tabel substitusi (*S-box*). Tabel substitusi dapat dilihat pada tabel 2, sedangkan ilustrasi *ByteSub* dapat dilihat pada gambar 9.
   b. *ShiftRow:* pergeseran baris-baris *array state* secara *wrapping*. Ilustarsi *ShiftRow* dapat dilihat pada gambar 10.
   c. *MixColumn*: mengacak data di masing-masing kolom *array state*. Ilustarsi *MixColumn* dapat dilihat pada gambar 11.
   d. *AddRoundKey:* melakukan *XOR* antara *state* sekarang dengan *round key*. Ilustarsi *AddRoundKey* dapat dilihat pada gambar 12.
3. *Final round*: proses untuk putaran terakhir:
   a. *ByteSub*.
   b. *ShiftRow*.
   c. *AddRoundKey*.

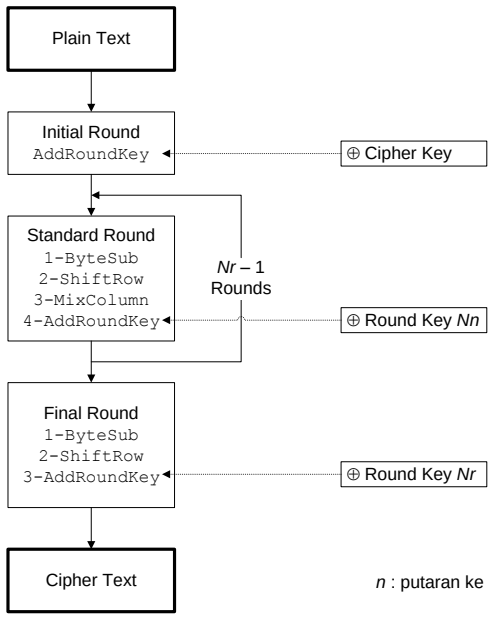
Diagram proses enkripsi *AES* dapat dilihat pada Gambar 8.

Algoritma *Rijndael* mempunyai 3 parameter sebagai berikut:

1. `plainteks` : *array* yang berukuran 16 *byte*, yang berisi data masukan.
2. `cipherteks` : *array* yang berukuran 16 *byte*, yang berisi hasil enkripsi.
3. `key` : *array* yang berukuran 16 *byte*, yang berisi kunci ciphering (disebut juga *cipher key*).

Dengan 16 *byte*, maka baik blok data dan kunci yang berukuran 128-bit dapat disimpan di dalam ketiga array tersebut (128 = 16 x 8).

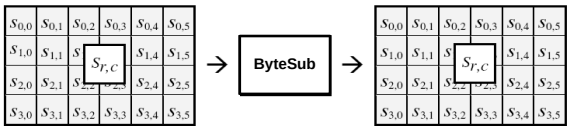
Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of byte* dua dimensi, `state`, yang berukuran `NROWS x NCOLS`. Elemen *array* `state` diacu sebagai *S*[*r*,*c*], dengan $0 \le r < 4$ dan $0 \le c < Nc$ (*Nc* adalah panjang blok dibagi 32). Pada *AES*, *Nc* = 128/32 = 4.



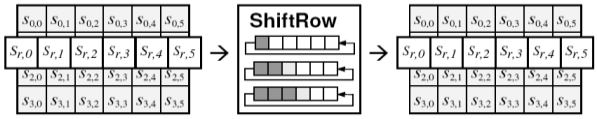
Gambar 8 Diagram Proses Enkripsi *AES*

Tabel 2 Tabel *S-box* yang digunakan dalam transformasi *ByteSub() AES*

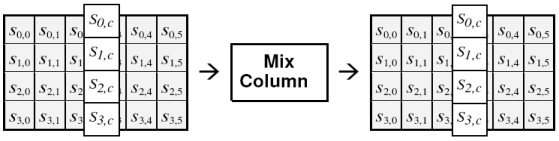
hex	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16



