RANCANG BANGUN APLIKASI ENKRIPSI *FILE* MENGGUNAKAN ALGORITMA *RIJNDAEL*

SKRIPSI

Oleh:

AFDAL YULIUS
NIM. 04550004

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) MALANG
2008

RANCANG BANGUN APLIKASI ENKRIPSI *FILE*
MENGGUNAKAN ALGORITMA *RIJNDAEL*

SKRIPSI

Diajukan Kepada :

Universitas Islam Negeri (UIN) Malang

Untuk Memenuhi Salah Satu Persyaratan Dalam

Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:

AFDAL YULIUS
NIM. 04550004

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN) MALANG
2008

LEMBAR PERSETUJUAN

Rancang Bangun Aplikasi Enkripsi *File* Menggunakan Algoritma *Rijndael*

SKRIPSI

Oleh :

Afdal Yulius

NIM : 4550004

Telah Disetujui

Dosen Pembimbing 1	Dosen Pembimbing 2
Ririen Kusumawati, M.Kom	**Achmad Nashichuddin, M.A**
NIP. 150 368 775	NIP 150 302 531

Mengetahui,

Ketua Jurusan Teknik Informatika

Suhartono, M.Kom

NIP.150 327 241

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI ENKRIPSI *FILE* MENGGUNAKAN ALGORITMA *RIJNDAEL*

SKRIPSI

Oleh :
Afdal Yulius
04550004

Telah dipertahankan Di Depan Dewan Penguji
Dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana Komputer (S. Kom)

Pada Tanggal, 29 Juli 2008

SUSUNAN DEWAN PENGUJI

Penguji Utama	**Ketua Penguji**
Fatchurrochman, M.Kom	**Ir. M Amin Hariyadi, M.T**
NIP. 150 368 774	**NIP. 150 368 791**
Sekretaris Penguji	**Anggota Penguji**
Ririen Kusumawati, M.Kom	**Achmad Nashichuddin, M.A**
NIP. 150 368 775	**NIP 150 302 531**

Mengetahui dan Mengesahkan

Ketua Jurusan Teknik Informatika

Universitas Islam Negeri Malang

Suhartono, M.Kom
NIP.150 327 241

Lembar Persembahan

Yang utama dari segalanya.....

Sembah sujud serta syukur kepada Allah SWT dzat pemilik penguasa segalanya. Sholawat dan salam semoga senantiasa terlimpahkan kepada Rosulullah saw

Kupersembahkan karya sederhana ini kepada orang yang sangat kukasihi dan kusayangi.

Ayah dan Mama.....

Tiada kata yang dapat ku tulis untuk menggambarkan segala pengorbanan dan kasih sayang beliau. Namun, hanya doa yang dapat kupersembahkan semoga kasih sayang dan rahmat Allah swt selalu tercurahkan..

Kakakku beserta Keluarganya

Yang telah memberikan dorongan semangat dan materil sehingga adinda dapat mencapai gelar sarjana. Terimakasih untuk semuanya....

Yusraini Jamil

"Wanita yang paling ku sayangi, yang setia menemani hari-hariku sejak aku masih di bangku sekolah dulu hingga saat ini dan sampai Tuhan memisahkan kita. Tak ada yang dapat menggantikan dirimu di hatiku."

MOTTO

"Ingatlah Allah Dimanapun dan Kapanpun"

PERNYATAAN KEASLIAN SKRIPSI

Judul Skripsi	:	Rancang Bangun Aplikasi Enkripsi *File* Menggunakan Algoritma *Rijndael.*
Nama Mahasiswa	:	Afdal Yulius
NIM	:	04550004
Jurusan	:	Teknik Informatika
Fakultas	:	Sains dan Teknologi
Jenjang	:	Strata-1

Saya yang bertanda tangan dibawah ini:

Menyatakan dengan sesungguhnya bahwa skripsi yang dibuat dengan judul:

"RANCANG BANGUN APLIKASI ENKRIPSI FILE MENGGUNAKAN ALGORITMA RIJNDAEL."

Adalah skripsi yang saya buat sendiri, bukan duplikat serta tidak mengutip atau menyadur seluruhnya karya orang lain kecuali disebut dari sumber aslinya.

Malang, 28 Juli 2008

Yang membuat pernyataan

Afdal Yulius

KATA PENGANTAR

Bissmillahirrahmanirrahim

Alhamdulillah, puji syukur kehadirat Allah SWT yang melimpahkan segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang menjadi salah satu syarat mutlak untuk menyelesaikan program studi Teknik Informatika jenjang Strata-1 Universitas Islam Negeri (UIN) Malang.

Dengan segala kerendahan hati, penulis menyadari bahwa dalam menyelesaikan skripsi ini tidak lepas dari peran berbagai pihak yang telah banyak memberikan bantuan, bimbingan dan dorongan. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang tak terhingga khususnya kepada:

1. Bapak Prof. DR. H. Imam Suprayogo, selaku Rektor Universitas Islam Negeri Malang beserta seluruh staf. Darma Bakti Bapak dan Ibu sekalian terhadap Universitas Islam Negeri Malang turut membesarkan dan mencerdaskan penulis.
2. Bapak Prof. Drs. Sutiman Bambang Sumitro, SU., DSc, selaku Dekan Fakults Sains dan Teknologi Universitas Islam Negeri Malang beserta staf . Bapak dan ibu sekalian sangat berjasa memupuk dan menumbuhkan semangat untuk maju kepada penulis.
3. Bapak Suhartono, M.Kom selaku Ketua Jurusan Teknik Informatika dan dosen pembimbing penulisan skripsi ini yang telah memotivasi, membantu dan memberikan penulis arahan yang baik dan benar dalam menyelesaikan penulisan skripsi ini .

4. Ibu Ririen Kusumawati, M.Kom selaku pembimbing skripsi sekaligus orang tua penulis di jurusan Teknik Informatika UIN Malang yang telah banyak memberikan bimbingan serta motifasi kepada penulis dalam menyelesaikan skripsi ini.
5. Seluruh Dosen Universitas Islam Negeri (UIN) Malang, khususnya dosen Teknik Informatika dan staf yang telah memberikan ilmu kepada penulis selama empat tahun lamanya, serta kepada bapak Cahyo dosen ITN selaku pembimbing penulis dalam menyelesaikan penulisan skripsi ini.
6. Orangtua, dan seluruh keluarga besar di Padang yang telah banyak memberikan doa, motivasi dan dorongan dalam penyelesaian skripsi ini.
7. Pak Angah dan Bunda yang sebagai orang tua penulis di Malang, yang telah memberikan kasih sayang nya selama ini.
8. Semua sahabat dan saudara-saudara yang telah membantu penulis hingga terselesaikanya skripsi ini, khususnya kepada Zaenal, Ajib, Siraj, Dhofar, Anif, Ana, Ivana, Ulfa, David, Adi, Tuhil, Hakim, Alfi, Ayoung, Iqbal, Mujib, Azwar, Zaenul, mba' pen, Ida, Nia, Indah, Toni, Rudi, Tika terimakasih printer-nya dan semua sahabat di TI-UIN Malang angakatan 2004 semoga *Allah* SWT memberikan balasan yang setimpal atas jasa dan bantuan yang telah diberikan.
9. Semua pegawai dan rekan-rekan di Puskom ITS yang telah memberikan kesempatan kepada penulis untuk belajar mengembangkan potensi diri dalam Praktek Kerja Lapangan (PKL).

10. Rekan-rekan di www.Delphi-id.org Bos Luri selaku pemilik forum Delphi-Id.org (terimakasih atas situs nya), Indra Gunawan /DE (Terimaksih atas ilmu nya selama ini, andai aku tidak dibantu mungkin skripsi ini tidak akan selesai tepat waktu), Syarif (Terimakasih atas link-link nya udah mengenalkan aku dengan para master Delphi), Mas Kofa, Yaya Retina, Eko Indriawan dan Mr.Lee (terimakasih atas sharing ilmunya), serta semua rekan-rekan programmer Delphi-Id.org (Walaupun kita tidak pernah bertemu tapi kita adalah pecinta setia Delphi, Maju terus Delphi Indonesia..!!)

Penulis menyadari sepenuhnya bahwa sebagai manusia biasa tentunya tidak akan luput dari kekurangan dan keterbatasan. Maka dengan segenap kerendahan hati, penulis mengharapkan saran dan kritik yang dapat menyempurnakan penulisan ini sehingga dapat bermanfaat dan berguna untuk pengembangan ilmu pengetahuan.

Malang, 28 Juli 2008

Penulis

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ISI

ABSTRAKSI

Afdal Yulius . 2008, **Rancang Bangun Aplikasi Enkripsi *File* menggunakan algoritma *Rijndael.***

Nama Mahasiswa :

NIM : 04550004

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Jenjang : Strata-1

Cryptography adalah ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan. Sedangkan Enkripsi adalah suatu proses yang melakukan perubahan suatu kode dari yang bisa dimengerti menjadi tidak bisa dimengerti (tidak terbaca). Dekripsi adalah suatu proses dengan algoritma yang sama untuk mengembalikan informasi teracak tadi menjadi bentuk aslinya

Rijndael adalah salah satu algoritma enktipsi simetris. Dimana proses enkripsi dan dekripsi nya menggunakan kunci yang sama. *Rijndael* merupakan algoritma yang ditetapkan sebagai standar metode enkripsi modern pengganti DES (*Data Encryption Standard*), dalam sayembara AES (*Advanced Encryption Standard*). oleh NIST (*National Institute of Standards and Technology*).

Rijndael diambil dari nama pembuatnya Dr. Vincent Rijmen dan Dr. Joan Daemen. *Rijndael* sendiri dipilih jadi pemenang bukan karena algoritma paling aman, namun karena memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai *paltform software* dan *hardware.*

.

Kata kunci: *Cryptography,* Enkripsi, *Rijndael,* dan AES.

BAB I

PENDAHULUAN

1.1 Latar Belakang

هُوَ ٱللَّهُ ٱلَّذِى لَآ إِلَٰهَ إِلَّا هُوَ ٱلْمَلِكُ ٱلْقُدُّوسُ ٱلسَّلَٰمُ ٱلْمُؤْمِنُ ٱلْمُهَيْمِنُ ٱلْعَزِيزُ ٱلْجَبَّارُ ٱلْمُتَكَبِّرُ ۚ سُبْحَٰنَ ٱللَّهِ عَمَّا يُشْرِكُونَ ﴿٢٣﴾

"Dialah Allah yang tiada Tuhan selain Dia, raja, yang Maha suci, yang Maha Sejahtera, yang Mengaruniakan Keamanan, yang Maha Memelihara, yang Maha Perkasa, yang Maha Kuasa, yang memiliki segala Keagungan, Maha Suci Allah dari apa yang mereka persekutukan." (QS : Al Hasyr : 23).

Dari ayat Al-Qur'an diatas di sebutkan bahwa Allah adalah Tuhan yang wajib untuk kita sembah. Allah yang maha memelihara, yang maha perkasa, yang maha kuasa, yang memiliki segala keagungan, juga yang mengaruniakan keamanan bagi manusia.

Keamanan merupakan karunia Allah yang diberikan kepada manusia yang wajib untuk kita syukuri. Allah juga yang telah mengaruniakan manusia akal untuk berpikir. Masalah keamanan merupakan hal terpenting dalam kehidupan di dunia ini. Oleh karena itu manusia wajib menggunakan akalnya untuk mempelajari dan menciptakan keamanan itu.

Perkembangan teknologi komputer dan teknologi telekomunikasi pada saat ini telah mengubah cara masyarakat dalam berkomunikasi. Dulu, komunikasi jarak jauh masih dilakukan dengan cara konvensional, yaitu dengan cara saling mengirim surat. Sekarang, dengan adanya *internet*, komunikasi jarak jauh bisa dilakukan dengan cara saling mengirim *email* atau *sms* (*short messaging service*). *Internet* juga telah membuat komunikasi semakin terbuka dan pertukaran informasi juga semakin cepat melewati batas-batas negara dan budaya. Namun tidak semua perkembangan teknologi komunikasi ini memberikan dampak yang menguntungkan bagi dunia komunikasi. Penyadapan data merupakan hal yang paling ditakuti oleh pengguna jaringan komunikasi pada saat ini. (Wibowo, 2004)

Masalah keamanan merupakan salah satu aspek terpenting dari suatu sistem informasi. Masalah keamanan sering kali kurang mendapat perhatian dari para perancang dan pengelola sistem informasi. Seringkali masalah keamanan berada di urutan setelah tampilan, atau bahkan di urutan terakhir dalam daftar hal-hal yang dianggap penting. Apabila mengganggu performansi dari sistem, seringkali masalah keamanan tidak begitu dipedulikan bahkan ditiadakan.

Dengan adanya kemungkinan penyadapan data, maka aspek keamanan dalam pertukaran informasi menjadi sangat penting karena suatu komunikasi data jarak jauh belum tentu memiliki jalur transmisi yang aman dari penyadapan sehingga keamanan informasi menjadi bagian penting dalam dunia informasi itu sendiri. Terdapat data-data yang tidak terlalu penting, sehingga apabila publik mengetahui data tersebut, pemilik data tidak terlalu dirugikan. Tetapi apabila

pemilik data adalah pihak militer atau pemerintah, keamanan dalam pertukaran informasi menjadi sangat penting karena data yang mereka kirim kebanyakan adalah data-data rahasia yang tidak boleh diketahui oleh publik.

Aspek-aspek keamanan komputer meliputi enam aspek diantaranya (Ariyus, 2006):

1. *Confidentiality*. Menjamin bahwa data-data tersebut hanya bisa diakses oleh pihak-pihak tertentu.
2. *Authentication*. Baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya.
3. *Integrity*. Menjamin setiap pesan pasti sampai kepada pihak yang dituju dengan data asli.
4. *Nonrepudation*. Mencegah pengirim ataupun penerima mengingkari bahwa pesan itu adalah kiriman dari / untuk mereka.
5. *Acces Control*. Membatasi sumber-sumber data hanya kepada orang-orang tertentu
6. *Avilability*. Semua pesan bisa dibuka setiap saat yang diinginkan.
7. *Secrecy*. Perlindungan terhadap kerahasiaan data informasi

Cryptography adalah salah satu teknik yang digunakan untuk meningkatkan aspek keamanan suatu informasi. *Cryptography* merupakan kajian ilmu dan seni untuk menjaga suatu pesan atau data informasi agar data tersebut

aman. *Cryptography* mendukung kebutuhan dari dua aspek keamanan informasi, yaitu *Confidentiality, Authentication, Integrity, Nonrepudation,* dan *Secrecy*.

Cryptography berasal dari dua kata Yunani, yaitu *Crypto* yang berarti rahasia dan *Grapho* yang berarti menulis. Secara umum *cryptography* dapat diartikan sebagai ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan. *Cryptography* pada dasarnya sudah dikenal sejak lama. Menurut catatan sejarah, *cryptography* sudah digunakan oleh bangsa Mesir sejak 4000 tahun yang lalu oleh raja-raja Mesir pada saat perang untuk mengirimkan pesan rahasia kepada panglima perangnya melalui kurir-kurinya (Ariyus, 2006).

Terdapat dua teknik dasar yang biasa digunakan pada *cryptography* pada masa itu :

1. Teknik subtitusi : Penggantian setiap karakter pesan dengan karakter lain. Algoritma yang menggunakan teknik ini diantaranya adalah : *Caesar Cipher, Hill Cpiher, Vigenere Cipher* dll.

2. Teknik Transposisi : Teknik ini menggunakan permutasi karakter.

Sedangkan pada masa sekarang *cryptography* mempunyai jenis kerumitan yang sangat kompleks, karena dalam pengoperasiannya telah menggunakan komputer.

Algoritma *cryptography* yang baik akan memerlukan waktu yang lama untuk memecahkan data yang telah disandikan. Seiring dengan perkembangan teknologi komputer maka dunia teknologi informasi membutuhkan algoritma *cryptography* yang lebih kuat dan aman. Saat ini, *AES* (*Advanced Encryption Standard*) digunakan sebagai standar algoritma *cryptography* yang terbaru. AES menggantikan DES (*Data Encryption Standard*) yang pada tahun 2002 sudah berakhir masa penggunaannya. DES juga dianggap tidak mampu lagi untuk menjawab tantangan perkembangan teknologi komunikasi yang sangat cepat. Dalam penelitian yang dilakukan oleh rinaldi Munir tahun 2004, algoritma *Rijndael* dalam di aplikasikan pada tipe data string, dan stream.(Daemen and Rijmen, 1998)

AES / *Rijndael* sendiri adalah algoritma *cryptography* simetris dengan menggunakan algoritma *Rijndael*, yang dapat mengenkripsi dan mendekripsi *block* data sepanjang 128 *bit* dengan panjang kunci 128 *bit* , 192 *bit* , atau 256 *bit*. (Kurniawan, 2004).

1.2 Rumusan Masalah

Bagaimana algoritma *Rijndael* melakukan proses enkripsi.

Bagaimana melindungi kerahasiaan *file* dengan melakukan proses enkripsi *Rijndael*.

1.3 Batasan Masalah

Enkripsi yang dipakai menggunakan algoritma *Rijndael*.

File yang di enkripsi adalah seluruh *type file*.

1.4 Tujuan dan Manfaat Penelitian

a. Tujuan Penelitian

Membangun aplikasi yang dapat melakukan proses enkripsi.

b. Manfaat Penelitian

Untuk menjaga kerahasiaan data agar tidak bisa dibaca oleh orang lain.

1.5 Metode Penelitian

Dalam mengembangkan aplikasi ini digunakan metode yang digunakan adalah :

1. Fase Analisis
    - Studi pustaka yaitu Mempelajari dan memahami landasan teori yang terkait dengan masalah yang akan dibahas, serta konsultasi bimbingan dengan dosen pembimbing dan dosen lainnya.
    - Konsultasi dan bimbingan, yaitu setelah mempelajari teori berdasarkan literatur lalu juga konsultasi dengan dosen pembimbing dan dosen lainnya.
2. Fase Pembuatan Program
    - Eksperimental

Yaitu ber *Eksperimen* membuat program, berdasarkan materi dan algoritma yang telah dipelajari. Fase ini dilakukan secara otodidak (belajar sendiri). Baik itu coding programnya maupun *lay out* program.

3. Fase Uji Coba Program

Dari fase uji coba ini didapat beberapa *level* untuk proses pengujian diantaranya:

- Uji enkripsi

Yaitu menguji apakah program yang telah di bangun sesuai dengan metode enkripsi *Rijndael*? Dan apakah dijebol/ di *hack* dengan mudah oleh *hacker*?

- Uji Dekripsi

Yaitu menguji apakah data yang dienkripsi dapat di dekripsikan?

- Uji Kelayakan

Yaitu dengan mendemonstrasikan program didepan para dosen dan teman-teman *programmer*.

4. Revisi Program

Setelah fase diatas, tahap ini adalah memperbaiki kesalahan- kesalahan dalam listing program maupun menambah kekurangan dari program yang dikerjakan.

- Dokumentasi dan penyusunan Laporan

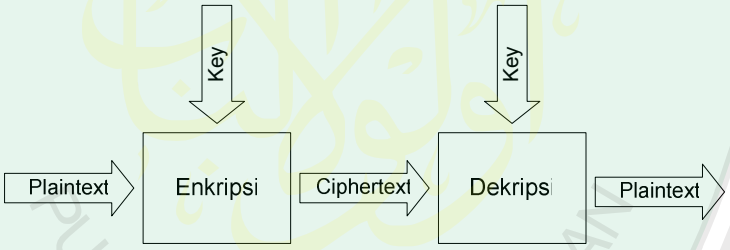
Dokumentasi dilakukan untuk jangka waktu sekarang dan yang akan datang agar memudahkan *maintenance* jika terjadi kesalahan program lagi akibat ketidak stabilan perangkat atau karena gangguan teknis lainnya. Penyusunan laporan adalah tahap dimana melaporkan semua hal dan data-data yang sudah dikerjakan selama penelitian.

