Studi mengenai Disk Encryption

menggunakan Mode Operasi LRW dan Algoritma Rijndael

Elfira Yolanda S – NIM : 13503087

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13087@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang studi mengenai *disk encryption*. *Disk Encryption* adalah enkripsi yang dilakukan pada *data at rest*. Istilah *data at rest* ditujukan pada data yang tersimpan pada media penyimpanan komputer. Data at rest Dengan demikian, dalam pengertian *disk encryption*, enkripsi tidak dilakukan pada data yang sedang ditransmisi.

Disk encryption juga berbeda dengan file encryption yang melakukan enkripsi terhadap file satu per satu. Pada *disk encryption*, enkripsi dilakukan pada area tertentu yang sudah dirancang sedemikian rupa sehingga data yang disimpan pada area tersebut secara otomatis dienkripsi. Area tersebut bisa berupa seluruh hard disk (*entire hard disk encryption*/*EHD encryption*), maupun berupa virtual disk yang memang diciptakan sebagai area enkripsi (*virtual hard disk encryption/VHD encryption*). Makalah yang akan dibuat merupakan studi terhadap *on-the-fly encryption* tersebut.

Selain itu akan dibahas juga mengenai salah satu mode operasi dan algoritma yang dapat digunakan dalam aplikasi *disk encryption*. Mode operasi yang dipilih untuk makalah adalah mode operasi LRW yang ditemukan oleh Moses Liskov, Ronald L. Rivest, dan David Wagner. Sedangkan algoritma enkripsi yang diambil adalah algoritma Rijndael (AES)

***Kata kunci**: on-the-fly encryption, disk encryption, mode operasi, LRW, Rijndael, AES.*

1. Pendahuluan

Data dapat menjadi salah satu aset penting dalam kelangsungan hidup perusahaan mana pun. Penyimpanan data memerlukan berbagai macam pertimbangan, terutama dari segi keamanannya. Penggunaan *laptop/notebook* saat ini menimbulkan masalah baru berkaitan dengan penyimpanan data tersebut. Mau tidak mau, perusahaan harus mengizinkan data penting mereka "berkeliaran" di luar. Untuk menyelesaikan masalah ini, enkripsi data sebaiknya dilakukan.

Teknik enkripsi yang umum digunakan adalah enkripsi yang dilakukan pada file. Satu file yang diinginkan dienkripsi pada satu waktu. Enkripsi file seperti ini sudah dilakukan bertahun-tahun dan masih memiliki manfaat dari segi keamanan. Namun, enkripsi file seperti ini juga memiliki kekurangan terutama bila file yang berisi informasi penting berjumlah banyak, seperti file-file yang dimiliki oleh perusahaan. Enkripsi file yang dilakukan membutuhkan penanganan khusus berkaitan dengan status file apa yang menggunakan kunci apa.

Disk encryption merupakan alternatif enkripsi data selain enkripsi file. Enkripsi ini disebut juga enkripsi volume (*volume encryption*). Penggunaan kata volume dikarenakan enkripsi ini seakan-akan dilakukan pada area tertentu. Area itu menjadi sebuah volume/ kontainer yang dapat digunakan untuk menyimpan file-file penting.

Disk encryption bekerja dengan blok bit. Oleh karena itu untuk membangun sistem enkripsi seperti ini, algoritma enkripsi yang dipakai adalah algoritma yang bekerja dengan blok bit juga. Jenis algoritma seperti itu dinamakan *cipher* blok (*block cipher*). Rangkaian bit plainteks/cipherteks yang akan dienkripsi atau didekripsi dibagi menjadi blok-blok bit yang panjangnya sama dan sudah ditentukan sebelumnya[6]. *Cipher* blok termasuk dalam tipe algoritma simetri, tipe algoritma kriptografi modern yang berarti proses enkripsi dan dekripsi memiliki struktur yang serupa.

Banyak algoritma kriptografi cipher blok yang sudah pernah dipublikasikan . Untuk menyebut beberapa di antaranya adalah DES(Data Encryption Standard), Triple DES(3DES), IDEA(International Data Encryption Algorithm), Blowfish, Gost, Safer, LOKI, FEAL, RC2, RC5, Serpent, dan lain-lain, hingga yang terbaru adalah AES (Advanced Encryption Standard) [6].

Pada *cipher* blok juga dikenal istilah mode operasi. Mode operasi yang paling sering digunakan untuk melakukan *disk encryption* adalah *Cipher Block Chaining* (CBC). Bagaimanapun juga, saat ini banyak sekali kelemahan yang ditemukan pada CBC untuk *disk encryption* sehingga mode operasi ini tidak menjadi kandidat dalam penentuan standard mode operasi untuk sistem enkripsi yang dilakukan pada media penyimpanan (*disk encryption*). Pihak IEEE, terutama salah satu *work group*-nya SISWG (Security In Storage Work Group), sampai saat ini masih berusaha menentukan mode operasi yang menjadi standard enkripsi media penyimpanan. Ada dua kandidat utama yaitu EME (ECB-Mix-ECB) dan LRW (Liskov Rivest Wagner). EME merupakan kandidat untuk *wide cipher mode*, sedangkan LRW untuk *narrow cipher mode*.