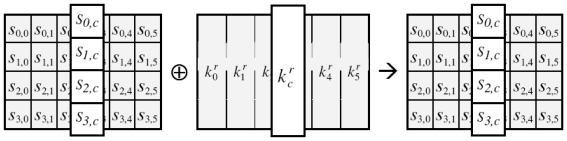
Gambar 9 Ilustrasi Transformasi *ByteSub() AES*



Gambar 10 Ilustrasi Transformasi *ShiftRow() AES*



Gambar 11 Ilustrasi Transformasi *MixColumn() AES*



Gambar 12 Ilustrasi Transformasi *AddRoundKey() AES*

3.3 Cipher Kebalikan (*Inverse Cipher*)

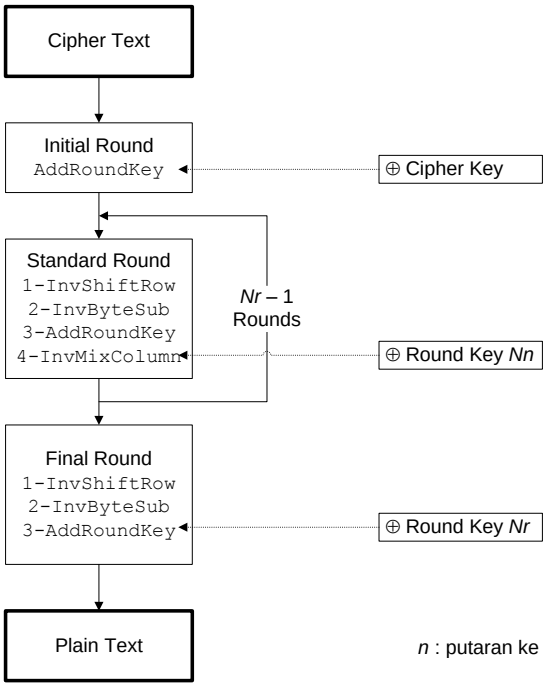
Cipher kebalikan merupakan algoritma kriptografi *AES* yang digunakan untuk melakukan proses dekripsi cipherteks menjadi plainteksnya. Secara garis besar, *cipher* kebalikan yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

1. *AddRoundKey*: melakukan *XOR* antara *state* awal (cipherteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
    a. *InvShiftRow:* pergeseran baris-baris *array state* secara *wrapping*.
    b. *InvByteSub*: substitusi byte dengan menggunakan tabel substitusi kebalikan (*inverse S-box*). Tabel substitusi dapat dilihat pada tabel 3.
    c. *AddRoundKey:* melakukan *XOR* antara *state* sekarang dengan *round key*.
    d. *InvMixColumn*: mengacak data di masing-masing kolom *array state*.
3. *Final round*: proses untuk putaran terakhir:
    a. *InvShiftRow*.
    b. *InvByteSub*.
    c. *AddRoundKey*.

Tabel 3 Tabel *S-box* yang digunakan dalam transformasi *InvByteSub() AES*

hex	y 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x 0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Diagram proses dekripsi *AES* dapat dilihat pada Gambar 13.



Gambar 13 Diagram Proses Dekripsi *AES*

4. Pengujian

4.1 Perancangan Kasus Uji Pengujian Perangkat Lunak *AESEncryptor*

Berdasarkan tata ancang dan teknik pengujian yang telah dijelaskan, maka dirancang kasus-kasus uji sebagai berikut:

