Studi dan Analisis Enkripsi Sistem File Pada Sistem Operasi Macintosh

Nikolaus Indra / 13508039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
nikolausindra@yahoo.com

***Abstract*—Saat ini hampir semua orang memiliki komputer pribadi atau laptop untuk membantu kegiatannya sehari-hari. Semua orang yang menggunakan komputer pasti akan menyimpan data-data dari yang bersifat penting hingga data-data yang sudah tidak terpakai / bersifat tidak penting sekalipun.**

Sistem operasi (Linux, Windows, Macintosh, dan lain-lain) yang digunakan oleh pengguna memberikan fasilitas untuk melakukan pengamanan data-data yang berada dalam komputer yang bersangkutan. Namun banyak pengguna (awam) yang tidak mengetahui apa yang dilakukan oleh fasilitas pengamanan data pada sistem operasi yang digunakannya.

Oleh karena itu pada makalah ini, penulis akan melakukan studi dan analisis enkripsi sistem file pada sistem operasi Macintosh. Pada sistem operasi Macintosh tersedia beberapa cara untuk melakukan pengamanan data seperti fitur FileVault, input secara langsung menggunakan Terminal, dan juga tersedia fungsi (API) untuk membangun aplikasi yang membutuhkan pengamanan data.

***Index Terms*—Macintosh, FileVault, Terminal, API, enkripsi.**

I. PENDAHULUAN

Sejak beberapa tahun yang lalu, pengguna sistem operasi Macintosh sudah semakin banyak. Sistem operasi Macintosh terkenal karena ringan (UNIX based), cepat, dan tidak dapat terserang virus layaknya sistem operasi Windows. Oleh karena alasan tersebut banyak orang menggunakan sistem operasi Macintosh ini.

Tentu saja orang-orang yang menggunakan Macintosh akan menyimpan banyak data baik yang bersifat penting, pribadi hingga data-data yang bersifat tidak terpakai / tidak penting. Walaupun terdapat data-data yang tidak begitu penting tetap saja pengguna Macintosh ingin semua data-data yang berada dalam komputer mereka terutama yang bersifat penting tersebut (pribadi) tidak dapat diakses atau dilihat oleh orang yang tidak berkepentingan. Atau bila pengguna kehilangan komputer / laptop miliknya, data-data penting tersebut akan tetap tidak dapat dibaca oleh orang lain yang menemukan komputer mereka. Dengan kata lain, keamanan data merupakan sesuatu yang sangat dibutuhkan pengguna komputer.

Terdapat tiga cara utama untuk melakukan perlindungan data tersebut: *invisibility*, kata sandi (*password*), dan enkripsi. *Invisibility* yaitu membuat file yang diinginkan tidak terlihat, disembunyikan agar pengguna lain tidak dapat melihat file tersebut. Kata sandi dapat melakukan pengamanan pada data dengan cara meminta pengguna untuk memberikan kata sandi untuk dapat mengakses file yang diinginkan. Dan yang terakhir adalah enkripsi, melakukan konversi data menjadi *cipher*.

Sistem operasi Macintosh dapat melakukan ketiga cara utama untuk mengamankan data. Namun tidak semua pengguna Macintosh mengetahui cara kerja / pakai ketiga metode tersebut. Oleh karena itu pada makalah ini penulis akan menjelaskan ketiga metode yang telah disebutkan di atas dengan melakukan studi dan analisis (serta implementasi) tersebut pada sistem operasi Macintosh

II. DASAR TEORI

A. Metode Pengamanan Data

Terdapat 3 cara / metode primer untuk melakukan pengamanan file / data pada computer, yaitu membuat data tidak terlihat (*invisibility*), penggunaan kata sandi (*password*), dan enkripsi.

a. Invisibility

Membuat agar file tidak terlihat merupakan cara pengamanan data yang paling mudah dan sederhana untuk dilakukan pada Macintosh. Para pengguna awam melakukan metode ini untuk mengamankan datanya. Cara melakukannya pada Macintosh sangat mudah. Namun sistem operasi Macintosh tidak memberikan fitur bawaan untuk melakukan penyembunyian file ini. Sehingga untuk melakukannya dapat dilakukan dengan mengunduh aplikasi untuk membuat file tidak terlihat terlebih dahulu (tersedia gratis dan berbayar) untuk melakukan

penyembunyian file. Dan dapat juga dilakukan melalui terminal untuk melakukan metode *invisibility* ini.

