PROTEKSI DATA DENGAN MENGGUNAKAN KOMBINASI SERIALI

by Santi Agustina

Submission date: 27-Oct-2021 04:51AM (UTC-0400)
Submission ID: 1685456136
File name: PROTEKSI_DATA_DENGAN_MENGGUNAKAN_KOMBINASI_SERIALI.pdf (3.36M)
Word count: 5031
Character count: 29645

PROTEKSI DATA DENGAN MENGGUNAKAN KOMBINASI SERIALISASI XML

Oleh: Rudianto, Siti Maimunah, Nur Endah Layutumi
Fakultas Teknologi Informasi
Institusi Teknologi Adhi Tama Surabaya

ABSTRACT

In this era of age information, the exchange of the data become the matter was very important. The exchange of the data could be also compared with the trade transaction, where inside had plenty of information that was secret between the sender in the recipient. Definitely very dangerous if this data fell to the hands of the wrong person or was not responsible. Then was needed enkripsi to maintain this data secrecy. Spoke concerning enkripsi definitely will not be free from standard that was used in enkripsi the data. At this time often sprang up algorithms enkripsi tgat was popular enough. Mentioned Des that was very famous until the newest Rijndael algorithm (the AES standard) that was discussed tho the thesis. This algorithm determined the strength in protecting secrecy of a data. And to stress the srength enkripsi generally was united with the certain method. One of the methods was united algorithm Rihndael with serialisasi XML and Flash the Disk to get enkripsi that was very strong and safe that was almost impossible to be taken apart. Here the writer also stressed that to make these methods not so hard that was imagined. We were only prosecuted to innovative in getting the method that was strong and safe by using the available algorithm.

Keywords : *Encryption, Flash Disk, Rijndael Algorthm, Serialization, XML*

1 LATAR BELAKANG

Keamanan data dalam suatu perusahaan merupakan salah satu hal yang penting dilakukan. Terutama bagi perusahaan yan mengasusmsikan bahwa data tersebut adalah aset penting yang harus dijaga kerahasiaanya. Banyak cara untuk melindungi kerahasiaan data tersebut. Mulai dari penggunaan *Password* hingga enkripsi terhadap data tersebut.

Tidak heran apabila sekarang ini banyak bermunculan "program khusus proteksi data yang memproteksi data dengan beberapa metode seperti DES ataupun metode *Proprietary* yang lebih cepat, yang sebenarnya kurang aman" (Sukamawan 1998, 1). *Proprietary* adalah metode yang dibuat oleh suatu organisasi atau perusahaan yang kemudian dipublikasikan ke umum dan bersifat komersil. Karena itu banyak usaha untuk mencari kelemahan dari program proteksi tersebut. Hal ini menyebabkan banyaknya program - program untuk membongkar *password* atau enkripsi.

Beberapa program tersebut ada yang secara otomatis membongkar proteksi program suatu data dengan sangat mudah. Bahkan ada program yang menambah *delay loop* sehingga seolah - olah program tersebut sedang bekerja keras membongkar *password, contohnya pada salah satu perusahaan yaitu Access Data. Mereka membuat software* yang dapat membongkar WordPefrect (versi 4.2-6.1, 'regular' atau 'enhanced', Microsoft Word (versi 2.0-6.1). Microsoft Excel (semua versi termasuk versi Macintosh), dll (Sukmawan 1998, 1).

Alasan utama kurang baiknya proteksi dari program - program di atas adalah mungkin untuk mendapatkan izin ekspor dari pemerintah Amerika Serikat tidaklah mudah, karena disana untuk mengekspor program enkripsi yang kuat memerlukan izin yang ketat dari pemerintah. Dan mengekspor program enkripsi sama dengan mengekspor amunisi sehingga sangat dibatasi bahkan dikenai hukuman bagi yang melanggarnya. Contoh klasik adalah apa yang menimpa Philip Zimmermann yang diadili karena program PGP yang ia buat dan menyebar ke seluruh dunia. (Sukmawan 1998,1).

"Untuk proteksi data yang cukup penting tidak ada jalan lain selain menggunakan program khusus proteksi/enkripsi data" (Sukmawan, 1998:1) dengan menggabungkan teknik, metode algoritma yang telah ada. Salah satu contoh adalah Proteksi data dengan Flash Disk menggunakan kombinasi serialisasi XML dan Algoritma Rijndael.

2 PERUMUSAN MASALAH

Berdasarkan latar belakang diatas,didapatkan rumusan permasalahan sebagai berikut :

1. Bagaimana merancang program enkripsi dengan Flash Disk menggunakan kombinasi serialisasi XML dan Algoritma Rijndael.
2. Bagaimana membangun suatu aplikasi proteksi data, sehingga data tersebut tidak dengan mudah diakses oleh orang lain yang tidak berkepentingan.

3 TUJUAN DAN MANFAAT

Tujuan dan manfaat dari penelitian ini adalah :

1. Merancang dan membuat program utilitas yang mempunyai kemampuan memproteksi suatu data dengan menggunakan *serial number* pada Flash Disk.
2. Melindungi data proses pertukaran data sehingga data yang akan ditukarkan akan tetap aman dan tidak bisa diakses oleh orang yang tidak berkepentingan.

4 BATASAN MASALAH

Batasan masalah dari pembuatan sistem ini adalah sebagai berikut :

1. Proteksi data dengan Flash Disk menggunakan kombinasi Serialisasi XML dan Algoritma Rijndael.
2. Proteksi data hanya pada lingkungan sistem operasi Microsofr Windows Xp/2000 Server dan Profesional dan Microsoft Windows 2003 Server.
3. Proteksi data hanya pada ruang lingkup pemakaian pada komputer personal tanpa ada komponen jaringan komputer di dalamnya.
4. Enkripsi dan dekripsi menggunakan *class* Rijndael Managed yang ada pada Visual Studio. NET dengan panjang kunci 128 bit.
5. Proses Enkripsi dan Dekripsi dilakukan di dokumen word, gambar (jpeg), *music* (mp3), dokumen PDF.

