BAB 1

PENDAHULUAN

1.1. Latar Belakang

Salah satu hal terpenting dalam komunikasi menggunakan komputer dan jaringan komputer adalah untuk menjamin keamanan pesan, data, ataupun informasi dalam proses pertukaran data, sehingga menjadi salah satu pendorong munculnya teknologi Kriptografi. Kriptografi berbasis pada algoritma pengkodean data informasi yang mendukung kebutuhan dari dua aspek keamanan informasi, yaitu *secrecy* (perlindungan terhadap kerahasiaan data informasi) dan *authenticity* (perlindungan terhadap pemalsuan dan pengubahan informasi yang tidak diinginkan.

Kriptografi merupakan studi matematika yang mempunyai hubungan dengan aspek keamanan informasi seperti integritas data, keaslian entitas dan keaslian data. Kriptografi menggunakan berbagai macam teknik dalam upaya untuk mengamankan data. Pengiriman data dan penyimpanan data melalui media elektronik memerlukan suatu proses yang dapat menjamin keamanan dan keutuhan dari data yang dikirimkan tersebut. Data tersebut harus tetap rahasia selama pengiriman dan harus tetap utuh pada saat penerimaan di tujuan. Untuk memenuhi hal tersebut, dilakukan proses penyandian (enkripsi dan dekripsi) terhadap data yang akan dikirimkan.

Enkripsi dilakukan pada saat pengiriman dengan cara mengubah data asli menjadi data rahasia, sedangkan dekripsi dilakukan pada saat penerimaan dengan

cara mengubah data rahasia menjadi data asli. Jadi data yang dikirimkan selama proses pengiriman adalah data rahasia, sehingga data asli tidak dapat diketahui oleh pihak yang tidak berkepentingan. Data asli hanya dapat diketahui oleh penerima dengan menggunakan kunci rahasia.

Disini enkripsi dapat diartikan sebagai kode atau *cipher*. Sebuah *system* pengkodean menggunakan suatu tabel atau kamus yang telah didefinisikan untuk kata dari informasi atau yang merupakan bagian dari pesan, data, atau informasi yang di kirim. Sebuah *cipher* menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) bit dari suatu pesan asli (*plaintext*) menjadi *cryptogram* yang tidak di mengerti. Karena *system cipher* merupakan suatu sistem yang telah siap untuk di outomasi, maka teknik ini digunakan dalam sistem keamanan jaringan komputer.

National Institute of Standard and Technology (NIST) untuk pertama kalinya mengumumkan suatu algoritma standar penyandian data yang telah dijadikan standard sejak tahun 1977 adalah *Data Encryption Standard* (*DES*). Kekuatan *DES* ini terletak pada panjang kuncinya yaitu 56-bit. Untuk menanggapi keinginan agar mengganti algoritma DES sebagai standar. Perkembangan kecepatan perangkat keras dan meluasnya penggunaan jaringan komputer terdistribusi mengakibatkan penggunaan *DES*, dalam beberapa hal, terbukti sudah tidak aman dan tidak mencukupi lagi terutama dalam hal yang pengiriman data melalui jaringan internet. Perangkat keras khusus yang bertujuan untuk menentukan kunci 56-bit *DES* hanya dalam waktu beberapa jam sudah dapat dibangun. Beberapa pertimbangan tersebut

telah manandakan bahwa diperlukan sebuah standard algoritma baru dan kunci yang lebih panjang. *Triple-DES* muncul sebagai alternative solusi untuk masalah-masalah yang membutuhkan keamanan data tingkat tinggi seperti perbankan, tetapi ia terlalu lambat pada beberapa penggunaan enkripsi.

Pada tahun 1997, *the U.S. National Institue of Standards and Technology* (*NIST*) mengumumkan bahwa sudah saatnya untuk pembuatan standard algoritma penyandian baru yang kelak diberi nama *Advanced Encryption Standard* (*AES*). Algoritma *AES* ini dibuat dengan tujuan untuk menggantikan algoritma *DES & Triple-DES* yang telah lama digunakan dalam menyandikan data elektronik. Setelah melalui beberapa tahap seleksi, algoritma *Rijndael* ditetapkan sebagai algoritma kriptografi *AES* pada tahun 2000.

Algoritma *AES* merupakan algoritma kriptografi simetrik yang beroperasi dalam mode penyandi blok (*block cipher*) yang memproses blok data 128-bit dengan panjang kunci 128-bit (*AES*- 128), 192-bit (*AES*-192), atau 256-bit (*AES*-256). Beberapa mode operasi yang dapat diterapkan pada algoritma kriptografi penyandi blok *AES* di antaranya adalah *Electronic Code Book* (*ECB*), *Cipher Block Chaining* (*CBC*), *Cipher Feedback* (*CFB*), dan *Output Feedback* (*OFB*). Implementasi *AES* dengan mode operasi *ECB, CBC, CFB,* dan *OFB* tentu saja memiliki kelebihan dan kekurangan tertentu dalam aspek tingkat keamanan data.

1.2. Rumusan Rancangan

Rumusan rancangan untuk program aplikasi kriptosistem ini menggunakan algoritma enkripsi AES (*Advanced Encryption Standard)* dengan menggunakan bahasa pemrograman visual basic 6.0.

1.3. Batasan Rancangan

Batasan rancangan pada program aplikasi kriptosistem dengan algoritma AES (*Advanced Encryption Standard)* yaitu:

1 Rancangan program aplikasi ini dibuat untuk mengamankan pesan, tetapi rancangan program ini tidak dibandingkan dengan program sejenis sebelumnya yang telah dibuat sehingga tidak diketahui program ini lebih baik atau tidak dari program sejenis sebelumnya.

2 Ukuran teks yang dapat dienkripsi senilai 2000 karakter, teks berupa angka, huruf dan tombol lain yang tersedia pada keyboard, hal ini dikarenakan keterbatasan bahasa pemograman yang digunakan yaitu visual basic 6.0.

3 Rancangan algoritma kriptosistem ini hanya dapat mengenkripsi dan mendekripsi data yang berupa teks atau tulisan, bukan suara maupun gambar.

1.4. Spesifikasi Rancangan

Spesifikasi rancangan program aplikasi AES terdiri dari sebagai berikut :

1 Unit Enkripsi dan Dekripsi

Pada unit ini digunakan algoritma AES menggunakan sistem *Block Cipher* yang memiliki panjang blok 128-bit. Panjang kunci bervariasi (128/ 192/ 256-bit) dan kunci ini menggunakan tipe *symetric key*. *Key* yang digunakan dalam proses enkripsi ini sama dengan key yang digunakan pada unit dekripsi. Text yang ingin dienkripsi juga dapat berupa text apa saja yang termasuk dalam bilangan ASCII baik yang 7 bit maupun 8 bit.

Bahasa pemrograman yang digunakan adalah bahasa pemrograman visual basic 6.0.

2 Unit Penyimpanan dan Unit Pembuka File

Unit ini digunakan untuk proses penyimpan data yang telah diubah dalam proses enkripsi sehingga menjadi bentuk *chipertext* dan membuka kembali data yang telah disimpan untuk diubah dalam proses dekripsi menjadi data asli.

3 Unit Pembanding

Unit ini digunakan penerima untuk memastikan keutuhan dan keaslian pesan yang diterima dari pengirim. Begitu pula dengan pengirim memastikan bahwa si-penerima pesan adalah orang yang benar dan berhak atas pesan yang dikirim.

1.5. Kegunaan Rancangan

Perancangan kriptosistem dengan menggunakan algoritma AES ini digunakan oleh bagian yang mengelola data penting diperusahaan, seperti data produk baru, data keuangan, data klien dan sebagainya. Untuk itu kriptosistem digunakan untuk mengamankan data-data tersebut.

1.6. Tujuan Rancangan

Perancangan kriptosistem dengan menggunakan algoritma AES ini memiliki tujuan untuk meningkatkan keamanan data. Algoritma AES memiliki ketahanan terhadap semua jenis serangan yang diketahui. Disamping itu kesederhanaan rancangan, kekompakan kode dan kecepatan pada berbagai platform dimiliki oleh algoritma AES ini.

BAB 2

LANDASAN TEORI

2.1. Pengertian Kriptografi

Kriptografi merupakan seni dan ilmu menyembunyikan informasi dari penerima yang tidak berhak. Kata cryptographi berasal dari kata Yunani *kryptos* (tersembunyi) dan *graphein* (menulis). *Cryptanalysis* adalah aksi untuk memecahkan mekanisme kriptografi dengan cara mendapatkan plaintext atau kunci dari ciphertext yang digunakan untuk mendapatkan informasi berharga kemudian mengubah atau memalsukan pesan dengan tujuan untuk menipu penerima yang sesungguhnya. *Encryption* adalah mentransformasi data kedalam bentuk yang tidak dapat terbaca tanpa sebuah kunci tertentu. Tujuannya adalah untuk meyakinkan privasi dengan menyembunyikan informasi dari orang-orang yang tidak ditujukan, bahkan mereka mereka yang memiliki akses ke data terenkripsi. Dekripsi merupakan kebalikan dari enkripsi, yaitu transformasi data terenkripsi kembali ke bentuknya semula.

Enkripsi dan dekripsi pada umumnya membutuhkan penggunaan sejumlah informasi rahasia, disebut sebagai kunci. Untuk beberapa mekanisme enkripsi, kunci yang sama digunakan baik untuk enkripsi dan dekripsi; untuk mekanisme yang lain, kunci yang digunakan untuk enkripsi dan dekripsi berbeda. Dua tipe dasar dari teknologi kriptografi adalah *symmetric key (secret/private key) cryptography* dan *asymmetric (public key) cryptography*. Pada *symmetric key cryptography*, baik

pengirim maupun penerima memiliki kunci rahasia yang umum. Pada *asymmetric key cryptography*, pengirim dan penerima masing-masing berbagi kunci publik dan privat. Kriptografi saat ini lebih dari enkripsi dan dekripsi saja. Otentikasi menjadi bagian dari kehidupan kita sama seperti privasi. Kita menggunakan otentikasi dalam kehidupan sehari-hari, sebagai contoh saat kita menandatangani sejumlah dokumen dan saat kita berpindah ke dunia dimana keputusan dan persetujuan kita dikomunikasikan secara elektronis, kita membutuhkan teknik-teknik untuk otentikasi. Kriptografi menyediakan mekanisme untuk prosedur semacam itu. *Digital signature* (tanda tangan digital) mengikat dokumen dengan kepemilikan kunci tertentu, sedangkan *digital timestamp* mengikat dokumen dengan pembuatnya pada saat tertentu. dengan kepemilikan kunci tertentu, sedangkan *digital timestamp* mengikat dokumen dengan pembuatnya pada saat tertentu.

2.2. Sejarah Kriptografi

Kriptografi memiliki sejarah yang panjang dan mengagumkan. Penulisan rahasia ini dapat dilacak kembali ke 3000 tahun SM saat digunakan oleh bangsa Mesir. Mereka menggunakan hieroglyphcs untuk menyembunyikan tulisan dari mereka yang tidak diharapkan. Hieroglyphcs diturunkan dari bahasa Yunani hieroglyphica yang berarti ukiran rahasia. Hieroglyphs berevolusi menjadi hieratic, yaitu *stylized script* yang lebih mudah untuk digunakan. Sekitar 400 SM, kriptografi militer digunakan oleh bangsa Spartan dalam bentuk sepotong papirus atau perkamen dibungkus dengan batang kayu. Sistem ini disebut Scytale.

Sekitar 50 SM, Julius Caesar, kaisar Roma, menggunakan *cipher substitusi* untuk mengirim pesan ke Marcus Tullius Cicero. Pada *cipher* ini, huruf-huruf alfabet disubstitusi dengan huruf-huruf yang lain pada alfabet yang sama. Karena hanya satu alfabet yang digunakan, cipher ini merupakan substitusi monoalfabetik. *Cipher* semacam ini mencakup penggeseran alfabet dengan 3 huruf dan mensubstitusikan huruf tersebut. Substitusi ini kadang dikenal dengan C3 (untuk Caesar menggeser 3 tempat). Secara umum sistem cipher Caesar dapat ditulis sebagai berikut :

$$Zi = Cn(Pi)$$

Dimana Zi adalah karakter-karekter ciphertext, Cn adalah transformasi substitusi alfabetik, n adalah jumlah huruf yang digeser, dan Pi adalah karakter-karakter *plaintext*. Disk mempunyai peranan penting dalam kriptografi sekitar 500 th yang lalu. Di Italia sekitar tahun 1460, Leon Battista Alberti mengembangkan disk *cipher* untuk enkripsi. Sistemnya terdiri dari dua disk konsentris. Setiap disk memiliki alfabet di sekelilingnya, dan dengan memutar satu disk berhubungan dengan yang lainnya, huruf pada satu alfabet dapat ditransformasi ke huruf pada alfabet yang lain.