b. Kata Sandi (*Password*) dan Enkripsi

Penggunaan kata sandi merupakan cara yang umum untuk digunakan dalam melakukan pengamanan file. Pada Macintosh terdapat fitur yang disebut *Disk Utility* yang tersedia untuk digunakan secara langsung untuk mempopulasikan sebuah / kumpulan file dan ditempatkan pada sebuah penampung (.dmg). Penampung tersebut akan mengenkripsi data-data yang ditampungnya dengan menggunakan algoritma *AES*. Penggunaan *Disk Utility* sangat mudah dan sederhana. Namun kekurangannya file akan diduplikasi sehingga akan memakan tempat pada *harddisk* apabila file / data yang ingin ditampung di dalam *Disk Utility* berukuran besar.

Enkripsi merupakan sebuah proses transformasi / konversi sebuah informasi / data (*plaintext*) dengan menggunakan algoritma sehingga informasi tersebut tidak dapat dibaca (*unreadable*) oleh siapapun yang tidak memiliki kunci (*key*). Hasil dari proses enkripsi adalah sebuah *ciphertext*. Proses untuk mengubah sebuah *ciphertext* ke dalam bentuk *plaintext* kembali disebut dengan dekripsi.

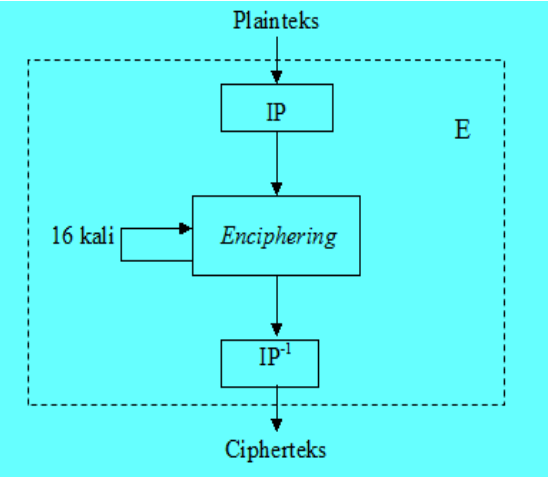
Dibutuhkan algoritma-algoritma tertentu dalam melakukan proses enkripsi data. Pada sistem operasi Macintosh pun tersedia aplikasi yang dapat melakukan enkripsi terhadap sistem dan juga dengan pilihan algoritma yang tersedia.

Pada makalah ini, penulis akan melakukan analisis dan implementasi algoritma enkripsi *AES* dan secara *DES* pada sistem operasi Macintosh.

B. Data Encryption Standard (DES)

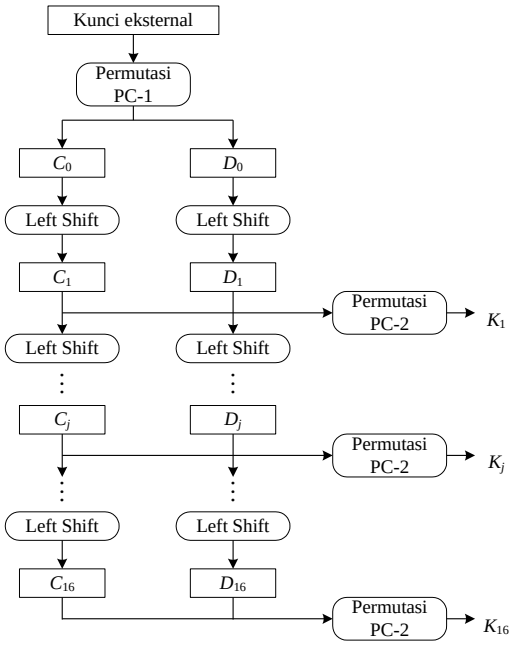
DES adalah sebuah algoritma standard yang termasuk ke dalam kriptografi kunci simetri dan merupakan algoritma *cipher* blok. *DES* dipilih oleh *National Bureau of Standards sebagai Federal Information Processing Standard (FIPS)* oleh Amerika Serikat pada tahun 1976.