2. Disk Encryption

2.1. Deskripsi Disk Encryption

Disk Encryption adalah enkripsi yang dilakukan pada *data at rest*. Istilah *data at rest* ditujukan pada data yang tersimpan pada media penyimpanan komputer. Data at rest Dengan demikian, dalam pengertian *disk encryption*, enkripsi tidak dilakukan pada data yang sedang ditransmisi.

Media penyimpanan yang dimaksud dalam *disk encryption* merupakan peralatan/*device* yang memiliki sektor-sektor dan alamat untuk tiap sektor tersebut. Contohnya adalah hard disk, baik pada PC (Personal Computer) maupun laptop. Satu sektor pada hard disk biasanya memiliki ukuran 512 byte.

Disk encryption sendiri dipicu oleh makin maraknya penggunaan laptop. Keuntungan mobilitas yang ditawarkan laptop ternyata membawa dampak lain. Keamanan data pada hard disk laptop harus diperhitungkan. *Disk encryption* merupakan solusi untuk hal tersebut.

Lama sebelum *disk encryption*, masyarakat mengenal *file encryption*. Jenis enkripsi ini sudah lama dikenal dan sangat umum dipakai. Yang menjadi perbedaan dengan *file encryption* atau enkripsi file biasa adalah pada disk

encryption, terlebih dahulu disediakan sebuah wilayah hard disk. Semua file yang disimpan pada wilayah tersebut akan terenkripsi. Sedangkan enkripsi file melakukan enkripsi pada tiap file, satu per satu. Kunci yang digunakan untuk membuka tiap data pada disk encryption hanya satu. Seluruh data tersebut dapat diakses dengan memberikan satu kunci tersebut. Dengan kata lain, kunci itu membuka wilayah hard disk yang disediakan oleh *disk encryption* tadi. Pada enkripsi file, masing-masing file memiliki kunci sendiri. Semakin banyak file yang dienkripsi, semakin banyak kunci yang harus diingat. Enkripsi file menuntut penanganan khusus akan hal tersebut. *Disk encryption* jelas lebih efisien dalam hal tersebut.

2.2. Keamanan *Disk Encryption*

Berbicara mengenai enkripsi manapun pasti menyangkut soal keamanan. Berkaitan dengan *disk encryption*, implementasi dikatakan aman bila pihak luar (secara khusus pihak penyerang), yang memasukkan plainteks, mendapatkan cipherteks dan melakukan modifikasi terhadap cipherteks tersebut tidak mendapatkan informasi apa pun mengenai plainteks pada tiap sektor.

Pengertian aman di atas merupakan pemenuhan kebutuhan akan kerahasiaan. Sedangkan aman dari segi integritas berarti modifikasi yang dilakukan oleh pihak tidak berwenang terdeteksi sebelum data yang termodifikasi tersebut dipakai.

2.3. On-the-fly Encryption

Salah satu yang harus dipertimbangkan dalam implementasi disk encryption adalah bahwa Implementasi enkripsi dan dekripsi haruslah efisien untuk setiap sektor.

Salah satu karakteristik hard disk sebagai media penyimpanan adalah akses acak pada level sektor. Data yang dimaksudkan untuk disimpan di hard disk tidak ditempatkan secara berurutan dari sektor pertama sampai sektor terakhir. Karena itu, poin pertama dalam implementasi *disk encryption* di atas muncul. Enkripsi dilakukan sebelum data terikat pada sektor tertentu di *hard disk*, dari pengertian inilah muncul istilah *on-the-fly encryption*.

2.5. Jenis *Disk Encryption*

Implementasi disk encryption memberikan pilihan menyangkut seberapa banyak dari disk yang menjadi tempat penyimpanan data yang terenkripsi. Enkripsi dapat dikenakan pada seluruh data yang terdapat pada hard disk. Jenis disk encryption yang seperti ini dinamakan *Entire Hard Disk Encryption*. Jenis *Virtual Hard Disk Encryption* akan seolah-olah menciptakan wilayah, yang tentu lebih kecil dari pada seluruh hard disk, dan mengizinkan data yang disimpan pada wilayah tersebut terenkripsi. Berikut merupakan perbandingan kedua jenis disk encryption tersebut.

2.5.1 Entire Hard Disk Encryption (EHD)

Hard disk merupakan salah satu hardware pada komputer/laptop. Melakukan enkripsi terhadap keseluruhan hard disk akan membawa faktor hardware untuk dipikirkan. Tiap produk hard disk mempunyai perbedaan satu sama lain yang cukup untuk membuat pembangunan suatu sistem EHD encryption akan cocok terhadap produk tertentu juga.

Selain itu, ketika isi seluruh hard disk terenkripsi, bukan hanya data biasa saja yang terlibat, tetapi juga data sistem operasi. Hal tersebut memperluas aspek keamanan penggunaan komputer/laptop. Sejak pengguna komputer/laptop menyalakan komputer dan melakukan *boot*, pengguna akan dimintai *username* dan atau *password*.