2.3. Taksonomi Primitif-primitif Kriptografi

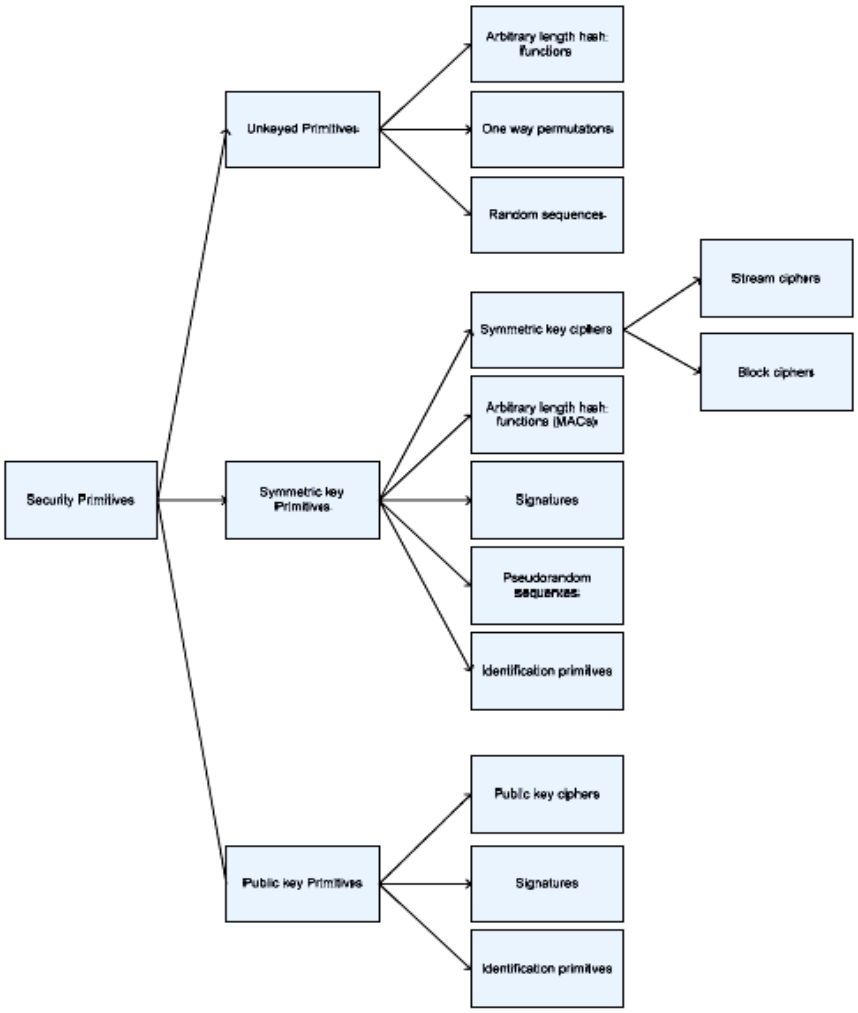
Ada beberapa dasar tool kriptografi (primitif) yang digunakan untuk mendukung keamanan informasi. Contoh dari primitif termasuk skema enkripsi, fungsi hash, dan skema tanda tangan digital. Primitif-primitif ini harus dapat dievaluasi berdasarkan beberapa kriteria seperti:

1. Level keamanan. Hal ini biasanya sulit untuk dihitung. Sering diwakili dengan jumlah operasi yang dibutuhkan (menggunakan metode terbaik yang diketahui) untuk melawan tujuan yang diharapkan. Level keamanan biasanya didefinisikan *work factor.*
2. *Fungsionalitas.* Primitif-primitif dibutuhkan untuk memenuhi tujuan keamanan informasi yang bermacam-macam. Primitif mana yang paling efektif untuk tujuan yang diberikan akan ditentukan dengan properti dasar dari primitif.
3. Metode operasi. Primitif, saat diterapkan dengan bermacam cara dan dengan bermacam input, biasanya akan menunjukkan karakteristik yang berbeda, sehingga satu primitif dapat menyediakan fungsionalitas yang sangat berbeda pada mode operasi atau penggunaannya.
4. Unjuk kerja. Merupakan efisiensi sebuah primitif pada mode tertentu. (sebagai contoh algoritma enkripsi dapat dihitung dengan jumlah bit per detik yang dapat dienkripsinya).
5. Kemudahan implementasi. Merupakan kesulitan dalam merealisasikan primitif pada prakteknya. Dapat meliputi kompleksitas pengimplementasian primitif dalam lingkungan *software* maupun *hardware*.

Kepentingan relatif dari bermacam kriteria ini sangat tergantung pada aplikasi dan sumber daya yang tersedia.

2.4. Enkripsi Kunci Rahasia

Secret-key cryptography kadang disebut sebagai *symmetric cryptography* merupakan bentuk kryptografi yang lebih tradisional, dimana sebuah kunci tunggal dapat digunakan untuk mengenkrip dan mendekrip pesan. *Secret-key cryptography* tidak hanya berkaitan dengan enkripsi tetapi juga berkaitan dengan otentikasi, disebut juga *message authentication codes*. (*Lihat Gambar.2.1. _Taksonomi primitif kriptografi)*



Gambar 2.1._Taksonomi primitif kriptografi

Masalah utama yang dihadapi *secret-key cryptography* adalah membuat pengirim dan penerima menyetujui kunci rahasia tanpa ada orang lain yang mengetahuinya. Ini membutuhkan metode dimana dua pihak dapat berkomunikasi tanpa takut akan disadap. Kelebihan *secret-key cryptography* dari *public-key cryptography* adalah lebih cepat. Teknik yang paling umum dalam *secret-key cryptography* adalah *block ciphers*, *stream ciphers*, dan message authentication codes.

Berdasarkan jenis kunci yang digunakannya, algoritma kriptografi dikelompokan menjadi dua bagian, yaitu

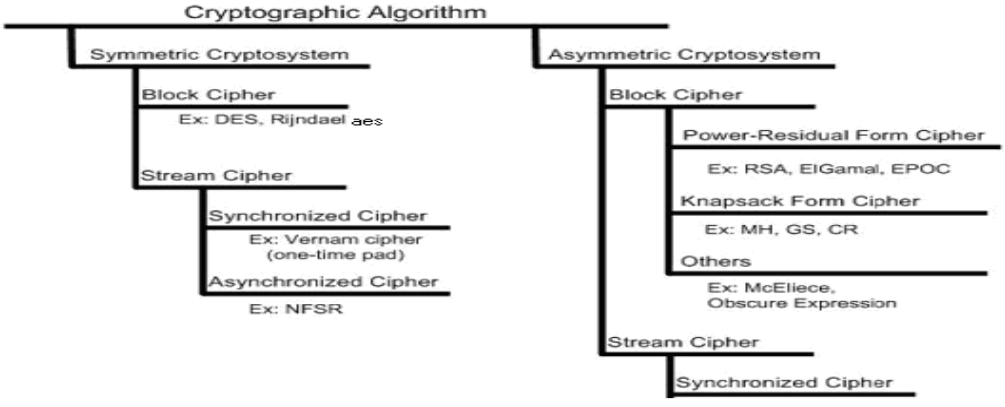
1. *Symmetric Algorithm*

Symmetric algorithm atau disebut *juga secret key algorithm* adalah algoritma yang kunci enkripsinya dapat dihitung dari kunci dekripsi dan begitu pula sebaliknya, kunci dekripsi dapat dihitung dari kunci enkripsi. Pada sebagian besar *symmetric algorithm* kunci enkripsi dan kunci dekripsi adalah sama. *Symmetric algorithm* memerlukan kesepakatan antara pengirim dan penerima pesan pada suatu kunci sebelum dapat berkomunikasi secara aman. Keamanan *symmetric algorithm* tergantung pada rahasia kunci. Pemecahan kunci berarti memungkinkan setiap orang dapat mengenkripsi dan mendekripsi pesan dengan mudah. *Symmetric algorithm* dapat dikelompokkan menjadi dua jenis, yaitu *stream cipher* dan *block cipher*. *Stream cipher* beroperasi bit per bit (atau byte per byte) pada satu waktu. Sedangkan *block cipher* beroperasi per kelompokkelompok bit yang disebut blok (*block*) pada satu waktu.

2. *Asymmetric Algorithm*

Asymmetric algorithm atau disebut juga *public key algorithm* didesain agar memudahkan dalam distribusi kunci yang digunakan untuk enkripsi dan dekripsi. Kunci dekripsi pada *public key algorithm* secara praktis tidak dapat dihitung dari kunci enkripsi. Algoritma ini disebut "*public key*" karena kunci dapat dibuat menjadi publik. Setiap orang dapat menggunakan kunci enkripsi untuk mengenkripsi pesan, tetapi hanya orang yang memiliki kunci dekripsi yang dapat mendekripsi pesan tersebut. Pada sistem ini kunci enkripsi sering disebut kunci publik (*public key*), dan kunci dekripsi disebut kunci rahasia (*private key*).

Teknik kriptografi modern yang ada saat ini dapat dikelompokkan sebagaimana ditunjukkan (*Gambar 2.2. Pengelompokkan enkripsi beserta contoh).* Pada bagian ini akan didiskusikan operasi-operasi penyandian dasar untuk memberikan dasar bagi pemahaman tentang evolusi metode-metode enkripsi dan usaha-usaha cryptanalysis yang berkaitan.



Gambar 2.2 . Pengelompokkan enkripsi beserta contoh

2.4.1. Substitusi

Caesar cipher adalah *cipher substitusi* sederhana yang mencakup pergeseran alfabet 3 posisi ke kanan. Caesar cipher merupakan subset dari cipher polialfabetik Vigenere. Pada Caesar cipher karakter-karakter pesan dan pengulangan kunci dijumlahkan bersama, modulo 26. Dalam penjumlahan modulo 26, huruf-huruf A-Z dari alfabet masing-masing memberikan nilai 0 sampai 25. Tipe *cipher* ini dapat diserang menggunakan analisis frekuensi. Dalam frekuensi analisis, digunakan karakteristik frekuensi yang tampak dalam penggunaan huruf-huruf alfabet pada bahasa tertentu. Tipe cryptanalysis ini dimungkinkan karena Caesar *cipher* adalah monoalfabetik *cipher* atau *cipher* substitusi sederhana, dimana karakter *ciphertext* disubstitusi untuk setiap karakter *plaintext*. Serangan ini dapat diatasi dengan menggunakan substitusi polialfabetik. Substitusi polialfabetik dicapai melalui penggunaan beberapa cipher substitusi. Namun substitusi ini dapat diserang dengan penemuan periode, saat substitusi berulang kembali.

2.4.2. Transposisi (Permutasi)

Pada *cipher* ini, huruf-huruf *plaintext* dipermutasi. Sebagai contoh, huruf-huruf plaintext A T T A C K A T D A W N dapat dipermutasi menjadi D C K A A W N A T A T T. *Cipher* transposisi kolumnar adalah *cipher* dimana *plaintext* ditulis secara horisontal pada kertas dan dibaca secara vertikal. *Cipher* transposisi dapat diserang melalui analisis frekuensi, namun *cipher* menyembunyikan properti statistik dari pasangan huruf-huruf, seperti IS dan TOO.

2.4.3. Vernam Cipher (One Time Pad)

Cipher ini diimplementasikan melalui sebuah kunci yang terdiri dari sekumpulan *random* karakter-karakter yang tidak berulang. Setiap huruf kunci dijumlahkan modulo 26 dengan huruf pada *plaintext.* Pada One Time Pad, tiap huruf kunci digunakan satu kali untuk satu pesan dan tidak digunakan kembali. Panjang *stream* karakter kunci sama dengan panjang pesan.

2.4.4. Book Key Cipher / Running Key Cipher

Cipher ini menggunakan teks dari sebuah sumber (misalnya buku) untuk mengenkripsi *plaintext*. Kunci, diketahui oleh pengirim dan penerima yang dimaksud, dapat berupa halaman dan jumlah baris dari teks pada buku. Teks ini adalah karakter yang sesuai untuk karakter dengan *plaintext*, dan penjumlahan modulo 26 dijalankan untuk memperngaruhi enkripsi. *Running key cipher* mengeliminasi periodisitas, namun masih dapat diserang dengan memanfaatkan redundansi pada kunci.

