# *WINDOWS VISTA BITLOCKER DRIVE ENCRYPTION*

Yudha Adiprabowo – NIM : 13506050

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail : if16050@students.if.itb.ac.id

## Abstrak

Makalah ini mengulas tentang salah satu sistem keamanan menggunakan kriptografi dalam sistem operasi *Windows Vista*, yaitu *Bitlocker Drive Encryption*. *Bitlocker Drive Encryption* merupakan sebuah metode *Disc Encryption* (enkripsi pada kandar komputer) yang memanfaatkan algoritma *Advanced Encryption Standard* (*AES*). Implementasi *AES* yang digunakan pada *Bitlocker Drive Encryption* memanfaatkan mode operasi *cipher block chaining* (*CBC*) dan sebuah *diffuser*, yaitu *Elephant Diffuser*. *Elephant Diffuser* menggunakan *diffusion*, salah satu dari prinsip penyandian Shannon, untuk menyembunyikan hubungan statistik antara plainteks, cipherteks, dan kunci.

***Kata kunci**: Windows Vista, Bitlocker Drive Encryption, Disk Encryption, Advanced Encryption Standard, cipher block chaining, Elephant Diffuser, diffusion*, enkripsi, dekripsi.

## 1. Pendahuluan

*Disk Encryption* adalah salah satu kasus proteksi data yang sedang tidak diakses ketika media penyimpanannya adalah sebuah perangkat yang memiliki alamat dan sektor penyimpanan (contohnya *hard disk* komputer). *Bitlocker Drive Encryption* merupakan salah satu metode *Disk Encryption*.

Versi *Enterprise* dan *Ultimate* dari *Windows Vista* memuat fitur baru bernama *Bitlocker Drive Encryption* yang mengenkripsi semua data pada sistem. *Bitlocker* menyediakan beberapa kelebihan dalam hal keamanan pada algoritma enkripsinya yang biasanya tidak dicapai dengan algoritma enkripsi biasa. Hal ini memunculkan masalah baru karena sebuah cipher baru tidak bisa dipercaya tanpa penilaian dari publik selama bertahun-tahun, dan cipher yang sudah ada yang memenuhi kebutuhan keamanan tambahan memiliki kelemahan dalam hal kecepatan atau kurang dianalisis secara mendalam.

*Bitlocker* menyelesaikan dilema yang ada dengan mengombinasikan cipher yang sudah mapan (*AES* dalam mode *CBC*) dengan komponen baru yang bernama *Elephant Diffuser*. Keamanan enkripsi mendasar disediakan oleh *AES-CBC* yang sudah dibahas dan digunakan secara luas dalam industri untuk enkripsi data. *Diffuser* menambahkan keamanan yang dibutuhkan dalam *Disk Encryption* yang tidak disediakan oleh metode *AES-CBC*. Selain itu pendekatan menggunakan *AES-CBC* dan sebuah *diffuser* memiliki kecepatan yang lebih baik daripada alternatif-alternatiif lainya, yang merupakan hal penting untuk aplikasi.

*Bitlocker* sebenarnya dibuat untuk mengatasi sebuah skenario spesifik, yaitu laptop yang hilang. Komputer bertipe laptop yang hilang memiliki informasi-informasi rahasia dalam bentuk dokumen, presentasi, *e-mail*, data sementara, dan hak akses ke jaringan. Informasi-informasi ini biasanya jauh lebih berharga dibandingkan dengan harga laptop tersebut jika dimanfaatkan orang yang tepat. Sebuah laptop mudah diganti dengan biaya secukupnya. Sementara, untuk mengganti data-data di dalamnya biaya yang dibutuhkan bisa berkali-kali lipat.

## 2. Penjelasan Mengenai Disk Encryption

Metode *Disk Encryption*, enkripsi pada media penyimpanan data, memiliki beberapa ciri-ciri yang hendak dicapai:

a. Data pada *disk* tetap rahasia.
b. Pengambilan dan penyimpanan data merupakan operasi yang cepat, tanpa mengindahkan posisi data disimpan pada *disk*.
c. Metode enkripsi tidak membuang-buang ruang kosong pada *disk*.