5 METODOLOGI BATASAN MASALAH

Untuk memperoleh hasil yang bagus dan bermanfaat bagi para pembaca dan penggunanya maka ada beberapa metode yang peneliti gunakan dalam penyusunan skripsi ini , yaitu sebagai berikut :

1. Study literatur

   Disini akan dipelajari beberapa literatur yang berhubungan dalam penulisan skripsi, diantaranya adalah referensi yang berkaitan mengenai Algoritma Rinjdael, Serialiasai XML dan Flash Disk sehingga memudahkan dalam melengkapi data serta memecahkan masalah.

2. Analisa

Pada tahap analisa ini akan dicoba untuk mencari beberapa metode dan akan dijelaskan kelebihan dan kelemahan dari metode tersebut. Metode tersebut antara lain :

a. BFA 97 (Blofish Advanced 97)

b. Kremlin

Adapun hasil dari penjelasan metode tersebut adalah untuk mencari kekurangan dari beberapa metode di atas sehingga dari kekurangan tersebut dapat diambil suatu metode atau cara baru yang lebih aman dalam proteksi data. Untuk teknik yang digunakan akan dijelaskan lebih rinci pada Bab 3 (Analisa dan Perancangan Sistem).

3. Perancangan Sistem

Merupakan desain dan implementasi mengenai perancangan proses programm proteksi data meliputi USB flash disk, Serial XML dan algoritma Rijndael dalam bentuk *flowchart* dan penjelasan sistem.

a. Perancangan Antar Muka

User interface dan *user friendly* dirancang untuk memudahkan bagi pengguna dalam menggunakan aplikasi dan mendapat informasi yang dibutuhkan dari aplikasi tersebut.

b. Implementasi Sistem

Mengimplementasikan hasil rancangan sistem yang telah dibuat sebelumnya untuk menjadi sebuah perangkat lunak dengan menggunakan Bahasa pemrograman Visual Studio, Net versi 2003.

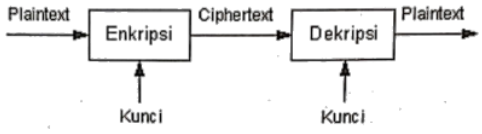
6 LANDASAN TEORI TENTANG PERMSALAHAN

6.1 Kriptosistem

"Kriptosistem adalah algoritma kriptografi, ditambah seluruh kemungkinan *plaintext*, *ciphertext* dan kunci -kuncinya" (Kurniawan 2004, 2)."Kriptografi merupakan ilmu yang mempelajari tentang penyembunyian huruf atau tulisan sehingga membuat tulisan tersebut tidak dapat dibaca oleh orang yang tidak berkepentingan" (Sofwan 2006, 23). "Kriptoanalis adalah orang yang berusaha merusak suatu metode enkripsi untuk mendapatkan teks yang asli secara ilegal" (Kurniawan, 2004 1). Seorang kriptoanalis selalu berusaha untuk memecahkan metode ernkripsi dari suatu teks dengan berbagai kemungkinan. Salah satu cara paling

populer dalam memecahkan metode enkripsi yaitu dengan menggunakan *Brute Force Attack* yang menggunakan segala kemungkinan untuk memecahkan kode enkripsi.

"Suatu pesan atau informasi yang dapat dibaca disebut *plaintext* sedangkan pesan yang tidak dapat dibaca yaitu *ciphertext"* (kurniawan 2004,1). Proses yang dilakukan untuk mengubah *plaintext* ke dalam *ciphertext* kembali ke *plaintext* disebut enkripsi atau *encipherment. "*Sedangkan proses utnuk mengubah *ciphertext* kembali ke *plaintext* disebut dekripsi atau *decipherment.* secara sederhana istilah - istilah di atas dapat digambarkan seperti Gambar 4.1



Gambar 4.1 Proses Enkripsi /Dekripsi Sederhana

Seperti telah disebut diatas, bentuk asli dari suatu dokumen yang akan dienkripsi disebut dengan *plaintext,* sedangkan dokumen hasil enkripsi lazim disebut *ciphertext.* Beberapa enkripsi menggunakan kunci, sehingga *chiphertext* yang di hasilkan bergantung pada nilai kunci. Kunci yanga ada bisa berupa sebuah kata ataupun rangkaian karakter acak yang dihasilkan oleh komputer.

6.2 Key

Dalam kriptografi, terdapat dua macam *key* atau kunci yang digunakan dalam proses enkripsi dan dekripsi. Dua macam *key* tersebut adalah :

1. *Public Key*

Diungkapkan pertama kali oleh Diffie dan Heellman pada tahun 1976, kunci jenis ini banyak digunakan dalam enkripsi data yang berhubungan dengan jaringan komputer. Kunci yang ada dengan sengaja disebarluaskan kapada publik secara bebas. Sehingga pada saat proses pengiriman data yang telah terenkrip, pihak pengirim dan penerima menggunakan *public key* sendiri yang nantinya dikombinasikan dengan *public key* yang telah ada.

2. *Single Key*

Seringkali disebut sebagai algoritma konvensional dalam proses enkripsi. Pada jenis ini kunci yang dibutuhkan harus tetap terjaga kerahasiaanya, dalam artian bahwa hanya pemilik kunci yang bisa menjalankan proses enkripsi dan dekripsi.

7 LANDASAN TEORI TENTANG ILMU TERKAIT

7.1 Advanced Encryption Standard

"DES terbukti menjadi algoritma enkripsi yang aman di dunia selama puluhan tahun. Namun panjang kunci DES yang hanya 56 bit dianggap terlalu pendek" (Kurniawan 2004, 51). Untuk menjawab tantangan dalam dunia kriptografi, National Institute of Standard and Technology (NIST) bersip -siap mengganti DES.NIST mengadakan kontes terbuka yang dapat diikuti oleh berbagai peserta dari seluruh dunia agar ikut berpartisipasi dalam algoritma standard pengganti DES. Januari 1997, NIST mengumumkan algoritma baru dalam kriptografi modern sebagai pengganti algoritma DES yang disebut sebagai AES sendiri dilombakan secara terbuka dan pada tanggal 2 Oktober 2000, NIST secara resmi menetapkan algoritma Rijndael sebagai pemenangnya. Algoritma Rijndael merupakan sebuah algoritma enkripsi berjenis block cipher yang diciptakan oleh dua orang ilmuwan Belgia bernama Joan Daemen dan Vincent Rijmen yang merupakan sebuah algoritma dengan panjang blok dan panjang kunci bervariasi antara 128 bit hingga 256 bit.