2.4.5. Codes

Codes berkaitan dengan kata-kata dan frase dan menghubungkan kata-kata ini sebagai frase untuk sekelompok angka atau huruf. Sebagai contoh, angka 526 dapat berarti “Attack at dawn”.

2.4.6. Steganography

Steganography adalah seni menyembunyikan keberadaan pesan. "steganography" berasal dari kata Yunani "steganos" yang berarti "terlindungi", dan "graphein" yang berarti "menulis". Sebuah contohnya adalah microdot, yang mengkompresi pesan kedalam ukuran period atau dot. Steganography dapat digunakan untuk membuat "*watermark*" digital untuk mendeteksi penyalinan image digital secara ilegal.

2.5. Tujuan Kriptografi

Dalam teknologi informasi telah dan sedang dikembangkan cara-cara untuk menangkal berbagai serangan, seperti penyadap dan pengubahan data yang sedang dikirimkan. Salah satu cara yang ditempuh untuk mengatasi masalah ini adalah dengan menggunakan kriptografi yang menggunakan transformasi data sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak yang tidak berhak mengakses. Transformasi ini memberikan solusi pada dua macam masalah keamanan data, yaitu masalah privasi *(privacy*) dan keotentikan (*authenticatioan*).

Privacy mengandung arti bahwa data yang dikirimkan hanya dapat dimengerti oleh penerima yang sah atau berhak. Sedangkan keotentikan mencegah pihak ketiga untuk mengirimkan data yang salah atau mengubah data yang dikirimkan.

Ada empat tujuan mendasar dari ilmu kriptografi yang merupakan aspek keamanan informasi yaitu :

1. Kerahasiaan

Merupakan layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atau mengupas informasi yang telah disandi.

2. Integritas data

Berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.

3. Autentikasi

Berhubungan dengan identifikasi / pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.

4. Non-repudiasi

Usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan / membuat.

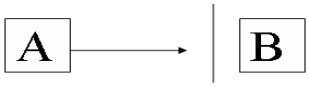
2.6. Pola-pola Penyaringan Data

Proteksi data dan informasi dalam komunikasi komputer menjadi penting karena nilai informasi itu sendiri dan meningkatnya penggunaan komputer di berbagai sektor. Melihat kenyataan semakin banyak data yang diproses dengan komputer dan dikirim melalui perangkat komunikasi elektronik maka ancaman

terhadap pengamanan data akan semakin meningkat. Beberapa pola ancaman terhadap komunikasi data dalam komputer dapat diterangkan sebagai berikut :

1. Interruption

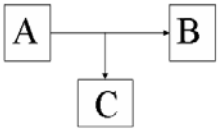
Interception terjadi bila data yang dikirimkan dari A tidak sampai pada orang yang berhak B. *Interruption* merupakan pola penyerangan terhadap sifat *availability* (ketersediaan data) seperti pada *(Gambar 2.3 . Interruption).*



Gambar 2.3 . Interruption

2. Interception

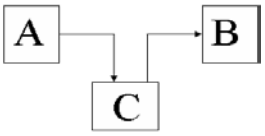
Serangan ini terjadi bila pihak ketiga C berhasil membaca data yang dikirimkan. *nterception* merupakan pola penyerangan terhadap sifat *confidentiality* (kerahasiaan data) seperti pada *(Gambar 2.4 Interception).*



Gambar 2.4. Interception

3. Modification

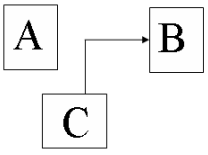
Pada serangan ini pihak ketiga C berhasil merubah pesan yang dikirimkan. *Modification* merupakan pola penyerangan terhadap sifat *integrity* (keaslian data) seperti pada (*Gambar* 2.5 *Modification).*



Gambar 2.5. Modification

4. Fabrication

Pada serangan ini, penyerang berhasil mengirimkan data ke tujuan dengan memanfaatkan identitas orang lain. *Fabrication* merupakan pola penyerangan terhadap sifat *authenticity* seperti pada *(Gambar 2.6 Fabrication).*



Gambar 2.6. Fabrication

Ancaman-ancaman tersebut di atas menjadi masalah terutama dengan semakin meningkatnya komunikasi data yang bersifat rahasia seperti: pemindahan dana secara elektronik kepada dunia perbankan atau pengiriman dokumen rahasia pada instansi pemerintah. Untuk mengantisipasi ancaman-ancaman tersebut perlu dilakukan usaha untuk melindungi data yang dikirim melalui saluran komunikasi salah satunya adalah dengan teknik enkripsi. Dan untuk masalah kekuatan pengamanannya tergantung pada algoritma metode enkripsi tersebut dan juga kunci yang digunakan di dalamnya.

2.7. Sejarah AES (Advance Encryption Standard)

Setelah hampir 20 Tahun lebih semenjak DES (*Data Encryption Standard*) diciptakan, DES menjadi tulang punggung bagi algoritma enkripsi standar yang digunakan di dunia internasional. Hingga akhirnya sebelum masa itu habis telah

banyak muncul tanda-tanda akan kejatuhannya, hal ini diawali oleh Eli Biham dan Adi Shamir yang memperkenalkan *Differential Cryptanalysis* pada tahun 1990 untuk melakukan *attack* terhadap DES dan hasilnya pun cukup mengejutkan *attack* ini dapat me*recover* 48 bit kunci untuk DES dengan jumlah round kurang dari 16-round. Selain itu attack ini terbukti efektif dibandingkan dengan *brute force attack* untuk jumlah round yang kurang dari 16-round.

Setelah terbukti bahwa DES dapat dipecahkan, banyak kalangan mulai meragukannya. Seiring dengan perkembangan tersebut banyak bermunculan algoritma baru yang diharapkan menjadi pengganti DES, diantaranya varian atau pengembangan dari DES sendiri (Triple DES, DES-x, LOKI, S2 DES), FEAL, IDEA, MISTY, RC5 dan lain sebagainya. Namun semuanya gagal mencuri hati NIST (*National Institute of Standards Technology*) untuk menjadikan salah satu diantaranya menjadi algoritma standar yang baru menggantikan DES.

Pada awal januari 1997, NIST mengumumkan diadakannya kontes pemilihan algoritma satandar terbaru menggantikan DES dengan harapan algoritma standar yang baru dapat bertahan paling tidak 30 tahun serta mampu memproteksi informasi rahasia selama 100 tahun. Kemudian pada bulan april pada tahun yang sama, NIST mengumumkan *requirements* yang harus dipenuhi algoritma enkripsi standar yang baru, antara lain :

1. *Block Cipher*
2. Panjang blok 128-bit
3. Panjang kunci yang bervariasi (128/ 192/ 256-bit)

4. Memiliki kekuatan yang sama atau lebih baik dari Triple DES serta memiliki tingkat efisiensi yang cukup signifikan
5. Dapat diimplementasikan dalam dua bahasa pemrograman yaitu bahasa C dan Java
6. Jika algoritma tersebut terpilih maka akan menjadi standar publik yang tidak menerima royalti

Setelah melalui perjalanan panjang terdapat 15 algoritma yang menjadi finalis AES, algoritma yang dibuat memiliki struktur umum *(Tabel 2.1. 15 algoritma finalis AES)*.

Tabel 2.1.

15 algoritma finalis AES

Nama algoritma	Pengirim	Negara	Tipe	Rounds	Penggunaan Fungsi
CAST - 256	Entrust	Kanada	Ext. Feistel	48	-
Crypton	Future Systems	Korea	Square	12	DES
Deal	Outerbridge	Kanada	Feistel	6, 8, 8	Decorelation modules
DFC	ENS – CNRS	Perancis	Feistel	8	
E_2	NTT	Jepang	Feistel	12	BombPermutation
Frog	Tec Apro	Kosta Rika	Special	8	Hasty Pudding
HPC	Schroeppel	USA	Omni	8	
LOKI 97^{TM}	Brown, Pieprzyk, Seberry	Australia	Feistel	16	
Magenta	Deutsche Telekom	Jerman	Feistel	6, 6, 8	
Mars	IBM	USA	Ext. Feistel	32	Var rot, multi, non-crypt, round
RC6	RSA	USA	Feistel	20	Var rot, multi,(perkalian)
Rijindael	Daeman, Rijmen	Belgia	Square	10, 12, 14	
Safer+	Cylink	USA	SP Network	8, 12, 16	PHT
Serpent	Anderson, Biham, Knudsen	UK, Israel , Norway	SP Network	32	Bitsslice
Twofish	Counterpane	USA	Feistel	16	

Setelah melalui perjalanan panjang sejak diumumkan pada tahun 1997, akhirnya dilakukan proses analisis antar para kontestan maupun publik yang *aware* terhadap pemilihan AES. Pada hari pertama tercatat bahwa pertemuan tersebut dihadiri oleh 180 partisipan (termasuk NIST) yang berasal dari 23 negara. Sekitar 28 paper yang diterima dan 21 paper dipresentasikan. Hampir seluruh kandidat algoritma saling mengungkapkan kelemahan satu sama lain namun hal yang cukup mengejutkan adalah Rijndael, tidak terdapat satupun komentar negatif yang ditujukan pada algoritma ini. Kemudian komentar datang dari Eli Biham mengenai estimasi jumlah round minimum yang harus dipenuhi supaya algoritma tersebut dinyatakan ”*secure”*, dalam artian bahwa attack yang dilakukan memiliki kompleksitas yang lebih kecil daripada ”*exhaustive* key search”*(Tabel 2.2. algoritma finalis AES – testing)*.

Tabel 2.2

Algoritma Finalis Aes – Testing

Cipher	Rounds	Cycles	Min.Rounds	Now Cycles
Serpent	32	1800	17	956
Mars	32	1600	20	1000
Rijindael	10	1276	8	1021
Twofish	16	1254	14	1097
Crypton	12	1282	11	1175
E_2	12	1808	10	1507
RC6	20	1436	21	1508
CAST - 256	48	2088	40	1740
Safer+	8	4424	7	3871
DFC	8	5874	9	6608
Deal	6	8950	10	14917
LOKI 97™	16	6376	38	15143
Magenta	6	23186	11	42508
Frog	8	2182	?	
HPC	8	2602	?	

Dari 15 algoritma yang masuk pada awalnya setelah melalui proses yang cukup ketat akhirnya algoritma yang dinyatakan sebagai finalis teridiri dari 5 algoritma. (*Tabel 2.3. 5 finalis AES)*.

Tabel 2.3

5 finalis AES

Algoritma	Original Round	Minimal Round	Speed With min. Rounds
Mars	32	20	1.00
RC6	20	21	0.663
Rijindael	10	8	0.98
Serpent	32	17	1.04
Twofish	16	14	0.911

Kemudian untuk menyederhanakan tugas yang sangat sulit untuk menentukan pemenang dari lima finalis algoritma *(Tabel 2.4 Metrik Penilaian 5 Finalis AES berdasarkan parameter NIST),* maka dibuatlah paremeter / metrik ukur yang menggunakan skala nilai 100.

Tabel 2.4

Metrik Penilaian 5 Finalis AES berdasarkan parameter NIST

metric / algoritma	MARS	RC6 ™	Rjindael	Serpent	Twofish
Algoritma design & presentation (10 poin)	8	10	8	8	8
Security (30 poin)	28	29	25	28	27
Ease of implementation (10 poin)	8	9	7	8	7
Usage flexibility (10 poin)	8	8	7	8	8
Performance / eficiency (10 poin)	8	9	8	7	9
Performance on smartcards (10 poin)	8	7	9	9	8
Strength against cryptanalisis (10 poin)	8	9	7	8	9
future resilience (10 poin)	8	8	7	9	8
total	84	89	78	85	84

Terdapat delapan parameter dengan bobot nilai yang terdistribusi pada masing – masing parameter berdasarkan tujuan NIST [4]. Parameter tersebut antara lain :

1. Desain Algoritma & *Presentation* (10 poin)

Penilaian pada kriteria ini mencakup kesederhanaan, *cryptographic core* yang mudah dipahami secara jelas, fungsi round dan desain rasional yang mendasari pembuatan algoritma tersebut

2. *Security* (30 poin)

Penilaian ini mencakup sifat menurun dari algoritma ketika digunakan sebagai *block cipher*. Hal ini meliputi penggunaan *cryptographic core* yang kuat, teruji dan terpercaya, yang terintegrasi dengan tepat pada fungsi enkripsi/ dekripsi, *key scheduling* yang tepat, tidak adanya operasi yang lemah, diffusi yang sesuai, *whitening*, tidak adanya trapdor, tidak adanya *weak keys* atau semi *weak keys*. Hal lain yang juga diperhatikan adalah jaminan mengenai pemetaan satu-satu plainteks ke cipherteks menggunakan kunci yang sama dan pemetaan satu – satu yang berbeda menggunakan kunci yang berbeda.

