Teknik-Teknik Enkripsi pada *File System*

Abdul Halim[1], Natan[2], Dziki Yazid Fahmi[3]

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : halim@students.itb.ac.id[1], if12034@students.if.itb.ac.id[2],
if12061@students.if.itb.ac.id[3]

Abstrak

Keamanan informasi digital terhadap pengaksesan oleh pihak yang tidak berhak menjadi masalah tersendiri dalam dunia elektronik saat ini. Berbagai ancaman dapat terjadi, misalnya pelanggaran privasi, pencurian data rahasia perusahaan, tindak kriminal terorganisasi, organisasi teroris dan sebagainya. Kriptografi adalah salah satu harapan yang paling menjanjikan sebagai alat pengaman di era informasi ini. Pengamanan untuk data sensitif yang disimpan pada media perlu mendapat perhatian khusus. Berbagai teknik atau algoritma dapat diterapkan untuk mengamankan data. Secara umum, pengamanan isi data dilakukan oleh program aplikasi terpisah (*stand alone*) sehingga setiap kali mengenkripsi atau mendekripsi berkas plainteks atau cipher harus dilakukan memalui program aplikasi tersebut. Cara ini harus melibatkan *user* sehingga bila frekuensi enkripsi / dekripsi tinggi maka akan merepotkan *user*. Cara lain yang dapat ditempuh adalah dengan melakukan enkripsi pada level *file system*. Cara ini memungkinkan user dapat melakukan enkripsi dan dekripsi berkas seperti layaknya berkas biasa tanpa enkripsi.

***Kata kunci**: enkripsi* file system, *kriptografi*

1. Pendahuluan

Digitalisasi informasi menyentuh hampir seluruh aspek kehidupan kita. Kenyataan ini menimbulkan permasalahan baru. Keamanan informasi digital terhadap pengaksesan oleh pihak yang tidak berhak menjadi masalah tersendiri terutama bila data atau informasi digital tersebut terletak pada *system* terkoneksi. Berbagai ancaman dapat terjadi, misalnya pelanggaran privasi, pencurian data rahasia perusahaan, tindak kriminal terorganisasi, organisasi teroris dan sebagainya.

Kriptografi adalah salah satu harapan yang paling menjanjikan sebagai alat pengaman di era informasi ini. Pengamanan untuk data sensitif yang disimpan pada media kurang aman dan rawan terhadap pencurian data perlu mendapat perhatian khusus. Sebagai contoh, pengguna komputer jinjing atau *laptop* sangat rawan kehilangan *laptop*nya. Bila data di dalam *laptop* dibiarkan tanpa dienkripsi, pencuri dapat dengan mudah melihat isi dokumen. Seringkali isi dokumen yang ada di dalam *laptop* jauh lebih berharga dari *laptop* itu sendiri.

Data atau informasi yang ada di dalam diska dapat diamankan dengan dua cara, yaitu mengamankan secara fisik atau mengamankan isi data tersebut pada saat disimpan di dalam diska. Berbagai teknik

atau algoritma kriptografi dapat diterapkan untuk mengamankan data atau informasi.

Berbagai teknik atau algoritma dapat diterapkan untuk mengamankan data. Secara umum, pengamanan isi data dilakukan oleh program aplikasi terpisah (*stand alone*) sehingga setiap kali mengenkripsi atau mendekripsi berkas plainteks atau cipher harus dilakukan memalui program aplikasi tersebut. Cara ini harus melibatkan *user* sehingga bila frekuensi enkripsi / dekripsi tinggi maka akan merepotkan *user*. Cara lain yang dapat ditempuh adalah dengan melakukan enkripsi pada level *file system*. Cara ini memungkinkan user dapat melakukan enkripsi dan dekripsi berkas seperti layaknya berkas biasa tanpa enkripsi. Enkripsi pada *file system* dapat dilakukan pada *usermode* maupun *kernelmode*. Bila dilakukan pada *kernelmode* berarti perlu melakukan modifikasi *kernel* pada sistem operasi.