1. Kasus Uji 1
   Kasus Uji 1 bertujuan untuk menguji kebenaran proses enkripsi dan dekripsi beserta lama waktu proses enkripsi dan dekripsi dengan menggunakan algoritma kriptografi *AES* dengan mode operasi *ECB* untuk panjang kunci 128-bit, 192-bit, dan 256-bit.
2. Kasus Uji 2
   Kasus Uji 2 bertujuan untuk menguji kebenaran proses enkripsi dan dekripsi beserta lama waktu proses enkripsi dan dekripsi dengan menggunakan algoritma kriptografi *AES* dengan mode operasi *CBC* untuk panjang kunci 128-bit, 192-bit, dan 256-bit.
3. Kasus Uji 3
   Kasus Uji 3 bertujuan untuk menguji kebenaran proses enkripsi dan dekripsi beserta lama waktu proses enkripsi dan dekripsi dengan menggunakan algoritma kriptografi *AES* dengan mode operasi *CFB* 8-bit untuk panjang kunci 128-bit, 192-bit, dan 256-bit.
4. Kasus Uji 4
   Kasus Uji 4 bertujuan untuk menguji kebenaran proses enkripsi dan dekripsi beserta lama waktu proses enkripsi dan dekripsi dengan menggunakan algoritma kriptografi *AES* dengan mode operasi *OFB* 8-bit untuk panjang kunci 128-bit, 192-bit, dan 256-bit.
5. Kasus Uji 5
   Kasus Uji 5 bertujuan untuk menguji tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *ECB* terhadap pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks.
6. Kasus Uji 6
   Kasus Uji 6 bertujuan untuk menguji tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *CBC* terhadap pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks.
7. Kasus Uji 7
   Kasus Uji 7 bertujuan untuk menguji tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *CFB* 8-bit terhadap pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks.
8. Kasus Uji 8
   Kasus Uji 8 bertujuan untuk menguji tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *OFB* 8-bit terhadap pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks.

4.2 Evaluasi Hasil Pengujian Perangkat Lunak *AESEncryptor*

Dari hasil pengujian Kasus Uji 1, 2, 3, dan 4, diketahui bahwa perangkat lunak *AESEncryptor* telah melakukan proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB* 8-bit, dan *OFB* 8-bit untuk panjang kunci 128-bit, 192-bit, dan 256-bit dengan benar. Proses enkripsi dengan menggunakan kunci tertentu dengan panjang tertentu akan menyandikan isi arsip asal. Proses dekripsi dengan menggunakan kunci yang sama dengan kunci yang digunakan dalam proses enkripsi (kunci simetris) akan mengembalikan isi arsip hasil dekripsi menjadi isi arsip asal. Sedangkan, kesalahan penggunaan kunci mengakibatkan isi arsip hasil dekripsi tidak sama dengan arsip asal. Kasus Uji 1, 2, 3, dan 4 juga menunjukkan bahwa:

1. Lama waktu yang digunakan untuk proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *ECB* dan *CBC* adalah relatif sama.
2. Lama waktu yang digunakan untuk proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *CFB* 8-bit dan *OFB* 8-bit adalah relatif sama.
3. Lama waktu yang digunakan untuk proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *CFB* 8-bit dan *OFB* 8-bit lebih besar dari pada lama waktu yang digunakan untuk proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *ECB* dan *CBC*.

Dari hasil pengujian Kasus Uji 5, diketahui bahwa tingkat keamanan algoritma kriptografi

AES dengan mode operasi *ECB* terhadap manipulasi cipherteks adalah sebagai berikut:

1. Pengubahan satu bit atau lebih blok cipherteks akan mengakibatkan terjadinya perubahan terhadap sebuah blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang diubah.
2. Penambahan sebuah blok cipherteks semu akan mengakibatkan terjadinya penambahan sebuah blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang ditambahkan.
3. Penghilangan satu atau lebih blok cipherteks akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang dihilangkan.

Hasil pengujian ini menunjukkan keuntungan sekaligus kelemahan mode operasi *ECB*. Dalam hal keuntungan, kesalahan atau perubahan satu atau lebih bit blok cipherteks hanya mempengaruhi blok cipherteks yang bersangkutan pada waktu proses dekripsi. Sedangkan kelemahan mode operasi *ECB* adalah blok plainteks yang sama akan menghasilkan blok cipherteks yang sama.