3. Kemudahan Implementasi (Ease of Implementation, 10 poin)

Hal ini sangat dipengaruhi oleh kesederhanaan dan kapabilitas dalam *coding* algoritma agar dapat berjalan pada *hardware*, *software*, dan *operating system* yang berbeda.

4. Fleksibiltas Penggunaan (Usage Flexibility, 10 poin)

Fleksibilitas yang dimaksud dalam proses ini adalah penggunaan algoritma tersebut sebagai fungsi Hash, *Pseudo Random Number Generator, Message Authentication Codes* dan *Stream Cipher*. Hampir sebagian besar finalis memenuhi kriteria ini.

5. Performance/ Computational Efficiency (10 poin)

Hal ini mengenai karakteristik performance algoritma pada platforms yang berbeda dengan konfigurasi yang berbeda (ukuran kunci, ukuran blok, jumlah round). Pada uji kriteria ini Twofish dan RC6™ menunjukkan performa yang baik pada semua platform.

6. Performance/ Adaptability on Smart Cards (10 poin)

Hal ini mencakup *performance* algoritma dan adaptibilitas pada Smart Cards. Rijndael dan Serpent bekerja lebih baik pada platform ini.

7. Demonstrated/Expected strength against Cryptanalysis (10 poin)

Kriteria ini mencakup penilaian mengenai kekuatan algoritma terhadap linier, non-linier, statistical dan *differential cryptanalysis* termasuk *power* dan *timing attack*. Selain itu hal ini juga mencakup *interpolation attack, partial key guessing, related key cryptanalysis, distributed key search, short-cut attacks seperti visual cryptanalysis*. Ini merupakan tambahan 30 poin untuk keseluruhan security

8. Future Resilience (10 poin)

Penilaian ini ini meliputi fleksibiltas untuk digunakan dengan ukuran blok/kunci yang disesuaikan, paralelisme menurun yang dapat dieksploitasi (*exploitable inherent parallelism*), kesesuaian dengan arsitektur masa depan, serta *operting system.*

Akhirnya pada tahun 2000 Rijndael ditetapkan sebagai AES kemudian pada tanggal 26 November 2001 ditetapkan Rijndael sebagai standar baru AES melalui FIPS 197 (*Federal Information Processing Standards Publications*). Pembaca dapat menyimpulkan sendiri dari data dukung yang disajikan apakah Rijndael yang terpilih cukup menjadi jawaban untuk menjadi algoritma enkripsi yang *reliable* atau hanya policy NSA yang memiliki peran dan kepentingan yang cukup besar untuk tetap dapat memonitor seluruh *traffic* pertukaran informasi.

2.8. Algoritma AES

Input dan output dari algoritma AES 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 AES terdiri dari *key* dengan panjang 128 bit, 192 bit, atau 256 bit.

2.8.1. Pengantar Matematis

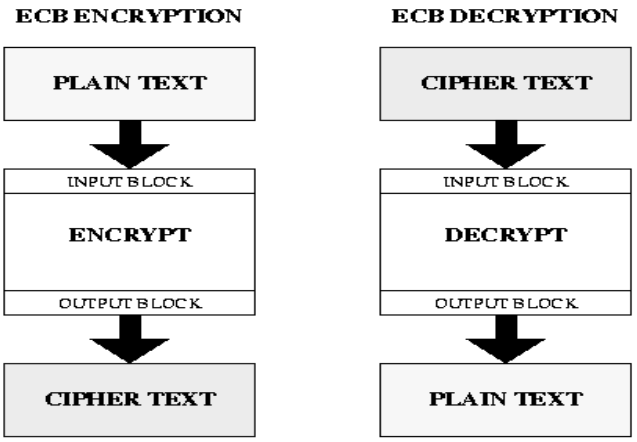
Seluruh byte dalam algoritma AES diinterpretasikan sebagai elemen *finite field*. Elemen *finite field* ini dapat dikalikan dan dijumlahkan, tetapi hasil dari penjumlahan dan perkalian elemen *finite field* sangat berbeda dengan hasil dari penjumlahan dan perkalian bilangan biasa.

2.8.2. Penyandian Blok

Penyandian blok pada dasarnya adalah proses penyandian terhadap blok data yang jumlahnya sudah ditentukan. Untuk sistem penyandian blok terdapat empat jenis mode operasi, yaitu *Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB).*

1. **Electronic Code Book (ECB)**

Mode ECB adalah mode yang paling umum dan paling mudah untuk diimplementasikan. Cara yang digunakan adalah dengan membagi data ke dalam blok-blok data terlebih dahulu yang besarnya sudah ditentukan. Blok-blok data inilah yang disebut *plaintext* karena blok data ini belum disandikan. Proses enkripsi akan langsung mengolah *plaintext* menjadi *ciphertext* tanpa melakukan operasi tambahan. Suatu blok *plaintext* yang dienkripsi dengan menggunakan kunci yang sama akan menghasilkan *ciphertext* yang sama.*(Gambar 2.7 Mode Operasi ECB).*

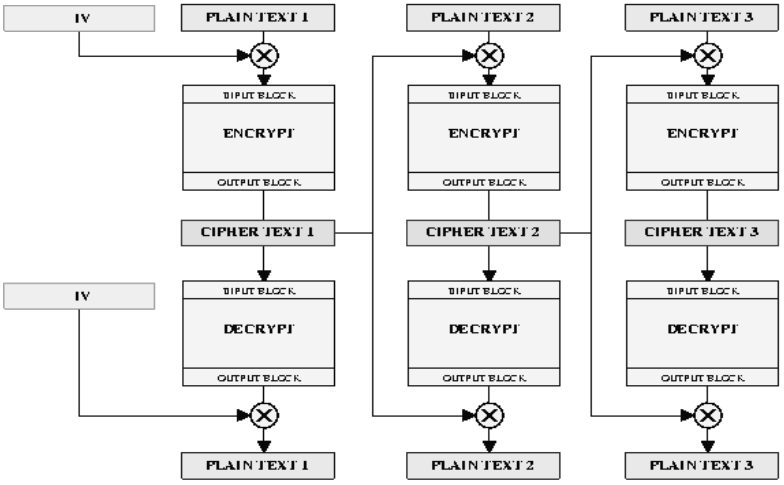


Gambar 2.7. Mode Operasi ECB

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 *kriptanalis* dapat menyusun suatu *code book* tanpa perlu mengetahui kuncinya.

2. Cipher Block Chaining (CBC)

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 blok berikutnya. Dengan kata lain, setiap blok *ciphertext* dipakai untuk memodifikasi proses enkripsi dan dekripsi pada blok berikutnya.(*Gambar 2.8 Mode Operasi CBC*).



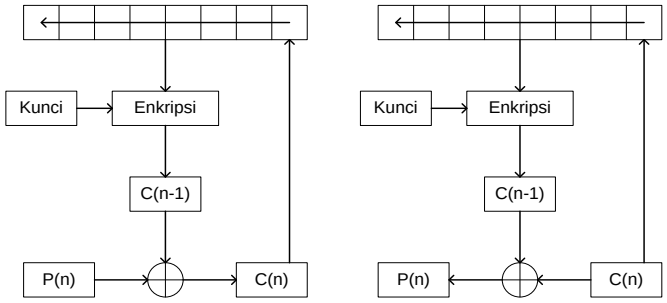
Gambar 2.8. Mode Operasi CBC

Pada CBC diperlukan data acak sebagai blok 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.

3. Cipher Feedback (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 blok. Misalkan pada CFB 8 bit, maka data akan diproses tiap 8 bit.*(Gambar 2.9 Mode Operasi CFB).*

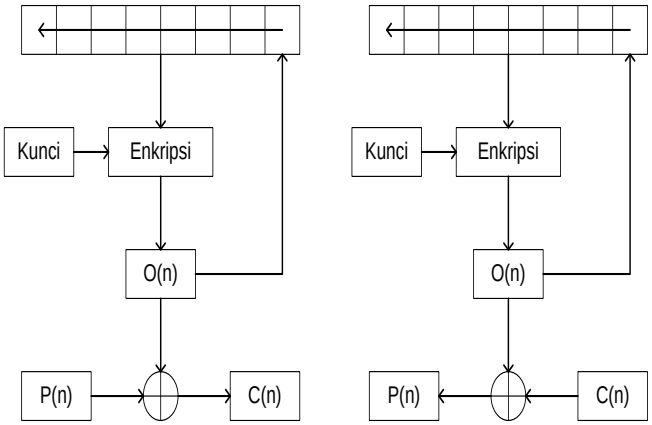


Gambar 2.9. Mode Operasi CFB

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.

4. Output Feedback (OFB)

Sama pada mode CFB, mode OFB juga memerlukan sebuah *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.*(Gambar 2.10 Mode Operasi OFB)*



Gambar 2.10. Mode Operasi OFB

2.8.3. Algoritma AES - *Rijndael*

Pada algoritma AES, jumlah blok input, blok output, dan *state* adalah 128 bit. Dengan besar data 128 bit, berarti *Nb* = 4 yang menunjukkan panjang data tiap baris adalah 4 byte. Dengan panjang kunci 128-bit, maka terdapat sebanyak $3{,}4 \times 1038 = 2128$ kemungkinan kunci. Jika komputer tercepat dapat mencoba 1 juta kunci setiap detik, maka akan dibutuhkan waktu $5{,}4 \times 1024$ tahun untuk mencoba seluruh kunci.

Jika tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu 5,4 × 1018 tahun untuk mencoba seluruh kunci

Dengan blok input atau blok data sebesar 128 bit, *key* yang digunakan pada algoritma AES tidak harus mempunyai besar yang sama dengan blok input. *Cipher key* pada algoritma AES bisa menggunakan kunci dengan panjang 128 bit, 192 bit, atau 256 bit. Perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan diimplementasikan pada algoritma AES ini. Di bawah ini adalah Tabel yang memperlihatkan jumlah *round* (*Nr*) yang harus diimplementasikan pada masing-masing panjang kunci (*Tabel 2.5. Perbandingan jumlah Round dan Key*).

Tabel 2.5

Perbandingan jumlah Round dan Key

	Panjang Kunci (*Nk words*)	Ukuran Blok (*Nb words*)	Jumlah Putaran (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Catatan: 1 *word* = 32 bit

Tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte*. Setiap putaran mengunakan kunci internal yang berbeda (disebut *round key*). *Enciphering* melibatkan operasi substitusi dan permutasi.

1. Ekspansi Kunci

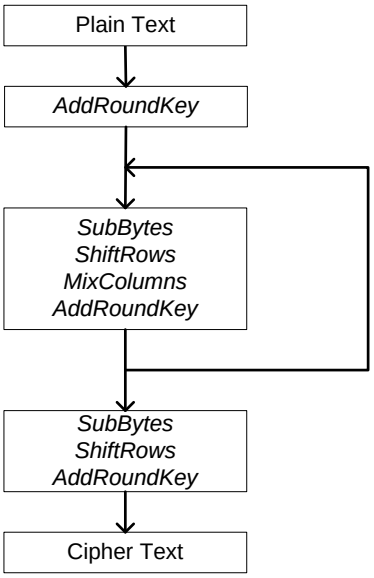
Algoritma AES mengambil kunci *cipher*, K, dan melakukan rutin ekspansi kunci (*key expansion*) untuk membentuk *key schedule*. Ekspansi kunci menghasilkan

total Nb(Nr+1) *word*. Algoritma ini membutuhkan set awal *key* yang terdiri dari *Nb word*, dan setiap *round Nr* membutuhkan data kunci sebanyak *Nb word*. Hasil *key schedule* terdiri dari array 4 byte word linear yang dinotasikan dengan [w_i]. *SubWord* adalah fungsi yang mengambil 4 byte *word* input dan mengaplikasikan S-Box ke tiap-tiap data 4 byte untuk menghasilkan *word* output. Fungsi *RotWord* mengambil *word* [a_0, a_1, a_2, a_3] sebagai input, melakukan permutasi siklik, dan mengembalikan *word* [a_1, a_2, a_3, a_0]. Rcon[i] terdiri dari nilai-nilai yang diberikan oleh [x^{i-1}, {00}, {00}, {00}], dengan x^{i-1} sebagai pangkat dari x (x dinotasikan sebagai {02} dalam *field* $GF(2^8)$. W*ord* ke *Nk* pertama pada ekspansi kunci berisi kunci *cipher*. Setiap word berikutnya, w[i], sama dengan XOR dari *word* sebelumnya, w[i-1] dan *word Nk* yang ada pada posisi sebelumnya, w[i-Nk]. Untuk *word* pada posisi yang merupakan kelipatan *Nk*, sebuah transformasi diaplikasikan pada w[i-1] sebelum XOR, lalu dilanjutkan oleh XOR dengan konstanta *round*, Rcon[i]. Transformasi ini terdiri dari pergeseran siklik dari byte data dalam suatu word *RotWord*, lalu diikuti aplikasi dari *lookup Tabel* untuk semua 4 byte data dari word *SubWord*.