DES beroperasi pada ukuran blok 64 bit (kunci eksternal). Namun hanya 56 bit saja yang dipakai sebagai kunci. Setiap blok dienkripsi dalam 16 putaran dan setiap putaran menggunakan kunci internal (56 bit) yang berbeda. Kunci internal tersebut dibangkitkan dari kunci eksternal. Setiap blok mengalami permutasi, 16 putaran *enciphering*, dan inversi permutasi.



Gambar 1. Skema Global *DES*

Seperti yang telah dijelaskan, *DES* menggunakan kunci internal dalam melakukan enkripsi blok. Kunci internal tersebut merupakan kunci setiap putaran (16 putaran) yang dilakukan untuk mengenkripsi blok. Karena dilakukan 16 kali putaran, maka ada 16 kunci internal (K1,K2, … K16). Kunci internal ini dibangkitkan dari kunci eksternal.



Gambar 2. Skema Pembangkitan Kunci Internal *DES*

Tujuan dilakukan permutasi adalah agar bit-bit pada plainteks teracak. Setelah dilakukan 16 kali putaran maka terakhir akan dilakukan permutasi menggunakan matriks permutasi awal balikan (IP^{-1}).

Proses dekripsi pada *DES* merupakan kebalikan dari proses enkripsinya. *DES* menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Pada proses dekripsi, urutan kunci yang digunakan adalah K16,

K15, … , K1.

Keamanan *DES* ditentukan oleh kunci. Panjang eksternal *DES* hanya 64 bit namun yang dipakai hanya 56 bit. Hal inilah yang membuat *DES* dianggap sudah tidak aman lagi. Panjang kunci yang hanya 56 bit dianggap sangat pendek. Maka sejak beberapa tahun yang lalu DES telah digantikan oleh *Advanced Encryption Standard (AES).*

C. Advanced Encryption Standard (*AES*)

AES merupakan algoritma kriptografi kunci simetri yang dikembangkan oleh pemerintah Amerika Serikat. AES terdiri dari tiga blok cipher, AES-128, AES-192, dan AES-256. Angka-angka 128, 192, dan 256 merupakan panjang kunci yang digunakan pada algoritma tersebut. Pada *AES* setiap blok dienkripsi dalam sejumlah putaran tertentu yang dapat dipilih secara independen tidak seperti *DES* yang melakukan 16 kali putaran.

Jenis AES	Panjang Kunci	Ukuran Blok	Jumlah Putaran
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabel 1. Jenis AES

Panjang kunci, ukuran blok, dan jumlah putaran dalam satuan *word*, setara dengan 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.

Dengan panjang kunci 128-bit, maka terdapat sebanyak

$2^{128} = 3{,}4 \times 10^{38}$ kemungkinan kunci.

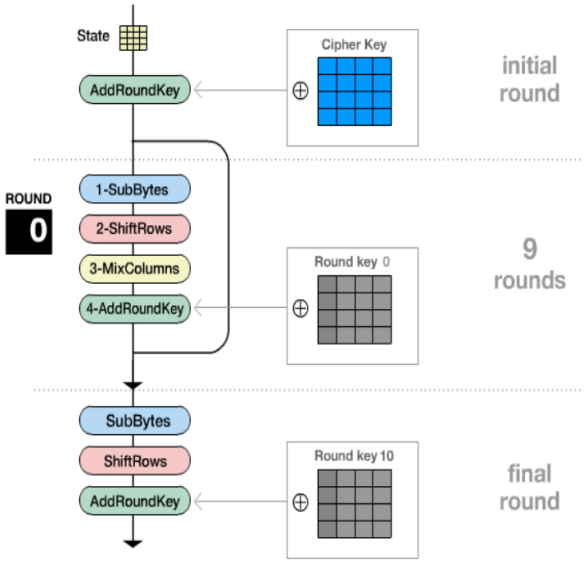
Jika komputer tercepat dapat mencoba 1 juta kunci setiap detik, maka akan dibutuhkan waktu $5{,}4 \times 10^{24}$ tahun untuk mencoba seluruh kunci.