Pada ciri pertama, dibutuhkan pendefinisian musuh yang hendak membongkar kerahasiaan data. Musuh terkuat pada bidang *Disk Encryption* memiliki kemampuan seperti membaca data mentah dalam *disk* kapan saja, melakukan enkripsi dan menyimpan arsip pilihan, mengubah sektor *disk* yang tidak digunakan dan melakukan dekripsi. Metode yang kerahasiaannya baik adalah jika satu-satunya yang bisa dilihat oleh musuh adalah adanya perubahan pada data dalam suatu sektor setelah pengawasan.

Ciri kedua membutuhkan pembagian *disk* menjadi beberapa sektor, biasanya sebesar 512 byte, yang dienkripsi dan didekripsi secara terpisah satu sama lain. Pada akhirnya, jika data hendak dirahasiakan, metode enkripsi harus bisa disesuaikan dengan kondisi. Seharusnya tidak ada dua sektor yang diproses dengan cara yang sama. Jika ada dua sektor yang diproses dengan cara yang sama, seorang musuh bisa saja melakukan dekripsi pada sektor manapun dengan memindahkannya ke sektor yang tidak terpakai dan melakukan dekripsi pada sektor yang tidak terpakai tersebut.

Ciri ketiga secara tidak langsung melarang penggunaan *stream cipher*, karena mereka menggunakan larangan di mana initial state tidak boleh digunakan dua kali, dan ini menyebabkan metode enkripsi menyimpan *initial state* yang berbeda-beda untuk tiap sektor pada *disk*. Sementara, *block cipher* yang merupakan alternatifnya memiliki batasan ukuran blok sebesar 128 atau 256 bit.

Jika suatu metode memenuhi ketiga ciri-ciri di atas, maka metode tersebut sudah memenuhi ciri-ciri mendasar *Disk Encryption*. Meskipun begitu, metode tersebut belum tentu menyediakan keamanan absolut karena seorang musuh selalu bisa melakukan serangan pada disk tersebut.

## 3. Penjelasan dan Prinsip Kerja *Bitlocker Drive Encryption*

*Bitlocker* menggunakan teknologi enkripsi berbasiskan perangkat lunak dan perangkat keras sekaligus. Perangkat keras yang digunakan dalam penggunaan *Bitlocker* adalah *chip* TPM yang akan digunakan secara luas pada komputer tahun-tahun mendatang. Walapun TPM memiliki banyak kegunaan, *Bitlocker* hanya menggunakan beberapa kegunaan dasarnya.

*Chip* TPM menyimpan beberapa *Platform Configuration Registers* (PCR). Saat komputer baru pertama kali dinyalakan, PCR diset ke nilai 0. PCR hanya diubah oleh sebuah fungsi *extend* yang secara efektif mengeset nilai PCR menjadi hasil hash dari nilai lamanya dengan sebuah kunci string. Sehingga ketika PCR bernilai *x* setelah beberapa penggunaan fungsi *extend*, tidak ada cara untuk mencapai nilai *x* itu kembali selain melakukan pemanggilan fungsi *extend* yang benar-benar serupa setelah komputer dinyalakan.

Sementara itu TPM juga menyimpan fungsi *seal/unseal* yang mengizinkan akses selektif pada kunci berdasarkan nilai yang tersimpan pada PCR. Fungsi *seal* digunakan untuk mengenkripsi kunci menjadi sebuah string yang hanya bisa didekripsi oleh TPM tersebut. Tambah lagi, TPM hanya mendekripsi string tersebut jika nilai PCR sama persis dengan nilai PCR saat fungsi *seal* dilakukan.

Saat proses *boot* komputer, PCR akan memantau kode-kode yang dijalankan. Saat *boot* normal, PCR akan selalu mencapai nilai yang sama sehingga kunci akan terbuka oleh TPM. Jika seorang musuh membuka *disk* dengan sistem operasi lain, komputer tetap akan jalan tetapi nilai yang tersimpan pada PCR akan berbeda dan TPM tidak akan membuka kuncinya. Akibatnya, sistem tidak bisa membaca data pada *disk* atau mencari cara mengubah isi *disk* menggunakan kata sandi milik *administrator*.

Untuk menjamin nilai PCR yang selalu sama, *Bitlocker* haruslah yang diakses pertama kali pada saat *boot* komputer. Jika tidak, nilai PCR masih mungkin akan berubah-ubah ketika kita melakukan *upgrade* atau *update* pada sistem operasi kita sendiri dan mengakibatkan seluruh data pada komputer terkunci karena TPM tidak menemukan nilai PCR yang tepat.