2. Enkripsi

Proses enkripsi pada algoritma AES 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 AES disebut sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *state* tidak mengalami transformasi MixColumns. (*Gambar 2.11 Diagram Alir Proses Enkripsi*).



Gambar 2.11. Diagram Alir Proses Enkripsi

a. SubBytes

SubBytes merupakan transformasi byte dimana setiap elemen pada state akan dipetakan dengan menggunakan sebuah Tabel substitusi (*S-Box*). Hasil yang didapat dari pemetaan dengan menggunakan Tabel *S-Box (Tabel 2.6 substitusi (S-Box))* ini sebenarnya adalah hasil dari dua proses transformasi bytes, yaitu :

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".

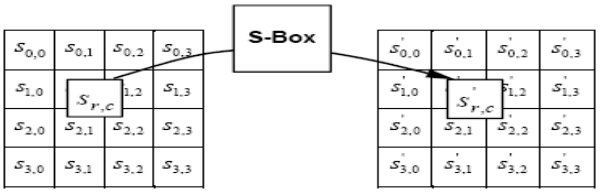
2. Transformasi *affine* pada *state* yang telah dipetakan. Transformasi *affine* ini apabila dipetakan dalam bentuk matriks adalah sebagai berikut :

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

$b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$ adalah urutan bit dalam elemen state atau array byte dimana b_7 adalah *most significant bit* atau bit dengan posisi paling kiri. (*Gambar 2.12 subbytes()*)

Tabel 2.6

Substitusi (S-Box)

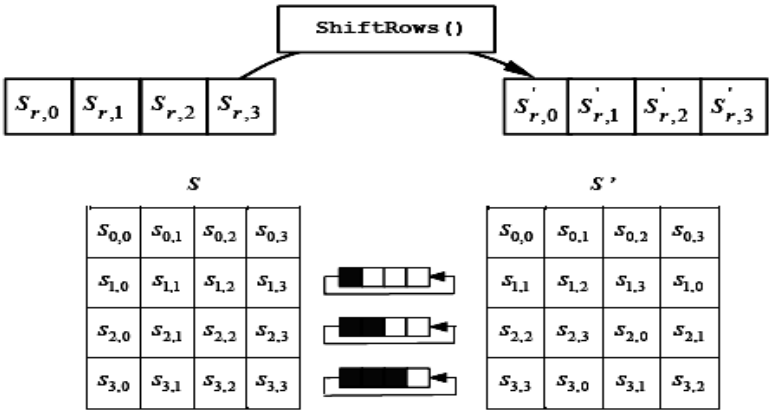


x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 2.12. Subbytes ()

b. ShiftRows

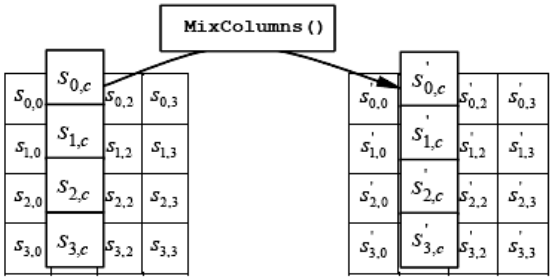
Transformasi Shiftrows pada dasarnya adalah proses pergeseran bit dimana bit paling kiri akan dipindahkan menjadi bit paling kanan (rotasi bit). Transformasi ini diterapkan pada baris 2, baris 3, dan baris 4. Baris 2 akan mengalami pergeseran bit sebanyak satu kali, sedangkan baris 3 dan baris 4 masing-masing mengalami pergeseran bit sebanyak dua kali dan tiga kali. (*Gambar 2.13 Transformasi shiftRows()*)



Gambar 2.13. Transformasi shiftRows()

c. MixColumns

MixColumns mengoperasikan setiap elemen yang berada dalam satu kolom pada *state* (*Gambar 2.14 : MixColumns()*). Elemen pada kolom dikalikan dengan suatu polinomial tetap a(x) = {03}x^3 + {01}x^2 + {01}x + {02}.



Gambar 2.14. MixColumns()

Secara lebih jelas, transformasi mixcolumns dapat dilihat pada perkalian matriks berikut ini :

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

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 :

$$s'_{0,c} = (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Mixcolumns membutuhkan tabel penunjang dalam memperoleh hasil tabel tersebut adalah tabel E dan tabel L. *(Tabel E dan Tabel L)*

Tabel 2.7

Tabel E dan Tabel L

E Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

L Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

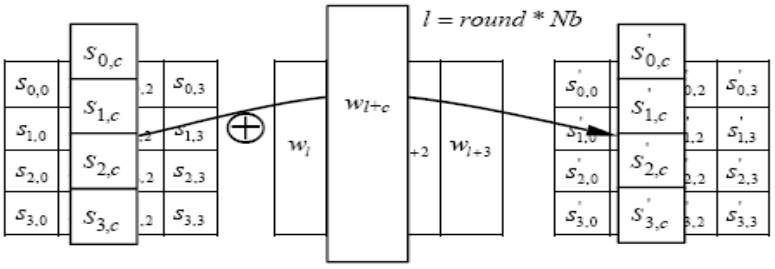
d. AddRoundKey

Pada proses AddRoundKey, sebuah *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 :

$$\left[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}\right] = \left[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}\right] \oplus \left[w_{round*Nb+c}\right] \text{untuk } 0 \le c \le Nb$$

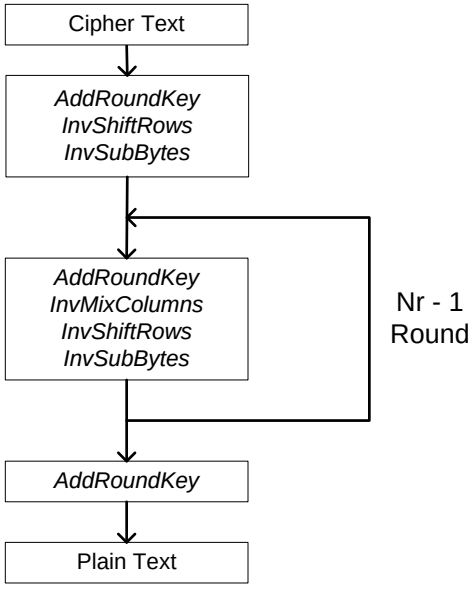
[w_i] adalah *word* dari key yang bersesuaian dimana *i = round*Nb+c*. Transformasi AddRoundKey diimplementasikan pertama kali pada *round* = 0, dimana *key* yang digunakan adalah *initial key* (*key* yang dimasukkan oleh kriptografer dan belum mengalami proses *key expansion*). (*Gambar 2.15. AddRoundKey ()*)



Gambar 2.15. AddRoundKey ()

3. Dekripsi

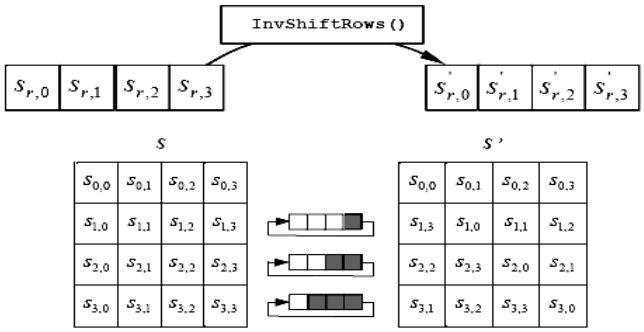
Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada invers *cipher* adalah InvShiftRows, InvSubBytes, InvMixColumns, dan AddRoundKey. Algoritma dekripsi dapat dilihat pada skema berikut ini : (*Gambar 2.16. Diagram Alir Proses Dekripsi*).



Gambar 2.16. Diagram Alir Proses Dekripsi

e. InvShiftRows

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi ShiftRows. Pada transformasi InvShiftRows, 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. (*Gambar 2.17 Transformasi InvShiftRows()*)



Gambar 2.17. Transformasi InvShiftRows()

f. InvSubBytes

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada InvSubBytes, 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 (*Tabel 2.8. Inverse S-box*), yaitu transformasi *affine* terlebih dahulu, baru kemudian perkalian *invers* dalam GF(2^8).

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

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

Tabel 2.8.

Inverse S-box

		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

g. InvMixColumns

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

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

Atau dalam matriks :

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

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

Hasil dari perkalian diatas adalah :

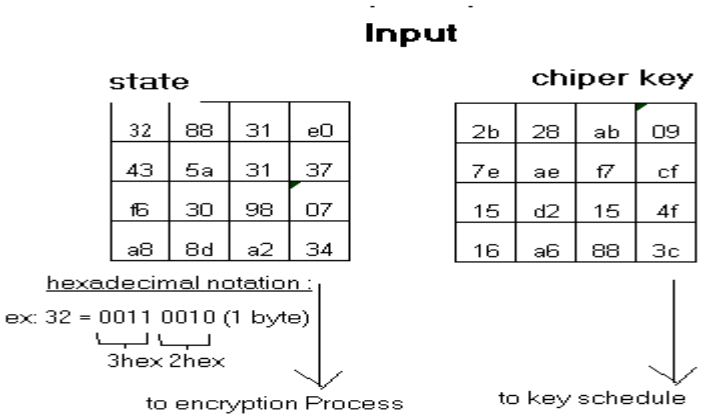
$$s'_{0,c} = (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,c} = (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c})$$
$$s'_{2,c} = (\{0D\} \bullet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

h. 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.

4. Contoh AES

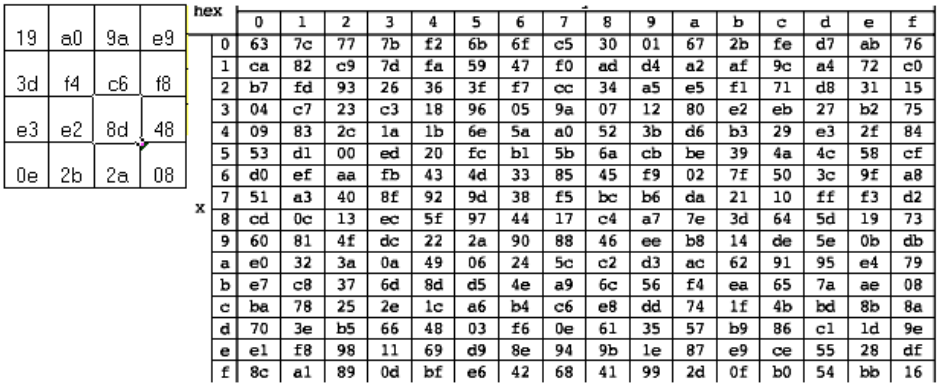
Berikut ini adalah contoh penerapan proses enkripsi AES lihat (*Gambar 19 : elemen state dan kunci dalam notasi HEX*). Pada rounds pertama *plaintext* dan *key* yang bernotasi *hexadecimal* di XOR.