7.2 Algoritma Rijndael

Algoritma Rijndael merupakan sebuah algoritma enkripsi berjenis *block cipher* yang diciptakan oleh dua orang ilmuwan Belgia berna Joan Daemen dan Vincent Rijmen yang merupakan sebuah algoritma dengan panjang blok dan panjang kunci bervariasi antara 128 bit hingga 256 bit.

"Sebenarnya Algoritma Rijndael hanya memiliki tinggkat keamanan yang tidak tinggi. Namun karaena kesedarhanaan desain Rihndael sehingga memiliki fleksibilitas dan kecepatan yang lebih tinggi dibanding dengan yang sangat aman namun lambat sekali untuk digunakan" (Kurniawan 2004,59)

1. Tidak seperti AES, Rijndael memiliki ukuran *block cipher* dan panjang kunci yang bervariasi antara 128, 192 bit hingga 256 bit.
2. Garis besar Algortima Rijndael yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut (di luar proses pembangkitan *round key*):
   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. *SubBytes:* subtitusi *byte* dengan menggunakan tabel subtitusi (S-box).
b. *ShiftRows:* pergeseran baris - baris *array state* secara *wrapping.*
c. *MixColumns:* mengacak data masing - masing kolom *array state.*
d. *AddRoundKey:* transformasi ini melakukan operasi XOR terhadap sebuah *round key* dengan *array state,* dan hasilnya disimpan di *array state.*

3. Final round: proses untuk putaran terakhir :
   a. *SubBytes*
   b. *ShiftRows*
   c. *AddRounKey*

3. Putaran terakhir dilakukan sebanyak 10 kali untuk yang 128 bit, 12 kali untuk yang 192 bit dan 14 kali untuk yang 256 bit. Pada putaran terakhir tidak melakukan Mixcolumns, dalam arti pada putaran terakhir dari langkah Shiftrows langsung ke AddRounkey.

7.3 Serialisasi XML

Sebuah XML atau dalam bahasa umumnya XML Serializer merupakan sebuah class dalam Visual Basic. NET dari namespace XML.Runtime. *Serialization* yang berfungsi untuk melakukan serialisasi sebuah obyek ke dalam sebuah file XML dan sebaliknya. Serialisasi obyek secara umum akan memanfaatkan kemampuan Visual Basic.NET untuk membaca *memory stream* dalam format tipe *byte.* Sehingga obyek yang dapat diserialisasi lebih bervariasi, misal: dokumen image, dokumen PDF hingga *runtime library*.

7.4 Algoritma BASE 64

Base64 adalah algoritma pemetaan (*encoding*) yang memungkinkan data binari dapat ditransfer melalui media yang hanya mendukung ASCII, karakter yang dapat dibaca oleh manusia terdiri dari ‘A’ sampi ‘Z’, ‘a’ sampai ‘z’, ‘0’ sampai ‘9’ , ‘+’ sampai ‘/’.

Base64 digunakan agar dapat diformat ke dalam bentuk dokumen XML. Ini karena dokumen XML hanya mau menerima ASCII karakter. Karena itu hampir tidak ada batasan file yang dapat diformat dalam bentuk dokumen XML. File dengan ekstensi Microsoft Word, file dengan format gambar bahkan hingga file kompres pun dapat diformat ke dalam bentuk XML.

7.5 Algoritma SHA 384

NIST bersama NSA mendesain *secure has algoritm* (SHA) untuk digunakan sebagai salah satu komponen *message digest*. Hingga sekarang NIST telah mengeluarkan tiga variasi SHA yaitu SHA-0, SHA-1 dan terakhir adalah SHA-2. SHA-0 dirilis pada tahun 1993. Tidak lama kemudian, NIST melakukan beberapa perbaikan pada SHA-0 dan kemudian dinamakan SHA-1. Namun banyak sekali terjadi serangan - serangan yang ditujukan ke SHA-1. Pada awal tahun 2005, Rijmen berhasil menembus algoritma yang digunakan oleh SHA-1 meskipun belum sampai mendapatkan pesan aslinya. Namun demikian, pada tahun 2006 Christian Rechberger dan Christopher De Canniere menyatakan bahwa mereka telah menemukan serangkaian serangan - serangan yang memungkinkan kriptoanalisis untuk mengambil beberapa pesan dari *message digest*. Karena alasan tersebut, NIST mengumumkan varian baru SHA yaitu SHA-2 dengan panjanga bit yang berbeda yaitu SHA-256, SHA-384, SHA-512. Hingga sekarang belum ada pengumuman resmi tentang adanya serangan yang berhasil sebagaimana terjadi pada SHA-1.

7.6 Flash Disk

USB merupakan suatu teknologi yang memungkinkan kita untuk menghubungkan alat eksternal (peripheral) seperti scanner, printer, mouse, papan ketik (*keyboard*), alat penyimpanan data (*zip driver*), flash disk, kamera digital atau perangkat lainnya ke komputer kita. USB sangat mendukung transfer data sebesar 12 Mbps (juta bit per detik). Komputer (PC) saat ini, umumnya sudah memiliki port USB. Biasanya disediakan minimal 2 port. Jika dibandingkan dengan paralel port dan serial port, penggunaan port USB lebih mudah.

8 PERANCANGAN SISTEM

Merupakan desain implementasi mengenai perancangan program proteksi data, yang meliputi perancangan *flowchart* dan *pseudocode*.