Hal tersebut tidak terlalu mengesankan karena kemudian timbul pertanyaan seperti jika boot dilakukan dari sebuah *floppy disk*. Apakah dengan demikian pihak luar (penyerang) dapat tetap masuk ke dalam sistem? Apakah ada cara untuk mengatur BIOS untuk melakukan boot dari hard disk saja? Kalaupun hal itu dapat terjadi, apa yang terjadi jika suatu hari nanti, pada kondisi yang khusus, *boot* perlu dilakukan dari *recovery diskette*?

Pengguna sistem EHD tentu tetap bersyukur karena tidak ada sejengkal wilayah pun dari hard disk komputer/laptopnya yang tidak terjangkau sentuhan sihir enkripsi. Namun, penggunaan sistem enkripsi seperti ini dalam suatu jaringan memiliki beban pada saat awal instalasi. Berkaitan dengan apa yang dapat di-share sedangkan seluruh hard disk terenkripsi, instalasi sistem pada jaringan membutuhkan adminstrator dan waktu berjam-jam untuk melakukan beberapa penyesuaian.

Proses instalasi yang sulit merupakan tanda bahwa proses pemeliharaan (*maintance*) juga tidak mudah. Hal-hal yang perlu dipertanyakan misalnya jika pengguna lupa password, atau apa yang terjadi kalau sistem operasi atau bahkan *hardware* rusak. Belum lagi masalah perangkat lunak versi baru yang muncul dalam kurun waktu tertentu. Apakah sistem mampu mengikuti perubahan-perubahan yang ada pada perangkat lunak tersebut, mengingat instalsi perangkat lunak itu dilakukan pada hard disk yang terenkripsi.

Semua kesulitan itu dibayar dengan kemudahan penggunaan sistem ini. Pengguna cukup memberikan user name dan atau password saat melakukan boot dan tidak perlu khawatir. Apabila hard disk jatuh ke tangan pihak luar, data yang ada di dalamnya tidak dapat terbaca dengan baik karena telah terenkripsi.

2.5.2. Virtual Hard Disk Encryption (VHD)

Tidak seperti sistem EHD encrypition, VHD encryption tidak bermasalah dengan produk hardware tertentu. Sepanjang ada kesesuaian sistem dengan sistem operasi tempat VHD encryption terinstal, semua akan berjalan lancar.

Pada dasarnya, sistem ini mengizinkan pengguna untuk membuat *virtual drive*. Tiap drive baru dapat dimunculkan dan diakses jika pengguna memberikan *password* yang tepat. Satu perbedaan lagi antara *virtual drive* ini dengan drive biasa adalah seluruh data yang disimpan pada drive ini secara otomatis dan transparan akan dienkripsi.

Kata *virtual* pada *virtual drive* dipakai karena sebetulnya drive ini berupa sebuah file. Pada saat membuat *drive* ini, sebenarnya pengguna membuat file yang terenkripsi. Proses dekripsi terhadap tersebut akan memunculkan data apa saja yang terdapat di dalamnya. Bagi pengguna, data yang dimunculkan tersebut seolah-olah berada pada satu *drive,* yang baru muncul saat proses dekripsi dilakukan. File yang dimaksud biasanya dinamakan *file volume*.

Karena bentuk aslinya berupa file, seluruh data yang disimpan di dalamnya bisa diduplikasi sebanyak file tersebut di-copy. Namun hal tersebut tidak menjadi masalah karena file yang di-copy juga *password protected*. Tanpa sistem enkripsi yang sama dan tentu saja, *password*, file tersebut tidak akan dapat dibuka. Kondisi ini dapat dihitung sebagai keuntungan, karena tiap *copy file* juga berarti *backup* data.

Proses instalasi sistem ini tidaklah terlalu rumit, termasuk instalasi pada jaringan. Instalasinya dapat dilakukan tanpa harus melakukan banyak penyesuaian pada komputer tempat instalasi terjadi atau pada komputer yang lain.

Keuntungan lain dari VHD encryption adalah banyak pengguna dapat membuat sendiri *virtual drive* mereka pada satu komputer yang sama, sepanjang kapasitas hard disk masih mencukupi. Mereka tidak perlu berbagi wilayah enkripsi dan berbagi *password* yang sama.

Kerusakan sistem operasi atau hardware juga tidak menjadi sesuatu yang meresahkan asalkan memiliki *back up file volume*. Proses *back up* sendiri, seperti yang dijelaskan di atas, amat mudah dilakukan.

3. Mode Operasi *Cipher* Blok untuk disk encryption

Mode operasi diperlukan dalam melakukan *disk encryption*. Hal tersebut dikarenakan apabila enkripsi ingin dilakukan pada satu sektor hard disk (512 byte), cipher blok tetap beroperasi dalam satuan blok (biasanya 8 atau 16 byte). Mode operasi akan mengatur bagaimana keseluruhan blok dalam satu sektor tersebut dienkripsi.