BAB II

TINJAUAN PUSTAKA

2.1 *Cryptography*

Cryptography adalah ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan. Sedangkan Enkripsi adalah suatu proses yang melakukan perubahan suatu kode dari yang bisa dimengerti menjadi tidak bisa dimengerti (tidak terbaca). Dekripsi adalah suatu proses dengan algoritma yang sama untuk mengembalikan informasi teracak tadi menjadi bentuk aslinya. (WAHANAKomputer, 2003)



Gambar2.1 Proses Enkripsi dan Dekripsi

Dalam sistem komputer, pesan terbuka (*plaintext*) diberi lambang M, yang merupakan singkatan dari *Message*. *Plaintext* ini dapat berupa tulisan, foto, atau video yang berbentuk data biner. *Plaintext* inilah yang nantinya akan dienkripsi menjadi pesan rahasia atau *ciphertext* yang dilambangkan dengan C (*ciphertext*). Secara matematis, fungsi enkripsi ini dinotasikan dengan :

$$E(M) = C \tag{2.1}$$

Sedangkan fungsi dekripsi adalah proses pembalikan dari *ciphertext* menjadi *plaintext* kembali. Secara matematis dinotasikan sebagai berikut :

$$D(C) = M$$

$$D(E(M)) = M \quad (2.2)$$

Dalam *cryptography* modern algoritma yang digunakan tidak lagi rahasia, yang dirahasiakan hanyalah *key* untuk melakukan enkripsi pesan (Kurniawan, 2004).

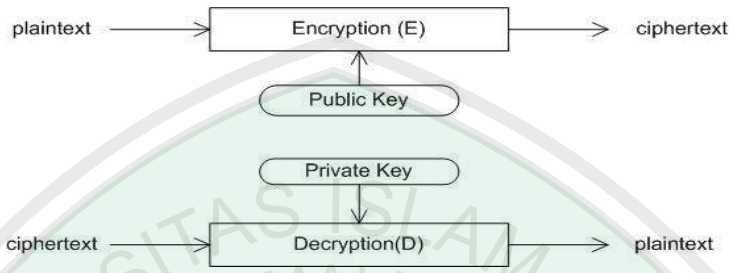
2.1.1 Pembagian Algoritma *Cryptography* Berdasarkan Jenis *Key* nya

2.1.1.1 *Asymmetric Cryptography* (*Cryptography* Asimetris)

Cryptography public key sering disebut dengan *cryptography* asimetris. *key* yang digunakan pada proses enkripsi dan proses dekripsi pada *cryptography public key* ini berbeda satu sama lain. Jadi dalam *cryptography public key*, suatu *key generator* akan menghasilkan dua *key* berbeda dimana satu *key* digunakan untuk melakukan proses enkripsi dan *key* yang lain digunakan untuk melakukan proses dekripsi. (Ariyus, 2006)

Key yang digunakan untuk melakukan enkripsi akan dipublikasikan kepada umum untuk dipergunakan secara bebas. Oleh sebab itu, *key* yang digunakan untuk melakukan enkripsi disebut juga sebagai *public key*. Sedangkan *key* yang digunakan untuk melakukan dekripsi akan disimpan oleh pembuat *key* dan tidak akan dipublikasikan kepada umum. *Key* untuk melakukan dekripsi ini

disebut *private key.* satu *key* digunakan untuk melakukan proses enkripsi dan *key* yang lain digunakan untuk melakukan proses dekripsi. (Wibowo, 2004)



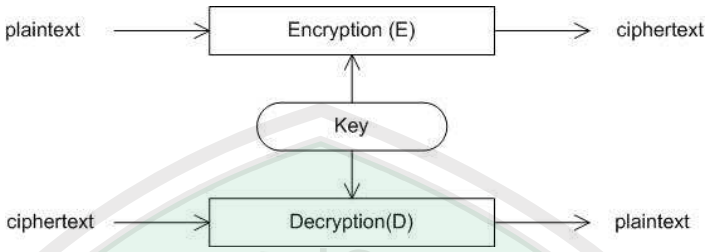
Gambar2.2 *Cryptography* Asimetris

Dengan cara demikian, semua orang yang akan mengirimkan pesan kepada pembuat *key* dapat melakukan proses enkripsi terhadap pesan tersebut, sedangkan proses dekripsi hanya dapat dilakukan oleh pembuat atau pemilik *key* dekripsi. Dalam kenyataannya, *cryptography* asimetris ini dipakai dalam *ssh*, suatu layanan untuk mengakses suatu *server*.

Algoritma-algoritma yang termasuk kedalaman *Cryptography* asimetris diantaranya adalah : *Elliptic Curve Cryptography* (ECC), *Digital Signature Algorithm* (DSA), *LUC, RSA, ELGAMAL, Diffie-Hellman* (DH). (www.wikipedia.org wiki/Advanced_Encryption_Standard.htm, 2007)

2.1.1.2 S*ymmetric Cryptography* (*Cryptography* Simetris)

Cryptography Simetris atau *secret key* adalah *cryptography* yang hanya melibatkan satu *key* dalam proses enkripsi dan dekripsi. Proses dekripsi dalam *cryptography secret key* ini adalah kebalikan dari proses enkripsi.



Gambar2.3 *cryptography* simetris

Cryptography simetris dapat dibagi menjadi dua, yaitu penyandian blok (*Block Cipher*) dan penyandian alir (*Stream Cipher*). Penyandian blok bekerja pada suatu data yang terkelompok menjadi blok-blok data atau kelompok data dengan panjang data yang telah ditentukan. Pada penyandian blok, data yang masuk akan dipecah-pecah menjadi blok data yang telah ditentukan ukurannya. Penyandian alir bekerja pada suatu data bit tunggal atau terkadang dalam satu *byte*. Jadi format data yang mengalami proses enkripsi dan dekripsi adalah berupa aliran *bit-bit* data. (Ariyus, 2006)

Algoritma-algoritma yang termasuk ke dalam *cryptography* simetris diantaranya adalah : *Data Encryption Standard* (DES), *International Data Encryption Algorithm* (IDEA), ***Rijndael*** (AES), *A5*, *RC2*, *RC4*, *RC5*, *RC6* dll.

2.1.1.2.1 *Stream Cipher*

Stream Cipher (Aliran *Cipher*) Merupakan suatu *Cipher* yang berasal dari hasil XOR. Setiap *bit plaintext* dengan setiap *bit key*. *Key* merupakan *key* utama (Kunci induk) yang digunakan untuk membangkitkan *key* acak semu yang

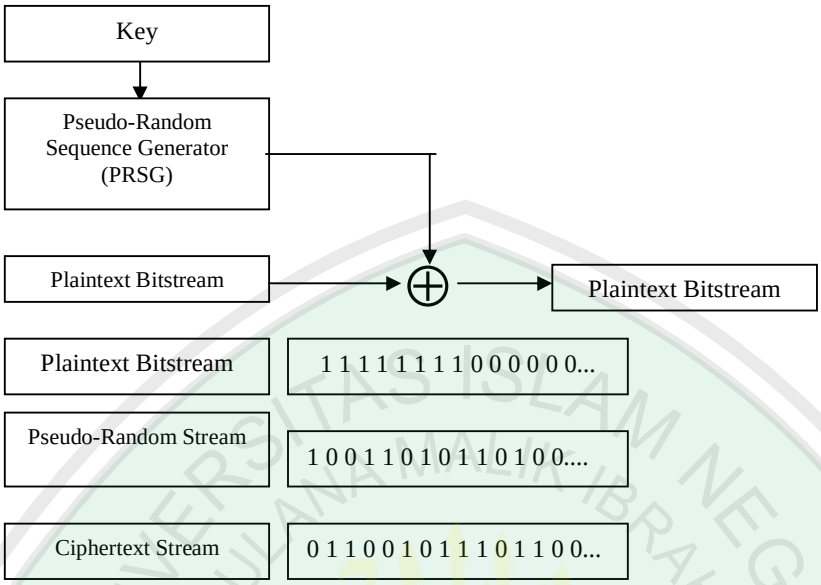
dibangkitkan dengan *Pseudo Random Sequence Generator* yang merupakan suatu nilai yang nampak seperti diacak, tetapi sesungguhnya nilai tersebut merupakan suatu urutan. Secara khusus urutan dari nilai yang dihasilkan oleh RNG (*Random Number Generator*), *computational* mekanisme *deterministic* atau FSM (*Finite State Machine*) merupakan kebalikan dari *Really Random*.

Salah satu pembangkit bilangan acak yang cocok untuk *cryptography* dinamakan *cryptographically secure pseudorandom generator* (CSPRNG) (Munir, 2006).

Persyaratan *CSPRNG* adalah:

1. Secara statistik ia mempunyai sifat-sifat yang bagus (yaitu lolos uji keacakan statistik).
2. Tahan terhadap serangan (attack) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan.

Random Number Generator secara umum adalah *Pseudo Random.* Yang memberikan *intial state* atau *seed* (nilai yang di-*input* kedalam *state*), seluruh urutan tersebut ditentukan secara keseluruhan, tetapi meskipun demikian banyaknya karakteristik yang ditampilkan dari suatu urutan yang diacak tersebut. *Pseudorandomness* menghasilkan urutan yang sama secara berulang-ulang pada penempatan yang berbeda. Kemudian kunci acak semu tersebut diberikan operasi *XOR* dengan *Plaintext* untuk mendapatkan *ciphertext*.



Gambar2.4 Proses *Stream Cipher*

2.1.1.2.2 *Block Cipher*

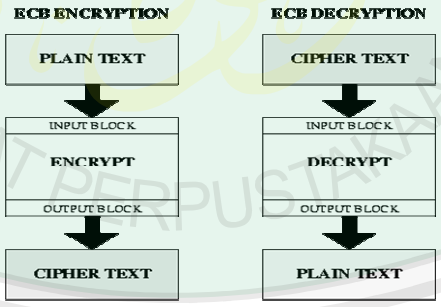
Block Cipher merupakan suatu logaritma yang mana *input* dan *output* nya berupa satu *block*, dan setiap *block* terdiri dari beberapa *bit* (1 *block* = 64 *bit* atau 128 *bit*). *Block cipher* mempunyai banyak aplikasi, aplikasi tersebut digunakan untuk memberikan layanan *confidentiality* (kerahasiaan), integritas data atau *authentication* (pengesahan pemakai), dan juga bisa memberikan layanan *keystream Generator* untuk *StreamCpiher*. (Munir, 2006)

Mode operasi dari *block cipher* terbagi menjadi empat bagian diantaranya adalah :

2.1.1.2.2.1 *Mode Electric Code Book* (ECB)

Menggunakan mode ini suatu *block cipher* yang panjang dibagi dalam bentuk *sequence biner* menjadi satu *block* tanpa mempengaruhi *block* yang lain, satu *block* terdiri dari 64 *bit* atau 128 *bit*, setiap *block* merupakan bagian dari pesan yang di enkripsi.

Mode ini merupakan suatu enkripsi yang sederhana, kerusakan satu *block* data tidak mempengaruhi *block* lainnya, sehingga jika penerima mendapatkan satu *block* yang rusak, maka penerima hanya minta dikirimkan kembali hanya *block* yang rusak, tanpa harus meminta semua *block*, hal ini membantu pengiriman *block* yang rusak dengan cepat. Pada dasarnya sifat yang paling mendasar dari mode ECB ini adalah *block plaintext* yang sama akan di kodekan menjadi *cipher* yang sama. (Wibowo, 2004)



Gambar2.5 Mode Operasi ECB

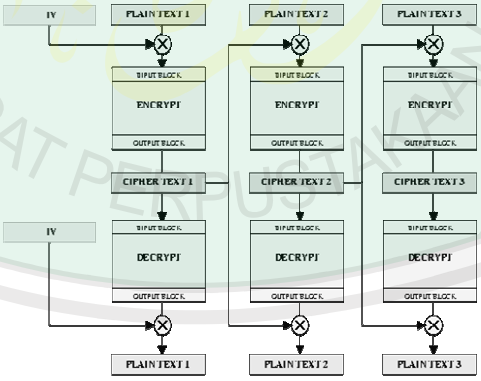
Keuntungan dari mode OBC ini adalah kemudahan dalam implementasi dan pengurangan resiko salahnya semua *plaintext* akibat kesalahan pada satu *plaintext*. Namun mode ini memiliki kelemahan pada aspek keamanannya.

Dengan mengetahui pasangan *plaintext* dan *ciphertext,* seorang *cryptanalist* dapat menyusun suatu *code book* tanpa perlu mengetahui kuncinya.

2.1.1.2.2.2 *Mode Cipher Block Chainning* (CBC)

Sistem dari mode *cipher block chainning* (CBC) adalah *plaintext* yang sama akan di enkripsi ke dalam bentuk *cipher* yang berbeda, disebabkan *block cipher* yang satu tidak berhubungan dengan *block cipher* yang lain. Melainkan tergantung pada *cipher* yang sebelumnya.

Pada CBC digunakan operasi umpan balik atau dikenal dengan operasi berantai (*chaining*). Pada CBC, hasil enkripsi dari blok sebelumnya adalah *feedback* untuk enkripsi dan dekripsi pada *block* berikutnya. Dengan kata lain, setiap *block ciphertext* dipakai untuk memodifikasi proses enkripsi dan dekripsi pada *block* berikutnya.



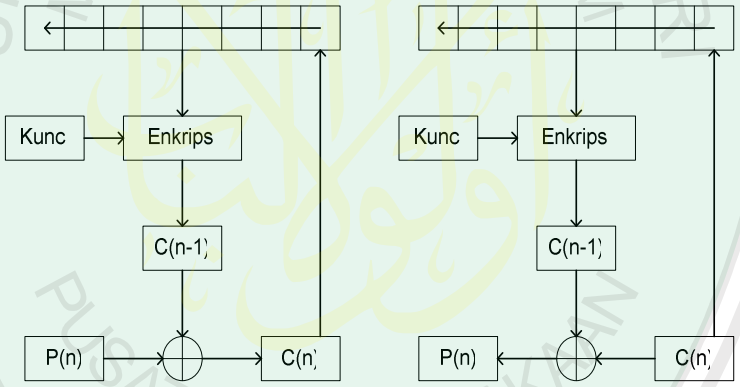
Gambar2.6 Mode Operasi CBC

Pada CBC diperlukan data acak sebagai *block* pertama. Blok data acak ini sering disebut *initialization vector* atau IV. IV digunakan hanya untuk membuat suatu pesan menjadi unik dan IV tidak mempunyai arti yang penting

sehingga IV tidak perlu dirahasiakan. (www.wikipedia.org wiki/Advanced_Encryption_Standard.htm, 2007)

2.1.1.2.2.2 *Mode Cipher Feed Back* (CFB)

Pada mode CBC, proses enkripsi atau dekripsi tidak dapat dilakukan sebelum blok data yang diterima lengkap terlebih dahulu. Masalah ini diatasi pada mode *Cipher Feedback* (CFB). Pada mode CFB, data dapat dienkripsi pada unit-unit yang lebih kecil atau sama dengan ukuran satu *block*. Misalkan pada CFB 8 *bit*, maka data akan diproses tiap 8 *bit*.



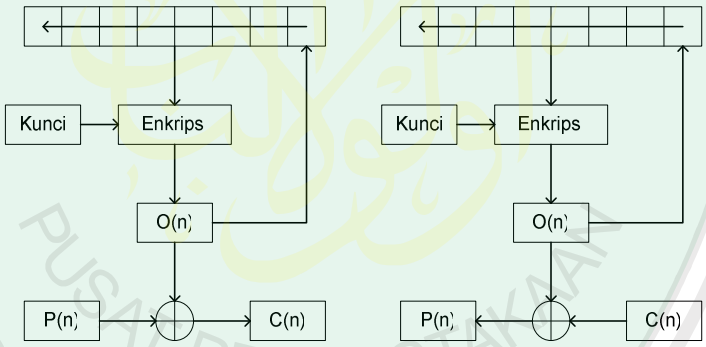
Gambar2.7 Mode Operasi CFB

Pada permulaan proses enkripsi, IV akan dimasukkan dalam suatu *register* geser. IV ini akan dienkripsi dengan menggunakan kunci yang sudah ada. Dari hasil enkripsi tersebut, akan diambil 8 bit paling kiri atau *Most Significant Bit* untuk di-XOR dengan 8 *bit* dari *plaintext*. Hasil operasi XOR inilah yang akan menjadi *ciphertext* dimana *ciphertext* ini tidak hanya dikirim untuk ditransmisikan

tetapi juga dikirim sebagai *feedback* ke dalam *register* geser untuk dilakukan proses enkripsi untuk 8 *bit* berikutnya. (wibowo, 2004)

2.1.1.2.2.3 *Mode Output Feedback* (OFB)

Sama pada mode CFB, mode OFB juga memerlukan suatu *register* geser dalam pengoperasiannya. Pertama kali, IV akan masuk ke dalam *register* geser dan dilakukan enkripsi terhadap IV tersebut. Dari hasil proses enkripsi tersebut akan diambil 8 *bit* paling kiri untuk dilakukan XOR dengan *plaintext* yang nantinya akan menghasilkan *ciphertext*. *Ciphertext* tidak akan diumpan balik ke dalam *register* geser, tetapi yang akan diumpan balik adalah hasil dari enkripsi IV.



Gambar2.8 Mode Operasi OFB

2.2 Pengantar Matematis

Seluruh *byte* dalam algoritma *Rijndael* diinterpretasikan sebagai elemen *finite field* GF(2^8). Elemen *finite field* ini dapat dikalikan dan dijumlahkan, tetapi hasil dari penjumlahan dan perkalian elemen *finite field* sangat berbeda dengan hasil dari penjumlahan dan perkalian bilangan biasa. (Wibowo, 2004)

2.2.1 *Finite Field* GF(2^8)

Pada *finite field* GF(2^8) 1 *Byte* terdiri dari 8 *bit* yang dapat dituliskan dengan $b_{7,}$ $b_{6,}$ $b_{5,}$...,b_0. dianggap *polynomials* dengan koefisien antara {0,1}.

Contoh : 63 = 01100011 = $x^6 + x^5 + x + 1$.