Jika computer tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu $5{,}4 \times 10^{18}$ tahun untuk mencoba seluruh kunci.

Dengan data tersebut, *AES* dapat dipercaya tingkat keamanannya. Untuk mendapat keamanan yang lebih tinggi lagi, dapat digunakan *AES* dengan keamanan kunci 256 bit.

Garis besar algoritma Rijndael (*AES*) yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut (di luar proses pembangkitan *round key*) :

- *AddRoundKey* : melakukan *XOR* antara keadaan awal (plainteks) dengan *cipher key*. Tahap ini disebut *initial round.*
- Putaran sebanyak Nr – 1 kali. Proses yang dilakukan pada setiap putaran adalah :
  - *SubBytes*: subtitusi *byte* dengan menggunakan table substitusi (S-box)
  - *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
  - *MixColumns*: mengacak data di masing-masing kolom *array state*
  - *AddRoundKey:* melakukan *XOR* antara *state* sekarang *round key*.
- *Final round:* proses untuk putaran terakhir:
  - *SubBytes*
  - *ShiftRows*
  - *AddRoundKey*



Gambar 3. Skema operasi algoritma Rijndael (*AES*)

D. Cocoa Framework

Cocoa adalah sebuah *API* pada sistem operasi Macintosh. Sistem operasi Macintosh sendiri dibangun di atas kerangka kerja ini.

Cocoa terdiri dari dua *library* primer Objective-C :

1. Foundation Kit

Foundation kit (*Foundation*) didasarkan pada *Core Foundation* pada sistem operasi Macintosh. *Foundation* adalah sebuah *generic object-oriented library* yang menyediakan manipulasi terhadap *string* dan nilai-nilai (*integer, real*, dan lain-lain), *cointainers* dan *iteration, distributed computing, run loops*, dan fungsi-fungsi lainnya yang tidak terikat secara langsung kepada *GUI*. Pada *foundation* digunakan awalan "NS" pada setiap kelas (*class*) dan konstanta (*constant*) pada *framework.*

2. Application Kit

*Application Kit (AppKit)*diturunkan secara

langsung dari *NeXTSTEP Application Kit*. *AppKit* berisi kode yang dapat membuat dan berhubungan dengan *GUI*. *AppKit* dibangun di atas *Foundation* dan juga menggunakan awalan "NS".

Kunci dari arsitektur *Cocoa* adalah *views model* yang komprehensif. Hal ini diatur oleh *application framework*, namun berdasarkan pada model gambar PDF yang disediakan oleh *Quartz*. Hal ini memungkinkan pembuatan gambar kustomisasi menggunakan perintah yang menyerupai *PostScript* untuk menggambar, yang juga memungkinkan untuk mendukung pencetakan secara otomatis.

Karena *Cocoa framework* mengelola semua *clipping*, *scaling*, dan tugas-tugas lainnya untuk menggambar sehingga *programmer* dibebaskan pada pengimplementasian infrastruktur dasar dan dapat berkonsentrasi dalam pembuatan aspek-aspek unik dari sebuah aplikasi.

III. Analisis Fitur-fitur Keamanan Pada Sistem Operasi Macintosh

A. Invisibility File

Terdapat beberapa cara untuk membuat sebuah file agar tidak terlihat oleh pengguna pada Macintosh. Pengguna dapat menggunakan aplikasi tambahan seperti *HideOut (freeware)* atau *SecretFolder*.