Pada awal bagian 3, akan dibahas mengenai informasi tambahan yang digunakan dalam mode operasi. Banyak mode operasi menyebut informasi tambahan itu sebagai *Initial Vector*), walaupun ada yang menggunakan istilah lain untuk maksud yang sama. Kemudian, bagian ini juga akan membahas mengenai mode operasi yang digunakan untuk *disk encryption* satu per satu.

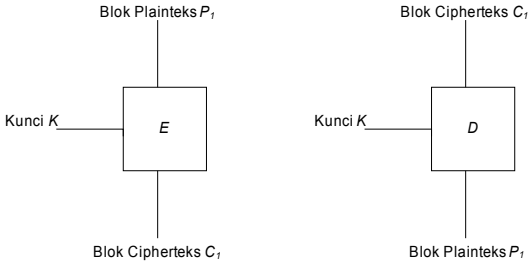
Mode operasi pertama yang akan dibahas adalah mode *Cipher Block Chaining* (CBC). CBC dapat digolongkan dalam mode operasi tradisional. Mode operasi tradisional biasanya sederhana dan sudah cukup lama digunakan. Mode CBC ini dapat digunakan untuk *disk encryption*, dan sudah banyak digunakan. Sayangnya, kelemahan mode ini ternyata cukup banyak.

Mode operasi kedua yang muncul adalah ECB-Mix-ECB (EME). Mode operasi ini ditemukan oleh Haveli dan Rogaway, yang juga menemukan mode operasi CBC-Mask-CBC (CMC). Perbedaan CMC dengan EME merupakan proses paralel yang terdapat pada mode EME. Dari kedua nama mode operasi tersebut dapat dilihat bahwa mode operasi ini menggunakan ulang mode operasi tradisional yang telah ada (ECB dan CBC).

Mode operasi EME merupakan kandidat SISWG untuk standar enkripsi pada media penyimpanan, khususnya sebagai *wide cipher mode.*
Mode terakhir yang dibahas adalah mode Liskov-Rivest-Wagner (LRW). Mode yang diberi nama sesuai nama tiga orang penemunya ini kandidat SISWG untuk *narrow cipher mode*.

3.1. Informasi tambahan untuk mode operasi

Mengapa diperlukan informasi tambahan (selain kunci) untuk sebuah mode operasi? Seperti yang tampak pada , penggunaan mode operasi seperti *Electronic Code Book* (ECB) yang tidak menggunakan informasi tambahan apa pun, sangatlah tidak aman, karena blok plainteks yang sama akan menjadi blok cipherteks yang sama dan menimbulkan pola data tertentu. Pola itu dapat menjadi petunjuk untuk penyerang.



Gambar 1 Skema mode operasi ECB

Mode operasi lain melakukan enkripsi pada blok secara independen, artinya kemungkinan blok plainteks yang sama menjadi blok cipherteks yang sama nyaris nol. Kondisi itu dimungkinkan dengan menggunakan bantuan informasi tambahan saat melakukan enkripsi.

Informasi tambahan tersebut tidak harus disediakan oleh pengguna. Bentuk penggunaan informasi tambahan tersebut misalnya sebagai bit-bit awal yang digunakan untuk melakukan xor terhadap plainteks. Banyak istilah yang muncul untuk informasi tambahan seperti ini. Istilah yang paling umum adalah *Initial Vector (IV)*. Sedangkan *tweaks, nonces, spice, counter, atau diversification parameter* adalah istilah lain yang pada dasarnya merupakan informasi tambahan tersebut. (Bagian makalah selanjutnya akan menyebut informasi tambahan ini dengan istilah IV). Kebanyakan mode operasi yang digunakan untuk disk encryption memerlukan informasi tambahan seperti ini.

Dua karakter penting untuk sebuah IV adalah *predictability* dan *uniqueness*. *Predictability* menyangkut kemungkinan IV untuk diprediksi. Pada umumnya, sistem disk encryption tidak men-*generate* atau menyimpan nilai IV. Nilai IV diambil dari nomor sektor (istilah sektor digunakan untuk blok pada hard disk, berukuran 512 *byte*), karena itu nilai IV yang sama dapat dipakai berulang kali. Hal itu juga berarti nilai IV dapat diperkirakan. Sedangkan uniqueness berarti IV sebisa mungkin harus unik.

Pemilihan IV seperti ini adalah salah satu faktor yang harus diperhatikan saat implementasi *disk encryption*. Ada banyak cara untuk mendapatkan IV ini. Beberapa di antaranya adalah:

- Plain-IV
  IV didapatkan dari nomor sektor, yang diberi padding berupa nilai ‘0’ bila dibutuhkan
- ESSIV
  IV didapatkan dari nilai sektor yang dienkripsi dengan fungsi hash dari kunci.

$$\mathrm{IV}(\mathrm{sector}) = \mathcal{E}_{\mathbf{salt}}(\mathrm{sector})$$
$$\mathrm{salt} = H(K)$$

- Plumb-IV
  Nilai IV didapat dari fungsi hash terhadap plainteks, mulai dari blok kedua sampai blok terakhir.

Pemilihan IV dapat menjadi sasaran serangan. Misalnya pada serangan watermark, penggunaan public IV yang dapat diprediksi (memiliki predictability yang rendah) dapat membocorkan keberadaaan data.