2.2.1.1 ***Addition*** (Penjumlahan)

Penjumlahan dari dua elemen dalam suatu *finite field* dilakukan dengan menjumlahkan koefisien dari pangkat *polynomials* yang bersesuaian dari dua elemen tersebut. Penjumlahan dilakukan dengan operasi *XOR* dan dinotasikan dengan $\oplus$. Dengan operasi ini, maka $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, dan $0 \oplus 0 = 0$. Pengurangan dari *polynomials* identik atau sama dengan penjumlahan *polynomials*.(Daemen and Rijmen, 1998)

Sebagai alternatif, penjumlahan elemen-elemen pada *finite field* dapat dijelaskan sebagai penjumlahan *modulo* 2 dari *bit* yang bersesuaian dalam *byte*. Untuk 2 *byte* $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ dan $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, hasil penjumlahannya adalah $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$ dimana setiap $c_i = a_i \oplus b_i$. Contoh dari operasi penjumlahan adalah sebagai berikut :

$(x^6+x^4+x^2+x+1) \oplus (x^7+x+1) = x^7+x^6+x^4+x^2$ (notasi *polynomials*)

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (notasi *biner*)

$\{57\} \oplus \{83\} = \{d4\}$ (notasi *exadecimal*)

2.2.1.2 *Multiplication* (Perkalian)

Dalam representasi *polynomials*, perkalian dalam GF(2^8) yang dinotasikan dengan • mengacu pada perkalian modulo *polynomials* suatu *irreducible polynomials* yang berderajat 8. Suatu *polynomials* bersifat *irreducible* jika satu-satunya pembagi adalah dirinya sendiri dan 1. Untuk algoritma *Rijndael*, *irreducible polynomials* ini adalah :

$$m(x) = x^8 + x^4 + x^3 + x + 1 = \text{‘11b’} \quad (2.3)$$

atau dalam notasi heksadesimal adalah {01}{1b}. Sebagai contoh: {57}•{83} = {c1}, karena

$$(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

dan

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)$$

$$= x^7 + x^6 + 1 \text{ (Fips197, 2001)}$$

Pengurangan *modulo* oleh *m(x)* memastikan bahwa hasilnya akan berupa *polynomials biner* dengan derajat kurang dari 8, sehingga dapat dipresentasikan dengan 1 *byte* saja. (Wibowo, 2004)

2.2.1.3 *Devided* (Pembagian)

Pembagian dalam GF(2^8) dengan representasi *polynomials* antara *polynomials* g(x) dengan h(x) menghasilkan *polynomials* dimana g(x) dan r(x) dimana :

$g(x) = g(x)\ h(x) + r(x)$ $\quad r(x) < h(x)$ (2.4)

g(x) adalah hasil bagi, r(x) adalah sisa bagi.

Contoh : 'GF' dibagi dengan '19' = x^2

$x^6 + x^5 + x^3 + x^2 + x + 1 = g(x)\ (x^4 + x^3 + 1) + r(x)$

$= x^2\ (x^4 + x^3 + 1) + r(x)$

$-r(x) = (x^2\ (x^4 + x^3 + 1)) - (x^6 + x^5 + x^3 + x^2 + x + 1)$

$r(x) = x^2\ (x^4 + x^3 + 1) + (x^6 + x^5 + x^3 + x^2 + x + 1)$

$r(x) = x^3 + x + 1$

dengan g(x) *modulo* h(x) adalah: $x^3 + x + 1$

2.3 *Rijndael*

Pada tahun 1972 dan 1974 *National Bureau of Standards* (sekarang dikenal dengan nama *National Institute of Standards and Technology*, NIST)

menerbitkan permintaan kepada publik untuk pembuatan standar enkripsi. Hasil dari permintaan pada saat itu adalah DES (*Data Encryption Standard*), yang banyak digunakan di dunia. DES adalah sebuat algoritma *cryptography* simetris dengan panjang *key* 56 *bit* dan blok data 64 *bit*. Dengan semakin majunya teknologi, para *cryptografer* merasa bahwa panjang kunci untuk DES terlalu pendek, sehingga keamanan algoritma ini dianggap kurang memenuhi syarat. Untuk mengatasi hal itu, akhirnya muncul *triple DES*. (Wibowo, 2004)

Triple DES pada waktu itu dianggap sudah memenuhi syarat dalam standar enkripsi, namun teknologi yang tidak pernah berhenti berkembang akhirnya juga menyebabkan standar ini dianggap kurang memenuhi syarat dalam standar enkripsi. Akhirnya NIST mengadakan kompetisi untuk standar *cryptography* yang terbaru, yang dinamakan AES (*Advanced Encryption Standard*). (Ariyus, 2006)

Persyaratan AES adalah :

- Algoritma harus dipublikasikan secara luas untuk diperiksa keamanannya.
- Algoritma haruslah merupakan simetris *Block Cipher.*
- Algoritma harus dapat diimplementasikan secara cepat dan tepat dalam *hardware* dan *software*.
- Algoritma harus memiliki input-an data 128*bit.*
- Algoritma memiliki kunci fleksibel : 128, 192 dan 256*bit.*

Dari hasil seleksi yang dilakukan oleh NIST, akhirnya NIST memilih 5 finalis AES, yaitu : *Mars* (IBM Amerika), *RC6* (RSA Corp, Amerika), *Rijndael* (Belgia), *Serpent* (Israel, Norwegia dan Inggris), dan *Twofish* (Counterpane Amerika). Kompetisi ini akhirnya dimenangkan oleh *Rijndael* dan secara resmi diumumkan oleh NIST pada tahun 2001. Sejak saat itu Algoritma *Rijndael* sering disebut juga dengan AES.

Rijndael diambil dari nama pembuatnya Dr. Vincent Rijmen dan Dr. Joan Daemen. *Rijndael* sendiri dipilih jadi pemenang bukan karena algoritma paling aman, namun karena memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai *paltform software* dan *hardware.* (Kurniawan, 2004)

2.3.1 Representasi Data

Input dan output dari algoritma *Rijndael* terdiri dari urutan data sebesar 128 *bit*. Urutan data yang sudah terbentuk dalam satu kelompok 128 *bit* tersebut disebut juga sebagai blok data atau *plaintext* yang nantinya akan dienkripsi menjadi *ciphertext*. *Cipher key* dari *Rijndael* terdiri dari *key* dengan panjang 128 *bit*, 192 *bit*, atau 256 *bit*. Urutan bit diberi nomor urut dari 0 sampai dengan *n*-1 dimana *n* adalah nomor urutan. Urutan data 8 *bit* secara berurutan disebut sebagai *byte* dimana *byte* ini adalah unit dasar dari operasi yang akan dilakukan pada blok data (Wihartantyo, 2004).

Dalam algoritma *Rijndael*, data sepanjang 128 *bit* akan dibagi-bagi menjadi *array byte* dimana setiap *array byte* ini terdiri dari 8 *bit* data input yang

saling berurutan. *Array byte* ini direpresentasikan dalam bentuk :

$a_0 a_1 a_2 ... a_{15}$

Dimana:

$a_0 = \{ input_0, input_1, ..., input_7 \}$

$a_1 = \{ input_8, input_9, ..., input_{15} \}$

$a_{15} = \{ input_{120}, input_{121}, ..., input_{127} \}$

$a_n = \{ input_{8n}, input_{8n+1}, ..., input_{8n+7} \}$

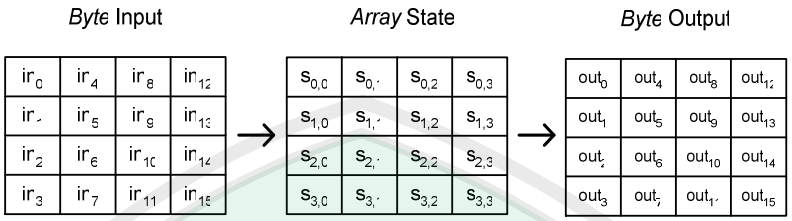
Operasi algoritma *Rijndael* dilakukan pada suatu *state* dimana *state* sendiri adalah suatu *array byte* dua dimensi. Setiap *state* pasti mempunyai jumlah baris yang tetap, yaitu 4 baris, sedangkan jumlah kolom tergantung dari besarnya blok data. Baris pada state mempunyai indeks nomor *row* (*r*) dimana $0 \leq r < 4$, sedangkan kolom mempunyai indeks nomor *column* (*c*) dimana $0 \leq c < Nb$. *Nb* sendiri adalah besarnya blok data dibagi 32.

Pada saat permulaan, input *bit* pertama kali akan disusun menjadi suatu *array byte* dimana panjang dari *array byte* yang digunakan pada *Rijndael* adalah sepanjang 8 *bit* data. *Array byte* inilah yang nantinya akan dimasukkan atau di-*copy* ke dalam *state* dengan urutan :

$$s[r,c] = in[r+4c] \text{ untuk } 0 \leq r < 4 \text{ dan } 0 \leq c < Nb \qquad (2.4)$$

sedangkan dari *state* akan di-*copy* ke *output* dengan urutan :

$$out[r+4c] = s[r,c] \text{ untuk } 0 \leq r < 4 \text{ dan } 0 \leq c < Nb \quad (2.5)$$



Gambar2.9 *Byte Input, Array State,* dan *Byte Output*

Dari gambar 2.9 kita dapat membaca matrik tersebut dari atas kebawah, maka kita akan mempunyai *array* 1 dimensi 32 *bit* sebagai berikut :

$w_0 = out_0out_1out_2out_3 \quad w_1 = out_4out_5out_6out_7$

$w_2 = out_8out_9out_{10}out_{11} \quad w_3 = out_{12}out_{13}out_{14}out_{15}$

Pada algoritma *Rijndael*, jumlah *block cipher*, *block output*, dan *state* adalah 128 *bit* . Dengan besar data 128 *bit* , berarti *Nb* = 4 yang menunjukkan panjang data tiap baris adalah 4 *byte* . Dengan *block cipher* atau *block* data sebesar 128 *bit* , *key* yang digunakan pada algoritma *Rijndael* tidak harus mempunyai besar yang sama dengan *block cipher*. *Cipher key* pada algoritma *Rijndael* bisa menggunakan kunci dengan panjang 128 *bit* , 192 *bit* , atau 256 *bit* . Perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan diimplementasikan pada algoritma *Rijndael* ini. Di bawah ini adalah tabel yang memperlihatkan jumlah *round* (*Nr*) yang harus diimplementasikan pada masing-masing panjang kunci. (Fips197, 2001)

Tabel 2.1 memperlihatkan jumlah *round* (*Nr*) yang harus diimplementasikan pada masing-masing panjang kunci.

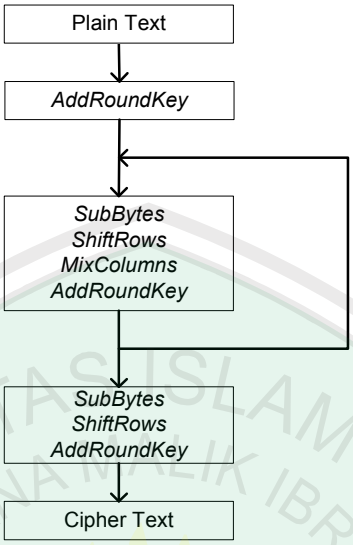
	Jumlah Key (Nk)	Besar Block (Nb)	Jumlah Round (Nr)
AES – 128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

Tabel 2.1 Perbandingan jumlah *Round* dan *Key*

Dari tabel 2.1 terlihat bahwa *Rijndael*- 128*bit* menggunakan panjang kunci N_k = 4 *word* yang setiap *word*-nya berisi 32*bit* sehingga total total kuncinya 128*bit*. Ukuran blok masukan 4 *word* (128*bit*), sedangkan jumlah *round*-nya (N_r) sebanyak 10 *round*. *Rijndael*-256 memiliki panjang kunci 256*bit*, blok masukan *plaintext* 128*bit*, dan jumlah *round*-nya 14.

2.3.2 Algoritma Enkripsi *Rijndael*

Proses enkripsi pada algoritma *Rijndael* terdiri dari 4 jenis transformasi *bytes*, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey*. Pada awal proses enkripsi, input yang telah dikopikan ke dalam *state* akan mengalami transformasi *byte AddRoundKey*. Setelah itu, *state* akan mengalami transformasi *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara berulang-ulang sebanyak *Nr*. Proses ini dalam algoritma *Rijndael* disebut sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *state* tidak mengalami transformasi *MixColumns*. (Wibowo, 2004)



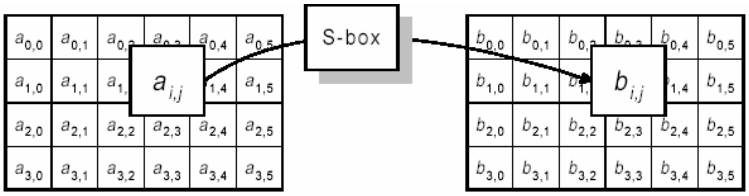
Gambar2.10 Diagram Alir Proses Enkripsi

2.3.2.1 *SubBytes*

SubBytes merupakan transformasi *byte* dimana setiap elemen pada *state* akan dipetakan dengan menggunakan suatu tabel substitusi (*S-Box*).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

Tabel 2.2 Contoh *S-Box*



Gambar 2.11 Proses *SubByte*

Hasil yang didapat dari pemetaan dengan menggunakan tabel *S-Box* ini sebenarnya adalah hasil dari dua proses transformasi *bytes*, yaitu : (Ariyus, 2006)

1. *Invers* perkalian dalam GF(2^8) adalah fungsi yang memetakan 8 *bit* ke 8 *bit* yang merupakan *invers* dari elemen *finite field* tersebut. Suatu *byte* a merupakan *invers* perkalian dari *byte* b bila a•b = 1, kecuali {00} dipetakan ke dirinya sendiri. Setiap elemen pada *state* akan dipetakan pada tabel *invers*. Sebagai contoh, elemen "01010011" atau {53} akan dipetakan ke {CA} atau "11001010".

2. Transformasi *affine* pada *state* yang telah dipetakan. Transformasi *affine* ini apabila dipetakan dalam bentuk matriks adalah sebagai berikut :

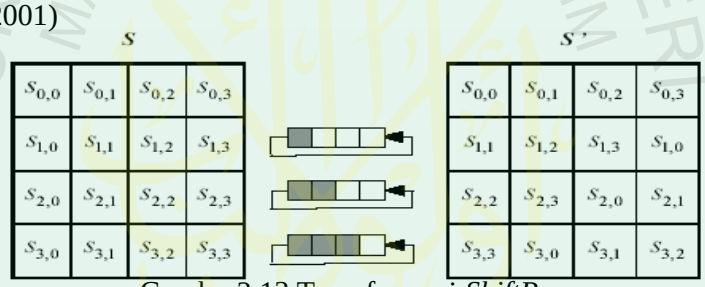
$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Gambar2.12 Matriks *Affine*

$b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$ adalah urutan *bit* dalam elemen *state* atau *array byte* dimana b_7 adalah *most significant bit* atau *bit* dengan posisi paling kiri.

2.3.2.2 *ShiftRows*

Transformasi *Shiftrows* pada dasarnya adalah proses pergeseran *bit* dimana *bit* paling kiri akan dipindahkan menjadi *bit* paling kanan (rotasi *bit*). Transformasi ini diterapkan pada baris 2, baris 3, dan baris 4. Baris 2 akan mengalami pergeseran *bit* sebanyak satu kali, sedangkan baris 3 dan baris 4 masing-masing mengalami pergeseran *bit* sebanyak dua kali dan tiga kali. (Fips197, 2001)



Gambar2.13 Transformasi *ShiftRows*

2.3.2.3 *MixColoumns*

Mixolumns mengoperasikan setiap elemen yang berada dalam satu kolom pada *state*. Elemen pada kolom dikalikan dengan suatu *polynomials* tetap $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. Secara lebih jelas, transformasi *mixcolumns* dapat dilihat pada perkalian matriks berikut ini : (Fips197, 2001)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar2.14 Matriks Transformasi *MixColumns*

Melakukan proses penambahan pada operasi ini berarti melakukan operasi *bitwise* XOR. Maka hasil dari perkalian matriks diatas dapat dianggap seperti perkalian yang ada di bawah ini (http://en.wikipedia.org/wiki/*Rijndael*_mix_columns, 2007)

$$\begin{aligned} s'_{0,c} &= (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}) \end{aligned} \quad (2.6)$$

2.3.2.4 *AddRoudkey*

Pada proses *AddRoundKey*, suatu *round key* ditambahkan pada *state* dengan operasi *bitwise* XOR. Setiap *round key* terdiri dari Nb *word* dimana tiap *word* tersebut akan dijumlahkan dengan *word* atau kolom yang bersesuaian dari *state* sehingga :

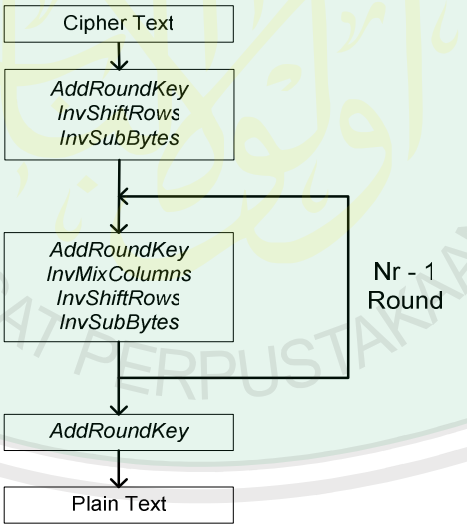
$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [w_{round*Nb+c}]$$

untuk 0d c < Nb (2.7)

Dimana [w_i] adalah *word* dari *key* yang bersesuaian dimana *i* = *round*Nb+c*. Transformasi *AddRoundKey* diimplementasikan pertama kali pada *round* = 0, dimana *key* yang digunakan adalah *initial key* (*key* yang dimasukkan oleh *cryptografer* dan belum mengalami proses *key expansion*). (Wibowo, 2004)

2.3.3 Algoritma Dekripsi *Rijndael*

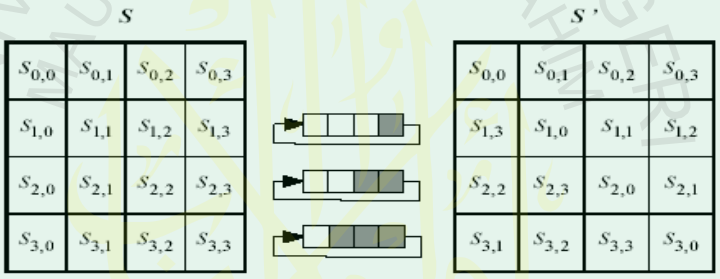
Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma *Rijndael*. Transformasi *byte* yang digunakan pada *invers cipher* adalah Inv*ShiftRows*, Inv*SubBytes*, Inv*MixColumns*, dan *AddRoundKey*. (Ariyus, 2006)



Gambar2.15 Diagram Alir Proses Dekripsi

2.3.3.1 Inv*ShiftRows*

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi ShiftRows. Pada transformasi Inv*ShiftRows*, dilakukan pergeseran bit ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran bit ke kiri. Pada baris kedua, pergeseran bit dilakukan sebanyak 3 kali, sedangkan pada baris ketiga dan baris keempat, dilakukan pergeseran bit sebanyak dua kali dan satu kali. (Kurniawan, 2004)



Gambar2.16 Transformasi *InvShiftRows*

2.3.3.2 Inv*SubBytes*

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada Inv*SubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan tabel *inverse S-Box*. Tabel ini berbeda dengan tabel *S-Box* dimana hasil yang didapat dari tabel ini adalah hasil dari dua proses yang berbeda urutannya, yaitu transformasi *affine* terlebih dahulu, baru kemudian perkalian *invers* dalam GF(2^8). (www.iaik.tugraz.at, 2008)

$$\begin{bmatrix} b_7' \\ b_6' \\ b_5' \\ b_4' \\ b_3' \\ b_2' \\ b_1' \\ b_0' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Gambar2.17 Matriks Invers*Affine*

Perkalian *invers* yang dilakukan pada transformasi InvSubBytes ini sama dengan perkalian *invers* yang dilakukan pada transformasi SubBytes.