2. User mode

User space adalah area memori yang digunakan oleh *user mode applications* dan area memori ini dapat di-*swap* kapan saja bergantung dari kubutuhan *system* saat itu. Aplikasi dengan *user mode* tidak dapat mengakses langsung *kernel space*, sebaliknya aplikasi kernel mode juga tidak dapat mengakses langsung *user space* tanpa mengecek terlebih dahulu apakah *page* yang diperlukan ada di memori atau sedang di-*swap*.

Pada enkripsi *file system* dengan *user mode* terdapat beberapa prinsip yang perlu diperhatikan yaitu:

1. Penggunaan *virtual file system*, yaitu layer abstrak yang berjalan di atas *file system* yang sebenarnya. Tujuan dari *virtual file system* ini adalah agar pengguna dapat mengakses berbagai *file system* yang berbeda dengan cara yang seragam. Dengan kata lain pengguna hanya mengenal satu *file system* yaitu *virtual file system* yang dibentuk, sedangkan akses ke *file system* yang sebenarnya diatur oleh *virtual file system* tersebut.
2. Proses enkripsi saat menulis dan proses dekripsi saat membaca dilakukan terhadap *file* pada *virtual file system*. Dalam hal ini *virtual file system* akan menerima kunci dari pengguna yang akan digunakan untuk melakukan proses kriptografi ataupun menurunkan kunci turunan. Kunci dari pengguna ini hanya akan diminta sekali yaitu saat pengguna pertama kali masuk. Pada saat itu pula *virtual file system* akan dibentuk untuk pengguna tersebut. Melalui mekanisme tersebut maka *file* yang disimpan pada *file system* fisik akan berupa *cipher* yang tidak akan dapat dimengerti oleh pengguna lain karena sudah dienkripsi oleh *virtual file system*, sedangkan saat akan digunakan maka *virtual file system* akan membaca dari *file system* fisik, mendekripsinya dengan kunci pengguna, baru kemudian menyerahkannya ke aplikasi pengguna.
3. Pada enkripsi *file system* dengan user mode maka keseluruhan proses dijalankan sebagai proses yang menggunakan user space. Penggunaan user space membuat pengembangan aplikasi ini dimasa yang akan datang menjadi lebih mudah begitu pula dalam pencarian kesalahan (*debugging*).

Pada linux untuk enkripsi *file system* sudah didukung beberapa pilihan dengan

tujuan memudahkan proses kriptografi pada level *file system*. Misalnya *Cryptographic File System* (CFS) dan *Transparent Cryptographic File System* (TCFS) menggunakan NFS dengan ext2. EncFS menggunakan Fuse untuk membentuk *file system* di user spacenya. Fuse yaitu *file system* in User Space. Fuse itu sendiri adalah modul yang mengijinkan pengguna untuk membuat *file system* tanpa perlu memodifikasi kode pada kernel.

3. Kernel mode

Kernel merupakan perangkat lunak bagian sistem operasi yang melakukan eksekusi di bagian *supervisor state* [3] . Kernel merupakan inti dari sistem operasi. Kernel dirancang sebagai perangkat lunak terpercaya yang mendukung operasi untuk perangkat lunak lainnya. Aplikasi yang berjalan pada *kernel mode* berarti mempunyai akses lebih luas karena dapat melakukan eksekusi pada *supervisor space* serta mampu melakukan pengaksesan terhadap bagian lain dari kernel.

Enkripsi pada mode ini mengharuskan adanya akses terhadap kernel serta melakukan modifikasi terhadap kernel tersebut. Untuk Sistem Operasi linux atau sistem operasi lainnya yang *opensource*, akses terhadap kernel ini dibuka untuk siapapun sementara untuk sistem operasi yang bersifat komersial, integrasi enkripsi *file* system dilakukan oleh vendornya, karena hanya vendor yang mengetahui secara detail struktur kernel dari sistem operasinya.