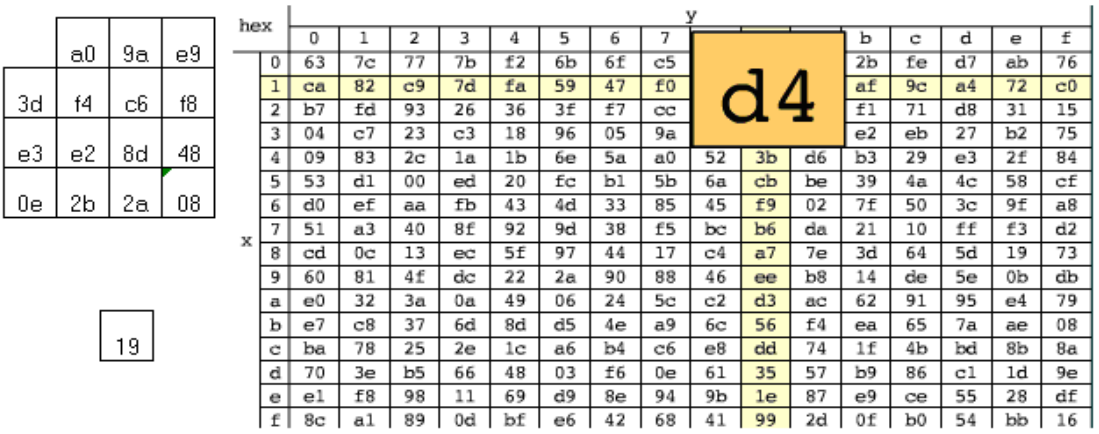
Gambar 2.18. elemen state dan kunci dalam notasi HEX

a. Transformasi *SubBytes()*

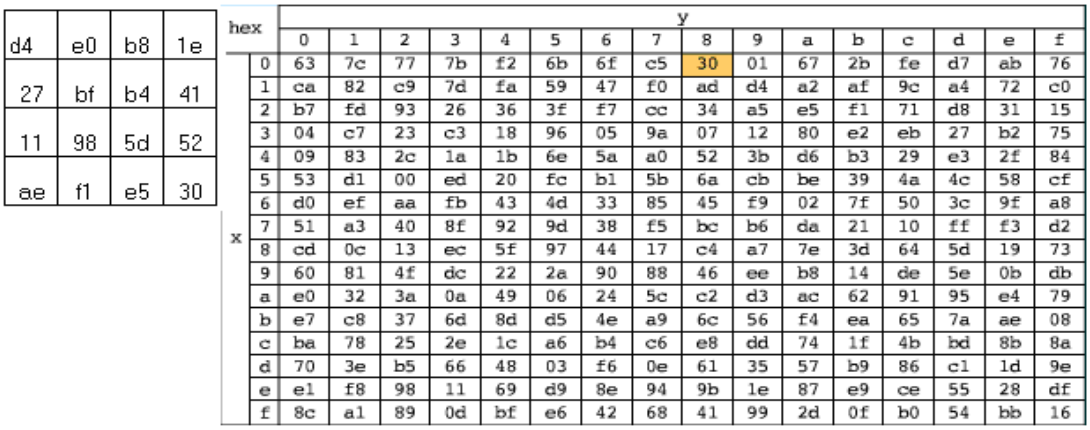
SubBytes() memetakan setiap byte dari *array state* dengan menggunakan *S-box (lihat gambar : 19, 20, 21).*



Gambar 2.19. input & S-box



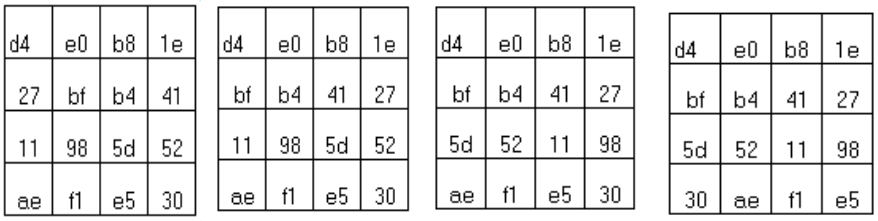
Gambar 2.20 . Proses pemetaan input dengan S-box



Gambar 2.21. Input yang telah dipetakan S-box

b. Transformasi *ShiftRows()*

1. Transformasi *ShiftRows()* melakukan pergeseran secara *wrapping* (siklik) pada 3 baris terakhir dari *array state*.
2. Jumlah pergeseran bergantung pada nilai baris (*r*). Baris *r* = 1 digeser sejauh 1 *byte*, baris *r* = 2 digeser sejauh 2 *byte*, dan baris *r* = 3 digeser sejauh 3 *byte*. Baris *r* = 0 tidak digeser. (*lihat Gambar 2.22*)



d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

d4	e0	b8	1e
bf	b4	41	27
11	98	5d	52
ae	f1	e5	30

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
ae	f1	e5	30

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Gambar 2.22. state awal ,rotate 1, rotate2 dan rotate 3

c. Transformasi *MixColumns()*

1. Transformasi *MixColumns()* mengalikan setiap kolom dari *array state* dengan polinom $a(x)$ mod $(x4 + 1)$.
2. Setiap kolom diperlakukan sebagai polinom 4-suku pada GF(28).
3. $a(x)$ yang ditetapkan adalah:

   $a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\}$

Input = D4 BF 5D 30

Output(0) = (D4 * 2) XOR (BF*3) XOR (5D*1) XOR (30*1)
= E(L(D4) + L(02)) XOR E(L(BF) + L(03)) XOR 5D XOR 30
= E(41 + 19) XOR E(9D + 01) XOR 5D XOR 30
= E(5A) XOR E(9E) XOR 5D XOR 30
= B3 XOR DA XOR 5D XOR 30
= **04**

Output(1) = (D4 * 1) XOR (BF*2) XOR (5D*3) XOR (30*1)
= D4 XOR E(L(BF)+L(02)) XOR E(L(5D)+L(03)) XOR 30
= D4 XOR E(9D+19) XOR E(88+01) XOR 30
= D4 XOR E(B6) XOR E(89) XOR 30
= D4 XOR 65 XOR E7 XOR 30
= **66**

Output(2) = (D4 * 1) XOR (BF*1) XOR (5D*2) XOR (30*3)

= D4 XOR BF XOR E(L(5D)+L(02)) XOR E(L(30)+L(03))

= D4 XOR BF XOR E(88+19) XOR E(65+01)

= D4 XOR BF XOR E(A1) XOR E(66)

= D4 XOR BF XOR BA XOR 50

= **81**

Output(3) = (D4 * 3) XOR (BF*1) XOR (5D*1) XOR (30*2)

= E(L(D4)+L(3)) XOR BF XOR 5D XOR E(L(30)+L(02))

= E(41+01) XOR BF XOR 5D XOR E(65+19)

= E(42) XOR BF XOR 5D XOR E(7E)

= 67 XOR BF XOR 5D XOR 60

= **E5**

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

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

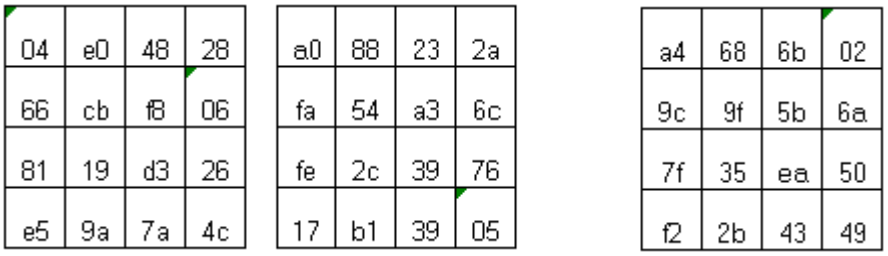
$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{3,c})$$

e0	b8	1e	d4		02	03	01	01		04		04	e0	48	28
b4	41	27	bf	*	01	02	03	01	=	66		66	cb	f8	06
52	11	98	5d		01	01	02	03		81		81	19	d3	26
ae	f1	e5	30		03	01	01	02		e5		e5	9a	7a	4c

Gambar 2.23. Proses mix columns

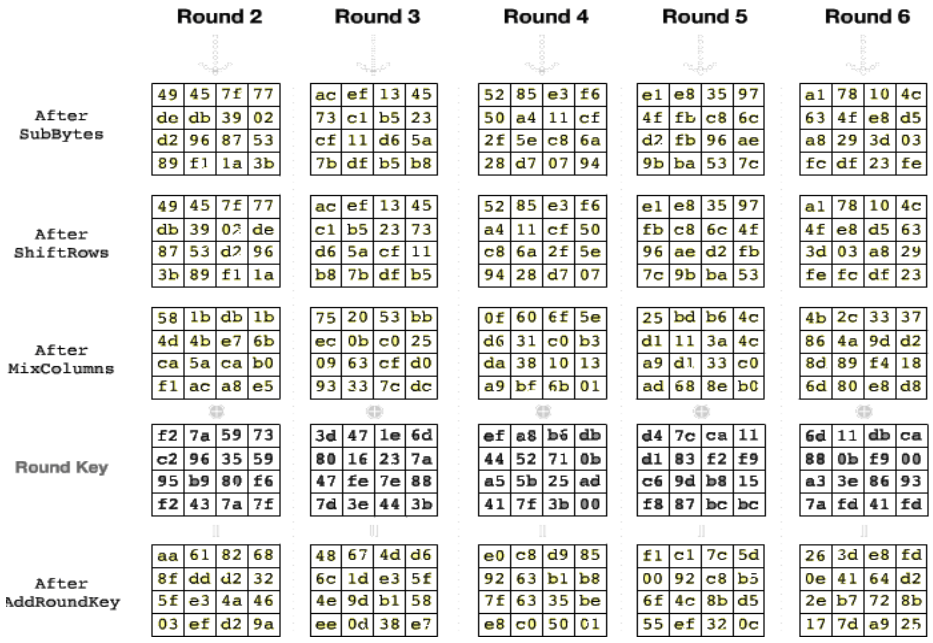
d. Transformasi *AddRoundKey*

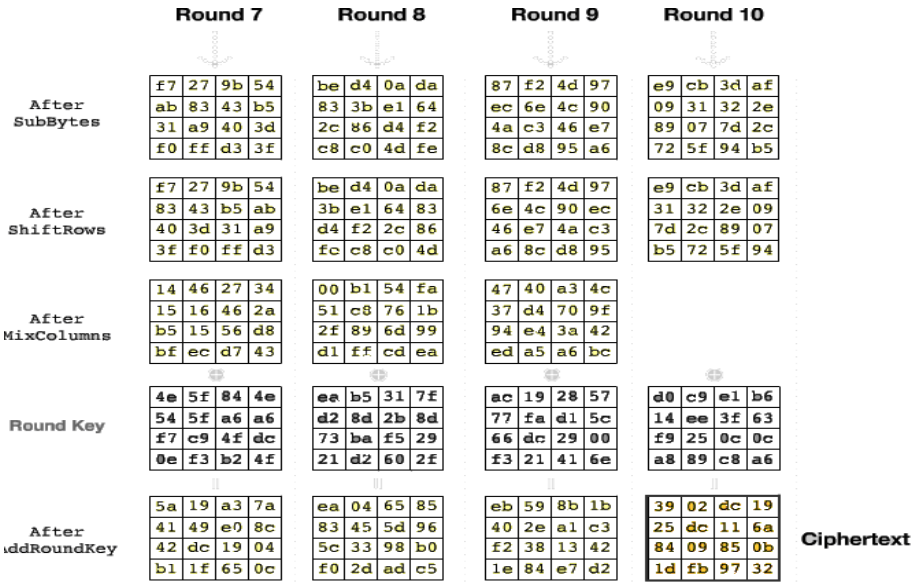
Transformasi ini melakukan operasi XOR terhadap sebuah *round key* dengan *array state*, dan hasilnya disimpan di *array state* (*Gambar 2.24 Proses add round keys*).



Gambar 2.24. Proses add round keys

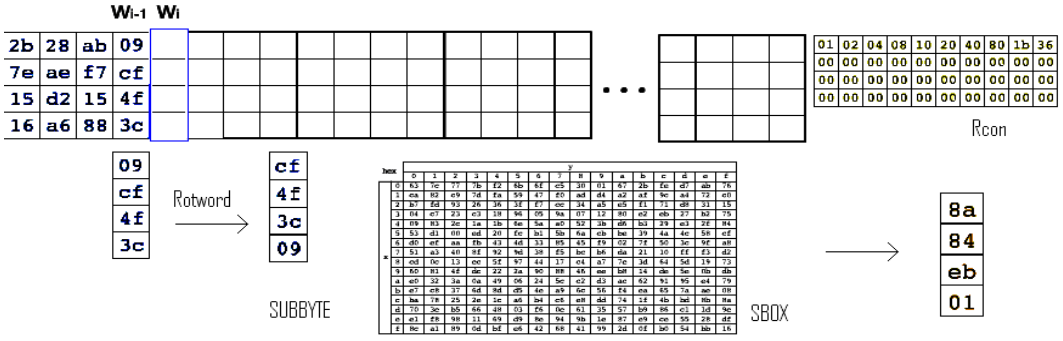
Berikut ini adalah tampilan *rounds* selanjutnya setelah penjelasan dari round pertama,dimulai dari round ke dua sampai kesepuluh. Round terakhir tidak melibatkan pengubahan dengan cara mix columns (*Gambar 2.25. proses round kedua sampai kesepuluh*).