		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

Tabel 2.3 Inverse *S-Box*

2.3.3.3 Inv*MixColumns*

Pada Inv*MixColumns*, kolom-kolom pada tiap *state* (*word*) akan dipandang sebagai *polynomials* atas GF(2^8) dan mengalikan *modulo* $x^4 + 1$ dengan *polynomials* tetap $a^{-1}(x)$ yang diperoleh dari : (Daemen dan Rijmen, 1998)

$$a^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}. \quad (2.8)$$

Atau dalam matriks :

$$s'(x) = a(x) \otimes s(x) \quad (2.9)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar2.18 Matriks Inv*MixColumns*

Hasil dari perkalian diatas adalah : (http://en.wikipedia.org/wiki/*Rijndael*_mix_columns, 2007)

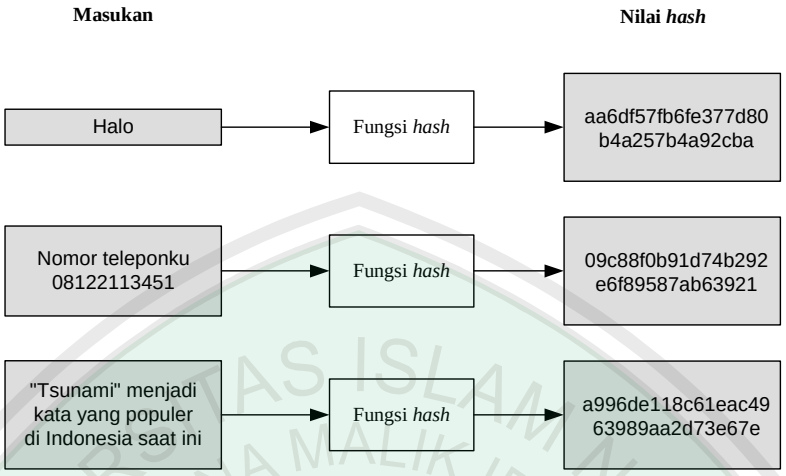
$$s'_{0,c} = (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,c} = (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c})$$
$$s'_{2,c} = (\{0D\} \bullet s_{o,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0E\} \bullet s_{2,c}) \oplus (\{0B\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{0B\} \bullet s_{o,c}) \oplus (\{0D\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0E\} \bullet s_{3,c})$$

2.3.3.4 *Inverse AddRoundKey*

Transformasi *Inverse AddRoundKey* tidak mempunyai perbedaan dengan transformasi *AddRoundKey* karena pada transformasi ini hanya dilakukan operasi penambahan sederhana dengan menggunakan operasi *bitwise* XOR. (Wibowo, 2004)

2.4 Fungsi *Hash*

Fungsi *hash* adalah: fungsi yang menerima masukan *string* yang panjangnya sembarang, lalu mentransformasikannya menjadi *string* keluaran yang panjangnya tetap (*fixed*) (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula). (Kurniawan, 2004)



Gambar 2.19 Contoh *Hash*

Fungsi *hash* satu-arah (*one-way function*) adalah: fungsi *hash* yang bekerja dalam satu arah. Pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula (*irreversible*) (Munir, 2006).

Persamaan fungsi *hash*:

$h = H(M)$ (2.10)

M = pesan ukuran sembarang

h = nilai *hash* (*hash value*) atau Pesan-ringkas (*message-digest*)

h <<<< *M*

Sifat-sifat fungsi *Hash* (*H*) satu arah adalah sebagai berikut (Munir, 2006):

1. Fungsi . *H* dapat diterapkan pada blok data berukuran berapa saja.
2. *H* menghasilkan nilai (*h*) dengan panjang tetap (*fixed-length output*).
3. *H* (*x)* mudah dihitung untuk setiap nilai x yang diberikan.

4. Untuk setiap *h* yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga $H(x) = h$. Itulah sebabnya fungsi *H* dikatakan fungsi *hash* satu-arah (*one way hash function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga $H(y) = H(x)$.
6. Tidak mungkin mencari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.
7. Ada beberapa fungsi *hash* satu-arah yang sudah dibuat orang, antara lain:

Beberapa contoh algoritma dari fungsi *Hash* satu arah : *MD2*, *MD4*, *MD5*, *Secure Hash Function* (*SHA*), *N-hash*, *RIPE-MD*, *Tiger*, *Haval* dan lain-lain.

Algoritma	**Ukuran (*bit*)**	**Ukuran Blok (*bit*)**	**Kolisi**
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD-128	128	512	Ya
RIPEMD-160	160	512	Tidak
SHA	160	512	Ya
SHA-1	160	512	Cacat
SHA-256	256	512	Tidak
SHA-512	512	1024	Tidak
WHIRPOOL	512	512	Tidak

Tabel 2.4 Perbandingan fungsi *Hash*

Manfaat dari aplikasi yang menggunakan fungsi *Hash* satu arah : (Munir, 2006)

1 Menjaga integritas data

- Fungsi *hash* sangat peka terhadap perubahan 1 bit pada pesan
- Pesan berubah 1 bit, nilai *hash* berubah sangat signifikan.
- Bandingkan nilai *hash* baru dengan nilai *hash* lama. Jika sama, pesan masih asli. Jika tidak sama, pesan sudah dimodifikasi.

2. Menghemat waktu pengiriman.

- Misal untuk memverifikasi suatu salinan arsip dengan arsip asli.
- Salinan dokumen berada di tempat yang jauh dari basisdata arsip asli
- Ketimbang mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih mangkus mengirimkan *message digest*-nya.
- Jika *message digest* salinan arsip sama dengan *message digest* arsip asli, berarti salinan arsip tersebut sama dengan arsip master.

3. Menormalkan panjang data yang beraneka ragam.

- Misalkan *password* panjangnya bebas (minimal 8 karakter)
- *Password* disimpan di komputer *host* (*server*) untuk keperluan otentikasi pemakai komputer.
- *Password* disimpan di dalam basisdata.
- Untuk menyeragamkan panjang *field password* di dalam basisdata, *password* disimpan dalam bentuk nilai *hash* (panjang nilai *hash* tetap).

2.5 Keamanan Menurut Islam Dan Integrasinya Dengan Enkripsi

"Dialah Allah yang tiada Tuhan selain Dia, raja, yang Maha suci, yang Maha Sejahtera, yang Mengaruniakan Keamanan, yang Maha Memelihara, yang Maha Perkasa, yang Maha Kuasa, yang memiliki segala Keagungan, Maha Suci Allah dari apa yang mereka persekutukan." (QS : Al Hasyr : 23).

Stabilitas keamanan sangat erat hubungannya dengan keimanan. Ketika keimanan lenyap, niscaya keamanan akan tergoncang. Dua unsur ini saling mendukung. Allah berfirman. (http://konsultasi.wordpress.com/2007/01/18/negara-islam-seperti-apa/, 2007)

" *Orang-orang yang beriman dan tidak mencampuradukkan iman mereka dengan dengan kezhaliman, mereka itulah orang-orang yang mendapatkan keamanan, dan mereka itu adalah orang-orang yang mendapat petunjuk"* [Al-An'am : 82]

Bagaimana mungkin seorang muslim dapat melaksanakan amalan sesuai dengan tuntunan petunjuk, jika ia merasa takut. Begitu pentingnya, sampai-sampai Nabi Ibrahim memohon kepada Allah curahan keamanan sebelum meminta kemudahan rizki. Sebab orang yang didera rasa takut, tidak akan bisa menikmati lezatnya makan dan minum. Allah menceritakan permohonan Nabi Ibrahim dalam ayat.

"*Dan (ingatlah) ketika Ibrahim bedo'a : Wahai, Rabbku, jadikanlah negeri ini negeri aman sentausa dan berikanlah rizki dari buah-buahan kepada*

penduduknya yang beriman diantara mereka kepada Allah dan hari kemudian".[Al-Baqarah : 126]

Secara eksplisit, beliau mendahulukan permohonan keamanan daripada permohonan rizki. Dari sini, generasi Salaf telah memaklumi betapa mahal nilai keamanan. Sesungguhnya Allah benar-benar telah memberikan anugerah besar kepada bangsa Arab, (yaitu) dengan menjadikan tanah mereka sebagai tanah haram (suci), membebaskan mereka dari rasa ketakutan, memberi makan mereka dari kelaparan. Allah berfirman.

"*Maka hendaklah mereka menyembah Rabb pemilik rumah ini (Ka'bah) yang telah memberi makanan kepada mereka untuk menghilangkan lapar dan mengamankan mereka dari ketakutan*". [Quraisy : 3-4]

Ada sekian mekanisme yang ditempuh berbagai negara demi terciptanya keamanan. Sebagian negara mempraktekkan bahasa pukulan, penganiayaan dan memaksakan kehendak kepada rakyat demi mengais kemanan. Pendekatan ini dikenal dengan diktatorisme. Sebaliknya, ada negara mengira dapat meraih keamanan dengan melepaskan kendali dan membebaskan para penjahat dan orang-orang perusak norma dengan slogan liberalisme. Negara lain mencoba merengkuh keamanan dengan pemanfaatan teknologi mutakhir dalam mendeteksi dan mengejar para pelaku kriminal. (http://konsultasi.wordpress.com/2007/01/18/negara-islam-seperti-apa/, 2007). Keamanan dalam dunia komputer dapat di buat salah satunya dengan metode enkripsi. Karena hidup didunia, manusia selalu membutuhkan keamanan.

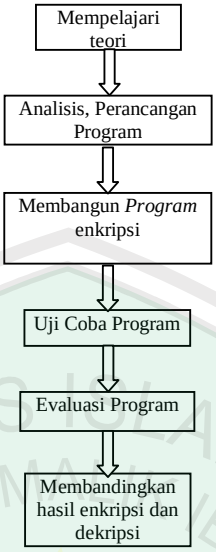
BAB III

METODE DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian *"Rancang Bangun Aplikasi Enkripsi File Menggunakan Algoritma Rijndael"*

Penelitian dilakukan dengan tahapan – tahapan berikut ini :

1. Mempelajari *cryptography* dan hal-hal yang berhubungan dengan algoritma *Rijndael*,. Metode–metode tersebut telah dijelaskan pada bab 2.
2. Menganalisa dan merancang sistem enkripsi dengan menggunakan Algoritma *Rijndael*.
3. Membuat sistem berdasarkan analisis dan perancangan yang telah dilakukan.
4. Uji coba sistem dengan menggunakan berbagai macam file. Seperti; *.txt, *.jpg, *.ppt dan lain-lain (*.*)
5. Evaluasi hasil enkripsi dan dekripsi dengan menggunakan sistem yang telah di buat, apakah sesuai dengan algoritma *rijndael*.
6. Membandingkan hasil enkripsi dan dekripsi. Baik itu dari segi waktu, dan tingkat keamanannya.



Gambar 3.1 Langkah-langkah penelitian

3.1 Dekripsi Umum Sistem

Tipe *file* inputan sistem yang akan dimasukkan adalah *all file* (*.*). Tipe *Key* untuk proses enkripsi dan dekripsi yang akan digunakan merupakan hasil dari fungsi *hash,* sedangkan tipe *file* hasil dekripsi adalah *all file* (*.*) dalam arti tipe dapat dipilih sesuai dengan keinginan *user*.

Rancangan fungsi *hash* yang akan digunakan menggunakan metode : *HAVAL, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA256,SHA512* dan *TIGER*. Hal ini dimasukkan untuk memberikan pilihan kunci kepada *user* dalam mengenkrip *file*. Ditinjau dari teori pada bab2 bahwa algoritma *Rijndael* merupakan algoritma yang bersifat terbuka dan siapa pun boleh mengetahui dan mempelajarinya. Keamanan *rijndael* terletak dari panjang kunci yang digunakan, yang dalam rancangan sistem ini akan di bangun berbagai macam algoritma *hash* sebagai kunci tersebut.

3.2 Perancangan Sistem

Berdasarkan analisis yang telah dilakukan, berikut ini akan dibahas mengenai arsitektur dan proses yang terjadi pada sistem enkripsi data yang akan dibangun ini.

3.2.1 Perancangan Proses

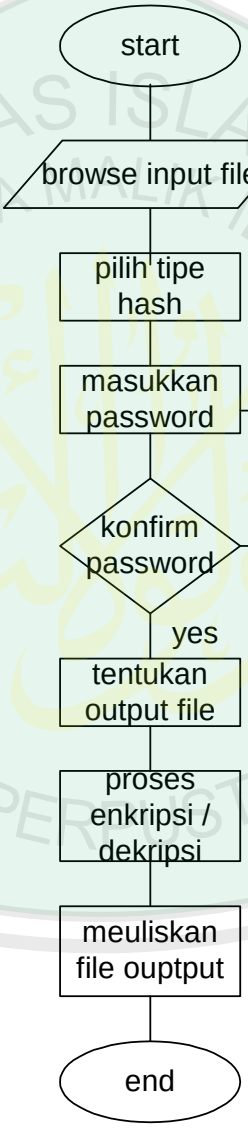
File yang akan diinputkan dalam sistem ini adalah *all file* maksudnya *file* dengan ekstensi apapun, sistem dapat mengenkrip *file* yang inputkan tersebut. Proses pembentukan enkripsi *file*, secara garis besar akan dijelaskan sebagai berikut:

Enkripsi dan dekripsi *file* dengan kunci *string password*

Langkah-langkah proses enkripsi *file* dengan *string password* sebagai berikut :

1. Masukkan *file* yang akan di enkripsi atau dekripsi. Dalam sistem ini semua tipe *file* file dapat diinputkan.
2. Pilih metode *hash* untuk mengacak password. fungsi *hash* yang digunakan dalam sistem ini ada delapan yaitu : *HAVAL, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA256, SHA512* dan *TIGER*. Dimana tiap metode mempunyai algoritma dan panjang data yang berbeda.
3. Masukkan *password*, dan konfirmasi *password*. *password* yang dimasukkan sesuai dengan keinginan *user*.
4. Tentukan direktori tempat menyimpan *file* hasil enkripsi atau dekripsi. Tipe *file* hasil enkripsi atau dekripsi dapat disimpan sesuai keinginan *user*. Misalnya, disimpan dalam tipe .doc, .jpg, .wav, .mp3, *.*. *File* hasil tidak boleh sama dengan *file input*.

5. Lakukan proses enkripsi atau dekripsi *file*. *File* hasil enkripsi atau dekripsi akan tersimpan secara langsung dalam direktori yang *user* tentukan (poin 2).
6. Program meuliskan file output ke dalam poin 4.

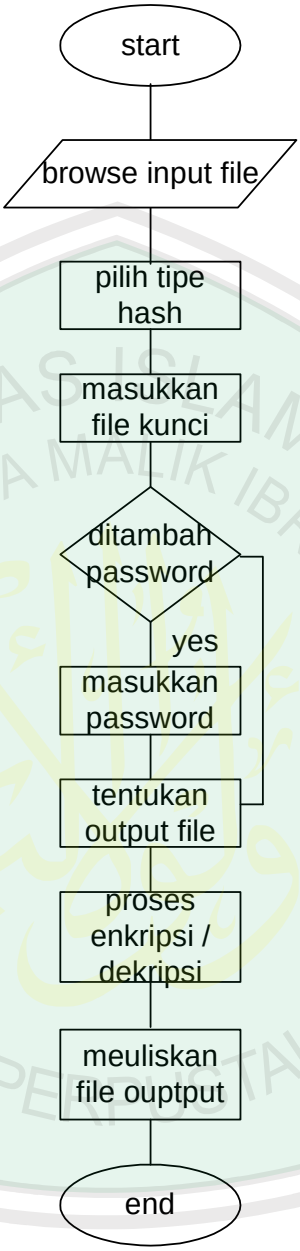


Gambar 3.2 Diagram alir enkripsi atau dekripsi *file* dengan *string password*

Enkripsi atau dekripsi *file* dengan *file* kunci

Langkah-langkah proses enkripsi *file* dengan *file* kunci sebagai berikut :

1. Masukkan *file* yang akan di enkripsi atau dekripsi. Dalam sistem ini semua tipe *file* file dapat diinputkan.
2. Pilih metode *hash* untuk mengacak *file* kunci. fungsi *hash* yang digunakan dalam sistem ini ada delapan yaitu : *HAVAL, MD4, MD5, RIPEMD-128, RIPEMD-160, SHA256, SHA512* dan *TIGER*. Dimana tiap metode mempunyai algoritma dan panjang data yang berbeda.
3. Masukkan *file* kunci. Dalam sistem ini semua tipe *file* dapat dimasukkan. *File* kunci tidak boleh sama dengan *file input* dan *file output.*
4. Terdapat pilihan untuk menggabungkan *file* kunci dengan *string password*. apabila pilihan ini tidak dipilih maka sistem hanya menggunakan *file* kunci.
5. Tentukan direktori tempat menyimpan *file* hasil enkripsi atau dekripsi. Tipe *file* hasil enkripsi atau dekripsi dapat disimpan sesuai keinginan *user*. Misalnya, disimpan dalam tipe .doc, .jpg, .wav, .mp3, *.*. *File* hasil tidak boleh sama dengan *fiel input*.
6. Lakukan proses enkripsi atau dekripsi *file*. *File* hasil enkripsi atau dekripsi akan tersimpan secara langsung dalam direktori yang *user* tentukan (poin 2).
7. Program meuliskan file output ke dalam poin 5.

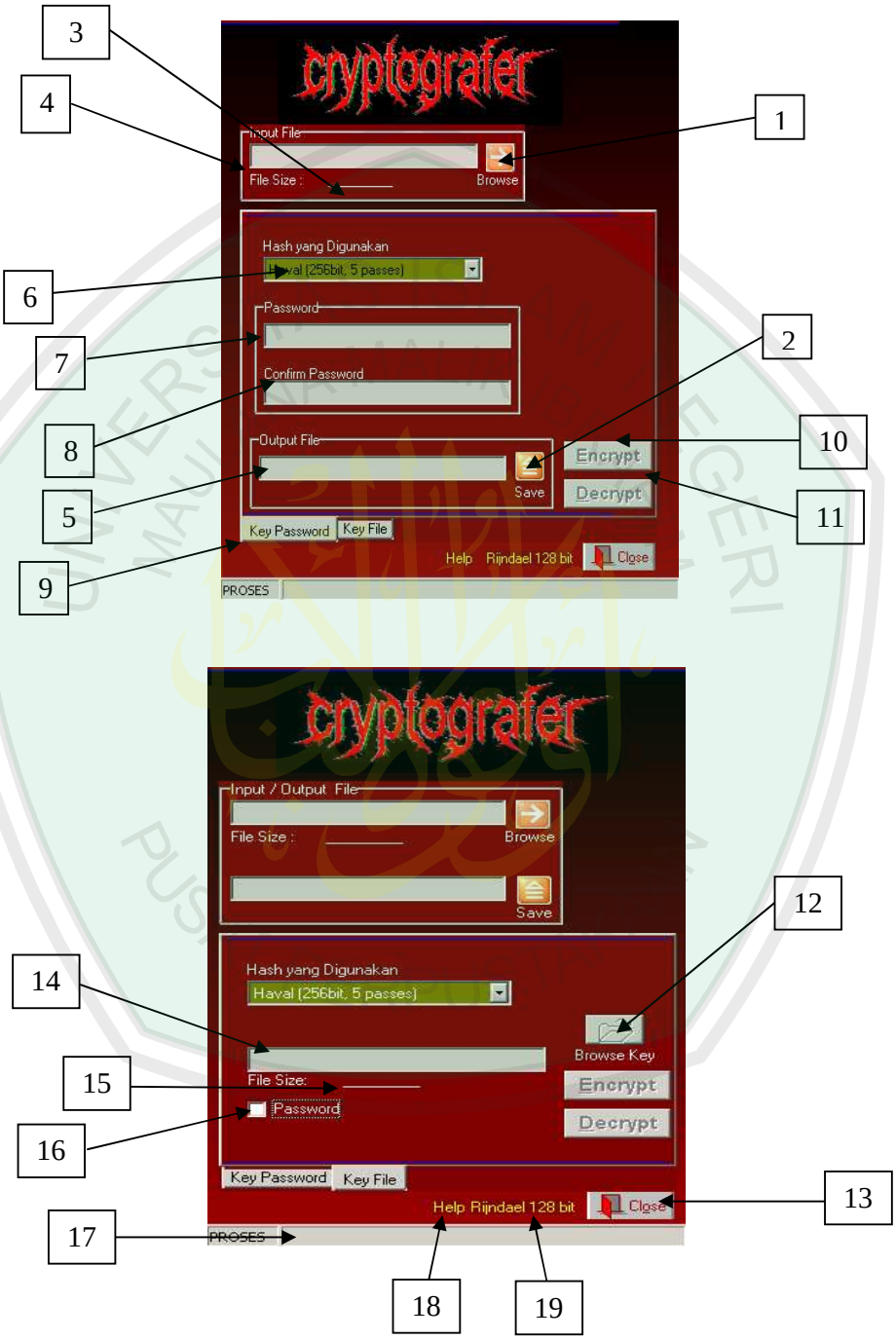


Gambar 3.3 Diagram alir enkripsi atau dekripsi *file* dengan *file* kunci

3.2.2 Perancangan Antarmuka

Berdasarkan hasil analisis secara keseluruhan terdapat beberapa bagian yang dibutuhkan dalam antarmuka sistem enkripsi *file rijndael,* yaitu :

1. Bagian untuk memasukkan *file input*.
2. Bagian untuk memasukkan *file output*.
3. Bagian yang menerangkan *size file input.*
4. Bagian yang menerangkan *path file input.*
5. Bagian yang menerangkan *path file output.*
6. Bagian untuk pilihan metode *hash*.
7. Bagian untuk memasukkan *string password.*
8. Bagian untuk konfirmasi *password.*
9. Bagian untuk pilihan tipe *key*.
10. Bagian untuk melakukan proses enkripsi.
11. Bagian untuk melakukan proses dekripsi.
12. Bagian untuk *browse file* kunci.
13. Bagian untuk keluar dari program.
14. Bagian yang menerangkan *path file* kunci *file.*
15. Bagian yang menerangkan *size file* kunci.
16. Bagian untuk pilihan menggambungkan *file* kunci dengan *password.*
17. Bagian yang menerangkan jalan nya proses.
18. Bagian untuk menampilkan *form help.*
19. Bagian untuk menampilkan *form about me*.