Pengguna juga dapat menggunakan *Terminal* untuk membuat sebuah file tidak terlihat (*invisible*). Cara untuk melakukannya adalah dengan membuka *Terminal* mengarahkan ke direktori tempat file yang diinginkan untuk dibuat tidak terlihat dengan mengetikkan cd <nama direktori>. Setelah itu ketikkan

mv <namafile> .<namafilebaru>

Jika ingin membuat file abc.txt tidak terlihat maka ketikkan :

mv abc.txt .abc.txt

Maka file abc.txt akan menjadi tidak terlihat (*invisible*). Selanjutnya untuk melihat file *invisible* tersebut dapat digunakan perintah sebagai berikut :

defaults write com.apple.Finder Apple ShowAllFiles TRUE

Lalu ketik baris perintah berikut :

killall Finder

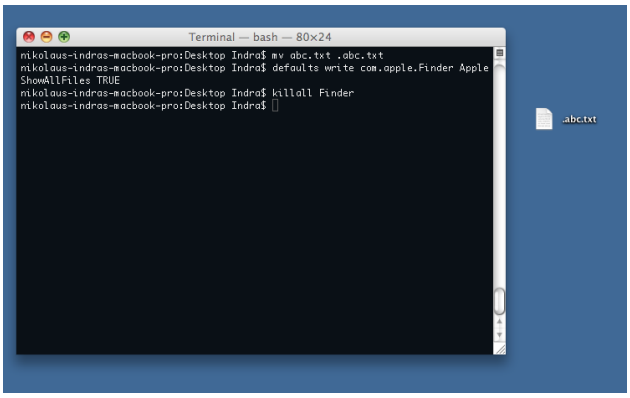
Maka file abc.txt yang sudah dibuat menjadi tidak terlihat akan nampak namun dengan keadaan masih tersembunyi (tidak terlihat).

Untuk membuat file kembali menjadi terlihat (*invisible*) adalah dengan mengetikkan baris perintah berikut :

mv .abc.txt abc.txt

Maka file abc.txt akan kembali terlihat (*invisible*).

Berikut contoh gambar penggunaan yang penulis lakukan.

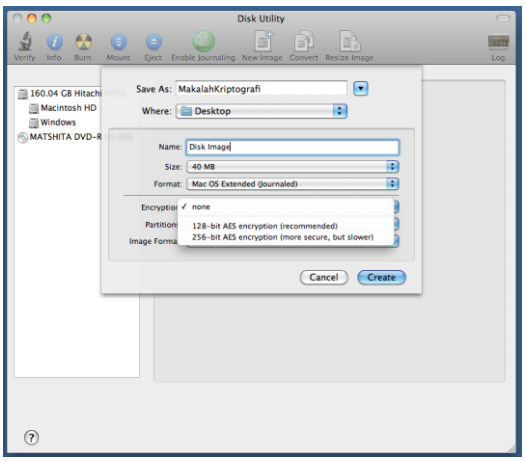


Gambar 4. Membuat File tidak terlihat

B. Passwords & Enkripsi menggunakan fitur aplikasi yang tersedia pada Macintosh

Seperti yang telah dijelaskan pada dasar teori, pada sistem operasi Macintosh terdapat satu buah fitur untuk mengelompokkan file / data-data dan disimpan ke dalam sebuah penampung (*Disk Image*) yang selanjutnya file / data-data yang tersimpan di dalamnya akan dienkripsi menggunakan algoritma *AES*. Terdapat dua pilihan algoritma AES yang dapat digunakan (AES-128 dan AES-256).

Untuk menggunakan *DiskUtility*, yang harus dilakukan pertama kali adalah menentukan besar ukuran (*size)* yang dapat ditampung oleh penampung tersebut. Terdapat beberapa pilihan ukuran yang telah disediakan oleh *DiskUtility*. Selanjutnya memilih algoritma enkripsi yang akan digunakan beserta kata kuncinya (*key*) untuk mengamankan file yang nantinya akan dimasukan ke dalam penampung tersebut. Setelah membuat penampung, maka hanya tinggal melakukan *drag and drop* file / data yang diinginkan untuk diamankan ke dalam penampung yang telah dibuat.