Sementara, dalam hal perangkat lunak, *Bitlocker Drive Encryption* memanfaatkan *Advanced Encryption Standard* (*AES*) dengan mode *Cipher Block Chaining* (*CBC*) dan ditambah dengan sebuah *diffuser* bernama *Elephant Diffuser*.

### 3.1 Cipher Block Chaining (CBC)

*Cipher Block Chaining* (*CBC*) merupakan salah satu mode operasi dari *cipher* blok. Secara umum, pada *cipher* blok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan

panjang sama, biasanya 64 bit (tapi ada kalanya 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 $i = 1, 2, \ldots, m$.

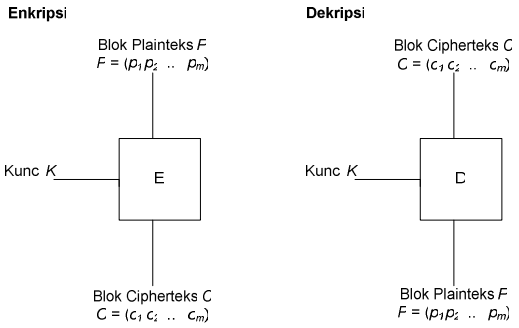
Bila plainteks dibagi menjadi *n* buah blok, barisan blok-blok plainteks dinyatakan sebagai

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

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

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

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




**Gambar 1 Skema Enkripsi dan Dekripsi dengan *Cipher* Blok**

Enkripsi dengan kunci *K* dinyatakan dengan persamaan

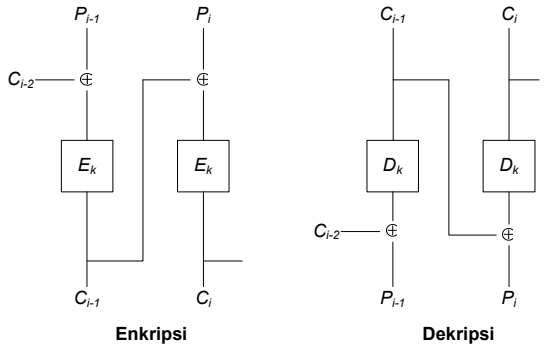
$$E_k(P) = C,$$

sedangkan dekripsi dengan kunci *K* dinyatakan dengan persamaan

$$D_k(C) = P$$

Mode *Cipher Block Chaining* menggunakan prinsip kerja *cipher* blok secara umum namun menggunakan cara yang berbeda dalam implementasinya. 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 2.




**Gambar 2 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.

Perhatikan bahwa enkripsi terhadap blok *i* adalah fungsi dari semua plainteks dari blok 0 sampai blok *i* - 1, sehingga blok plainteks yang sama menghasilkan blok cipherteks yang berbeda hanya jika blok-blok plainteks 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 digunakan untuk membuat tiap blok cipherteks menjadi unik.

### 3.2 *Advanced Encryption Standard* (*AES*)

#### 3.2.1 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 sebanyak
$2^{128} \approx 3{,}4 \times 10^{38}$
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.

#### 3.2.2 Algoritma Rijndael

*Rijndael* menggunakan substitusi dan permutasi, dan sejumlah putaran. Untuk setiap putarannya, *Rijndael* menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. *Rijndael* beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware*.
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*.

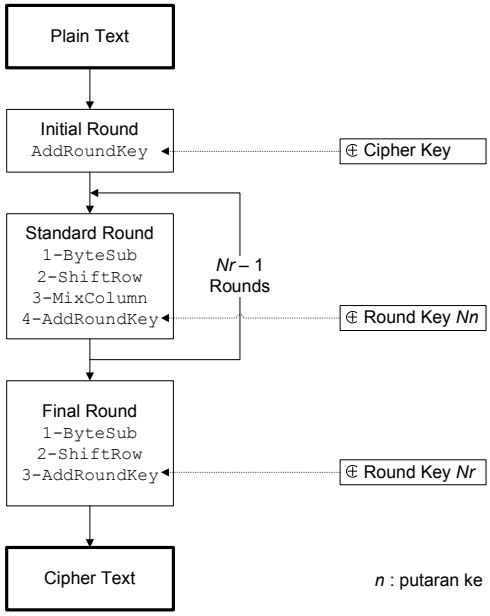
Diagram proses enkripsi *AES* dapat dilihat pada Gambar 8.

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).