Gambar 3.4 Gambar Rancangan Antarmuka

3.2.3 Algoritma Program

Operasi enkripsi *Rijndael* dapat dinyatakan dengan kode semu (*pseudo code*) berikut ini :

```
Cipher (byte in [4*Nb], byte out [4*Nb], word w[Nb*(Nr+1)]) //nama fungsi
Begin
Byte state [4, Nb]
State = in                          //masukkan input
      AddRoundkey (state, w)
            For Nr = 1 to Nr -1
                  SubByte (state)
                  ShiftRows (state)
                  MixColoumns (state)
                  AddRoundkey (state, w + round *Nb )
            End
      SubByte (state)           //proses terakhir
      ShiftRows(state)
AddRoundkey (state, w)
Out = state                         //output
End
```

(Kurniawan, 2004)

Dari *Pseudo code* diatas dapat kita menarik kesimpulan bahwa enkripsi dilakukan dengan fungsi *cipher* yang memiliki parameter masukkan in = 16 *byte*, keluaran out = 16 *byte* dan *array* 1 dimensi w 44 *byte* untuk *Rijndael-128bit*. Proses yang dilakukan pada setiap *round* identik (dari Nr ke 0 sampai Nr-1), kecuali untuk *round* terakhir Nr. Proses yang identik itu terdiri dari *SubByte(),*

ShiftRows(), *MixColoumns*() *and AddRoundKey*(). Sedangkan pada *round* terakhir(Nr) tidak dilakukan proses *MixColoumns*(). (Wikipedia, 2007)

Operasi dekripsi *rijndael* dapat dituliskan dengan *pseudo code* / kode semu beikut ini :

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5.1

   for round = Nr-1 step -1 downto 1
     InvShiftRows(state)                     // See Sec. 5.3
     InvSubBytes(state)                      // See Sec. 5.3
     AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
     InvMixColumns(state)                    // See Sec. 5.3
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

(www.adeptscience.co.uk, 2008)

3.3 Perancangan Uji Coba

Pada subbab ini akan dilakukan perancangan uji coba dari sistem enkripsi ini, baik pengujian terhadap sistem apakah metode telah sesuai dengan perancangan, maupun evaluasi yang dihasilkan. Hasil enkripsi akan dievaluasi berdasarkan teori yang ada pada bab2.

3.3.1 Bahan Pengujian

Bahan dasar yang akan digunakan untuk pada proses pengujian ini, yaitu *file* *.txt. Karena file ini mempunyai *size* yang kecil sehingga sistem cepat dalam melakukan proses. Namun *file* dengan tipe lain juga menjadi acuan bahan pengujian. Yaitu, *.mp3, *.jpg, *.exe *.*

3.3.2 Tujuan Pengujian

Beberapa hal yang menjadi tujuan dari pelaksanaan pengujian terhadap sistem ini, yaitu :

1. Memeriksa kesesuaian hasil implementasi dengan hasil analisis dan perancangan.
2. Memeriksa perangkat lunak apakah telah berjalan baik (tidak terjadi *error*).
3. Mengevaluasi hasil sistem, dengan menghitung kecepatan enkripsi dan dekripsi.

BAB IV

IMPLEMENTASI DAN UJI COBA

Implementasi merupakan proses transformasi representasi rancangan ke bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini akan dibahas hal-hal yang berkaitan dengan implementasi sistem enkripsi *file*

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dipaparkan disini meliputi lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem enkripsi ini adalah sebagai berikut :

1. Prosesor Intel Pentium 4, 2.6 GHz
2. RAM 1024 MB
3. *HardDisk* dengan kapasitas 80 GB
4. *VGA* Ati Radeon 9250se
5. Monitor 15”
6. Keyboard
7. Mouse

Notebook Toshiba J10 :

1. Processor Pentium Pentium 2 GHz

2. RAM 750 Mbyte

3. Monitor 15”

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem enkripsi *file* ini adalah :

1. Sistem Operasi Windows Xp Service pack 2.
2. Delphi 7.0
3. Install Shield 5.0 *include on* Delphi 7.
4. TafreNenc *cryptographic component.*

4.2 Implementasi Sistem

Sistem yang akan di implementasikan menggunakan bahasa pemrograman *pascal* / Delphi 7.0 sebagai antar muka *platform.*

4.2.1 *Procedure* untuk memasukkan *file input*

```
procedure TForm1.btnInputBrowseClick(Sender: Tobject
begin
OpenDialog1.Execute;
inFile.Text:= OpenDialog1.FileName;
inFile.OnExit(inFile);
if not FileExists(InFile.Text) then
  begin
    MessageDlg('File Tidak Ada!',mtInformation,[mbOK],0);
    inFile.Text:='';
    Exit;
  end;
end;
```

Dalam *procedure* btnInputBrowseClick memiliki fitur-fitur sebagai berikut :

- *OpenDialog1* sebagai komponen untuk *browse file input.*

- *Infile* adalah *textfield* untuk menampilkan *path file input.*
- *MessageDlg* sebagai konfirmasi kepada *user*.

4.2.2 *Procedure* untuk meuliskan *file output*

```
procedure TForm1.btnOutputBrowseClick(Sender: TObject);
begin
  if inFile.Text=''   //textfile input kosong
  then   begin
  ShowMessage('    Silahkan Masukkan File Input!!');
  OpenDialog1.Execute;
  inFile.Text:= OpenDialog1.FileName;
  inFile.OnExit(inFile);
  end;

  if
  SaveDialog1.Execute
  then
  outFile.Text := SaveDialog1.FileName;

   if fileexists (outFile.Text) then
if
MessageDlg('  Over Write
File??',mtConfirmation,[mbYes,mbNo],0)=mrNo then
begin outFile.Text:='';
end;
   if (inFile.Text='') and (outFile.Text='')
  then
    begin
    ShowMessage('       Lakukan Pengisian Dengan Benar!!');
    OpenDialog1.Execute;
    inFile.Text:= OpenDialog1.FileName;
    inFile.OnExit(inFile);
    SaveDialog1.Execute;
    SaveDialog1.DefaultExt:= SaveDialog1.Filter;
    outFile.Text := SaveDialog1.FileName;
    end;
  if inFile.Text=outFile.Text
    then  begin
      ShowMessage('       File Tidak Boleh Sama!!'#13#13'
Lakukan Penyimpanan Ulang');
      outFile.Text:='';
      SaveDialog1.Execute;
SaveDialog1.DefaultExt:= SaveDialog1.Filter;
outFile.Text := SaveDialog1.FileName;
end;
end;
```

Dalam *procedure* OutputBrowseClick memiliki fitur-fitur sebagai berikut :

- *Showmessage* sebagai peringatan kepada *user*
- *Outfile* adalah *textfield* untuk menampilkan *path file output*.
- *SaveDialog1* adalah komponen untuk menyimpan *fieloutput*.
- *OpenDialog1* sebagai komponen untuk *browse file input*.
- *Infile* adalah *textfield* untuk menampilkan *path file input.*
- *MessageDlg* sebagai konfirmasi kepada *user*.

4.2.3 Fungsi Untuk Menghitung Ukuran *File*

```
function AddCommas(const S: string): string;  //fungsi menghitung
ukuran file
var
  i, j: integer;
begin
  i := Length(S) mod 3;
  if ((i <> 0) and (Length(S) > 3)) then
    Result := Copy(S,1,i) + ',';
  for j := 0 to ((Length(S) div 3) - 2) do
    Result := Result + Copy(S,1 + i + j*3,3) + ',';
  if (Length(S) > 3) then
    Result := Result + Copy(S,Length(S) - 2,3)
  else
    Result := S;
end;
```

Fungsi diatas memiliki *integer* i dan j sebagai variabel untuk menghitung panjang *file*.

4.2.4 fungsi Untuk Menampilkan Pilihan *Hash*

```
  for i := 0 to (ComponentCount - 1) do  //membaca Hash ke cbxHash
   if (Components[i] is TDCP_hash) then

cbxHash.Items.AddObject(TDCP_hash(Components[i]).Algorithm,Compone
nts[i]);
```

```
  for o:= 0 to (ComponentCount - 1) do   //membaca Hash ke
cbxHashFile
   if (Components[o] is TDCP_hash) then

CbxHashFile.Items.AddObject(TDCP_hash(Components[o]).Algorithm,Com
ponents[o]);
```

Fitur-fitur yang ada dalam fungsi menampilkan pilihan *hash* :

- *TDCPhash* adalah kompnen *hash* dari *afreNenc*.
- *CbxHash* adalah combobox sebagai antar muka menampilkan pilihan.

4.2.5 *Procedure* Untuk Menampilkan Pilihan Tipe Kunci

```
procedure PageControl1Change(Sender: TObject);
```

Didalam *procedurepageControl1Change* diatas memiliki 2 *TabSheet*. Dimana

Tabsheet1 merupakan untuk pilihan kunci dengan *string password*.

Tabsheet2 merupakan untuk pilihan kunci dengan *file* kunci.

4.2.6 *Procedure* Untuk *Browse File* Kunci

```
procedure  TForm1.BrowseKeyClick(Sender:  TObject);       //pilih
keyFile
begin
if CbxHashFile.Text<>''
  then begin
  OpenkeyFile.Execute;
  KeyFile.Enabled:=true;
  KeyFile.Text:= OpenkeyFile.FileName;
  KeyFile.OnExit(KeyFile);
  end;
    if (KeyFile.Text=inFile.Text) or (KeyFile.Text=outFile.Text)
  then begin
    ShowMessage('      Key  File  Tidak  Boleh  Sama  dengan  File
I/O..!!!');
    KeyFile.Text:='';
    CbxHashFile.SetFocus;

    end;
  if not FileExists(KeyFile.Text) then
  begin    MessageDlg('File Tidak Ada!',mtInformation,[mbOK],0);
    KeyFile.Text:='';
```

```
    Exit;
    KeyFile.SetFocus;
    end;
end;
```

Dalam *procedure* diatas memiliki fitur-fitur :

- CbxHashFile adalah combobox untuk menampilkan pilihan *hash.*
- *OpenKey* adalah komponen untuk *browse file* kunci.
- *KeyFile* adalah *textfield* untuk menampilkan *path file* kunci.
- Infile adalah *textfield* untuk menampilkan *path file input.*
- *Showmessage* sebagai peringatan kepada *user.*

4.2.7 *Procedure* Untuk Menambahkan *file* Kunci Untuk Menggabungkan Dengan *String Password*

```
procedure  TForm1.CheckBoxPassClick(Sender:  TObject);   //pilihan
penambahan password
begin
  if
   CheckBoxPass.Checked=false
   then
   begin
   PassFile.Visible:=false;
   end;
  if
   CheckBoxPass.Checked=true
   then  begin
        if KeyFile.Text=''
          then
          PassFile.Visible:=false;
        if KeyFile.Text<>''
        then
        begin
        PassFile.Visible:=true;
        PassFile.SetFocus;
        end;
    end;
end;
```

Pada *procedure checkBoxPasssClick* diatas terdapat fitur-fitur :

- *CheckBoxPass* adalah komponen *checkBox*
- *PassFile* adalah *text field* untuk memasukkan *password*.
- *KeyFile* adalah *textfield* untuk menampilkan *path file* kunci.

4.2.8 *Procedure* Untuk Mengacak *File* kunci konversi menjadi *String*

```
procedure TForm1.KeyFileExit(Sender: TObject); //keyFile
var
    HashDigest: array of byte;   // Matriks Hash
    w : integer;
    s: string;
begin
  if (KeyFile.Text = '') then
    label10.Caption := '________'
    else if FileExists(KeyFile.Text) then
      begin
    // jika file ada lakukan fungsi ini..!!
      strmInput := nil;
      try
      strmkey    :=   TFileStream.Create(KeyFile.Text,fmOpenRead);
//membaca Keyfile (namaFile, size 0..127)
      label10.Caption  :=  AddCommas(IntToStr(strmkey.Size))  +  '
bytes';          //menuliskan Ukuran File

     Hash                                                    :=
TDCP_hash(CbxHashFile.Items.Objects[CbxHashFile.ItemIndex]);
//identifikasi Hash
     SetLength(HashDigest,Hash.HashSize div 8);  //membaca panjang
Hash

      Hash.Init;                                          // initialize
it
      Hash.UpdateStream(strmkey,strmkey.Size);             //  Acak
KeyFile dg Hash
      Hash.Final(HashDigest[0]);                          // Hasil
Hash

      s:= '';
      for w:= 0 to Length(HashDigest) -1 do
        s:=         s         +         IntToHex(HashDigest[w],2);
//hexatoint to string
       Label8.Caption:=                                          s;
//tampilkan hasil
    strmkey.Free;
//kosongkan strmkey


  except                    //kesalahan user yg mungkin terjadi
  MessageDlg('  file Error!!',mtError,[mbOK],0);
  strmkey.Free;
```

```
  label10.Caption := 'File tidak bisa di baca!!';
  end;
 end;
end;
```

Pada *procedure* diatas mempunyai fitur-fitur sebagai berikut :

- Variabel *HashDigest: array of byte* adalah Matriks *Hash*, w adalah : *integer*, s adalah *string.*
- *KeyFile* adalah *textfield* untuk menampilkan *path file* kunci.
- *StrmInput* adalah variabel untuk membaca *file input.*
- *Strmkey* adalah variabel untuk membaca *file* kunci.
- *Label10* adalah komponen untuk menampilkan *size file* kunci.
- *TDCPhash* adalah kompnen *hash* dari *afreNenc*.
- *MessageDlg* sebagai konfirmasi kepada *user*.

4.2.9 *Procedure* Untuk Konfirmasi *Password*

```
procedure TForm1.Edit2Exit(Sender: TObject);    //textfile
confirmPassword
begin
 if Edit1.Text<> Edit2.Text
 then
 begin
   MessageDlg('Password Harus Sama !!',mtError,[mbOK],0);
    btnDecrypt.Enabled:=false ;
    encrypt.Enabled:=false;
    Edit1.SetFocus;
    end;
end;

procedure TForm1.Edit2Change(Sender: TObject);
begin
  if (Edit1.Text= Edit2.Text) and (inFile.Text<>'')and
(outFile.Text<>'')
    then
   begin
    btnDecrypt.Enabled:=true ;
    encrypt.Enabled:=true;
    encrypt.SetFocus;
    end;
```

```
end;

procedure TForm1.Edit2Enter(Sender: TObject);
begin
  if
  Edit1.Text=''
  then begin
  ShowMessage('Password Pertama Harus Diisi!!');
  Edit1.SetFocus;
end;
end;
```

Pada *procedure- procedure* diatas mempunyai fitur-fitur sebagai berikut :

- Edit1 adalah *textfield* untuk memasukkan *password.*
- Edit2 adalah *textfield* untuk konfirmasi ulang *password.*
- Btndecrypt adalah komponen tombol untuk melakukan proses dekripsi.

4.2.10 *Procedure* Untuk Proses Enkripsi

```
procedure TForm1.encryptClick(Sender: TObject);    //event tombol
encrypt
var
   HashDigest: array of byte;   // Matriks Hash
   Salt: array[0..7] of byte;    // Matriks bilangan acak untuk
mencegah serangan lawan
   i: integer;
   interval : integer;           // interval progressBar
   CipherIV: array of byte;     // initialisation vector /IV (CBC
model)
begin
  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;
  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);

  try
    waktuMulai := DateTimeToTimeStamp(time);              //mulai
menghitung waktu
    strmInput    :=    TFileStream.Create(InFile.Text,fmOpenRead);
//membaca input file
    strmOutput    :=    TFileStream.Create(OutFile.Text,fmCreate);
//menulis output file

    Hash  :=  TDCP_hash(cbxHash.Items.Objects[cbxHash.ItemIndex]);
//memilih hash dari combobox
```

```
    SetLength(HashDigest,Hash.HashSize           div           8);
//identifikassi panjang hash (div8: 1byte=8bit)

    for i := 0 to 7 do                    //perulangan untuk tiap
byte
      Salt[i] := Random(256);   // bilangan acak (random values)
crypto secure Pseudo Random Generator (CSPRNG)
      strmOutput.WriteBuffer(Salt,Sizeof(Salt));  // membuat Salt
dalam FileOut untuk proses decrypt

      Hash.Init;
      Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
      Hash.UpdateStr(Edit1.Text);  // Hash Password
      Hash.Final(HashDigest[0]);            // hasil akhir Hash ke
HashDigest

        rijndael:= TDCP_rijndael.Create(Self);      //identifikasi
rijndael
        SetLength(CipherIV,rijndael.BlockSize       div       8);
//identifikassi panjang rijndael (div8: 1byte=8bit)
        for   i   :=   0   to   (Length(CipherIV)   -   1)   do
//perulangan untuk tiap block
        CipherIV[i] := Random(256);           // Bilangan acak
untuk Initial Vektor/IV
        strmOutput.WriteBuffer(CipherIV[0],Length(CipherIV));   //
membuat IV dalam FileOutput untuk proses decrypt


rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // identifikasi key

rijndael.EncryptStream(strmInput,strmOutput,strmInput.Size);    //
encrypt file
        rijndael.Burn;   // mengambil dan menyimpan Key

        strmInput.Free; //mengosongkan strminput
        strmOutput.Free;//mengosongkan strmoutput
        waktuAkhir  :=  DateTimeToTimeStamp(time);  //akhir  waktu
proses
        totalWaktu       :=       waktuAkhir.Time-waktuMulai.Time;
//hitung waktu

        MessageDlg('Enkripsi  File  Sukses!!'#13#13'Waktu  :  '  +

IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
```

Procedure diatas mempunyai fitur-fitur sebagai berikut :

- *Encrypt* adalah komponen tombol untuk proses enkripsi.
- Variabel *HashDigest*: array of byte adalah Matriks Hash, *Salt: array[0..7] of byte* adalah : Matriks bilangan acak untuk mencegah

serangan lawan, *i,interval* adalah *integer, CipherIV: array of byte* adlah initialization vector /IV pada model CBC.