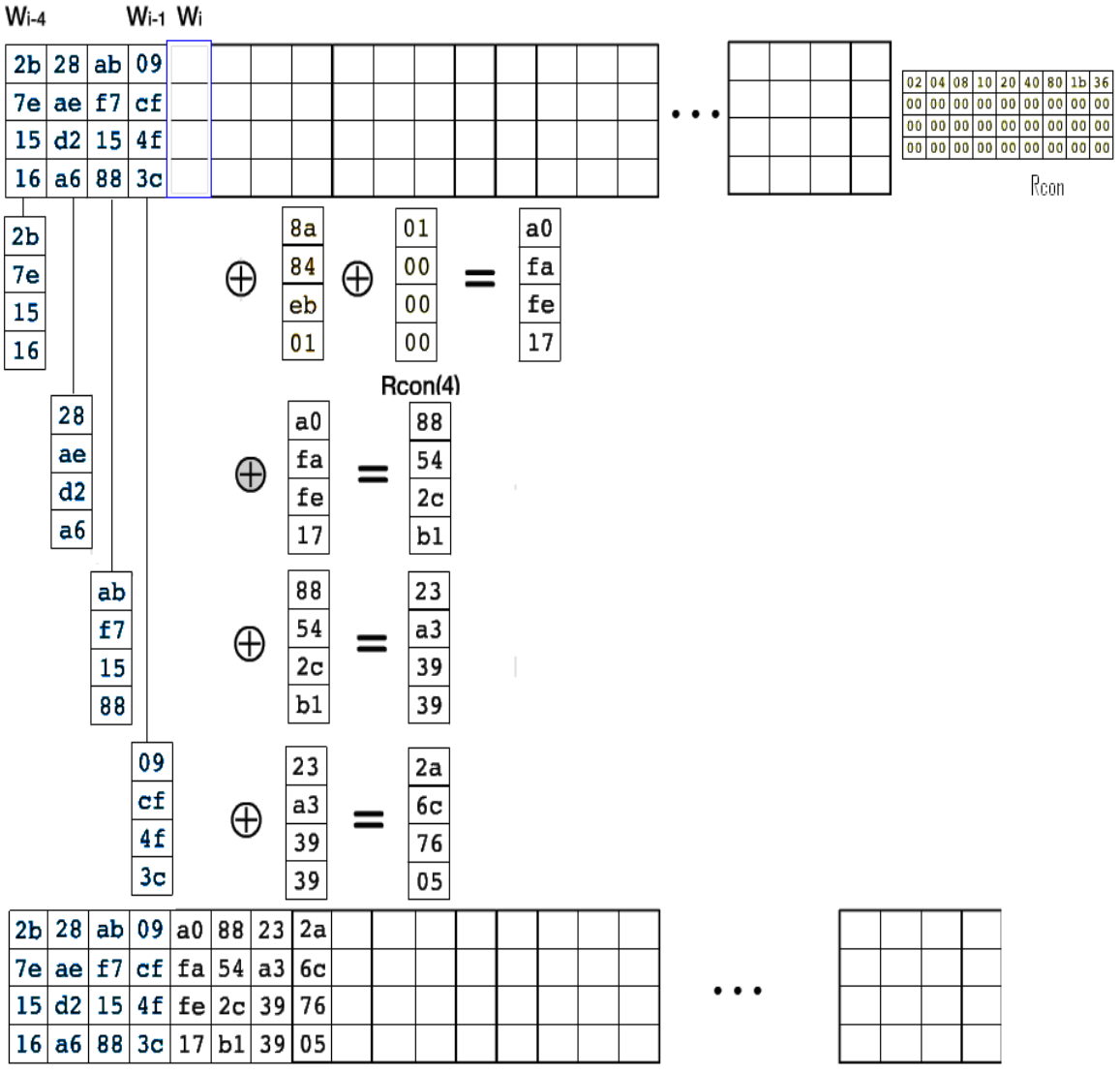


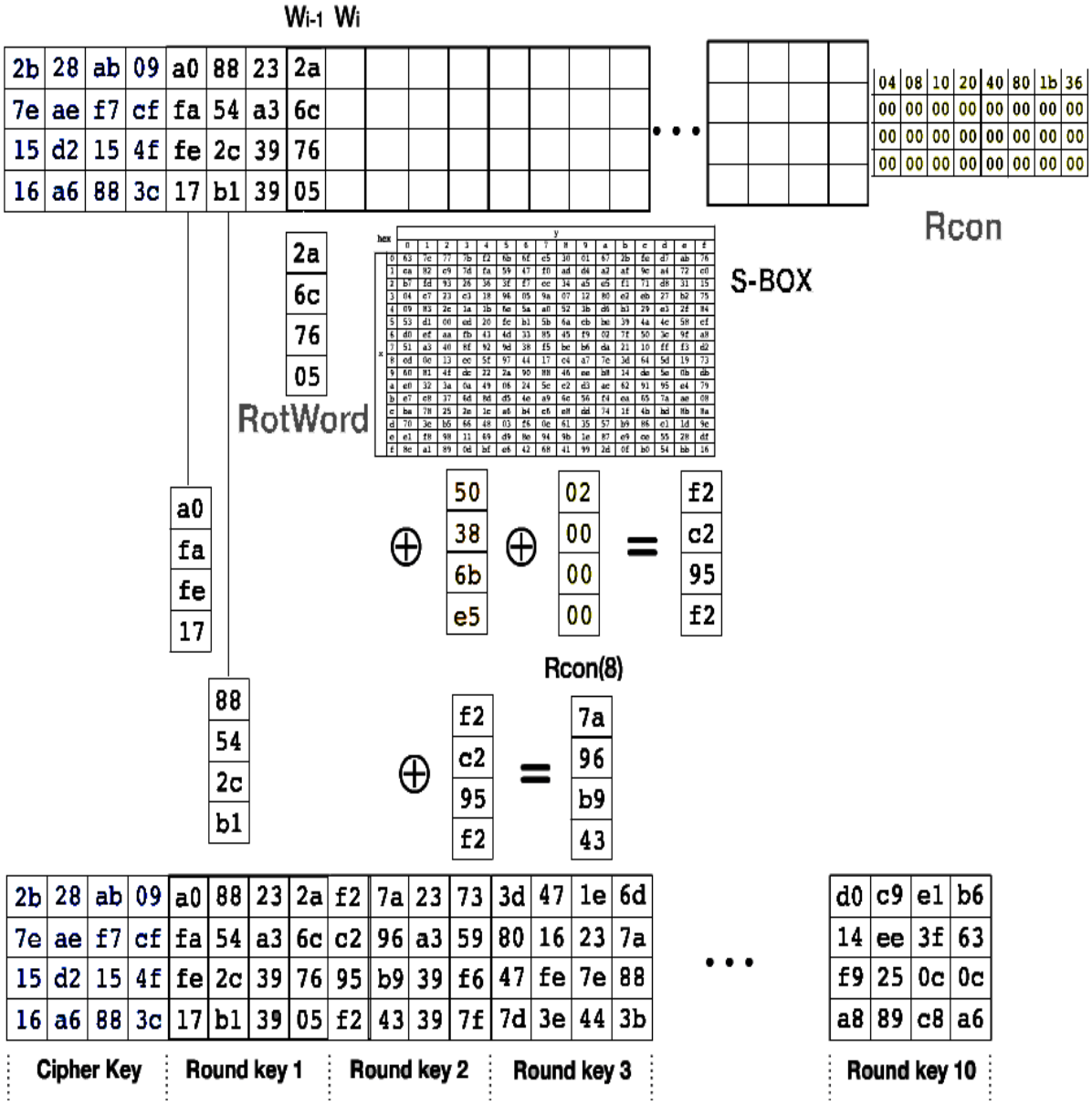


Gambar 2.25. Proses Round Kedua Sampai Kesepuluh

Pada AES memiliki sistem penjadwalan kunci sehingga pada setiap perputaran atau rounds kunci selalu berubah- ubah seperti pada (*Gambar 2.26. penjadwalan kunci*).




Wi-4
Wi-1
Wi
Rcon
Rcon(4)



Gambar 2.26. Penjadwalan Kunci

BAB 3

METODE PERANCANGAN

3.1. Pembuatan Program Aplikasi

Pembuatan program merupakan proses penerapan sistem secara nyata dari rancangan program yang telah dibuat sebelumnya. Setelah rancangan program telah selesai dibuat, maka tahap selanjutnya yaitu tahap pembuatan program yang dapat dijalankan.

3.2. Prosedur Pembuatan Program Aplikasi

Langkah – langkah pembuatan program aplikasi AES ini adalah sebagai berikut :

1. Mempersiapkan dan melakukan instalasi perangkat keras berupa komputer dengan spesifikasi sebagai berikut:
    a. Prosesor minimal pentium IV.
    b. Harddisk minimal 20 GB.
    c. Memori DDR minimal 128 MB.
    d. Monitor minmal 15’.
    e. Satu buah keyboard dan mouse standard.
    f. Media penyimpanan External jika diperluukan

g. Printer bila diperlukan.

2. Mempersiapkan dan melakukan instalasi perangkat lunak berupa komputer dengan spesifikasi sebagai berikut :
   a. Sistem operasi window XP.
   b. Program aplikasi Microsoft office, Microsoft office Word akan dibutuhkan untuk tempat menyimpan data.
   c. Program aplikasi Microsoft Visual Studio 6.0 , Microsoft Visual Basic 6.0 akan dibutuhkan untuk aplikasi.

3.3. Proses Pembuatan Program Aplikasi

Pembuatan program aplikasi ini dilakukan setelah selesai mempersiapkan perangkat keras dan lunak.

Langkah- langkah proses pembuatannya adalah sebagai berikut :

1. Mengumpulkan materi –materi yang dibutuhkan.
2. Melakukan pembatasan terhadap materi tersebut.
3. Membuat STD (*State Transition Diagram*) yang menggambarkan alur program.
4. Melakukan pengkodean dengan menggunakan Microsoft Visual Basic .

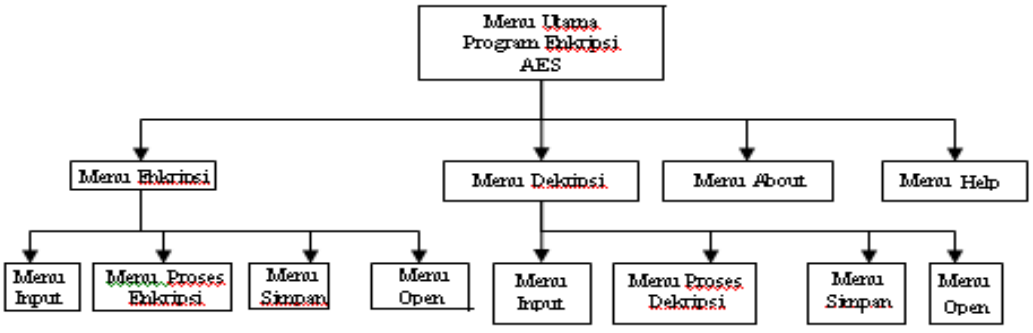
3.4. Rancangan Sistem

Perancangan program aplikasi kriptografi dengan menggunakan algoritma *Advanced Encryption Standard*. Rancangan ini digunakan untuk meningkatkan

keamanan dalam proses pengiriman atau pertukaran data. Rancangan ini dilakukan dalam beberapa tahap yaitu dimulai dari pembuatan diagram hirarki, yang dilanjutkan dengan menggunakan *State Transition Diagram* (STD), namun sebelumnya telah dibuat rancangan *Flowchart* dan *System Development Life Cycle* (SDLC) dan dilanjutkan lagi dengan perancangan antar muka program. setelah rancangan sistem ini selesai dilanjutkan dengan pembuatan program aplikasi. Setelah selesai, program aplikasi tersebut diuji.