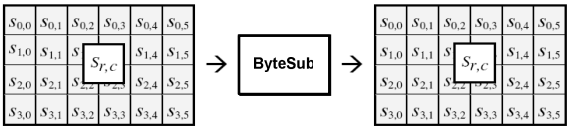
Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of byte* dua dimensi, `state`, yang berukuran `NROWS x NCOLS`. Elemen *array* `state` diacu sebagai $S[r,c]$, dengan $0 \leq r < 4$ dan $0 \leq c < Nc$ ($Nc$ adalah panjang blok dibagi 32). Pada *AES*, $Nc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



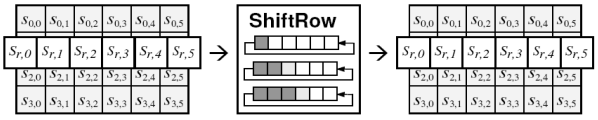

**Gambar 3 Diagram Proses Enkripsi *AES***

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

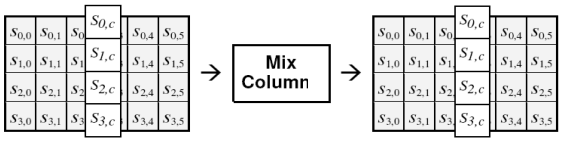
| hex | y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 1 | ca | 82 | c9 | 7d | fa | 59 | 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 |



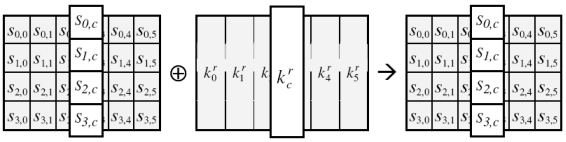

**Gambar 4 Ilustrasi Transformasi *ByteSub() AES***




**Gambar 5 Ilustrasi Transformasi *ShiftRow() AES***



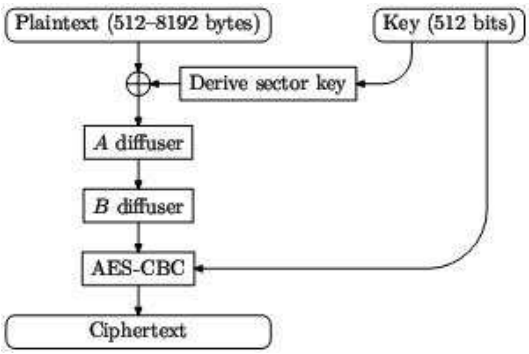

**Gambar 6 Ilustrasi Transformasi *MixColumn() AES***



**Gambar 7 Ilustrasi Transformasi *AddRoundKey() AES***

### 3.3 Implementasi pada Bitlocker Drive Encryption

Gambar 8 emberikan gambaran akan cara kerja Bitlocker secara keseluruhan. *A diffuser* dan *B diffuser* merupakan *Elephant Diffuser* yang disinggung di awal pembahasan. Plainteks di-XOR-kan dengan sebuah *sector key*, lalu melewati dua *diffuser* dan akhirnya dienkripsi dengan AES pada mode CBC.




**Gambar 8 Ilustrasi *Bitlocker Drive Encryption* secara keseluruhan**

Komponen *sector key* dan komponen AES-CBC mempunyai kunci yang saling independen yang membuat kita mudah menyimpulkan bahwa konstruksi ini setidaknya sama amannya dengan AES-CBC. Kedua komponen diberikan 256 bit kunci sehingga kunci secara penuh adalah 512 bit. Tergantung dengan versi yang digunakan, kedua komponen tersebut bisa menggunakan kunci yang kurang dari 256 bit sehingga ada bit yang kosong. Kunci secara penuh selalu 512 bit untuk mendukung ukuran kunci yang besar tanpa pengubahan dalam manajemen kunci. Ukuran blok dalam konstruksi ini juga berbeda-beda. Bisa saja berukuran antara dua kalinya 512-8192 bytes (4096 – 65536 bits).

Komponen AES-CBC cukup sederhana. Kunci AES $K_{\text{AES}}$ antara berukuran 128 bit atau 256 bit tergantung versi yang digunakan. Ukuran blok selalu berupa kelipatan 16 bytes sehinggga *padding* (tambahan bit untuk menutupi kekurangan) tidak dibutuhkan. *IV* untuk sebuah sektor *s* dihitung dengan rumus:

$$IV_s := E(K_{\text{AES}}, e(s))$$

Di mana $E()$ adalah fungsi enkripsi AES dan $e()$ adalah fungsi *encode* yang memetakan setiap nomor sektor *s* ke nilai 16 byte unik. Harap diperhatikan bahwa $IV_s$ hanya bergantung dengan kunci dan nomor sektor, tetapi tidak bergantung dengan data.