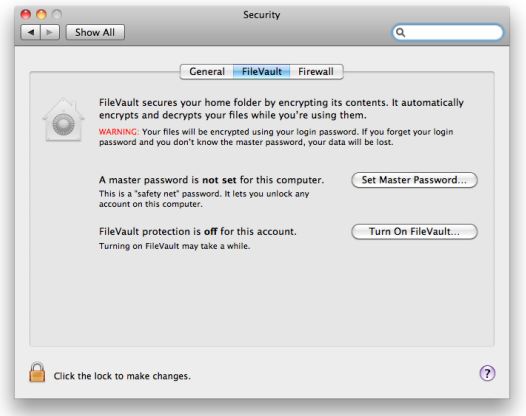
Gambar 5. DiskUtility

Fitur lainnya yang dimiliki sistem operasi untuk melakukan pengamanan data adalah *FileVault*. Fitur ini baru tersedia pada Mac OS X v.10.3 (*Panther*).

FileVault melakukan enkripsi terhadap seluruh sistem file pada laptop / komputer ketika pengguna melakukan *log out*. Direktori utama pada sistem operasi akan dienkripsi menggunakan algoritma *AES* dengan menggunakan kata kunci (*key*) yang digunakan pengguna untuk melakukan *log in*. Namun dapat juga digunakan kunci lainnya yang berperan sebagai kunci utama (*master key*).

Pada sistem operasi Macintosh versi terbaru (Mac OS X v.10.6), *FileVault* menyimpan hasil enkripsi sistem file ke dalam sebuah penampung yang disebut *sparse bundle*. Penampung ini membagi sistem file menjadi file-file yang berukuran lebih kecil (8MB) yang disebut dengan *bands*. Sehingga akan memudahkan pengguna untuk melakukan *backup* data menggunakan fitur *TimeMachine* apabila ingin melakukan *backup* data dengan tingkat keamanan yang cukup tinggi (menggunakan algoritma enkripsi AES).

Cara menggunakan *FileVault* sangat mudah, pengguna hanya perlu menyalakan *FileVault* dan membuat sebuah kunci untuk melakukan pengamanan sistem file pada laptop / komputer yang bersangkutan.



Gambar 6. FileVault

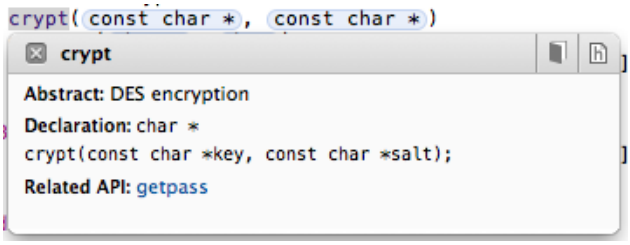
Kekurangan *FileVault* adalah menghabiskan kapasitas penyimpanan yang besar dan juga jika pengguna kehilangan kunci maka data akan terkunci selamanya.

C. Pembuatan Aplikasi Pengenkripsi teks dengan menggunakan *Cocoa Framework*

Penulis membuat aplikasi berbasis *Cocoa Framework*, dengen menggunakan bahasa pemrograman Objective-C dan menggunakan *IDE Xcode* v.3.2.3.

Algoritma enkripsi yang dipilih untuk diimplementasikan adalah algoritma DES karena fungsi *native* enkripsi yang tersedia adalah algoritma tersebut.

Tahapan yang dilakukan adalah dengan memanggil fungsi :



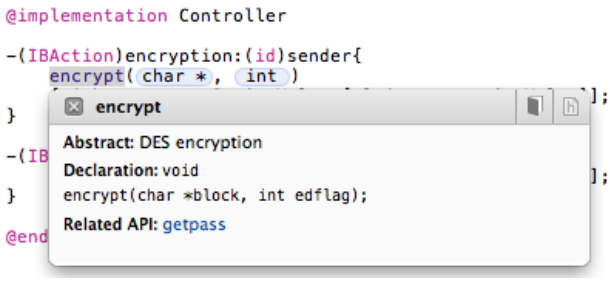
Gambar 7. Fungsi *crypt* pada IDE Xcode

Fungsi ***crypt(const char *key, const char *salt)*** melakukan enkripsi *password* berdasarkan *NBS Data Encryption Standard* (*DES*). Fungsi tersebut memunyai dua argumen. Argumen pertama adalah input sebuah kunci dengan tipe *string* (*char**) dan argumen kedua adalah dapat berarti dua : jika diawali dengan karakter *underscore* ('_') maka akan dilakukan *extended crypt* dan jika tidak maka akan dilakukan *traditional crypt*.