Dari hasil pengujian Kasus Uji 6, diketahui bahwa tingkat keamanan algoritma kriptografi *AES* dengan mode operasi *CBC* manipulasi cipherteks adalah sebagai berikut:

1. Pengubahan satu bit atau lebih blok cipherteks akan mengakibatkan terjadinya perubahan terhadap sebuah blok plainteks dan satu bit atau lebih pada blok plainteks berikutnya (pada posisi bit yang berkoresponden dengan bit cipherteks yang diubah) pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang diubah.
2. Penambahan sebuah blok cipherteks semu di awal atau tengah akan mengakibatkan terjadinya penambahan sebuah blok plainteks semu pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang ditambahkan disertai dengan perubahan terhadap sebuah blok plainteks berikutnya, sedangkan penambahan sebuah blok cipherteks semu di akhir akan mengakibatkan terjadinya penambahan sebuah blok plainteks semu pada arsip hasil dekripsi yang letaknya di akhir arsip hasil dekripsi.
3. Penghilangan satu atau lebih blok cipherteks di awal atau tengah akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan satu atau lebih blok cipherteks yang dihilangkan disertai dengan perubahan terhadap sebuah blok plainteks berikutnya, sedangkan penghilangan satu atau lebih blok cipherteks di akhir akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya di akhir arsip hasil dekripsi.

Hasil pengujian ini menunjukkan keuntungan sekaligus kelemahan mode operasi *CBC*. Dalam hal keuntungan, blok-blok plainteks yang sama tidak menghasilkan blok-blok cipherteks yang sama sehingga proses kriptanalisis menjadi lebih sulit. Sedangkan kelemahan mode operasi *CBC* adalah kesalahan satu bit pada blok cipherteks mempengaruhi blok plainteks yang berkoresponden dan satu bit pada blok plainteks berikutnya (pada posisi bit yang berkoresponden berkoresponden dengan bit cipherteks yang diubah).

Dari hasil pengujian Kasus Uji 7, diketahui bahwa tingkat keamanan algoritma kriptografi *AES* dengan mode operasi *CFB* 8-bit manipulasi cipherteks adalah sebagai berikut:

1. Pengubahan satu bit atau lebih blok cipherteks akan mengakibatkan terjadinya perubahan terhadap blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan blok cipherteks yang diubah dan diikuti dengan perubahan terhadap seluruh blok plainteks berikutnya.
2. Penambahan sebuah blok cipherteks semu di awal atau tengah akan mengakibatkan terjadinya penambahan sebuah blok plainteks semu pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang ditambahkan diikuti dengan perubahan terhadap seluruh blok plainteks berikutnya, sedangkan penambahan sebuah blok cipherteks

semu di akhir akan mengakibatkan terjadinya penambahan sebuah blok plainteks semu pada arsip hasil dekripsi yang letaknya di akhir arsip hasil dekripsi.

3. Penghilangan satu atau lebih blok cipherteks di awal atau tengah akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan satu atau lebih blok cipherteks yang dihilangkan diikuti dengan perubahan terhadap seluruh blok plainteks berikutnya, sedangkan penghilangan satu atau lebih blok cipherteks di akhir akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya di akhir arsip hasil dekripsi.

Hasil pengujian ini menunjukkan keuntungan sekaligus kelemahan mode operasi *CFB* 8-bit. Dalam hal keuntungan, blok-blok plainteks yang sama tidak menghasilkan blok-blok cipherteks yang sama sehingga proses kriptanalisis menjadi lebih sulit. Selain itu, pada mode operasi *CFB*, data dapat dienkripsikan dalam ukuran yang lebih kecil. Sedangkan kelemahan mode operasi *CFB* 8-bit adalah kesalahan satu bit pada blok cipherteks mempengaruhi blok plainteks yang berkoresponden dan seluruh blok plainteks berikutnya.