Untuk sistem operasi *open source*, enkripsi pada *file* system dengan mode kernel dilakukan melalui tahapan sebagai berikut :

1. Penambahan paket enkripsi ke dalam kernel. Langkah ini sekaligus melakukan konfigurasi terhadap kernel configuration *file*
2. Penambahan hard drive baru terhadap *system* yang akan menampung partisi tunggal yang terenkripsi.
3. Penambahan direktori untuk menampung *file* kunci. Tiap *file* kunci berasosiasi dengan partisi yang terenkripsi.
4. Menyertakan partisi terenkripsi pada kernel
5. Membuat *file* system pada perangkat terenkripsi.
6. Melakukan mounting terhadap enkripsi terpartisi.
7. Melakukan verifikasi terhadap ketersediaan *file* system tersebut.

Untuk sistem operasi komersial, apabila menyediakan fasilitas enkripsi berbasis kernel, maka paket enkripsinya sudah terintegrasi dengan sistem sehingga tidak diperlukan lagi modifikasi terhadap kernel..

Paket enkripsi yang disediakan menyediakan metoda *ciphers* serta algoritma enkripsi yang beragam. Metoda *ciphers* yang biasa dipakai pada enkripsi *file system kernel mode* adalah *simmetric block ciphers*. Algoritma yang biasa dipakai diantaranya adalah Triple DES, Blowfish serta Rijndael. Ketiga algoritma tersebut banyak digunakan untuk mengenkripsi *file system* karena biasa dipakai untuk enkripsi *file* serta secara matematis, masih sangat aman. Selain itu, algoritma lain yang dapat diimplementasikan untuk enkripsi *file system kernel mode* adalah algoritma CAST, GOST, IDEA, MARS, Serpent, RC5, RC6 serta algoritma TwoFish.

Peran yang terdapat di sistem ini dibagi menjadi tiga meliputi [1] :

1. *System administrator*

System administrator melakukan mount terhadap partisi terenkripsi serta bertanggung jawab untuk melakukan modifikasi kernel

2. *Owner*
   Adalah user yang melakukan kontrol terhadap kunci enkripsi dari data
3. *Readers and writers*
   Perbedaan yang paling mendasar antara owner dengan reader/writer adalah bahwa owner memberikan input kunci enkripsi sementara reader atau writer mengakses *file* secara transparan. Seorang owner secara implisit adalah reader atau writer

4. Kelebihan dan Kekurangan

Penggunaan enkripsi *file system* memiliki kelebihan yaitu:

1. Pengguna akan merasa lebih nyaman karena proses memasukkan kunci rahasia hanya sekali pada awal login atau saat proses mount direktori yang akan dijadikan *virtual file system*, berbeda dengan aplikasi enkripsi mandiri yang setiap kali enkripsi / dekripsi memerlukan kunci masukan dari pengguna.
2. Meningkatkan keamanan non-kriptanalis. Dengan mengenkripsi *file system*, informasi yang ada dalam media penyimpanan tetap aman walaupun media tersebut dicuri dan media tersebut dibuka dengan menggunakan boot dari CDROM atau device lain sehingga fitur keamanan lumpuh. Hal ini tidak menjadi masalah karena yang tersimpan dalam *file system* sudah dalam bentuk cipher.

Selain kelebihan, enkripsi *file system* juga memiliki kelemahan, yaitu penggunaan enkripsi seluruh *file system* akan menghabiskan banyak resource dan mengurangi performansi dari *system* itu sendiri. Hal ini diakibatkan karena proses baca dan menulis ke media akan selalu melibatkan proses enkripsi dan dekripsi. Belum lagi masalah yang ditimbulkan untuk melindungi data yang sudah diload ke memori atau page *file*. Hal ini perlu dilakukan karena secara umum data tersebut dalam bentuk plainteks. Untuk mengatasi hal tersebut maka performansi *system* akan mengalami penurunan lebih lanjut.

Enkripsi *file* system dengan kernel mode lebih baik dibandingkan user mode dalam hal performansi. Peningkatan performansinya mencapai 12 – 52 % dibandingkan enkripsi pada user mode atau enkripsi pada level aplikasi.[2]. Adanya peningkatan performansi ini disebabkan jumlah context switches yang lebih sedikit serta eksekusi pada privileged mode. Enkripsi *file system* berbasis kernel mode lebih aman dibandingkan user mode apabila menggunakan algoritma serta metode enkripsi yang sama. Hal ini dikarenakan informasi pada kernel lebih sulit untuk didapatkan serta akses terhadap kernel bersifat private.