8.1 Serial number USB Disk

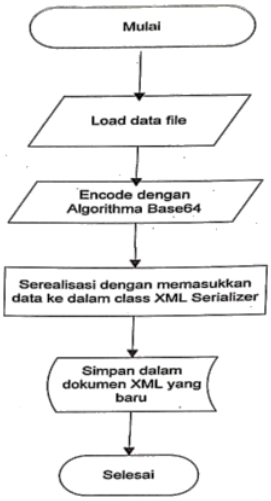
Setiap USB disk pasti memiliki *serial number* yang bersifat unik, artinya *serial number* tersebut pasti berbeda antara yang satu dengan lain bahkan untuk produsen yang sama atau merek yang sama. *Serial number* dari USB disk sendiri berguna apabila *user* memasang peralatan berbasis USB lebih dari satu ke komputer. Komputer menggunakan *serial number* agar tetap dapat menjaga proses *request* dan

response antara masing - masing peralatan USB dengan sistem operasi bahkan setelah *reebot*. *Serial number* juga memungkinkan komputer untuk menentukan peralatan USB mana yang telah digunakan atau baru (pertama kali) digunakan meski dari produsen yang sama. Dan apabila peralatan USB tersebut dipasang pada port lainnya pada komputer yang sama maka sistem operasi tidak perlu memanggil ulang *driver* dari peralatan USB tersebut. *Serial number* pada USB disk berubah saat kita melakukan format ulang pada USB disk itu sendiri.

8.2 Proses serialisasi

Proses serialisasi terhadap data dimulai dari pemanggilan data ke dalam blok memori. Serialisasi terhadap data tersebut *output*-nya akan dijadikan format XML, namun sebelumnya akan dimasukkan ke dalam sebuah class tersendiri yang memiliki properti pengaturan agar data saat dideserialiasai mampu kembali ke bentuk semula dengan sempurna. Perlu diperhatikan bahwa dalam melakukan serialisasi terhadap data berupa file terlebih dahulu data tersebut di konversi ke dalam bentuk *byte*. Ini karena file tersusun dari kode - kode binary yang tidak secara langsung diurai menjadi file XML. Untuk melakukan konversi tersebut digunakan algoritma Base64 yang memiliki fungsi untuk mengubah format *binary* ke bentuk *byte (array)*. Dan karena serialisasi XML hanya mengizinkan pembacaan data berua ASCII maka data yang berbentuk *byte* tersebut dikonversi sekali lagi ke dalam bentuk *string*.

Agar lebih jelas, Gambar 8.1 adalah alur program yang ditulis dalam bentuk *flowchart* dan *pseudocode*.



Gambar 8.1 Alur Serialisasi serial number USB disk

Adapun *pseudocode* dari *flowchart* di atas adalah sebagai berikut :

Start;

Step 1 [Load data file] Set strFilename = Filename;

Step 2 [Konversi ke byte dengan algoritma Base64] Set byFilenam() = base(64)strFilename;

Step 3 [Konversi ke base64String] set byFilename = base64ToString;

Step 4 [Serialisasi terhadap data yang dimasukkan ke dalam sebuah class] XMLSerializer(byFilename).

Stop.

Dari hasil *flowchart* dan *pseudocode* di atas, akan dibuat sebuah contoh yang menggambarkan dari proses di atas.

8.3 Enkripsi dokumen XML

Hasil dri proses serialisasi yang berupa dokumen XML selanjutkan akan dienkripsikan dengan mengguunakan algoritma Rijndael. Dokumen XML dalam proses enkripsi pada dasarnya dipandang sebagai sebuah dokumen teks normal atau *plaintext* yang tidak memiliki header file tertentu seperti halnya dokumen berjenis executable ataupun dokumen yang berasal dari sebuah database.

Dalam enkripsi yang dilakukan nantinya akan langsung menggunkan class Rijndael yang telah disediakan oleh Visual Basic.NET. Class Rijndael tersebut merupakan turunann dari namespace Cryptograpy dengan struktur sebagai berikut:

```
System.Object
    System.Security.Cryptography.SymmetricAlgorithm
        System.Security.Cryptography.Rijndael
```

Dalam *class* tersebut, *key* untuk enkripsi dengan algoritma Rijndael diambil dari *serial number* USB. *Key* yang terbentuk akan mengikuti salah satu standar keamanan yang diizinkan oleh Rijndael, yaitu 128 bit atau 16 *byte*. Karena itu *serial number* akan diacak terlebih dahulu dengan menggunakan algoritma SHA 384 dan 128 bit pertama hasil dari proses tersebut akan menjadi *key* . Dengan demikian *serial number* yang akan dijadikan *key* akan selalu memiliki panjang kunci 128 bit meskipun panjang rata - rata *serial number* adalah 12 karakter atau 96 bit. Ini akan menghasilkan keamanan data yang lebih kuat.

Proses enkripsi tersebut tidak hanya mengenkripsi elemen dokumen XML tetapi akan langsung mengenkripsi dokumen secara keseluruhan termasuk header dan atribut yang ada dalam XML, sehingga pihak yang tidak bertanggung jawab atau pihak yang mencoba memecahkan enkripsi akan lebih sulit menembus keamanan data.

Untuk mencegah hasil enkripsi sama dari dua data yang sama maka digunakan mode operasi *chiper block chaining* (CBC). Panjang *key* untuk mode operasi CBC yang disebut *intialization* Vektor atau IV harus 16 *bytes*. Apabila *key* diambil dari 128 bit pertama pada proses hash *serial number* dengan algoritma SHA 384, maka IV diambil dari 128 bit berikutnya. Dengan demikian IV yang dihasilkan akan berbeda - beda. Dengan cara ini dua buah data yang sama persis sekalipun akan menghasilkan file enkrip yang berbeda. Konsekuensi apabila tidak menggunakan IV tentunya pada dua data yang sama persis akan menghasilkan file enkripsi yang sama pula.

Setelah proses enkripsi berhasil dilakukan makan program akan mengahapus file asli. Ini dilakukan agar keamanan yang diperoleh lebih maksimal. Untuk itu sebelum melakukan enkripsi data program akan menganjurkan agar data back up terlebih dahulu. Agar lebih jelas, Gambar 8.2 adalah alur program yang ditulis dalam bentuk *flowchart* dan *pseudocode* .

Gambar 8.2 Alur enkripsi data dengan algoritma Rijndael

Adapaun *pseudocode* dari *flowchart* diatas adalah sebagai berikut :

Start ;

Step 1	[Panggil data] Set strFilename = Filename
Step 2	[Serialisasi terhadap data dengan XML Serializer] XMLSerializer(strFilename).
Step 3	[Ambil serial number dari USB Disk] strSerial = Serialnumber
Step 4	[Lakukan enkripsi terhadap serial number dengan algoritma SHA384 dan ambil 128 bit pertama sebagai key] Set strKey SHA384(128) = (SHA384)strSerial.
Step 5	[Ambil 128 bit selanjutnya sebagai IV] Set strVSHA384(128) = (SHA384)strSerial.
Step 6	[Enkripsi data] Do encrypt.
Step 7	[Apakah data original dihapus?] if data deleted then delete.
Step 8	[Terbentuk data baru hasil dari enkripsi dokumen XML] new XML document.