- WaktuMulai adalah komponen DateTimeToTimeStamp(time); untuk mulai menghitung waktu proses, waktuAkhir adalah komponen DateTimeToTimeStamp(time) untuk menghitung akhir waktu proses totalWaktu adalah fungsi untuk menghitung total waktu proses.
- *StrmInput* adalah variabel untuk membaca *file input.*
- *StrmOutput* adalah variabel untuk menulis *file input.*
- *TDCPhash* adalah kompnen *hash* dari *afreNenc*.
- *Rijndael* adalah bagian dari komponen *afreNenc* (*TDCP_rijndael*).
- *MessageDlg* sebagai konfirmasi kepada *user*.

4.2.11 *Procedure* Untuk Proses Dekripsi

```
procedure TForm1.btnDecryptClick(Sender: TObject); //event tombol
decrypt!!
var
  Hash: TDCP_hash;              // Hash
  HashDigest: array of byte;    // Matriks Hash
  Salt: array[0..7] of byte;    // bilangan acak untuk mencegah
serangan lawan
  CipherIV: array of byte;      // initialisation vector /IV (CBC
model)
  interval : integer;           // var untuk interval progressBar
begin
  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;
  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);

    try
    waktuMulai := DateTimeToTimeStamp(time);  // penghitung waktu
di mulai
        strmInput:=    TFileStream.Create(InFile.Text,fmOpenRead);
//membaca file input
        strmOutput:=    TFileStream.Create(OutFile.Text,fmCreate);
//menulis file output
```

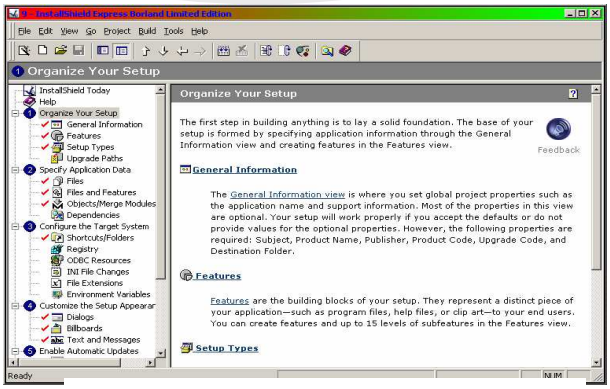
```
    Hash  :=  TDCP_hash(cbxHash.Items.Objects[cbxHash.ItemIndex]);
// memilih Hash dari combo box!!
    SetLength(HashDigest,Hash.HashSize div 8);    // identifikassi
panjang hash (div8: 1byte=8bit)
    strmInput.ReadBuffer(Salt[0],Sizeof(Salt));   //  membaca  Salt
dari File
    Hash.Init;
    Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
    Hash.UpdateStr(Edit1.Text);  // Hash Password
    Hash.Final(HashDigest[0]);           // hasil akhir Hash ke
HashDigest

     rijndael:=   TDCP_rijndael.Create(Self);      //identifikasi
rijndael
     SetLength(CipherIV,rijndael.BlockSize   div   8);         //
identifikassi panjang rijndael (div8: 1byte=8bit)
      strmInput.ReadBuffer(CipherIV[0],Length(CipherIV));       //
Membaca Initial Vektor/IV dari File

rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // Membaca key dari File
     rijndael.CipherMode  :=  cmCBC;         //menggunakan  cipher
model CBC

    rijndael.DecryptStream(strmInput,strmOutput,strmInput.Size  -
strmInput.Position); // Proses decrypt!
    rijndael.Burn;
       strmInput.Free; //mengosongkan strminput
       strmOutput.Free;//mengosongkan strmoutput
       waktuAkhir  :=  DateTimeToTimeStamp(time);  //  akhir  waktu
proses
       totalWaktu      :=      waktuAkhir.Time-waktuMulai.Time;
//menghitung waktu

       MessageDlg('Dekripsi  File  Sukses!!'#13#13'Waktu  :  '  +
IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
       ProgressBar1.Position := 0;
       inFile.Text:='';
       outFile.Text:='';
       Edit1.Text:='';
       Edit2.Text:='';
       cbxHash.Text:='';
       Label1.Caption:='________';
       Edit1.Enabled:= false;
       Edit2.Enabled:= false;
       encrypt.Enabled:= false;
       btnDecrypt.Enabled:= False;
       cbxHash.ItemIndex:=0;
       CbxHashFile.ItemIndex:=0;

       except //kesalahan user yang mungkin terjadi
       MessageDlg('File IO error',mtError,[mbOK],0);
```

- BtnDecrypt adalah komponen tombol untuk proses dekripsi.

- Variabel *HashDigest*: array of byte adalah Matriks Hash, *Salt: array[0..7] of byte* adalah : Matriks bilangan acak untuk mencegah serangan lawan, *i,interval* adalah *integer, CipherIV: array of byte* adalah initialisation vector /IV pada model CBC.
- WaktuMulai adalah komponen DateTimeToTimeStamp(time); untuk mulai menghitung waktu proses, waktuAkhir adalah komponen DateTimeToTimeStamp(time) untuk menghitung akhir waktu proses totalWaktu adalah fungsi untuk menghitung total waktu proses.
- *StrmInput* adalah variabel untuk membaca *file input.*
- *StrmOutput* adalah variabel untuk menulis *file input.*
- *TDCPhash* adalah kompnen *hash* dari *afreNenc*.
- *Rijndael* adalah bagian dari komponen *afreNenc* (*TDCP_rijndael*).
- *MessageDlg* sebagai konfirmasi kepada *user*.

4.2.12 *Install Shield*

Install Shield adalah program yang *include* dengan Delphi 7.0. *Install Shield* digunakan sebagai program untuk membangun program *setup* dari aplikasi hasil *running* yang sudah dibuat dari Delphi.



Gambar 4.1 Antar Muka *Install Shield*

Langkah-langkah membangun *setup* menggunakan *Install Shield* :

1. Jalankan *program Install Shield*. Pilih *create new project.*
2. Pilih *blank setup project.*
3. Pada *menu organize your setup* lakukan input sesuai dengan keinginan *user*.
4. Pada *menu files,* masukkan target *file*.exe* hasil *compile*-an Delphi.
5. Pada *menu configure the target system* pilih *shortcuts/folder* untuk membuat *shortcuts* pada *startmenu*.
6. Pada *menu customize the setup appearance.* Masukkan tipe *setup* sesuai dengan keinginan.
7. *Build* dengan tombol 'F7'.
8. Lihat hasil *build* pada direktori yang telah di setting pada no 1 atau pada *default install shield 'C:\Documents and Settings\afre_N\My Documents\MySetups'*

4.3 Implementasi Antarmuka

Berdasarkan rancangan antarmuka pada subbab 3.2.2 dihasilkan antarmuka berikut. Tampilan berikut terbentuk dari sistem enkripsi *file Rijndael*. Warna yang mayoritas hitam menggambarkan bahwa *cryptography* adalah seni dan teknik penyembunyian pesan.

Gambar 4.2 Hasil program enkripsi *file*

Pada gambar 4.2 mempunyai fitur-fitur waktu proses sedang berjalan :

1. Nama dan alamat *path file input.* (F:\tes\buka hidden.txt)
2. Metode *hash* yang dpilih adalah MD5.
3. *Password* "asa" dan konfirmasi *password* "asa".
4. Nama dan alamat *path file output* (F:\tes\hasil2.txt).
5. Tombol untuk melakukan proses enkripsi.

Gambar 4.3 *file* buka hidden.txt

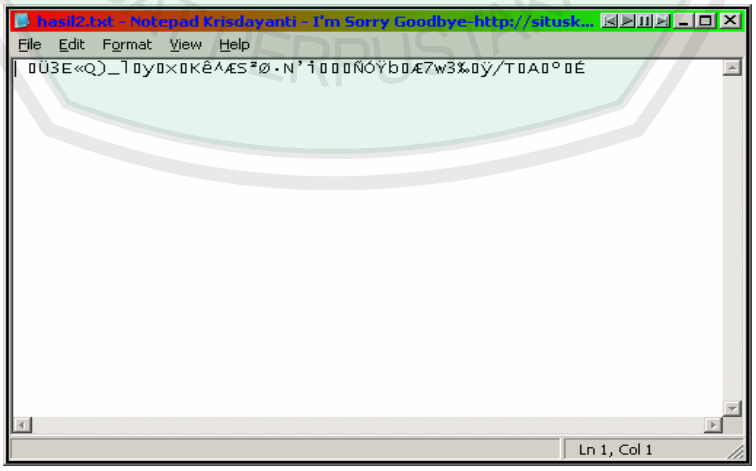
Keterangan isi di dalam *file input* F:\tes\buka hidden.txt :

```
attrib -s -r -h /s /d *.*
```

setelah proses enrkipsi menjadi :

□Ü3E«Q)_l□y□×□Kê^ÆS²Ø·N'i□□□ÑÓŸb□Æ7w3‰□ÿ/T□A□º□É

File ini kemudian disimpan dalam F:\tes\hasil2.txt.



Gambar 4.4 *file* hasil2.txt

Gambar 4.5 Hasil program dekripsi *file*.

Pada gambar 4.5 mempunyai fitur-fitur waktu proses sedang berjalan :

1. Nama dan alamat *path file input*. (F:\tes\hasil2.txt).
2. Metode *hash* yang dpilih adalah MD5.
3. *Password* "asa" dan konfirmasi *password* "asa".
4. Nama dan alamat *path file output* (F:\tes\dekripsihasil2.txt).
5. Tombol untuk melakukan proses dekripsi.

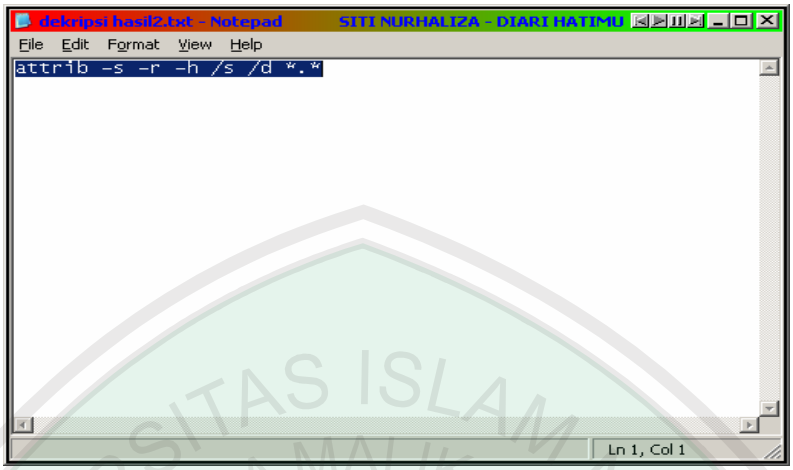
Keterangan isi di dalam *file input* F:\tes\hasil2.txt:

```
□Ü3E«Q)_l□y□×□Kê^ÆS²Ø·N'i□□□ÑÓŸb□Æ7w3‰□ÿ/T□A□º□É
```

setelah proses dekripsi menjadi :

```
attrib -s -r -h /s /d *.*
```

File ini kemudian disimpan dalam F:\tes\dekripsihasil2.txt.



Gambar 4.6 *File* dekripsi hasil2.txt

4.3.1 Implementasi Uji Coba

Uji coba dilakukan bukan hanya pada *file* **.txt*, tetapi dapat di implementasikan pada *all file* (*.*). Sistem dapat melakukan proses enkripsi pada semua jenis tipe *file*,namun *file hasil enkripsi* tidak dapat di buka atau dijalankan oleh aplikasi apapun. Namun dalam proses dekripsi sistem membutuhkan jenis tipe *file* yang asli, serta semua kunci yang diberikan *user* pada waktu proses enkripsi awal.

Pilihan tipe kunci dengan *password* ataupun dengan *file* kunci hanya terletak pada perbedaan jenis kunci saja. Namun, dalam proses nya tetap sama. Yaitu dengan menggunakan *string.*

Jika menggunakan *password* sistem melakukan pengacakkan password menggunakan metode *hash* yang diberikan *user* untuk melakukan proses. Namun pada tipe kunci menggunakan *file* kunci, sistem melakukan konversi terlebih

dahulu dari *stream* ke *string.* Lalu *string* diacak oleh metode *hash* yang diberikan *user*.

Sesuai pada pembahasan bab 2 bahwa *Rijndael* adalah algoritma *block cipher* simetris. Artinya dalam proses enkripsi dan dekripsi *Rijndael* menggunakan kunci yang sama. Maka begitu juga dalam sistem yang telah dibangun ini menggunakan kunci yang sama untuk melakukan proses kembalinya / dekripsi.

Sistem yang telah dibangun di uji coba keamanan nya oleh rekan-rekan programmer Delphi-id.org dan para sahabat di jurusan T.informatika UIN Malang. Hasil uji coba menyatakan bahwa *file* hasil enrkipsi *Rijndael* tidak dapat dibuka, dibaca, atau dijalankan oleh aplikasi manapun.

Berikut tabel-tabel hasil uji coba sistem :

Proses Enkripsi dengan key password

Nama file	Size File	Tipe Hash	Waktu	Size File Hasil
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	0 msec	105 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	47 msec 31 msec 31 msec 47 msec 31 msec 47 msec 47 msec 47 msec	761 KB
Kepastian yang kutungggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	297 msec 234 msec 234 msec 219 msec 266 msec 297 msec 297 msec 266 msec	4251 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	328 msec 265 msec 266 msec 265 msec 297 msec 344 msec 328 msec 328 msec 297 msec	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3203 msec 3219 msec 2578 msec 2531 msec 2829 msec 3125 msec 3110 msec 2812 msec	45,304 KB

Tabel 4.1 Tabel Uji coba enkripsi menggunakan *password.*

Keterangan *Hardware:* CPU Intel Pentium 4 (2.66 GHz)
RAM 1 GB

Proses Enkripsi dengan *File* Kunci

Nama file	Size File	Hash	Lama Proses	Size file Hasil Enkripsi
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	20 ms 10 ms 20ms 10ms 20ms 10ms 10ms 10ms	105 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	51ms 41ms 50ms 50ms 50ms 50ms 70ms 50ms	761 KB
Kepastian yang kutungggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	490ms 241ms 260ms 241ms 280ms 320ms 340ms 310ms	4,251 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	391ms 331ms 280ms 300ms 341ms 370ms 360ms 320ms	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3405ms 2734ms 3215ms 2764ms 3125ms 3275ms 3625ms 3055ms	45,304 KB

Tabel 4.2 Tabel Uji coba enkripsi menggunakan *file* kunci.

Keterangan *hardware* : *Notebook* Toshiba J10

CPU 2 GHz

RAM 750 MByte

Proses Dekripsi dengan key password

Nama file Asli	Size File	Hash	Lama Proses
Itb makalah.doc	105 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	0 msec
Kali.jpg	761 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	47 msec 46 msec 31 msec 32 msec 31 msec 47 msec 47 msec 32 msec
Kepastian yang kutungggu.mp3	4,250 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	281 msec 219 msec 218 msec 219 msec 250 msec 281 msec 282 msec 250 msec
Setup.exe	4,993 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	344 msec 297 msec 282 msec 282 msec 313 msec 344 msec 344 msec 313 msec 313 msec
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3219 msec 2625 msec 2581 msec 2535 msec 2830 msec 3125 msec 3118 msec 2820 msec

Tabel 4.3 Uji Coba Dekripsi Menggunakan *password*.

Keterangan *Hardware:* CPU Intel Pentium 4 (2.66 GHz)
RAM 1 GB

Proses Dekripsi dengan *File* Kunci

Nama file Asli	Size File	Hash	Lama Proses
Itb makalah.doc	105 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	10 msec 10 msec 30 msec 20 msec 30 msec 50 msec 20 msec 30 msec
Kali.jpg	761 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	80 msec 90 msec 60 msec 71 msec 80 msec 90 msec 70 msec 271 msec
Kepastian yang kutungggu.mp3	4,250 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	381 msec 320 msec 320 msec 311 msec 340 msec 410 msec 391 msec 351 msec
Setup.exe	4,993 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	491 msec 381 msec 400 msec 371 msec 461 msec 551 msec 461 msec 421 msec
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval(256bit,5 passes) MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	5438 msec 4636 msec 4517 msec 4717 msec 4757msec 5198 msec 4827 msec 4897msec

Tabel 4.4 Uji Coba Dekripsi Menggunakan *file* kunci.

Keterangan *hardware* : *Notebook* Toshiba J10
CPU 2 GHz
RAM 750 Mbyte

4.4 Evaluasi dan Analisis

Perbandingan dari hasil uji coba sistem dan perbandingan tabel 4.1 sampai 4.4 terdapat beberapa kesimpulan diantaranya :

1. *File* yang dihasilkan antara proses enkripsi dan dekripsi relatif hampir semuanya sama.
2. Lama proses yang dilakukan sistem sangat tergantung dari *hardware* yang digunakan dan proses yang sedang dilakukan oleh sistem operasi.
3. Proses dekripsi yang dilakukan menunjukkan rata-rata perbandingan lebih cepat daripada proses enkripsi.
4. Kunci yang digunakan untuk proses enkripsi dan dekripsi harus sama.
5. *File* yang di hasilkan dari proses enkripsi *all file* *.* yang pengisiannya terserah dari *user*. Hal ini juga menjadi bagian dari penyembunyian tipe *file* asli nya.
6. Tipe *file yang* harus di simpan dari proses dekripsi adalah tipe dari *file* asli sebelum proses enkripsi. Agar *file* yang di hasilkan dapat kembali dibaca.

Nama file	Size File	Hash	Lama Proses Enkripsi	Ukuran File Hasil Enkripsi	Lama Proses Dekripsi	Ukuran File Hasil Dekripsi
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	0 msec	105 KB	0 msec	104 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	47 msec 31 msec 31 msec 47 msec 31 msec 47 msec 47 msec 47 msec	761 KB	47 msec 46 msec 31 msec 32 msec 31 msec 47 msec 47 msec 32 msec	761 KB
Kepastian yang kutungggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	297 msec 234 msec 234 msec 219 msec 266 msec 297 msec 297 msec 266 msec	4,251 KB	281 msec 219 msec 218 msec 219 msec 250 msec 281 msec 282 msec 250 msec	4,250 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	328 msec 265 msec 266 msec 265 msec 297 msec 344 msec 328 msec 328 msec 297 msec	4,993 KB	344 msec 297 msec 282 msec 282 msec 313 msec 344 msec 344 msec 313 msec 313 msec	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3203 msec 3219 msec 2578 msec 2531 msec 2829 msec 3125 msec 3110 msec 2812 msec	45,304 KB	3219 msec 2625 msec 2581 msec 2535 msec 2830 msec 3125 msec 3118 msec 2820 msec	45,304 KB

Tabel 4.5 Tabel Perbandingan Proses menggunakan *Password.*

Nama file	Size File	Hash	Lama Proses Enkripsi	Size file Hasil Enkripsi	Lama Proses Dekripsi	Size File Hasil Dekripsi
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	20 ms 10 ms 20ms 10ms 20ms 10ms 10ms 10ms	105 KB	10 msec 10 msec 30 msec 20 msec 30 msec 50 msec 20 msec 30 msec	104 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	51ms 41ms 50ms 50ms 50ms 50ms 70ms 50ms	761 KB	80 msec 90 msec 60 msec 71 msec 80 msec 90 msec 70 msec 271 msec	761 KB
Kepastian yang kutungggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	490ms 241ms 260ms 241ms 280ms 320ms 340ms 310ms	4,251 KB	381 msec 320 msec 320 msec 311 msec 340 msec 410 msec 391 msec 351 msec	4,250 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	391ms 331ms 280ms 300ms 341ms 370ms 360ms 320ms	4,993 KB	491 msec 381 msec 400 msec 371 msec 461 msec 551 msec 461 msec 421 msec	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3405ms 2734ms 3215ms 2764ms 3125ms 3275ms 3625ms 3055ms	45,304 KB	5438 msec 4636 msec 4517 msec 4717 msec 4757msec 5198 msec 4827 msec 4897msec	45,304 KB

Tabel 4.6 Tabel Perbandingan Proses menggunakan *File* kunci.