Meskipun demikian, enkripsi *file system* berbasis *kernel mode* lebih rumit dibandingkan dengan enkripsi *file system* pada user mode. *File system* driver tidak dapat mengakses *OS services* tradisional yang terdapat pada *user mode*. Selain itu terdapat batasan bagi program yang berjalan pada *kernel mode* dalam hal *kernel paging* serta *IO architecture*. Proses debug terhadap program juga lebih sulit dilakukan karena berjalan pada *address space* tunggal bersama dengan seluruh komponen *kernel mode* dari OS. Proses *debugging* juga membutuhkan *debugger* spesifik untuk *kernel mode.*

5. Contoh Implementasi

Enkripsi dan dekripsi pada level *file system* telah diimplementasikan oleh berbagai kalangan baik bersifat komersial maupun non-komersial untuk berbagai sistem operasi. Masing-masing pihak yang melakukan implementasi kriptografi pada *file system* memiliki tujuan atau *goal* tersendiri. Berikut adalah beberapa contoh implementasi enkripsi pada *file system*:

5.1 Cryptographic File System(CFS)

CFS adalah jalan tengah antara kebutuhan enkripsi pada level *system* tetapi user masih memiliki kontrol terhadap *system* enkripsi tersebut. Bila enkripsi terlalu dekat dengan *system* maka user harus memberikan kepercayaan penuh terhadap komponen yang melakukan enkripsi. Sebaliknya, bila letak enkripsi terlalu dekat dengan user maka akan terlalu merepotkan user.

CFS dikembangkan oleh Matt Blaze[BLAZE] dari AT&T Laboratories. Tujuan utama yang ingin dicapai pengembangan CFS adalah menyediakan layanan penyimapanan *file* secara transparan tanpa merepotkan user.

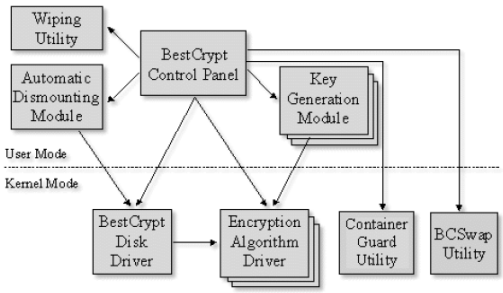
Prototipe CFS yang dikembangkan oleh Matt Blaze seluruhnya diimplementasikan dalam *usermode* pada sistem operasi Linux. *File system* CFS menyediakan diska virtual yang di*mount* pada direktori tertentu tetapi secara fisik berwujud seperti berkas biasa pada *file system* asal. Untuk berkomunikasi dengan diska virtual tersebut digunakan program khusus bernama `cfsd` yang berjalan terus-menerus di *background.* `Cfsd` sebenarnya adalah client NFS yang menerjemahkan operasi-operasi standar *file* system seperti *read, write, open* pada direktori titik *mount.*

5.2 BestCrypt

BestCrypt menyediakan diska virtual dan menganggapnya seperti diska biasa tetapi berkas disimpan dalam keadaan terenkripsi pada saat tidak digunakan. Secara fisik wujud diska tersebut seperti layaknya berkas biasa pada *file system* asal. BestCrypt dapat berjalan pada sistem operasi Microsoft Windows atau Linux dengan format yang sama dan fitur-fitur yang sama pula.