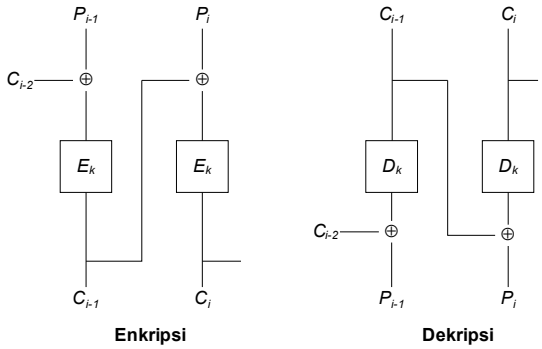
3.2. Mode Operasi CBC

3.2.1. Deskripsi mode 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 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.

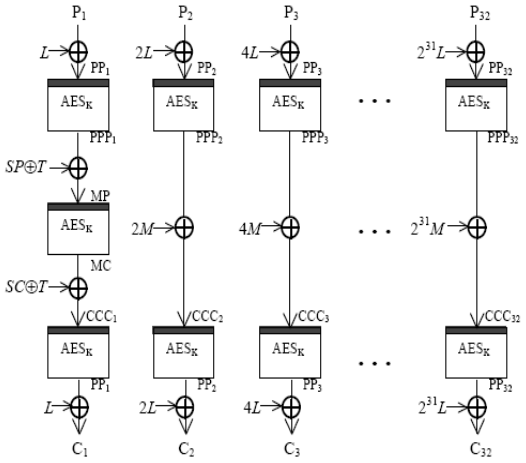
Pada mode *CBC*, blok plainteks yang sama menghasilkan blok cipherteks yang berbeda hanya jika blok-blok plainteks sebelumnya berbeda.

3.2.1. Kelemahan mode CBC

Sampai saat ini, sudah banyak ditemukan kelemahan dalam mode CBC, di antaranya adalah content leak attack, watermark attack, dan data modification leak. Sementara watermark attack dikarenakan pemilihan IV, content leak attack menyerang pengaruh bentuk *chaining* yang sederhana pada proses dekripsi. *Chaining* juga menjadi pokok utama dalam *data modification leak*, di mana penyerang dapat mempelajari *data history*, terutama saat terjadi perubahan terhadap plainteks dan kemudian cipherteks.

3.3. Mode Operasi EME

Mode operasi EME menerima masukan blok dengan ukuran besar, 512 byte. Oleh karena itu mode operasi ini disebut *wide cipher mode*.



Gambar 3 Skema Enkripsi Mode EME

Langkah –langkah enkripsi dengan mode EME:

1. menghitung nilai L, yaitu sebuah blok berukuran 16 byte. Cara menghitungnya adalah dengan mengenkripsi blok 0 dengan algoritma enkripsi dan kunci K (blok 0 yaitu satu blok 16 byte yang diisi dengan bit 0 semua). Hasil enkripsi dengan blok 0 disimpan pada L. Kemudian dilakukan perkalian polinomial antara L dan blok 0.
2. Membagi blok 512 byte menjadi 32 blok yang masing-masing berukuran 16 byte (P1 – P 32).
3. Setiap blok plainteks Pn di-XOR dengan nilai 2^nL menjadi blok PPn
4. PPn kemudian dienkripsi dengan algoritma dan kunci K menjadi blok PPPn
5. Untuk blok pertama (P1), menghitung nilai SP (PPP2 ⊕ PPP3 ⊕ ... ⊕ PPP32).

Kemudian nilai MP didapat dari SP ⊕ **T** ⊕ PPP1, di mana T adalah *tweak*.

6. Masih untuk blok pertama, MP kemudian dienkripsi kembali menjadi MC. Kemudian dilakukan SP ⊕ **T** ⊕ MC, sehingga didapatkan CCC1.

7. Sementara itu, untuk blok ke-2 sampai ke-32, CCCn didapat dari XOR yang dilakukan antara PPPn dengan 2^nM. Nilai M = MP ⊕ MC.

8. Enkripsi dilakukan sekali lagi terhadap CCCn menjadi PPn.

9. Langkah terakhir adalah melakukan XOR antara PPn dengan 2^nL, hingga didapatkan C, cipherteks final.

Langkah –langkah proses dekripsi mode EME:

1. menghitung nilai L, yaitu sebuah blok berukuran 16 byte. Cara menghitungnya adalah dengan mengenkripsi blok 0 dengan algoritma enkripsi dan kunci K (blok 0 yaitu satu blok 16 byte yang diisi dengan bit 0 semua). Hasil enkripsi dengan blok 0 disimpan pada L. Kemudian dilakukan perkalian polinomial antara L dan blok 0.
2. Membagi blok 512 byte menjadi 32 blok yang masing-masing berukuran 16 byte (C1 – C 32).