BAB V

KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Kesimpulan yang didapat selama pengerjaan Tugas Akhir ini adalah :

1. *File* Yang dihasilkan oleh proses enkripsi rata-rata hampir sama dengan *file* asli (99%), dan *file* yang dihasilkan proses dekripsi pasti sama (100%) dengan *file* asli sebelum proses enkripsi.
2. Lama proses yang dibutuhkan oleh sistem sangat tergantung dari *hardware* yang digunakan, dan kinerja sistem *processor*.
3. Tipe *File* yang dihasilkan oleh proses enkripsi dapat menjadi tipe apa pun saja, proses dekripsi nama *file* yang dihasilkan dapat ditulis sesuai keinginan *user*, namun agar dapat dibaca tipe *file* harus sama dengan *file* asli sebelum proses enkripsi.

5.2 SARAN

1. *File* hasil enkripsi dapat di *hide lock* dari sistem operasi, agar keamanannya lebih kuat.
2. Proses yang dilakukan dapat dikembangkan terhadap direktori / *folder*, dan *drive*.

DAFTAR PUSTAKA

Kurniawan Yusuf, Ir.MT. 2004. *Kriptografi Keamanan Internet dan Jaringan*. Bandung: Informatika.

Ariyus, Doni. 2006. *Kriptografi Keamanan Data dan Konunikasi*. Yogyakarta : Graha Ilmu.

Daemen and Rijmen. 1998. *AES submission document on Rijndael.*

WAHANAKomputer Semarang. 2003. *Memahami Model Enkripsi dan SecurityData* , Yogyakarta : ANDI.

Joan Daemen, Vincent Rijmen. 1999, *AES Proposal : Rijndael, Document Version 2.* NIST.

http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.htm. akses : 04 April 2008.

http://en.wikipedia.org/wiki/*Rijndael*_mix_columns. akses : 27-*DES* - 2007

Federal Information Processing Standards (FIPS) Publication 197. 2001. *Announcing the Advanced Encryption Standards* (AES) *RIJNDAEL.*

Wibowo , Wihartantyo Ari. 2004. *ADVANCED ENCRYPTION STANDARD, ALGORITMA RIJNDAEL.* Bandung : DEPARTEMEN TEKNIK ELEKTRO ITB.

Munir, Rinaldi. 2006. *Kumpulan Bahan Kuliah Keamanan Sistem Informasi*(*IF5054*). Bandung : DEPARTEMEN TEKNIK ELEKTRO ITB.

http://www.adeptscience.co.uk/products/mathsim/maple/powertools/cryptography /HTML/AES-Encryption.html Akses : 15 April 2008.

http://www.iaik.tugraz.at/RESEARCH/krypto/AES/old/~rijmen/rijndael/ Akses : 23 Februari 2008.

http://konsultasi.wordpress.com/2007/01/18/negara-islam-seperti-apa/ Akses : 30 Juli 2008

LAMPIRAN

Nama file	Size File	Hash	Lama Proses Enkripsi	Ukuran File Hasil Enkripsi	Lama Proses Dekripsi	Ukuran File Hasil Dekripsi
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	0 msec	105 KB	0 msec	104 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	47 msec 31 msec 31 msec 47 msec 31 msec 47 msec 47 msec 47 msec	761 KB	47 msec 46 msec 31 msec 32 msec 31 msec 47 msec 47 msec 32 msec	761 KB
Kepastian yang kutungggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	297 msec 234 msec 234 msec 219 msec 266 msec 297 msec 297 msec 266 msec	4,251 KB	281 msec 219 msec 218 msec 219 msec 250 msec 281 msec 282 msec 250 msec	4,250 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	328 msec 265 msec 266 msec 265 msec 297 msec 344 msec 328 msec 328 msec 297 msec	4,993 KB	344 msec 297 msec 282 msec 282 msec 313 msec 344 msec 344 msec 313 msec 313 msec	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3203 msec 3219 msec 2578 msec 2531 msec 2829 msec 3125 msec 3110 msec 2812 msec	45,304 KB	3219 msec 2625 msec 2581 msec 2535 msec 2830 msec 3125 msec 3118 msec 2820 msec	45,304 KB

Tabel Perbandingan Proses menggunakan *Password*.

Nama file	Size File	Hash	Lama Proses Enkripsi	Size file Hasil Enkripsi	Lama Proses Dekripsi	Size File Hasil Dekripsi
Itb makalah.doc	104 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	20 ms 10 ms 20ms 10ms 20ms 10ms 10ms 10ms	105 KB	10 msec 10 msec 30 msec 20 msec 30 msec 50 msec 20 msec 30 msec	104 KB
Kali.jpg	761 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	51ms 41ms 50ms 50ms 50ms 50ms 70ms 50ms	761 KB	80 msec 90 msec 60 msec 71 msec 80 msec 90 msec 70 msec 271 msec	761 KB
Kepastian yang kutunggu.mp3	4,250 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	490ms 241ms 260ms 241ms 280ms 320ms 340ms 310ms	4,251 KB	381 msec 320 msec 320 msec 311 msec 340 msec 410 msec 391 msec 351 msec	4,250 KB
Setup.exe	4,993 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	391ms 331ms 280ms 300ms 341ms 370ms 360ms 320ms	4,993 KB	491 msec 381 msec 400 msec 371 msec 461 msec 551 msec 461 msec 421 msec	4,993 KB
Sonia kau sbut Namaku,SONIA.DAT	45,304 KB	Haval MD 4 MD 5 RipeMD-128 RipeMD-160 SHA 256 SHA 512 Tiger	3405ms 2734ms 3215ms 2764ms 3125ms 3275ms 3625ms 3055ms	45,304 KB	5438 msec 4636 msec 4517 msec 4717 msec 4757msec 5198 msec 4827 msec 4897msec	45,304 KB

Tabel Perbandingan Proses menggunakan *File* kunci.

LISTING PROGRAM

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Buttons, DCPcrypt2, DCPmd5, DCPblockciphers,
  DCPrijndael, DCPsha512, DCPripemd128, DCPmd4, DCPsha1,
DCPripemd160,
  DCPhaval, ExtCtrls, DCPtiger, jpeg, Gauges, ComCtrls,
CoolTrayIcon,
  TextTrayIcon, Menus, DCPsha256;

type
  TForm1 = class(TForm)
    rijndael: TDCP_rijndael;
    DCP_haval1: TDCP_haval;
    DCP_md51: TDCP_md5;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    DCP_ripemd1601: TDCP_ripemd160;
    DCP_tiger1: TDCP_tiger;
    DCP_sha5121: TDCP_sha512;
    DCP_md41: TDCP_md4;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    btnInputBrowse: TSpeedButton;
    inFile: TEdit;
    Image1: TImage;
    StatusBar1: TStatusBar;
    ProgressBar1: TProgressBar;
    Label6: TLabel;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    btnDecrypt: TButton;
    encrypt: TButton;
    TabSheet2: TTabSheet;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    cbxHash: TComboBox;
    Label3: TLabel;
    GroupBox3: TGroupBox;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    BrowseKey: TSpeedButton;
    Label12: TLabel;
    CbxHashFile: TComboBox;
    KeyFile: TEdit;
    OpenkeyFile: TOpenDialog;
    SpeedButton1: TSpeedButton;
```

```
    Label5: TLabel;
    CoolTrayIcon1: TCoolTrayIcon;
    EncryptFile: TButton;
    DecryptFile: TButton;
    Label8: TLabel;
    CheckBoxPass: TCheckBox;
    PassFile: TEdit;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    DCP_ripemd1281: TDCP_ripemd128;
    DCP_sha2561: TDCP_sha256;
    Label17: TLabel;
    Label18: TLabel;
    GroupBox2: TGroupBox;
    btnOutputBrowse: TSpeedButton;
    Label7: TLabel;
    outFile: TEdit;
    procedure btnInputBrowseClick(Sender: TObject);
    procedure inFileExit(Sender: TObject);
    procedure btnOutputBrowseClick(Sender: TObject);
    procedure btnDecryptClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit2Exit(Sender: TObject);
    procedure encryptClick(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure Label5Click(Sender: TObject);
    procedure cbxHashChange(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Edit1Exit(Sender: TObject);
    procedure Edit2Enter(Sender: TObject);
    procedure StatusBar1DrawPanel(StatusBar: TStatusBar;
      Panel: TStatusPanel; const Rect: TRect);
    procedure TabSheet2Enter(Sender: TObject);
    procedure CbxHashFileChange(Sender: TObject);
    procedure PageControl1Change(Sender: TObject);
    procedure BrowseKeyClick(Sender: TObject);
    procedure KeyFileExit(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure CoolTrayIcon1DblClick(Sender: TObject);
    procedure CoolTrayIcon1Click(Sender: TObject);
    procedure CoolTrayIcon1Startup(Sender: TObject;
      var ShowMainForm: Boolean);
    procedure CheckBoxPassClick(Sender: TObject);
    procedure EncryptFileClick(Sender: TObject);
    procedure KeyFileChange(Sender: TObject);
    procedure DecryptFileClick(Sender: TObject);
    procedure PassFileChange(Sender: TObject);
    procedure Label16Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Hash : TDCP_Hash;
  Form1: TForm1;
  strmInput, strmOutput, strmkey: TFileStream;   //membaca inputan
& keluaran
  waktuMulai, waktuAkhir : TTimeStamp;        //var penghitung
waktu proses
  totalWaktu : LongWord;
implementation

uses Math, Unit2, Unit3;

{$R *.dfm}

function AddCommas(const S: string): string;  //fungsi menghitung
ukuran file
var
  i, j: integer;
begin
  i := Length(S) mod 3;
  if ((i <> 0) and (Length(S) > 3)) then
    Result := Copy(S,1,i) + ',';
  for j := 0 to ((Length(S) div 3) - 2) do
    Result := Result + Copy(S,1 + i + j*3,3) + ',';
  if (Length(S) > 3) then
    Result := Result + Copy(S,Length(S) - 2,3)
  else
    Result := S;
end;

function Min(a, b: integer): integer;  // untuk menginisialisasi
kan chiper.size(a) hash.size(b)
begin
  if (a < b) then
    Result := a
  else
    Result := b;
end;


procedure TForm1.FormCreate(Sender: TObject);  //tampilan form
saat program dijalankan
var i, o: integer;
ProgressBarStyle: integer;

begin
CoolTrayIcon1.HideTaskbarIcon;  //menghide task bar dengan
component TcooltrayIcon
CoolTrayIcon1.ShowBalloonHint('afre_N_enc.exe', 'Program Enkripsi
yg menggunakan algoritma'#13'Rijndael / Advanced Encryption
Standard(AES) serta menggunakan berbagai macam metode Hash sebagai
KEY-nya',bitInfo, 60);

//tampilan progressBar & windowProgram
StatusBar1.Panels[1].Style := psOwnerDraw;
```

```
  ProgressBar1.Parent := StatusBar1;
  ProgressBarStyle := GetWindowLong(ProgressBar1.Handle,
                                    GWL_EXSTYLE);
  ProgressBarStyle := ProgressBarStyle
                      - WS_EX_STATICEDGE;
  SetWindowLong(ProgressBar1.Handle,
                GWL_EXSTYLE,
                ProgressBarStyle);

ShowWindow(Application.Handle, sw_Hide);
SetWindowLong(Application.Handle,
gwl_exstyle, GetWindowLong(Application.Handle, GWL_EXSTYLE)
or WS_EX_TOOLWINDOW);
ShowWindow(Application.Handle, sw_show);

   Edit1.Enabled:= false;
   Edit2.Enabled:= false;
   BrowseKey.Enabled:=false;
   encrypt.Enabled:=false;
   btnDecrypt.Enabled:=false;
   PassFile.Visible:=false;
   EncryptFile.Enabled:=false;
   DecryptFile.Enabled:=false;


  for i := 0 to (ComponentCount - 1) do  //membaca Hash ke cbxHash
   if (Components[i] is TDCP_hash) then

cbxHash.Items.AddObject(TDCP_hash(Components[i]).Algorithm,Compone
nts[i]);

  for o:= 0 to (ComponentCount - 1) do  //membaca Hash ke
cbxHashFile
   if (Components[o] is TDCP_hash) then

CbxHashFile.Items.AddObject(TDCP_hash(Components[o]).Algorithm,Com
ponents[o]);
end;


procedure TForm1.CoolTrayIcon1DblClick(Sender: TObject);
begin
CoolTrayIcon1.ShowMainForm;
end;

procedure TForm1.CoolTrayIcon1Click(Sender: TObject);
begin
CoolTrayIcon1.HideMainForm;
end;

procedure TForm1.CoolTrayIcon1Startup(Sender: TObject;
  var ShowMainForm: Boolean);
begin
ShowMainForm:=true;
end;
```

```
procedure TForm1.FormPaint(Sender: TObject);   //memberi warnaform
var row, Ht : Word;
begin
  Ht := (ClientHeight + 255) div 245;
  for row := 0 to 215 do
  with Canvas do begin
  Brush.Color := RGB(row, 0, 0);
  FillRect(Rect(0, row * Ht, ClientWidth, (row + 3) * Ht));
  end;
end;


procedure TForm1.btnInputBrowseClick(Sender: TObject);        //
tombol browse click
begin
OpenDialog1.Execute;
inFile.Text:= OpenDialog1.FileName;
inFile.OnExit(inFile);
if not FileExists(InFile.Text) then
  begin
    MessageDlg('File Tidak Ada!',mtInformation,[mbOK],0);
    inFile.Text:='';
    Exit;
  end;
end;

procedure TForm1.inFileExit(Sender: TObject);
begin
  if (inFile.Text = '') then
    label1.Caption := '________'
  else if FileExists(inFile.Text) then
  begin
    // jika file ada lakukan fungsi ini..!!
    strmInput := nil;
    try
      strmInput := TFileStream.Create(inFile.Text,fmOpenRead); //
(namaFile, size 0..127)
      label1.Caption := AddCommas(IntToStr(strmInput.Size)) + '
KB'; //menampilkan ukuran file
      strmInput.Free; //mengkosongkan strminput
    except
      strmInput.Free;
      label1.Caption := 'File tidak bisa di baca!!';
    end;
  end
  else
    label1.Caption := '  File yang anda masukkan salah!!';
end;


procedure TForm1.btnOutputBrowseClick(Sender: TObject); // tombol
output file clik
begin
  if
  inFile.Text<>''
  then  begin
```

```
  SaveDialog1.Execute;
  outFile.Text := SaveDialog1.FileName;
  encrypt.Enabled:=true;
  btnDecrypt.Enabled:=true;
  end;

  if inFile.Text=''   //textfile input kosong
  then   begin
  ShowMessage('    Silahkan Masukkan File Input!!');
  OpenDialog1.Execute;
  inFile.Text:= OpenDialog1.FileName;
  inFile.OnExit(inFile);
  end;

     if fileexists (outFile.Text) then
if
MessageDlg('  Over Write
File??',mtConfirmation,[mbYes,mbNo],0)=mrNo then
begin outFile.Text:='';
end;
   if (inFile.Text='') and (outFile.Text='')
  then
    begin
    ShowMessage('      Lakukan Pengisian Dengan Benar!!');
    OpenDialog1.Execute;
    inFile.Text:= OpenDialog1.FileName;
    inFile.OnExit(inFile);
    SaveDialog1.Execute;
    SaveDialog1.DefaultExt:= SaveDialog1.Filter;
    outFile.Text := SaveDialog1.FileName;
    end;
  if inFile.Text=outFile.Text
    then  begin
     ShowMessage('      File Tidak Boleh Sama!!'#13#13'
Lakukan Penyimpanan Ulang');
      outFile.Text:='';
      SaveDialog1.Execute;
end;
end;


procedure TForm1.cbxHashChange(Sender: TObject); //combobox Hash
begin
  if (inFile.Text<>'')
    then begin
    Edit1.Enabled:= true;
    Edit2.Enabled:=true;
    //outFile.Text:='';
    end;
end;

procedure TForm1.Edit2Exit(Sender: TObject);    //textfile
confirmPassword
begin
 if Edit1.Text<> Edit2.Text
 then
```

```
 begin
   MessageDlg('Password Harus Sama !!',mtError,[mbOK],0);
    btnDecrypt.Enabled:=false ;
    encrypt.Enabled:=false;
    Edit1.SetFocus;
    end;
end;

procedure TForm1.Edit2Change(Sender: TObject);
begin
   if Edit1<>Edit2
  then
  begin
        encrypt.Enabled:=false;
        btnDecrypt.Enabled:=false;
        outFile.Text:='';
  end;
  if edit1=edit2
  then begin
  outFile.SetFocus;
  end;
end;

procedure TForm1.Edit2Enter(Sender: TObject);
begin
  if
  Edit1.Text=''
  then begin
  ShowMessage('Password Pertama Harus Diisi!!');
  Edit1.SetFocus;
end;
end;


procedure TForm1.Edit1Exit(Sender: TObject);
begin
  if
  Edit1.Text<>''
  then
  edit2.SetFocus;
end;


procedure TForm1.encryptClick(Sender: TObject);   //event tombol
encrypt
var
   HashDigest: array of byte;   // Matriks Hash
   Salt: array[0..7] of byte;   // Matriks bilangan acak untuk
mencegah serangan lawan
   i: integer;
   interval : integer;           // interval progressBar
   CipherIV: array of byte;     // initialisation vector /IV (CBC
model)
begin
  if
  outFile.Text=''
```

```
  then begin
  ShowMessage('File output kosong!!');
  Exit;
  end;

  if Edit1.Text<> Edit2.Text
  then begin
  ShowMessage('Password Harus Sama!!');
  Exit;
  end;


  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;


  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);

  try
    waktuMulai := DateTimeToTimeStamp(time);          //mulai
menghitung waktu
    strmInput := TFileStream.Create(InFile.Text,fmOpenRead);
//membaca input file
    strmOutput := TFileStream.Create(OutFile.Text,fmCreate);
//menulis output file

    Hash := TDCP_hash(cbxHash.Items.Objects[cbxHash.ItemIndex]);
//memilih hash dari combobox
    SetLength(HashDigest,Hash.HashSize div 8);
//identifikassi panjang hash (div8: 1byte=8bit)

    for i := 0 to 7 do                //perulangan untuk tiap
byte
     Salt[i] := Random(256);  // bilangan acak (random values)
crypto secure Pseudo Random Generator (CSPRNG)
      strmOutput.WriteBuffer(Salt,Sizeof(Salt));  // membuat Salt
dalam FileOut untuk proses decrypt