End.

Dari hasil *flowchart* dan *pseudocode* diatas, akan dibuat sebuah contoh yang menggambarkan proses diatas.

```
<?xml version="1.0" encoding=utf-8"?>
<string>SGVsbG8gAA==</STRING>
```

Dokumen XML diatas, dilanjutkan dengan enkripsi terhadap dokumen XML tersebut. Kunci diambil dari *serial number* USB Disk yang terpasang pada komputer, misal *serial number* tersebut "12345678901234". Agar kunci tersebut tidak dapat dibaca keasliannya maka dienkripsi dengan menggunakan algoritma hash SHA384. Dari hasil proses tersebut ambil 128 bit pertama sebagai kunci untuk enkripsi dan 128 bit selanjutnya untuk IV. Kunci untuk enkripsi tersebut menjadi

```
"omlqzHsuxSqcjx6xPS+nj5VOXJ8s/SsdSStxSQzKPwWT
L2AFIQfmfgMlSXXjRwle".
```

Kemudian dokumen XML tersebut di enkripsi dengan menggunakan algoritma Rjndael. Data dengan nama C:\Endah dengan isi "Hello" setelah diserialisasi dengan *class* XML Serializer dan kemudian dienkripsi dengan algoritma Rijndael, menjadi;

⌈Ùì,"Ì€ÊJ$m¶ñaú3ÝnÔ–è¹:ç,ÇÛ▯▯QÉ>Š▯bÁ ÐÀÅt“®^¯
®vmv/OÔ▯ß▯^z▯▯ŒÝ«ß:▯?í¤¨†¡▯”▯L▯Ó“¢V

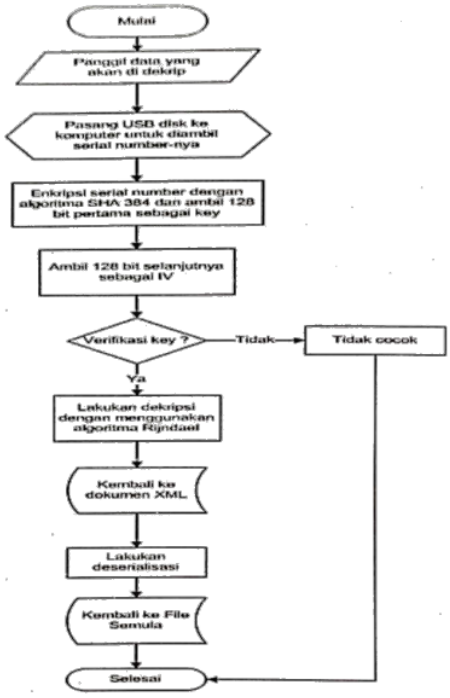
8.4 Proses Dekripsi

Untuk mengembalikan data yang telah terenkripsi maka harus melalui proses yang dinamakan dekripsi. Proses dekripsi sendiri secara umum merupakan kebalikan dengan proses enkripsi. Algoritma yang digunakan termasuk jenis algoritma simetris dan menggunakan *key* yang sama dengan saat proses enkripsi.

Saat pengguna melakukan proses dekripsi, maka akan diverifikasi terlebih dahulu key yang digunakan. *Key* ini tentunya diambil dari *serial number* USB disk yang dipasang pada komputer fimana langkah - langkahnya sama dengan saat proses enkripsi. Sama seperti pada saat enkripsi, *serial number* yang diambil juga dienkripsi telebih dahulu dengan menggunakan algoritma SHA 384 untuk dijadikan *key.* Langkah - langkahnya pun sama dengan saat proses enkripsi data . Hasil dari proses enkripsi *serial number* inilah yang akan dicocokan dengan *key* saat melakukan enkripsi data. Apabila *key* yang dipakai pada saat proses dekirpsi data tidak sama dengan saar proses enkripsi data maka dipastikan tidak akan berjalan lancar. Proses dekirpsi untuk algoritma Rijndael juga menggunakan *class* yang sama dengan saat proses enkripsi dilakukan.

Setelah proses dekrip telah selesai dan data berhasil didekrip maka data kembali kebentuk dokumen XML terlebih dahulu. Data dalam bentuk dokumen XML tersebut tetap belum bisa digunakan sebagaimana apabila data itu ada dalam bentuk aslinya.

Agar lebih jelas, Gambar 8.3 adalah alur program ditulis dalam bentuk *flowchart* dan *pseudocode.*



Gambar 8.3 Alur Dekripsi dengan Algoritma Rijndael

Adapun *pseudocode* dari *flowchart* pada gambar 8.3 adalah sebagai berikut:

Start ;

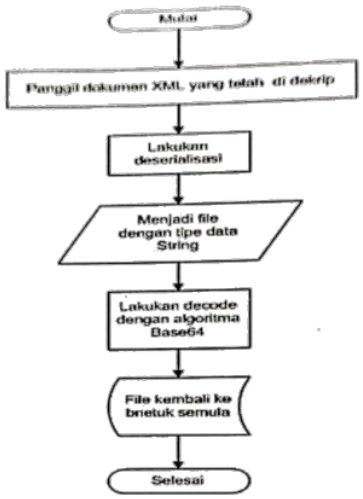
Step 1	[Panggil data yang akan didekrip] Call data.
Step 2	[Pasang USB disk] Plug USB disk.
Step 3	[Lakukan enkripsi *serial number* dengan algoritma SHA384 dan ambil 128 bit pertama sebagai *key*] Set srtKey SHA384(128) = (SHA384)strSerial.
Step 4	[Ambil 128 bit selanjutnya sebagai IV] Set strIVSHA384(128) =(SHA384)strSerial.
Step 5	[Verifikasi Key] if strKeySHA384 is match then Step 6 else STOP.
Step 6	[Lakukan dekripsi dengan menggunakan algoritma Rijndael] Do decrypt.
Step 7	[Kembali ke file XML] XML document.
Step 8	[Lakukan deserialisasi] (Deserializer) XML document.
Step 9	[Kembali ke data semula] Original data.