Secara garis besar, BestCrypt untuk Microsoft Windows terdiri atas modul-modul:

- BestCrypt Disk Driver
- BestCrypt Control Panel
- Encryption Algorithm Driver
- Container Guard Utility
- BCSwap Utility
- Wiping Utility
- Automatic Dismount Module
- Key Generation Module



Gambar 1 Skema Modul BestCrypt

Beberapa modul yaitu *BestCrypt Control Panel*, *Wiping Utility*, *Automatic Dismounting Module* dan *Key Generation Module* berjalan di *usermode* sebagai program aplikasi. Modul lainnya diimplementasikan dalam kernelmode yang berguna untuk:

- membuat diska virtual sehingga dapat dikenali oleh *file system* asal.
- melakukan enkripsi pada saat ada operasi pada diska virtual dan transparan terhadap user.

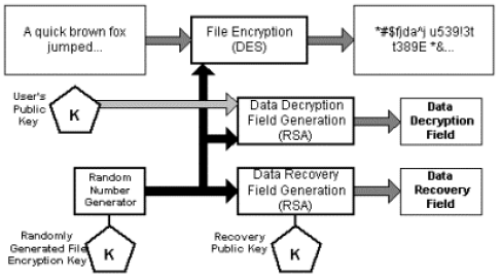
- menjaga agar *file* fisik yang berfungsi sebagai penampungan tidak terhapus secara tidak sengaja
- melakukan enkripsi pada *file* swap secara transparan
- mendapatkan dukungan sistem operasi agar diska virtual dapat melakukan *caching* sehingga performasi bisa meningkat

BestCrypt Key Generation Module dapat menggunakan lebih dari satu modul dalam membangkitkan kunci, misalnya modul yang satu memperoleh kunci dari input string dari user sedangkan modul lain dari *smart card*.

5.3 EFS

Encrypting File System(EFS) merupakan fitur tambahan yang terdapat pada NTFS di Sistem operasi Microsoft Windows 2000 atau turunannya. EFS menggunakan kriptografi kunci publik dengan memanfaatkan arsitektur CryptoAPI yang terdapat di sistem operasi.

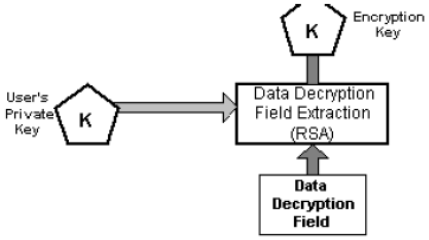
Antarmuka EFS yang disediakan meliputi dua macam yaitu melalui Windows Explorer dan *command line utility*. Setiap berkas atau direktori dienkrip menggunakan kunci acak yang berbeda sehingga kriptanalis semakin sulit melakukan serangan. EFS secara otomatis membuat pasangan kunci publik dan kunci privat untuk user yang melakukan enkripsi bila user tersebut belum memiliki pasangan kunci. Setelah user menandai bahwa berkas atau direktori terenkripsi, proses enkripsi dan dekripsi selanjutnya berlansung transparan, setiap aliran keluar/ masuk byte menuju diska dalam keadaan terenkripsi. Untuk menghindari kehilangan pasangan kunci di kemudian hari sehingga berkas atau direktori yang terenkripsi tidak dapat didekrip maka EFS menyediakan fasilitas yang terintegrasi dengan Local Policy pada komputer *stand alone* atau Domain Controller Policy pada komputer yang terhubung dengan jaringan untuk melakukan *recovery* pasangan kunci. Kunci dapat di*import* atau *export*.



Gambar 2 Proses Enkripsi EFS

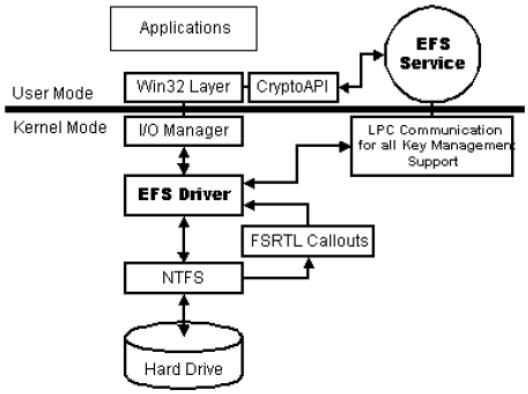
Proses enkripsi adalah sebagai berikut:

- setiap berkas dienkripsi menggunakan kunci acak yang disebut *File Encryption Key* (FEK). Setiap blok pada berkas tersebut dienkripsi dengan FEK yang berbeda.
- FEK independen terhadap pasangan kunci user dan disimpan dalam *Data Decryption Field*(DDF) dan *Data Recovery Field*(DRF). Kunci tersebut dienkrip terlebih dahulu dengan menggunakan kunci publik user sebelum disimpan.