Penulis memilih untuk menggunakan *traditional crypt*. Maka proses yang terjadi adalah *key* (argumen pertama) akan dibagi menjadi kelompok-kelompok 8 karakter dan 7 bit *low-order* untuk setiap karakter akan digunakan untuk membentuk 56-bit *DES key*. Untuk argumen kedua, *salt* merupakan 2-*character array of ASCII*. Maka hanya 12 bit *salt* yang digunakan.

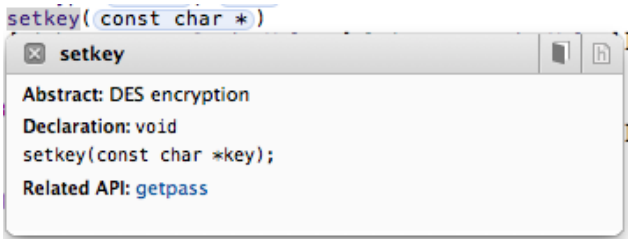
Dari kedua argumen di atas maka algoritma yang terjadi adalah *salt* akan mengacak bit dengan 4096 kemungkinan (12 bit *salt* yang digunakan). Kunci *DES* digunakan untuk mengenkripsi 64-bit *constant*, dengan menggunakan perhitungan iterasi perputaran dari *DES*. Nilai kembalian dari fungsi ini adalah sebuah string kosong, diikuti dengan enkripsi *encoded* 64-bit.

Setelah pemanggilan fungsi ***crypt(const char *key, const char *salt),*** dibutuhkan fungsi berikut untuk mengakses algoritma *DES* :



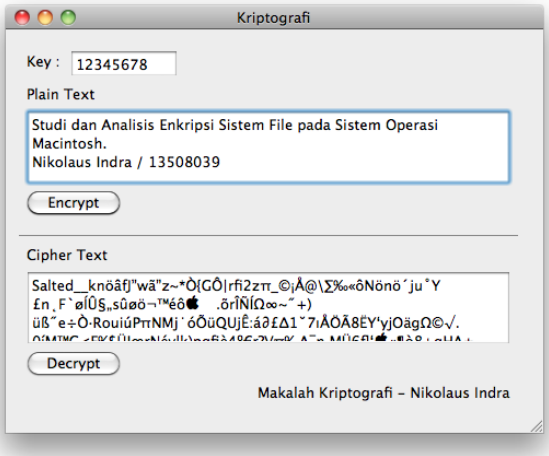
Gambar 8. Fungsi *encrypt* pada IDE Xcode

Argumen pertama merupakan 64-bit *array of bit*. Jika nilai *edflag* adalah 0 maka blok akan dienkripsi, jika tidak maka blok akan didekripsi. Hasil dari fungsi ini adalah *array* blok setelah melakukan pemanggilan fungsi *setkey()* untuk memprosesnya.



Gambar 9. Fungsi *setkey* pada IDE Xcode

Aplikasi yang penulis buat adalah sebagai berikut :



Gambar 10. Aplikasi sederhana algoritma enkripsi DES

D. Enkripsi file dengan menggunakan *Terminal*

Cara lain untuk menggunakan fungsi enkripsi DES yang disediakan oleh sistem operasi Macintosh adalah dengan menggunakan perintah langsung pada *Terminal*.

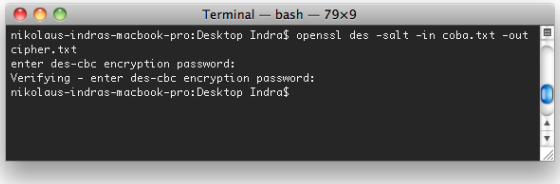
Algoritma yang digunakan untuk melakukan enkripsi file teks yang digunakan oleh penulis adalah *DES*.