Stop.

8.5 Proses Deserialisasi

Apabila proses dekrip berjalan dengan sukses maka data akan kembali ke bentuk dokumen XML. Agar data dapat kembali ke bentuk semula diperlukan proses deserialisasi, yang juga kebalikan dri proses serialisasi. Setelah proses deserialisasi berhasil dilakukan, data tidak kembali ke dalam bentuk semula melainkan ke bentuk *byte* yang akan di-*decode* menggunakan algoritma Base64. Proses ini akan mengembalikan data ke bentuk semula.

Agar lebih jelas, di bawah ini adalah alur program yang ditulis dalam bentuk *flowchart* dan *pseudocode.*



Gambar 8.4 Alur Deserialisasi Dokumen XML

Adapun *pseudocode* dari *flowchart* pada Gambar 8.4 adalah sebagai berikut:

Start ;

Step 1 [Panggil dokumen XML yang telah di dekrip] Load XML document.

Step 2 [Lakukan deserialisasi] (Deserializer0 XML document.

Step 3 [Menjadi data dengan tipe string] String data.

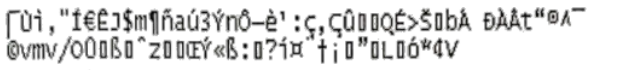
Step 4 [Lakukan decode dengan algoritma base64] (Base64)Data.

Stop.

Dari hasil *flowchart* dekripsi dan *flowchart* deserealisasi serta *pseudocode* diatas, akan dibuat sebuah contoh yang menggambarkan dari proses diatas;

```
<?xml version=1.0" encoding="utf-8"?>
<string>SGVsbG8gAA==</string>
```

Dari data yang terenkripsi diatas maka saat di dekrip akan kembali dokumen XML terlebih dahulu.



Dari dokumen XML tersebut dideserialisasi ke data orisinil, melalui proses decode dengan algoritma base64 terlebih dahulu, prosesnya adalah sebagai berikut SGVsbG8gAA kita buat menjadi nilai ASCII yaitu S(18), G (8), V(21), s(44), b(27), G(6), 8 (60), g (32), dari nilai ASCII tersebut dirubah menjadi *file binary* yaitu 01001000110010101 101100011011 000110 111100 100000, dari proses *file binary* , lalu kelompokkan menjadi 8 bit, yaitu 01001000 (48), 01100101 (65), 01101100 (6C), 01101100 (6C), 0110111 (6F). Setelah mengelompokan menjadi 8 bit, setelah itu akan dicocokan dengan tabel Algorithma Base 64, dan hasilnya sebagai berikut 48 (H), 65 (e), 6C(I), 6C(I), 6F (o). Proses ini dilakukan dalam satu langkah, sama seperti pada saat proses encode. Data kembali ke bentuk asalnya dengan isi data “Helo”.

9 IMPLEMENTASI SISTEM

Sebelum menjalankan program ini ada hal yang harus diperhatikan yaitu kebutuhan sistem. Tujuan pokok dari sistem komputer adalah mengolah data dari *input* kemudian diolah menjadi *output.* Dalam melaksanakan tujuan pokok tersebut diperlukan adanya elemen - elemen yang mendukung. Elemen - elemen dari sistem tersebut antara lain adalah *hardware* (perangkat keras komputer). dan *software* (perangkat lunak komputer).

9.1 Kebutuhan Sistem

Program proteksi data dengan flash disk yang menggunakan kombinasi serialisasi XML dan Algoritma Rijndael ini selanjutnya dinamakan dengan Dexia. Dalam merancang dan membangun program Dexia membutuhkan spesifikasi tertentu baik perangkat keras maupun perangkat keras maupun perangkat lunak, spesifikasi tersebut akan dijelaskan dibawah ini.

A Kebutuhan perangkat keras

Sifat umum dari perangkat keras adalah dapat dilihat dan dipegang bentuk fisiknya. Adapun perangkat kerang yang dibutuhkan untuk menjalankan aplikasi ini yaitu:

1. Prosesor minimal Pentium II 350 MHz
2. Memori minimal 256 MB
3. Monitor yang dapat menampilkan gambar dengan resolusi 800 x 600.
4. *Mouse* dan *keyboard*

Kecepatan proses enkripsi dan dekripsi data pada Dexia sangat tergantung dari *prosesor* dan besar memori yang dimiliki oleh suatu komputer. Semakin tinggi spesifikasi komputer maka semakin cepat pula proses enkripsi dan dekripsi data.

B Kebutuhan perangkat lunak

Perangkat lunak merupakan kebalikan dari perangkat keras di mana bentuk fisiknya tidak dapat dipegang. Adapun perangkat lunak yang dibutuhkan adalah :

1. Microsoft Windows 2000/XP atau Microsoft Windows 2000/2003 *server.*
2. Microsoft .Net Framework 2.0
3. Dexia.axe yang digunakan untuk menjalankan program Dexia. Pada Dexia.exe umumnya dapat digunakan pada Microsoft Windows 2000/XP atau Microsoft 2000/2003 server. Dan apabila menggunakan Microsoft 98/ME pastikan bahwa USB Disk dapat berjalan dengan baik pada sistem operasi tersebut.

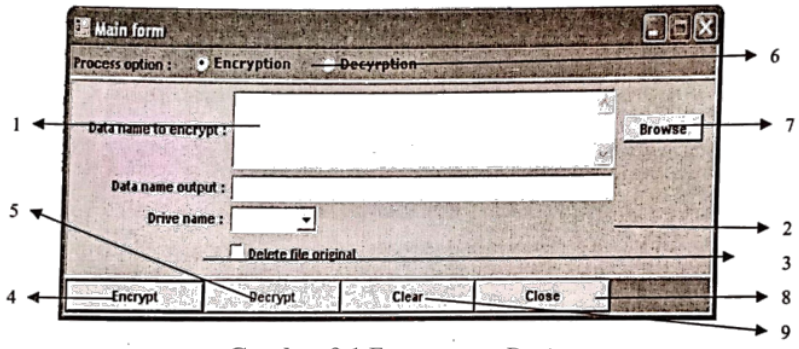
Setelah semua perangkat lunak di atas sudah ada, maka selanjutnya dapat menggunakan program Dexia.