3. Setiap blok plainteks Cn di-XOR dengan nilai 2^nL menjadi blok CCn

4. CCn kemudian dienkripsi dengan algoritma dan kunci K menjadi blok CCCn

5. Untuk blok pertama (C1), menghitung nilai SC (CCC2 ⊕ CCC3 ⊕ **...** ⊕ CCC32). Kemudian nilai MC didapat dari SC ⊕ **T** ⊕ PPP1, di mana T adalah *tweak*.

6. Masih untuk blok pertama, MC kemudian dienkripsi kembali menjadi MP. Kemudian dilakukan SP ⊕ **T** ⊕ MP, sehingga didapatkan PPP1.

7. Sementara itu, untuk blok ke-2 sampai ke-32, PPPn didapat dari XOR yang dilakukan antara CCCn dengan 2^nM. Nilai M = MC ⊕ MP.

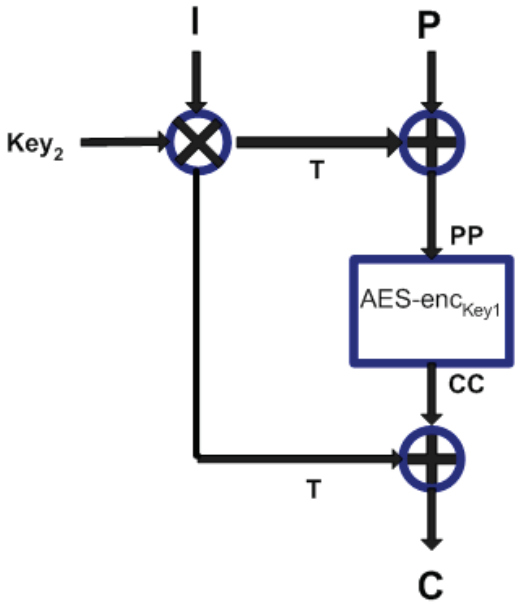
8. Enkripsi dilakukan sekali lagi terhadap PPPn menjadi CCn.

9. Langkah terakhir adalah melakukan XOR antara CCn dengan 2^nL, sehingga didapatkan P, plainteks final.

Struktur proses enkripsi dan dekripsi pada mode EME simetris.

3.4. Mode Operasi LRW

Mode operasi LRW bekerja sebagai *narrow cipher mode*. Mode operasi ini menerima blok dengan ukuran lebih kecil dari pada mode EME. Ukuran blok yang diterima biasanya 16 byte. Selain blok plainteks/cipherteks mode operasi LRW juga menerima masukan berupa 2 buah kunci yang memiliki panjang blok yang sama dengan plainteks/cipherteks (Key1 dan Key2), juga 16 byte blok I yang merepresentasikan indeks dari block/posisi logical data. Kunci pertama dan kedua haruslah tidak terikat satu sama lain, artinya kunci 2 tidak dibangkitkan dengan cara apa pun dari kunci 1.



Gambar 4 Skema Enkripsi LRW

Keterangan:

- XOR
- Perkalian dari 2 polinomial pada medan Galois 2^7/128 (sesuai dengan panjang blok: 128 bit/16 byte)
  [keterangan mengenai medan Galois terdapat pada bagian 4]

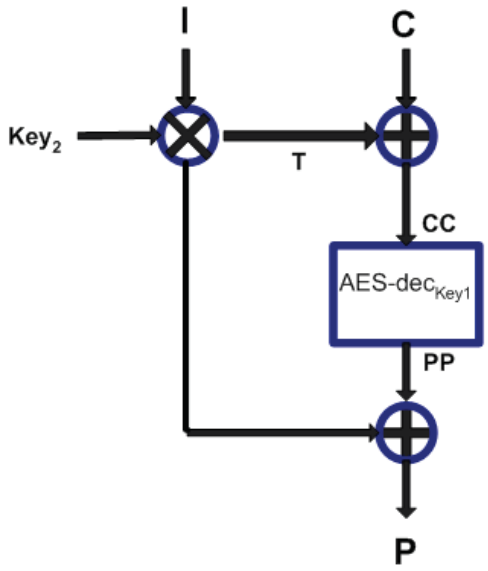
Langkah-langkah enkripsi dengan mode LRW, sesuai dengan gambar di atas adalah:

1. menghitung hasil perkalian 2 polinomial antara Key2 dan n, hasilnya kemudian disimpan di T
2. melakukan XOR antara plainteks P dan T, hasilnya dimasukkan dalam PP.

3. melakukan proses enkripsi dengan algoritma tertentu (misalnya Rijndael/AES) dan kunci Key1, dan mendapatkan cipherteks CC.
4. melakukan XOR antara cipherteks CC dan T, hasil inilah yang merupakan cipherteks final (C).

Dengan demikian, sebuah cipherteks pada blok ke-n didapat dari rumus berikut:

$$C_n = \mathcal{E}_k(P_i \oplus H(T)) \oplus H(T)$$



Gambar 5 Skema dekripsi mode LRW

Langkah-langkah dekripsi dengan mode LRW:
1. menghitung hasil perkalian 2 polinomial antara Key2 dan n, hasilnya kemudian disimpan di T
2. melakukan XOR antara cipherteks C dan T, hasilnya dimasukkan dalam CC.
3. melakukan proses dekripsi dengan algoritma tertentu (misalnya Rijndael/AES) dan kunci Key1, dan mendapatkan plainteks PP.
4. melakukan XOR antara plainteks PP dan T, hasil inilah yang merupakan plainteks final (P).