Perintah yang digunakan pada *Terminal* (dengan sebelumnya mengarahkan ke direktori dimana tempat file yang diinginkan berada) adalah :

openssl des –salt –in coba.txt –out cipher.txt

Ketika menekan tombol enter maka *Terminal* akan meminta pengguna memasukan sebuah kata kunci dan lalu untuk memverifikasi kata kunci tersebut.

Jika telah berhasil maka file coba.txt akan dienkripsi dengan menggunakan algoritma *DES* menjadi file bernama cipher.txt.

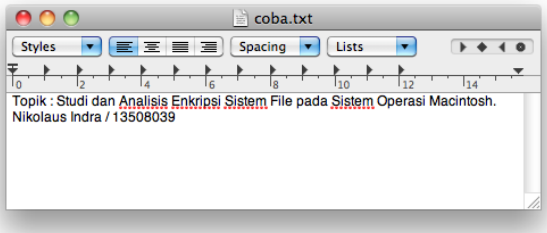


Gambar 11. Enkripsi file menggunakan algoritma *DES* melalui *Terminal*

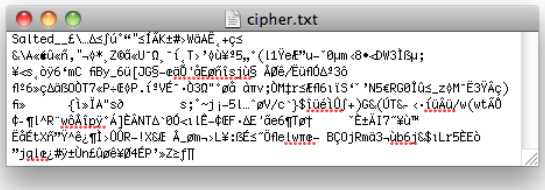
Untuk mendekripsi file cipher.txt maka gunakan baris perintah sebagai berikut :

openssl des –d –salt –in cipher.txt –out coba.txt

Masukan kata sandi yang digunakan sebelumnya untuk melakukan enkripsi file maka cipher.txt akan kembali menjadi file coba.txt.



Gambar 12. File coba.txt



Gambar 13. File cipher.txt

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Seiring dengan berkembangnya dunia teknologi informasi, maka semakin banyak orang-orang yang

memilki laptop / komputer personal dan juga semakin banyak tindak kejahatan yang dapat dilakukan seperti pencurian data pribadi.

Oleh karena itu sistem operasi Macintosh menyediakan berbagai macam fitur yang dapat digunakan untuk melakukan pengamanan data seperti membuat file tidak terlihat (*invisible*), melakukan enkripsi pada sekumpulan data/file dengan menggunakan *DiskUtility* dan bahkan melakukan enkripsi terhadap seluruh sistem file menggunakan fitur *FileVault*.

Juga tersedia fitur untuk membuat aplikasi enkripsi dan dekripsi sendiri dengan menggunakan *API* yang tersedia pada *Cocoa Framework*.

B. Saran

Masih banyak fungsi *native* untuk mengimplementasi algoritma enkripsi dan dekripsi pada pemrograman berbasis *Cocoa Framework*. Dengan melakukan *import library openssl* maka akan tersedia fungsi algoritma enkripsi *AES*, *blowfish*, *SHA-1*, *MD5*, dan lain-lain Dengan demikian tingkat keamanan suatu file akan semakin tinggi apabila programmer mencoba mengkombinasikan algoritma-algoritma tersebut untuk membuat sebuah aplikasi pengenkripsian file.

REFERENCES

[1] http://developer.apple.com
(Waktu akses : 18 Maret 2011)

[2] http://en.wikipedia.org/wiki/Cocoa
(Waktu akses : 18 Maret 2011)

[3] http://en.wikipedia.org/wiki/FileVault
(Waktu akses : 18 Maret 2011)

[4] http://aplawrence.com/Unixart/file_encryption.html
(Waktu akses : 18 Maret 2011)

[5] http://macclubindonesia.com
(Waktu akses : 19 Maret 2011)

[6] http://www.informatika.org/~rinaldi
(Waktu akses : 16 Maret 2011)

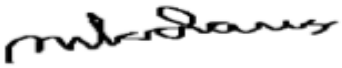
[7] http://macosx.com/forums/archive/t-42367.html
(Waktu akses : 19 Maret 2011)

[8] http://www.ehow.com/how_2274100_hidden-files-mac-os-x.html
(Waktu akses : 18 Maret 2011)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21 Maret 2011



Nikolaus Indra / 13508039