9.2 Implementasi program

Setelah Dexia selesai dibuat maka tahap selanjutnya adalah mengimplementasikan Dexia ke komputer. Dexia hanya memiliki satu *form* dimana proses enkripsi dan dekripsi dilakukan dalam satu *form* tersebut. Berikut ini penjelasan *form* Dexia tersebut.

1. *Form* Dexia

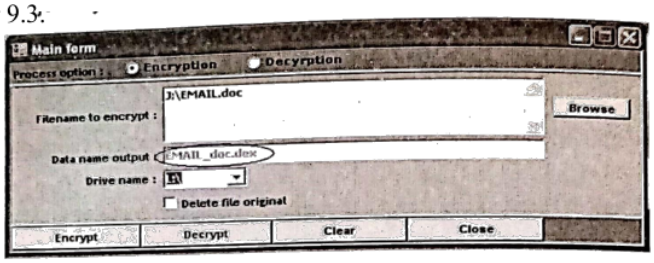
   Form Dexia adalah *form* yang digunakan untuk melakukan enkripsi data, dekripsi data dan juga memilih USB disk yang akan digunakan. Jadi semua proses dilakukan di *form* ini. Berikut ini penjelasan dari *form* utama dengan nomor urut sesuai dengan nomor yang ada dipenjelasan Gambar 9.1



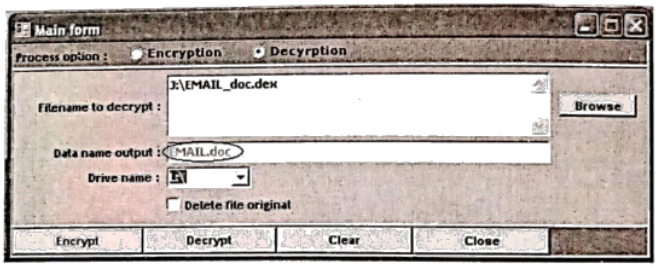
Gambar 9.1 Form utama *Dexia*

a. *Field Data name to encrypt* (Gambar 9.1 nomor 1) untuk meletakkan data yang akan dienkripsi. Data ini dipilih dengan menekan tombol Browse (nomor 7). Hal yang sama juga berlaku pada saat memilih data yang akan dienkripsi. Untuk memilih proses enkripsi atau dekripsi adalah dengan menekan *radio button* (nomor 6)

b. *Field Data name to output* (Gambar 9.1 nomor 2) untuk menampung data *output* di mana berupa data yang telah terenkripsi. Data tersebut setelah terenkripsi berubag ekstensi menjadi .dex. Agar dapat kembali ke bentuk semula maka ekstensi dari data asli tetap akan di simpan dan dijadikan satu sebagai nama data *output.* Sebagai contoh apabila data *input* bernama J:\Email.doc maka nantinya data *output* akan memiliki nama process Email_doc.dex . Dan apabila melakukan dekripsi maka data tersebut kembali menjadi Email.doc. Untuk lebih jelasnya dapat melihat pada Gambar 9.2 dan Gambar 9.3.



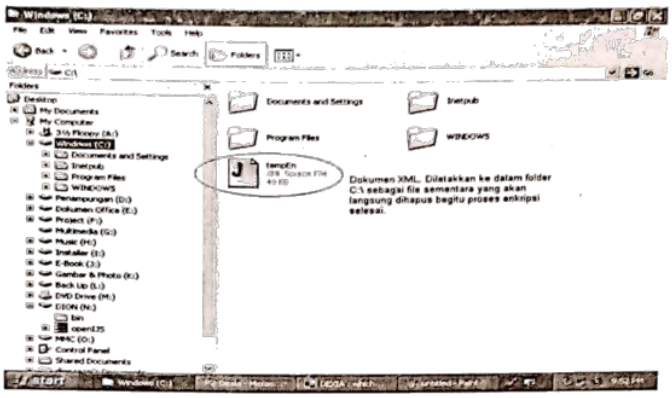
Gambar 9.2 Nama Data Saat Akan di Enkrip



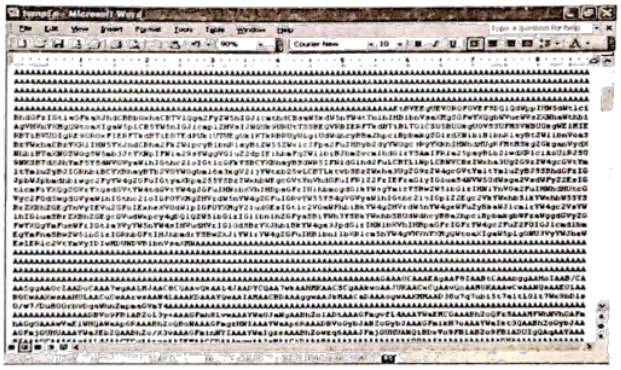
Gambar 9.3 *Nama Data Saat Akan di Dekrip*

c. *Field drive name* (gambar 9.1 nomor 3) untuk menentukan USB disk yang akan dignakan. Pada Dexia *field drive name* berisi *drive* yang *removable disk.*
d. Tombol *encrypt* (gambar 9.1 nomor 4) digunakan untuk melakukan enkripsi. Saat proses enkripsi ini data akan diserialisasi terlebih dahulu dan *output*nya adalah dokumen XML akan diletakkan ke tempat penampungan sementara di C\:tempEn yang langsung dihapus oleh Dexia begitu proses enkripsi selesai.

Bentuk dokumen XML ini dapat dilihat dengan menggunakan program teks editor biasa seperti Notepad, Wordpad atau Microsoft Word. Untuk lebih jelasnya dapat dilihat pada Gambar 9.5 dan Gambar 9.6.