Seperti yang dapat dilihat pada Gambar 3 dan Gambar 4, struktur proses enkripsi dan dekripsi pada mode LRW simetris.

4. Galois Field (Medan Galois)

Medan Galois merupakan sebutan lain untuk medan berhingga, sebagai penghargaan terhadap Evariste Galois yang menemukan hubungan antara grup dan persamaan polinomial pada tahun 1832. Medan Galois adalah medan berhingga dengan p^n elemen, yang dalam hal ini p adalah bilangan prima dan n >= 1.

Notasi : $GF(p^n)$

Dimungkinkan bagi kita membentuk GF(q) di mana $q = p^n$, n = 1, 2, ... Misalnya GF(8) = $GF(2^3)$ mengandung 8 elemen. Tidak ada medan Galois untuk q = 6 karena 6 bukan perpangkatan dari bilangan prima.

Kasus paling sederhana adalah bila n = 1 sehingga menjadi GF(p) di mana elemen-elemennya dinyatakan di dalam himpunan {0,1,2,..., p-1} dan operasi penjumlahan dan perkalian dilakukan dalam modulus p. Medan semacam ini dinamakan medan prima (*prime field*). Untuk $GF(p^n)$ penjumlahan dan perkalian tidak dilakkan dalam modulus p^n tetapi dalam modulo f(x) yang dalam hal ini f(x) adalah polinom derajat m.

Bentuk umum polinom berderajat m adalah:

$$F(x) = b_m x^m + b_{m-1} x^{m-1} + ... + b_1 x + b_0$$

Yang dalam hal ini b_m, b_m-1, .. b_0 adalah koefisien-koefisien polinom. Di dalam medan prima GF(p) koefisien-koefisien polinom adalah bilangan-bilangan di dalam {0,1,2,.. p-1} dan operasi aritmetika dilakukan terhadap koefisien polinom dalam modulo p. Sebagai contoh, polinom-polinom berikut adalah di dalam GF(2):

$F1(x) = x^3 + x^2 + 1$
$F2(x) = x^5 + x3$

Melakukan operasi perkalian pada polinom di dalam GF(2) sama seperti perkalian biasa, misalnya:

$$x^3 . x^2 = x^5$$

Untuk polinom di dalam $GF(p^n)$ operasi aritmetika pada koefisien polinom tidak dilakukan dalam modulo p, tetapi terhadap polinom yang tidak dapat direduksi lagi (irreducible) dalam GF(p) Sebuah polinom dikatakan tidak dapat direduksi jika ia tidak dapat dinyatakan sebagai perkalian dari dua buah polinom lain.

5. 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*.

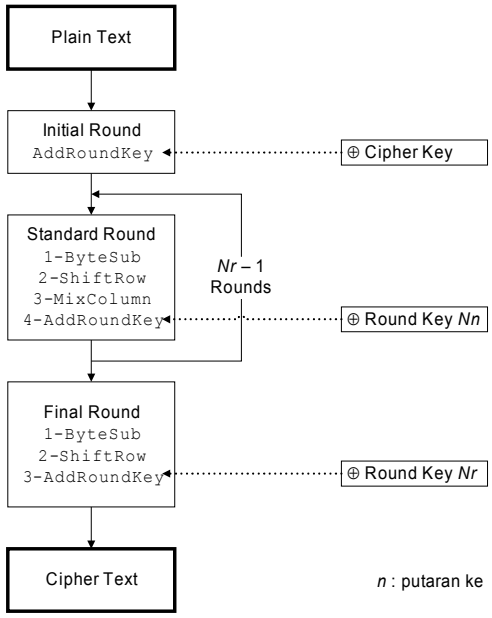
Diagram proses enkripsi *AES* dapat dilihat pada Gambar 8.

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).

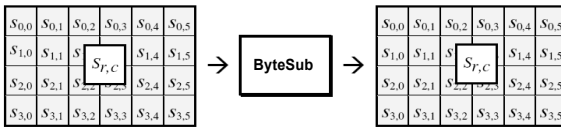
Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of byte* dua dimensi, `state`, yang berukuran `NROWS x NCOLS`. Elemen *array* `state` diacu sebagai *S*[*r*,*c*], dengan $0 \le r < 4$ dan $0 \le c < Nc$ (*Nc* adalah panjang blok dibagi 32). Pada *AES*,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



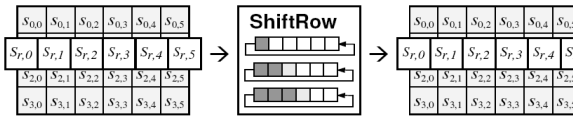
Gambar 6 Diagram Proses Enkripsi *AES*

Tabel 1 Tabel *S-box* yang digunakan dalam transformasi *ByteSub() AES*

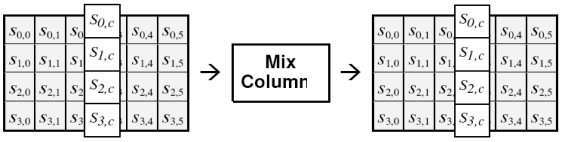
hex	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	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