Gambar 3 Proses Dekripsi EFS

Proses dekripsi tidak jauh berbeda dengan proses enkripsi. FEK diperoleh dengan melakukan dekripsi masing-masing FEK yang terdapat di DDF dengan menggunakan kunci privat user. Untuk melakukan dekripsi hanya pada blok tertentu pada berkas tidak perlu melakukan dekripsi keseluruhan berkas karena masing-masing blok dienkrip dengan FEK yang berbeda.



Gambar 4 Arsitektur EFS

EFS diimplementasikan dalam kernel Microsoft Windows 2000 dan menggunakan *non-paged pool* dalam menyimpan kunci untuk menghindari penggunakan *paging*. Komponen-komponen EFS terdiri atas:

- EFS driver: melayani semua permintaan layanan mengenai *key management service* seperti mendapatkan kunci enkripsi. EFS driver terletak di atas NTFS. Informasi ini diteruskan ke EFS *file system run-time library* (FSRTL) untuk melakukan operasi-operasi standar *file* system (*open, read, write, append*, dsb) secara transparan.
- EFS FSRTL: komponen ini mengimplementasikan semua operasi stanar *file* system seperti *open, append, read, write*.
- EFS service: merupakan bagian dari sub*system* keamanan, digunakan untuk berkomunikasi dengan EFS driver. EFS service juga menyediakan dukungan untuk Win32 API untuk melakukan enkripsi, dekripsi, *recovery*, *import* dan *export* kunci.
- Win32 API: menyediakan antar muka bagi bahasa pemrograman untuk melakukan enkripsi, dekripsi dan *recovery*.

6. Kesimpulan

Enkripsi pada *file* system menawarkan sisi keamanan transparan untuk pengguna sistem operasi. Enkripsi *file* s*ystem* ini dapat diimplementasikan pada *user mode* dan kernel mode. Tiap mode implementasi memiliki kelebihan dan kekurangannya masing-masing.

Saat ini, sudah banyak aplikasi-aplikasi sebagai perangkat enkripsi pada *file* system baik itu menggunakan *user mode* ataupun *kernel mode*. Contoh dari aplikasi itu adalah CFS, BestCrypt, dan EFS. CFS merupakan contoh enkripsi *file* system user mode. Sementara itu, BestCrypt serta EFS merupakan contoh implementasi enkripsi *file system kernel mode* meskipun BestCrypt mengimplementasikan beberapa fitur pada *user mode*.

Referensi

[1] Charles P.Wright, Dave Jay, Zadok Erez .NCryptfs: A Secure and Convenient Cryptographic *File* System. http:// www.fsl.cs.sunysb.edu/docs/ncryptfs/ncryptfs.html. Diakses tanggal 4 Januari 2006

[2] Erez Zadok, Ion Badulescu, and Alex Shender. *Cryptfs: A Stackable Vnode Level Encryption File System.* 1 : 18

[3] Nutt, Gary.*Operating Systems A modern perspective.* 2000.65 :21.

[4] Blaze, Matt. *A Cryptographic File System for Unix*. Proc ACM Conference on Communication and Computing Security. November. 1993.

[5] Petullo, W Michael. *Implementing Encrypted Home Directories*. July. 2003.

[6] Hindistan, KIVILCIM. *Userspace Filesystem Encryption with EncFS.* April. 2005.

[7] Bauer, Mick. *BestCrypt: Cross-Platform Filesystem Encryption*. June. 2002.

[8] *Windows 2000 Encrypted File System Weaknesses*.

[9] Wreski, Dave. *Filesystem Encryption*. August. 2000.

[10] *Encrypting File System in Windows XP and Windows Server 2003*. April. 2003.