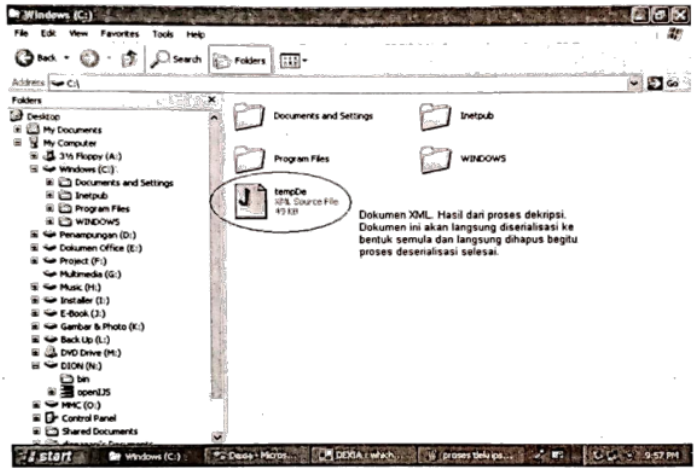


Gambar 9.4 *Dokumen XML pada Proses Enkripsi*

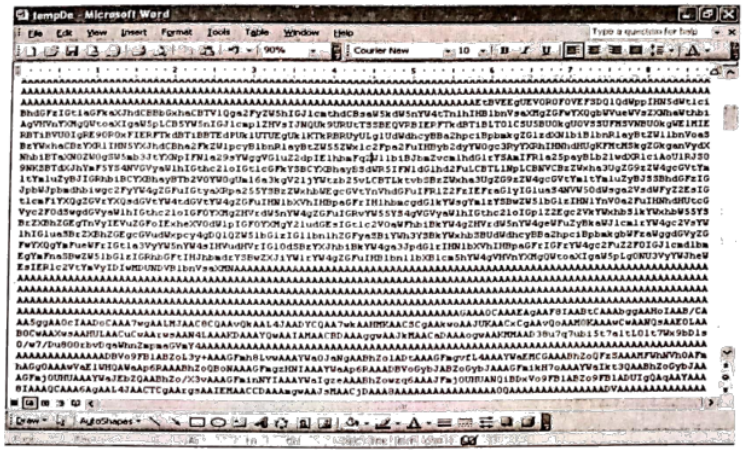


Gambar 9.5 *Dokumen XML C:\tempEn dilihat Menggunakan Microsoft Word*

e. Tombol *decrypt* (gambar 9.1 nomor 5)digunakan untuk melakukan dekripsi. Pada proses dekripsi data kembali ke bentuk dokumen XML terlebih dahulu. Setelah itu Dexia akan melakukan deserialisasi dokumen XML tersebtu ke bentuk data semula. Dokumen XML ini disimpan ke dalam folder sementara di C:\tempDe yang akan langsung dihapus begitu proses dekripsi dan deserialisasi selesai. Bentuk dokumen XML ini juga dapat dilihat dengan menggunakan perangkat lunak teks editor biasa seperti Notepad, Wordpad atau Microsoft Word. Untuk lebih jelasnya dapat dilihat pada Gambar 9.7 dan Gambar 9.8.



Gambar 9.6 *Dokumen XML pada Proses Dekripsi*



Gambar 9.7 *Dokumen XML C:\tempDe dilihat Menggunakan Microsoft Word*

f. Tombol *Close* (Gambar 9.1 nomor 8) digunakan untuk menutup tampilan dari form Dexia.

g. Tombol Clear (Gambar 9.1 nomor 9) digunakan untuk menghapus data pada *text field* atau mengosongkan *text field*.

10 KESIMPULAN

Setelah melakukan analisis, perancangan dan pembuatan aplikasi proteksi data atau yang dinamakan Dexia ini seta evaluasi hasil penelitiannya, maka dapat diambil kesimpulan sebagai berikut:

1. Data yang dapat diproteksi menggunakan aplikasi Dexia ini adalah data dokumen word, gambar (jpeg), musik (mp3), Pdf.
2. Pada aplikasi Dexia ini, ukuran file setelah proses enkripsi selalu lebih besar dibandingkan sebelum proses enkripsi.
3. Penggunaan *serial number* USB disk sebagai *password* atau *key* sendiri meminimalisasi penggandaan *password,* ini karena *serial number* USB disk hampir mustahil digandakan.
4. Karena digunakan sebagai proteksi adalah *serial number* pada flashdisk maka yang perlu diperhatikan adalah kondisi fisik pada flashdisk maka yang perlu diperhatikan adalah kondisi fisik pada flashdisk harus benar - benar baik. Karena apabila terjadi kerusakan maka *serial number* flashdisk tidak dapat dibaca oleh sistem. Dan yang terjadi adalah file yang telah di enkripsi tidak dapat dikembalikan lagi.

11 DAFTAR PUSTAKA

1. Ariyus , Dony(2005). kriptografi keamanan Data dan Komunikasi. Yogyakarta: Graha Ilmu.
2. Daemen Joan dan Rijmen Vincent. (1999) The Rijndael Block Chiper AES Proposal. URL: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.Maret.2007
3. Firdaus (2006) 7 Jam Belajar Visual Basic.NET untuk orang awam. Palembang:Maxikom
4. Gladman, Brian, Dr (2001). *A Sprecification for the AES Algortihm*. Amerika Serikat. Maret 2007
5. Kurniawan, yusuf (2004). Kriptografi Keamanan Internet dan Jaringan Komunikasi. Bandung: Informatika Bandung.
6. Melnick, Chris. (2004). What is Base 64. URL :www,aardwulf.com. Oktober 2006

7. Probo , Bayu (2004). Belajar Sendiri dalam 21 Hari Visual Basic.Net (terjemahan Mackenzie, Duncan, &Sharkey, Kent) Yogyakarta: Penerbit Andi. Buku asli diterbitkan tahun 2002
8. Sukmawan, Budi (1998). Keamanan Data dan Metoda Enkripsi. Agustus 2006.
9. Wibowo, Ari, Wihartantyo (2004). Advanced Encryption Standard, Algorithma Rijdael . Tugas mata kuliah Keamanan sistem informasi Bandung. Maret 2007.

PROTEKSI DATA DENGAN MENGGUNAKAN KOMBINASI SERIALI

ORIGINALITY REPORT

11%	11%	16%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	simdos.unud.ac.id Internet Source	11%

Exclude quotes	Off	Exclude matches	< 6%
Exclude bibliography	Off		