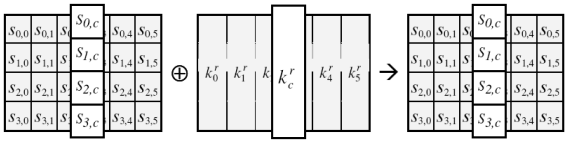
Gambar 7 Ilustrasi Transformasi *ByteSub() AES*



Gambar 8 Ilustrasi Transformasi *ShiftRow() AES*



Gambar 9 Ilustrasi Transformasi *MixColumn() AES*



Gambar 10 Ilustrasi Transformasi *AddRoundKey() AES*

3.3 Cipher Kebalikan (*Inverse Cipher*)

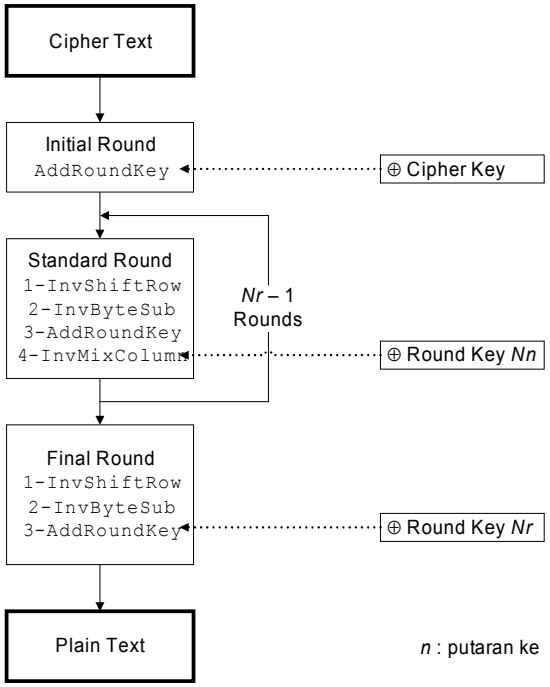
Cipher kebalikan merupakan algoritma kriptografi *AES* yang digunakan untuk melakukan proses dekripsi cipherteks menjadi plainteksnya. Secara garis besar, *cipher* kebalikan yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

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 2 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 11 Diagram Proses Dekripsi *AES*

6. Kesimpulan

Kesimpulan yang dapat diambil dari studi mengenai *disk encryption* dengan menggunakan mode operasi LRW dan algoritma Rijndael adalah :

1. Disk encryption merupakan solusi yang baik untuk meningkatkan keamanan data seiring dengan tren pemakaian laptop yang terjadi di mana-mana.
2. Disk encryption lebih efisien, dan dengan demikian lebih unggul dibandingkan dengan file encryption/enkripsi file yang umumnya digunakan.
3. Dari dua jenis *disk encryption* yang ada, virtual hard disk encrytion (VHD encryption) lebih banyak memberikan keuntungan dari pada entire hard disk encryption (EHD encryption). Kemudahan instalasi, compatibility, kemudahan mendapatkan *back up*, dan kemudahan *maintanance* menjadi faktor pendukung keunggulan *VHD encryption*.
4. Mode operasi LRW lebih mudah untuk diimplementasikan dan cukup kuat. Sekalipun struktur enkripsi dan dekripsi sama simetrisnya dengan mode LRW, mode EME melibatkan banyak operasi sehingga memberi dampak pada performansi sistem enkripsi. Selain itu, masukan blok untuk EME tergolong besar sehingga memakan tempat memori pada saat dijalankan.
5. Algoritma Rijndael merupakan salah satu solusi terbaik sebagai algoritma enkripsi, termasuk dalam penyimpanan data seperti pada *disk encryption*.

DAFTAR PUSTAKA

[1] Fruhwirth, Clemens. *Linux Hard disk Encryption Settings*. http://clemens.endorphin.org/ Tanggal akses: 5 Oktober 2006 pukul 18.00

[2] Fruhwirth, Clemens. 2005. New Methods in Hard Disk Encryption. Institute for Computer Languages Theory and Logic Group – Vienna University of Technology.

[3] IEEE P1619. Draft Proposal for Tweakable Wide-block Encryption. 2004. Tanggal akses: 5 Oktober 2006 pukul 18.00

[4] IEEE P1619. Draft Proposal for Tweakable Narrow-block Encryption. (2004). Tanggal akses: 5 Oktober 2006 pukul 18.00

[5] Liskov, Moses, Ronald L. Rivest, dan David Wagner. Tweakable Block Chiper.

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

[7]http://www.pcdynamics.com/SafeHouse/Encryption.asp, diakses tanggal 25 September 2006

[8] Reflex Magnetic Ltd. Entire Hard Disk Encryption vs Virtual Hard Disk Encryption. June 2004.

[9] http://www.truecrypt.org/ , diakses mulai tanggal 15 September 2006