Dari hasil pengujian Kasus Uji 8, diketahui bahwa tingkat keamanan algoritma kriptografi *AES* dengan mode operasi *OFB* 8-bit manipulasi cipherteks adalah sebagai berikut:

1. Pengubahan satu bit atau lebih blok cipherteks akan mengakibatkan terjadinya perubahan terhadap blok plainteks yang berkoresponden dengan blok cipherteks yang diubah.
2. Penambahan sebuah blok cipherteks semu akan mengakibatkan terjadinya penambahan sebuah blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang ditambahkan.
3. Penghilangan satu atau lebih blok cipherteks akan mengakibatkan terjadinya penghilangan satu atau lebih blok plainteks pada arsip hasil dekripsi yang letaknya berkoresponden dengan sebuah blok cipherteks yang dihilangkan.

Hasil pengujian ini menunjukkan keuntungan mode operasi *OFB* 8-bit. Dalam hal keuntungan, blok-blok plainteks yang sama tidak menghasilkan blok-blok cipherteks yang sama sehingga proses kriptanalisis menjadi lebih sulit. Selain itu, pada mode operasi *OFB*, data dapat dienkripsikan dalam ukuran yang lebih kecil.

5. Kesimpulan

Kesimpulan yang dapat dimbil dari studi dan implementasi AES dengan empat mode operasi *block cipher* ini ini adalah:

1. *Advanced Encryption Standard* (*AES*) merupakan salah satu solusi yang baik untuk mengatasi masalah keamanan dan kerahasiaan data yang pada umumnya diterapkan dalam pengiriman dan penyimpanan data melalui media elektronik.
2. *AES* dapat diimplementasikan secara efisien sebagai perangkat lunak dengan implementasi menggunakan tabel sebab setiap langkah transformasi telah disimpan ke dalam tabel-tabel, sehingga komputer hanya perlu melakukan operasi melihat (*look up*) tabel-tabel untuk melakukan sebuah langkah transformasi dan serangkaian operasi *XOR* untuk melakukan proses enkripsi *AES* atau proses dekripsi *AES*.
3. Urutan lama waktu yang digunakan untuk proses enkripsi dan dekripsi algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB* 8-bit, *OFB* 8-bit secara berturut-turut mulai dari yang tercepat adalah sebagai berikut:
   ECB ≈ CBC, *CFB* 8-bit ≈ *OFB* 8-bit
4. Kesalahan 1-bit pada blok plainteks/cipherteks *AES* dengan mode operasi *CFB* akan merambat pada blok-blok plainteks/cipherteks yang berkoresponden dan blok-blok plainteks/cipherteks selanjutnya pada proses enkripsi/dekripsi.
5. Urutan tingkat keamanan data algoritma kriptografi *AES* dengan mode operasi *ECB*, *CBC*, *CFB* 8-bit, dan *OFB* 8-bit terhadap pengubahan satu bit atau lebih blok cipherteks, penambahan blok cipherteks semu, dan penghilangan satu atau lebih blok cipherteks secara berturut-turut mulai dari yang teraman adalah sebagai berikut:
   OFB 8-bit, *CBC*, *ECB*, *CFB* 8-bit

DAFTAR PUSTAKA

[1] Daemen, Joan, Vincent Rijmen. (2004). The *Rijndael* Specification. *http://csrc.nist.gov/encryption/AES/Rijndael/Rijndael.pdf*. Tanggal akses: 4 Desember 2004 pukul 20:00.

[2] Lidl & Niederreiter. (1986). Introduction to Finite Fields and Their Applications. Cambridge University Press.

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

[4] *NIST*. (2004). National Institute of Standards and Technology. *http://www.nist.gov*. Tanggal akses: 4 Desember 2004 pukul 20:00.

[5] Schneier, Bruce. (1996). Applied Cryptography 2nd. John Wiley & Sons.

[6] The Square Page. (2004), *http://www.esat.kuleuven.ac.be/~rijmen/square*. Tanggal akses: 4 Desember 2004 pukul 20:00.

[7] Tanenbaum, Andrew S. (2003). Computer Networks Fourth Edition. Pearson Education International.

[8] Trustcopy. (2004). Trustcopy - The premier provider of Brand Protection and Secured Trade Documentation Solutions. *http://www.trustcopy.com/*. Tanggal akses: 4 Desember 2004 pukul 20:00.