      Hash.Init;
      Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
      Hash.UpdateStr(Edit1.Text);  // Hash Password
      Hash.Final(HashDigest[0]);           // hasil akhir Hash ke
HashDigest

        rijndael:= TDCP_rijndael.Create(Self);     //identifikasi
rijndael
        SetLength(CipherIV,rijndael.BlockSize div 8);
//identifikassi panjang rijndael (div8: 1byte=8bit)
        for i := 0 to (Length(CipherIV) - 1) do
//perulangan untuk tiap block
        CipherIV[i] := Random(256);           // Bilangan acak
untuk Initial Vektor/IV
        strmOutput.WriteBuffer(CipherIV[0],Length(CipherIV));  //
membuat IV dalam FileOutput untuk proses decrypt
```

```
rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // identifikasi key

rijndael.EncryptStream(strmInput,strmOutput,strmInput.Size); //
encrypt file
        rijndael.Burn;   // mengambil dan menyimpan Key

        strmInput.Free; //mengosongkan strminput
        strmOutput.Free;//mengosongkan strmoutput
        waktuAkhir := DateTimeToTimeStamp(time); //akhir waktu
proses
        totalWaktu := waktuAkhir.Time-waktuMulai.Time;
//hitung waktu

        MessageDlg('Enkripsi File Sukses!!'#13#13'Waktu : ' +
IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        Edit1.Text:='';
        Edit2.Text:='';
        cbxHash.Text:='';
        Label1.Caption:='_________';
        encrypt.Enabled:= false;
        btnDecrypt.Enabled:= False;
        Edit1.Enabled:= false;
        Edit2.Enabled:= false;
        Cursor:= crDefault;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;

        except //kesalahan user yang mungkin terjadi
        MessageDlg('File IO ERROR!!',mtError,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        Edit1.Text:='';
        Edit2.Text:='';
        cbxHash.Text:='';
        Label1.Caption:='_________';
        encrypt.Enabled:= false;
        btnDecrypt.Enabled:= False;
        Edit1.Enabled:= false;
        Edit2.Enabled:= false;
        strmInput.Free;
        strmOutput.Free;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
      end;
end;


procedure TForm1.btnDecryptClick(Sender: TObject); //event tombol
decrypt!!
```

```
var
  Hash: TDCP_hash;              // Hash
  HashDigest: array of byte;    // Matriks Hash
  Salt: array[0..7] of byte;    // bilangan acak untuk mencegah
serangan lawan
  CipherIV: array of byte;      // initialisation vector /IV (CBC
model)
  interval : integer;           // var untuk interval progressBar
begin

if
  outFile.Text=''
  then begin
  ShowMessage('File output kosong!!');
  Exit;
  end;
  if Edit1.Text<> Edit2.Text
  then begin
  ShowMessage('Password Harus Sama!!');
  Exit;
  end;

if
MessageDlg('  Anda Yakin Dengan Kunci yang Anda
Masukkan??',mtConfirmation,[mbYes,mbNo],0)=mryes then
  begin
  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;
  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);

    try
    waktuMulai := DateTimeToTimeStamp(time);  // penghitung waktu
di mulai
       strmInput:= TFileStream.Create(InFile.Text,fmOpenRead);
//membaca file input
       strmOutput:= TFileStream.Create(OutFile.Text,fmCreate);
//menulis file output

    Hash := TDCP_hash(cbxHash.Items.Objects[cbxHash.ItemIndex]);
// memilih Hash dari combo box!!
    SetLength(HashDigest,Hash.HashSize div 8);    // identifikassi
panjang hash (div8: 1byte=8bit)
    strmInput.ReadBuffer(Salt[0],Sizeof(Salt));  // membaca Salt
dari File
    Hash.Init;
    Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
    Hash.UpdateStr(Edit1.Text);  // Hash Password
    Hash.Final(HashDigest[0]);           // hasil akhir Hash ke
HashDigest

     rijndael:= TDCP_rijndael.Create(Self);  //identifikasi
rijndael
     SetLength(CipherIV,rijndael.BlockSize div 8);   //
identifikassi panjang rijndael (div8: 1byte=8bit)
```

```
      strmInput.ReadBuffer(CipherIV[0],Length(CipherIV));       //
Membaca Initial Vektor/IV dari File

rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // Membaca key dari File
      rijndael.CipherMode := cmCBC;      //menggunakan cipher
model CBC

     rijndael.DecryptStream(strmInput,strmOutput,strmInput.Size -
strmInput.Position); // Proses decrypt!
     rijndael.Burn;
        strmInput.Free; //mengosongkan strminput
        strmOutput.Free;//mengosongkan strmoutput
        waktuAkhir := DateTimeToTimeStamp(time); // akhir waktu
proses
        totalWaktu := waktuAkhir.Time-waktuMulai.Time;
//menghitung waktu

        MessageDlg('Dekripsi File Sukses!!'#13#13'Waktu : ' +
IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        Edit1.Text:='';
        Edit2.Text:='';
        cbxHash.Text:='';
        Label1.Caption:='_________';
        Edit1.Enabled:= false;
        Edit2.Enabled:= false;
        encrypt.Enabled:= false;
        btnDecrypt.Enabled:= False;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;

        except //kesalahan user yang mungkin terjadi
        MessageDlg('File IO error',mtError,[mbOK],0);
        ProgressBar1.Position := 0;inFile.Text:='';
        outFile.Text:='';
        Edit1.Text:='';
        Edit2.Text:='';
        cbxHash.Text:='';
        Label1.Caption:='_________';
        encrypt.Enabled:= false;
        btnDecrypt.Enabled:= False;
        Edit1.Enabled:= false;
        Edit2.Enabled:= false;
        strmInput.Free;
        strmOutput.Free;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
 end;
end;  end;


procedure TForm1.StatusBar1DrawPanel(StatusBar: TStatusBar;
```

```
  Panel: TStatusPanel; const Rect: TRect);
//panelStatus Bar
begin
  if Panel = StatusBar.Panels[1] then
  with ProgressBar1 do begin
    Top := Rect.Top;
    Left := Rect.Left;
    Width := Rect.Right - Rect.Left - 0;
    Height := Rect.Bottom - Rect.Top;
  //  StatusBar1.Color := clRed;
  end;
end;

procedure TForm1.TabSheet2Enter(Sender: TObject);
begin
Edit1.Text:='';
Edit2.Text:='';
encrypt.Enabled:= false;
btnDecrypt.Enabled:=false;
end;

procedure TForm1.PageControl1Change(Sender: TObject);
begin
   outFile.Text:='';
   Edit1.Enabled:= false;
   Edit2.Enabled:= false;
   encrypt.Enabled:=false;
   btnDecrypt.Enabled:=false;

Label8.Caption:='_____________________________________________________
___';
   EncryptFile.Enabled:=false;
   DecryptFile.Enabled:=false;
   CheckBoxPass.Checked:=false;
   BrowseKey.Enabled:=false;
   KeyFile.Text:='';
   Label10.Caption:='_________';
   cbxHash.ItemIndex:=0;
   CbxHashFile.ItemIndex:=0;
end;

procedure TForm1.CbxHashFileChange(Sender: TObject);
//HashComboboxfile
begin
if inFile.Text<>''
    then begin
    BrowseKey.Enabled:=true;
end;
end;


procedure TForm1.BrowseKeyClick(Sender: TObject);   //pilih
keyFile
begin
if CbxHashFile.Text<>''
  then begin
```

```
  OpenkeyFile.Execute;
  KeyFile.Enabled:=true;
  KeyFile.Text:= OpenkeyFile.FileName;
  KeyFile.OnExit(KeyFile);
  end;
    if (KeyFile.Text=inFile.Text) or (KeyFile.Text=outFile.Text)
  then begin
    ShowMessage('   Key File Tidak Boleh Sama dengan File
I/O..!!!');
    KeyFile.Text:='';
    CbxHashFile.SetFocus;

Label8.Caption:='_____________________________________________
___';
    end;
  if not FileExists(KeyFile.Text) then
  begin    MessageDlg('File Tidak Ada!',mtInformation,[mbOK],0);
    KeyFile.Text:='';
    Exit;
    KeyFile.SetFocus;
    end;
end;

procedure TForm1.KeyFileChange(Sender: TObject);    //keyFile
begin
  if KeyFile.Text=''
  then begin
  EncryptFile.Enabled:=false;
  DecryptFile.Enabled:=false;
  CheckBoxPass.Checked:=false;
end;

  if KeyFile.Text<>''
  then begin
  EncryptFile.Enabled:=true;
  DecryptFile.Enabled:=true;
  CheckBoxPass.Checked:=false;
end;
end;

procedure TForm1.KeyFileExit(Sender: TObject); //keyFile
var
    HashDigest: array of byte;   // Matriks Hash
    w : integer;
    s: string;
begin
  if (KeyFile.Text = '') then
    label10.Caption := '_________'
    else if FileExists(KeyFile.Text) then
      begin
    // jika file ada lakukan fungsi ini..!!
      strmInput := nil;
      try
      strmkey := TFileStream.Create(KeyFile.Text,fmOpenRead);
//membaca Keyfile (namaFile, size 0..127)
```

```
      label10.Caption := AddCommas(IntToStr(strmkey.Size)) + '
KB';          //menuliskan Ukuran File

     Hash :=
TDCP_hash(CbxHashFile.Items.Objects[CbxHashFile.ItemIndex]);
//identifikasi Hash
     SetLength(HashDigest,Hash.HashSize div 8);  //membaca panjang
Hash

      Hash.Init;                                      // initialize
it
      Hash.UpdateStream(strmkey,strmkey.Size);       // Acak
KeyFile dg Hash
      Hash.Final(HashDigest[0]);                            // Hasil
Hash

      s:= '';
      for w:= 0 to Length(HashDigest) -1 do
        s:= s + IntToHex(HashDigest[w],2);
//hexatoint to string
       Label8.Caption:= s;
//tampilkan hasil
    strmkey.Free;
//kosongkan strmkey


  except                    //kesalahan user yg mungkin terjadi
  MessageDlg('  file Error!!',mtError,[mbOK],0);
  strmkey.Free;
  label10.Caption := 'File tidak bisa di baca!!';
  end;
 end;
end;


procedure TForm1.CheckBoxPassClick(Sender: TObject);  //pilihan
penambahan password
begin
  if
   CheckBoxPass.Checked=false
   then
   begin
   PassFile.Visible:=false;
   end;
  if
   CheckBoxPass.Checked=true
   then  begin
        if KeyFile.Text=''
          then
          PassFile.Visible:=false;
        if KeyFile.Text<>''
        then
        begin
        PassFile.Visible:=true;
        PassFile.SetFocus;
        end;
```

```
    end;
end;

procedure TForm1.PassFileChange(Sender: TObject); //tambah key dg
password
begin
Label8.Caption:= PassFile.Text+Label8.Caption;
end;

procedure TForm1.EncryptFileClick(Sender: TObject);  //tombol
encrypt File
var
   HashDigest: array of byte;   // Matriks Hash
   Salt: array[0..7] of byte;   // Matriks bilangan acak untuk
mencegah serangan lawan
   i: integer;
   interval : integer;
   CipherIV: array of byte;     // initialisation vector /IV (CBC
model)
  begin
if
  outFile.Text=''
  then begin
  ShowMessage('File output kosong!!');
  Exit;
  end;

  if KeyFile.Text='' //jika keyFile kosong
  then begin
  ShowMessage('    Silahkan Isikan Key File ');
  Exit;
  CbxHashFile.SetFocus;
  EncryptFile.Enabled:=false;
  DecryptFile.Enabled:=false;
  end;

  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;
  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);
    try
    waktuMulai := DateTimeToTimeStamp(time); //penghitung waktu
dimulai
    strmInput := TFileStream.Create(InFile.Text,fmOpenRead);
//membaca inputfile
    strmOutput := TFileStream.Create(OutFile.Text,fmCreate);
//menulis outputfile

    Hash :=
TDCP_hash(CbxHashFile.Items.Objects[CbxHashFile.ItemIndex]);
//identifikasi Hash
    SetLength(HashDigest,Hash.HashSize div 8);  //identifikasi
panjang Hash (1Byte=8bit)
     for i := 0 to 7 do                        //perulangan untuk
tiap byte
```

```
      Salt[i] := Random(256);  // bilangan acak (random values)
crypto secure Pseudo Random Generator (CSPRNG)
      strmOutput.WriteBuffer(Salt,Sizeof(Salt));  // membuat Salt
dalam FileOut untuk proses decrypt

      Hash.Init;
      Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
      Hash.UpdateStr(Label8.Caption);  // Hash Password
      Hash.Final(HashDigest[0]);           // hasil akhir Hash ke
HashDigest

        rijndael:= TDCP_rijndael.Create(Self);
//identifikasi rijndael
        SetLength(CipherIV,rijndael.BlockSize div 8);
//identifikasi panjang rijndael
        for i := 0 to (Length(CipherIV) - 1) do       //perulangan
untuk tiap block
        CipherIV[i] := Random(256);         // Bilangan acak
untuk IV
        strmOutput.WriteBuffer(CipherIV[0],Length(CipherIV));  //
membuat IV dalam FileOut untuk proses decrypt


rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // identifikasi key

rijndael.EncryptStream(strmInput,strmOutput,strmInput.Size); //
encrypt file
        rijndael.Burn;   // mengambil dan menyimpan Key
        strmInput.Free;         //mengosongkan strminput
        strmOutput.Free;       //mengosongkan strmouptput
        waktuAkhir := DateTimeToTimeStamp(time); //akhir waktu
proses
        totalWaktu := waktuAkhir.Time-waktuMulai.Time;
//menghitung waktu

        MessageDlg('Enkripsi File Sukses!!'#13#13'Waktu : ' +
IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        KeyFile.Text:='';
        PassFile.Text:='';
        Label10.Caption:='_________';
        EncryptFile.Enabled:= false;
        DecryptFile.Enabled:= False;
        CheckBoxPass.Checked:=false;
        KeyFile.Enabled:=false;

Label8.Caption:='_____________________________________________________
___';
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
        Label1.Caption:='_________';
```

```
        except          //kesalahan user yg mungkin terjadi
        MessageDlg('File IO ERROR!!',mtError,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        KeyFile.Text:='';
        PassFile.Text:='';
        Label10.Caption:='________';

Label8.Caption:='_____________________________________________________
___';
        EncryptFile.Enabled:= false;
        DecryptFile.Enabled:= False;
        CheckBoxPass.Checked:=false;
        KeyFile.Enabled:=false;
        strmInput.Free;
        strmOutput.Free;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
        Label1.Caption:='_________';
      end;
end;


procedure TForm1.DecryptFileClick(Sender: TObject);
//tombol DecryptFile
var
  Hash: TDCP_hash;              // Hash
  HashDigest: array of byte;    // Matriks Hash
  Salt: array[0..7] of byte;    // bilangan acak untuk mencegah
serangan lawan
  CipherIV: array of byte;      // initialisation vector /IV (CBC
model)
  interval : integer;
begin

if
  outFile.Text=''
  then begin
  ShowMessage('File output kosong!!');
  Exit;
  end;

  if KeyFile.Text=''
  then begin
  ShowMessage('    Silahkan Isikan Key File');
  Exit;
  CbxHashFile.SetFocus;
  EncryptFile.Enabled:=false;
  DecryptFile.Enabled:=false;
  end;

if
MessageDlg('  Anda Yakin Dengan Kunci yang Anda
Masukkan??',mtConfirmation,[mbYes,mbNo],0)=mryes then
  begin
```

```
  ProgressBar1.Position := 0;
  ProgressBar1.Max := 100;
  for interval := 0 to 100 do
    ProgressBar1.Position := interval;
    Sleep(25);
    try
        waktuMulai := DateTimeToTimeStamp(time);  //memulai
menghitung waktu
        strmInput:= TFileStream.Create(InFile.Text,fmOpenRead);
//membaca fileinput
        strmOutput:= TFileStream.Create(OutFile.Text,fmCreate);
//menulis fileoutput

    Hash :=
TDCP_hash(CbxHashFile.Items.Objects[CbxHashFile.ItemIndex]);
//pilih Hash dari combo box!!
    SetLength(HashDigest,Hash.HashSize div 8);  //identifikasi
panjang Hash (1byte=8bit)
    strmInput.ReadBuffer(Salt[0],Sizeof(Salt));  // membaca Salt
dari File
    Hash.Init;
    Hash.Update(Salt[0],Sizeof(Salt));   // hash salt
    Hash.UpdateStr(Label8.Caption);  // Hash Password
    Hash.Final(HashDigest[0]);          // hasil akhir Hash ke
HashDigest

     rijndael:= TDCP_rijndael.Create(Self);   //identifikasi
rijndael
     SetLength(CipherIV,rijndael.BlockSize div 8);  //identifikasi
panjang rijndael
     strmInput.ReadBuffer(CipherIV[0],Length(CipherIV));      //
Membaca IV dari File

rijndael.Init(HashDigest[0],Min(rijndael.MaxKeySize,Hash.HashSize)
,CipherIV);  // Membaca key dari File
     rijndael.CipherMode := cmCBC;

     rijndael.DecryptStream(strmInput,strmOutput,strmInput.Size -
strmInput.Position); // decrypt!
     rijndael.Burn;
        strmInput.Free; //mengosongkan strminput
        strmOutput.Free;//mengosongkan strmoutput
        waktuAkhir := DateTimeToTimeStamp(time);  //akhir waktu
proses
        totalWaktu := waktuAkhir.Time-waktuMulai.Time;
//menghitung waktu

        MessageDlg('Dekripsi File Sukses!!'#13#13'Waktu : ' +
IntToStr(totalWaktu) + ' msec',mtInformation,[mbOK],0);
        ProgressBar1.Position := 0;
        KeyFile.Enabled:=false;
        inFile.Text:='';
        outFile.Text:='';
        KeyFile.Text:='';
        PassFile.Text:='';
        Label10.Caption:='_________';
```

```
Label8.Caption:='_____________________________________________________
___';
        EncryptFile.Enabled:= false;
        DecryptFile.Enabled:= False;
        CheckBoxPass.Checked:=false;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
        Label1.Caption:='_________';


        except  //kesalahan user yg mungkin terjadi
        MessageDlg('File IO error',mtError,[mbOK],0);
        ProgressBar1.Position := 0;
        inFile.Text:='';
        outFile.Text:='';
        KeyFile.Text:='';
        PassFile.Text:='';
        Label10.Caption:='_________';

Label8.Caption:='_____________________________________________________
___';
        EncryptFile.Enabled:= false;
        DecryptFile.Enabled:= False;
        CheckBoxPass.Checked:=false;
        KeyFile.Enabled:=false;
        strmInput.Free;
        strmOutput.Free;
        strmkey.Free;
        cbxHash.ItemIndex:=0;
        CbxHashFile.ItemIndex:=0;
        Label1.Caption:='_________';
        end;
end;
     end;

procedure TForm1.Label16Click(Sender: TObject);
begin
form3.show;   //form Help
end;

procedure TForm1.Label5Click(Sender: TObject);
begin
form2.show; //form aboutMe
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
Application.Terminate;   //Exit
end;

procedure TForm1.Edit1Change(Sender: TObject);
begin
 if Edit1<>Edit2
  then
  begin
```

```
        encrypt.Enabled:=false;
        btnDecrypt.Enabled:=false;
        outFile.Text:='';
  end;


if (Edit1.Text= Edit2.Text) and (inFile.Text<>'')and
(outFile.Text<>'')
    then
   begin
    btnDecrypt.Enabled:=true ;
    encrypt.Enabled:=true;
    encrypt.SetFocus;
    end;
end;

end.
```