Plainteks dienkripsi dengan AES-CBC dan *IV* untuk sektor yang bersangkutan. Dekripsi merupakan proses sebaliknya.

Sementara untuk *sector key* didefinisikan sebagai:

$$K_s := E(K_{\text{sec}}, e(\text{s})) \,||\, E(K_{\text{sec}}, e'(s))$$

Di mana $E()$ merupakan fungsi enkripsi AES, $K_{\text{sec}}$ merupakan kunci 128 atau 256 bit untuk komponen ini, $e()$ merupakan fungsi *encode* yang digunakan pada AES-CBC dan $e'(s)$ sama saja dengan $e(s)$ kecuali memiliki byte terakhir bernilai 128.

*Sector key* diulang berkali-kali untuk sampai sebesar ukuran blok lalu di-XOR-kan dengan plainteks.

*Diffuser A* dan B sangat mirip, tetapi bekerja dengan sebaliknya. Setiap *diffuser* menginterpretasikan sector data sebagai *array* 32-bit berisi *word* dan setiap *word* menggunakan *least-significant-byte first*. Diberikan *n* sebagai jumlah *word* dalam sektor dan ($d_0$, $d_1$,..., $d_{n-1}$) sebagai *word-word* dalam sektor tersebut.

Fungsi dekripsi untuk *diffuser* A:

**for** $i = 0,1,2,...,n \cdot A_{cycles} - 1$
$d_i \leftarrow d_i + (d_{i-2} \text{ XOR } (d_{i-5} <<< R_i^{(a)}{}_{\text{mod } 4}))$

$R^{(a)} := [9,0,13,0]$ adalah array berisi 4 constraint yang menspesifikasikan jumlah rotasi.

Fungsi enkripsi untuk *diffuser* A :

**for** $i = n \cdot A_{cycles} - 1,...,2,1,0$
$d_i \leftarrow d_i - (d_{i-2} \text{ XOR } (d_{i-5} <<< R_i^{(a)}{}_{\text{mod } 4}))$

Fungsi dekripsi untuk *diffuser* B:

**for** $i = 0,1,2,...,n \cdot B_{cycles} - 1$
$d_i \leftarrow d_i + (d_{i+2} \text{ XOR } (d_{i+5} <<< R_i^{(b)}{}_{\text{mod } 4}))$

Fungsi enkripsi untuk *diffuser* B:

**for** $i = n \cdot B_{cycles} - 1,...,2,1,0$
$d_i \leftarrow d_i - (d_{i+2} \text{ XOR } (d_{i+5} <<< R_i^{(b)}{}_{\text{mod } 4}))$

**4. Kesimpulan**

Kesimpulan yang dapat diambil dari *Bitlocker Drive Encryption* ini adalah:

1. *Disk Encryption* merupakan salah satu kasus proteksi data yang sedang tidak diakses ketika media penyimpanannya adalah sebuah perangkat yang memiliki alamat dan sektor penyimpanan (contohnya *hard disk* komputer). *Bitlocker Drive Encryption* merupakan salah satu metode *Disk Encryption*.
2. *Bitlocker Drive Encryption* memanfaatkan perangkat lunak dan perangkat keras dalam implementasinya. Perangkat keras yang dibutuhkan adalah sebuah *chip* TPM.
3. Dalam implementasi perangkat lunak, *Bitlocker Drive Encryption* memanfaatkan metode *Advanced Encryption Standard* (AES) mode *Cipher Block Chaining* (CBC) dan menggunakan sebuah *diffuser* bernama *Elephant Diffuser*.

**DAFTAR PUSTAKA**

[1] Ferguson, Niels. (2006). AES-CBC + Elephant diffuser A Disk Encryption Algorithm for Windows Vista. *http://www.microsoft.com/downloads/details.aspx?FamilyID=131dae03-39ae-48be-a8d6-8b0034c92555&displaylang=en*. Tanggal Akses: 12 Maret 2009 pukul 21.00.

[2] Munir, Rinaldi. (2006). Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika, Institut Teknologi Bandung.