3.5. Rancangan Diagram Hirarki

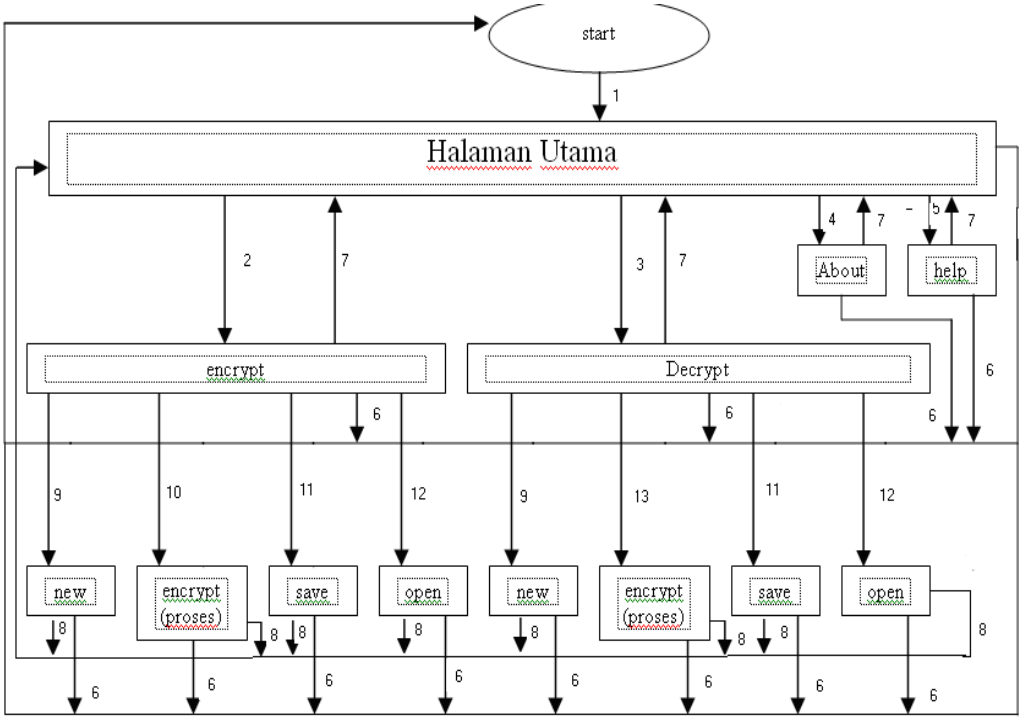
Rancangan ini dibuat untuk memudahkan proses perancangan aplikasi kriptosistem dengan menggunakan algoritma enkripsi *Advanced Encryption Standard* (AES). Diagram hirarki ini memiliki empat sub menu, yaitu: menu Encrypt, menu Decrypt , menu About dan menu Help. Sub menu Encrypt terbagi lagi menjadi New, enkripsi(proses), Save, dan Open . Sub menu Decrypt hampir sama dengan menu encrypt antara lain menjadi New, Dekripsi(proses), Save, dan Open. Gambar diagram hirarki dapat dilihat pada *Gambar 3.1*



Gambar 3.1. Diagram Hirarki

3.6. Rancangan State Transition Diagram

Rancangan ini digunakan untuk mengetahui apa saja yang terjadi pada sistem pada saat timbul perubahan – perubahan antara satu *state* dan *state* yang lain, apa yang menyebabkan timbulnya perubahan itu, dan apa akibat yang ditimbulkan dari perubahan itu. Rancangan *State Transition Diagram* untuk program aplikasi ini dapat dilihat pada *Gambar 3.2*



Gambar 3.3. Diagram STD

Keterangan gambar :

1. Klik AES.exe

   Untuk memasuki program AES sederhana.

2. Klik Menu Encrypt

Untuk masuk kedalam modul Enkripsi untuk proses menyandikan data.

3. Klik Menu Decrypt

   Untuk masuk kedalam modul Dekripsi untuk membalikkan proses menyandikan data.

4. Klik Menu About

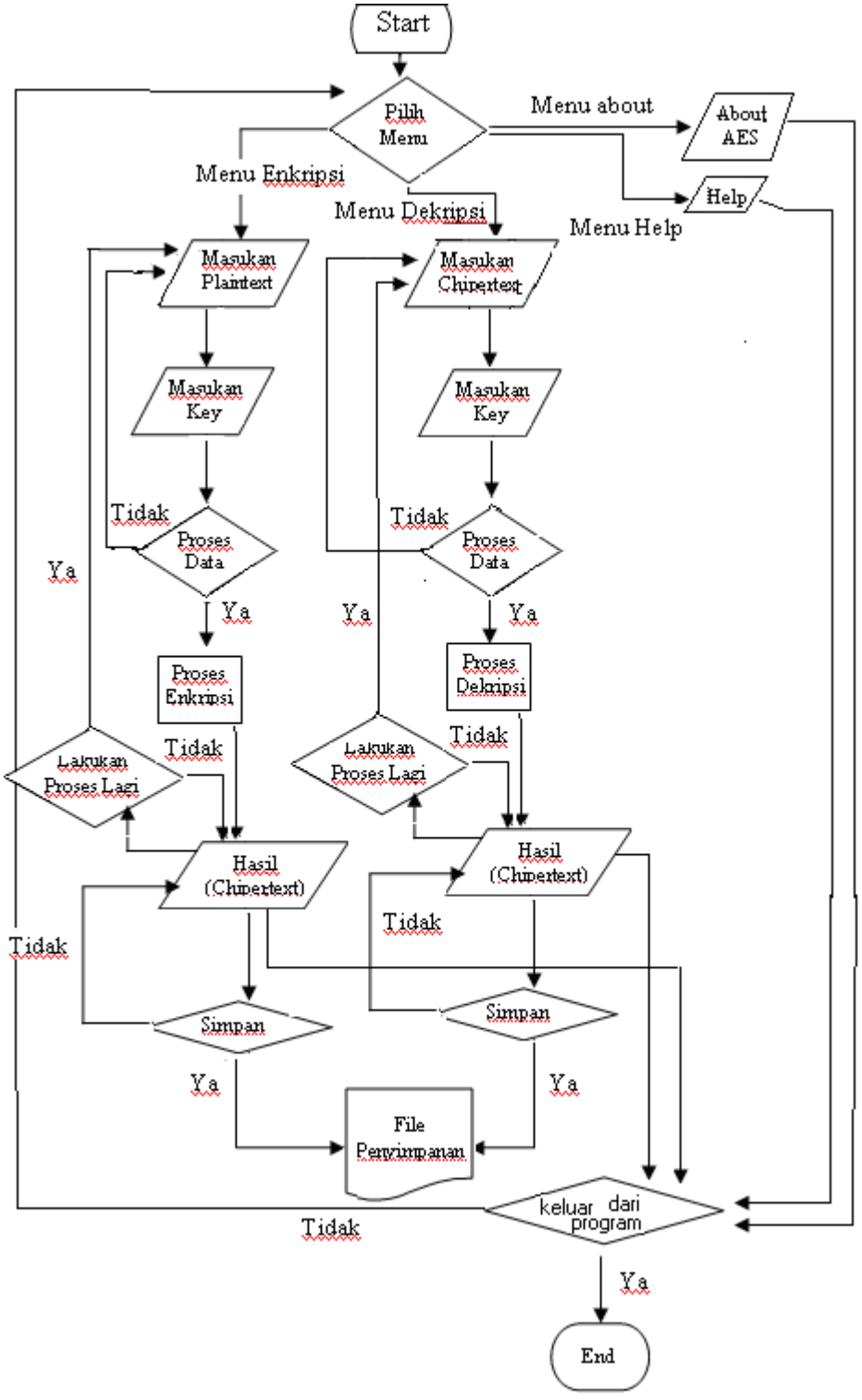
   Untuk masuk kedalam modul About yang memiliki informasi tentang program dan pembuatnya.

5. Klik Menu Help

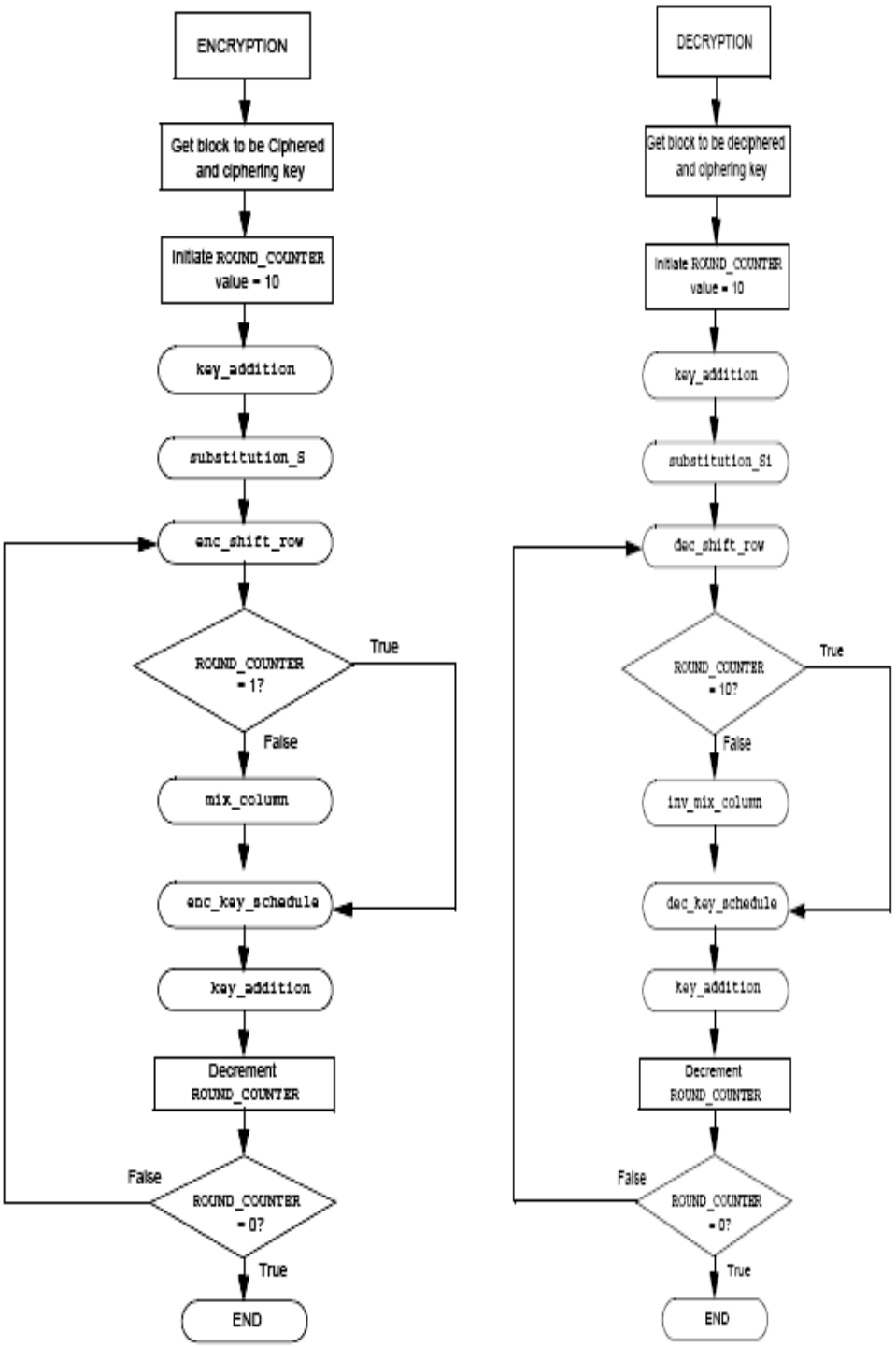
   Untuk masuk kedalam modul Help sebagai tempat mencari informasi dalam menjalankan program.

6. Klik Menu Exit
7. Klik Tombol Cancel
8. Klik Tombol Back to Menu
9. Klik Tombol New
10. Klik Tombol Encrypt (proses)
11. Klik Tombol Save
12. Klik Tombol Open
13. Klik Tombol Decrypt (proses)

Rancangan ini digunakan untuk mendesain dan merepresentasikan program. Sebelum pembuatan program, fungsinya adalah mempermudah *programmer* dalam menentukan alur logika program yang akan dibuat. Sesudah pembuatan program fungsinya adalah untuk menjelaskan alur program kepada orang lain atau user. Rancangan ini dapat dilihat pada *Gambar 3.3 dan Gambar 3.4.*



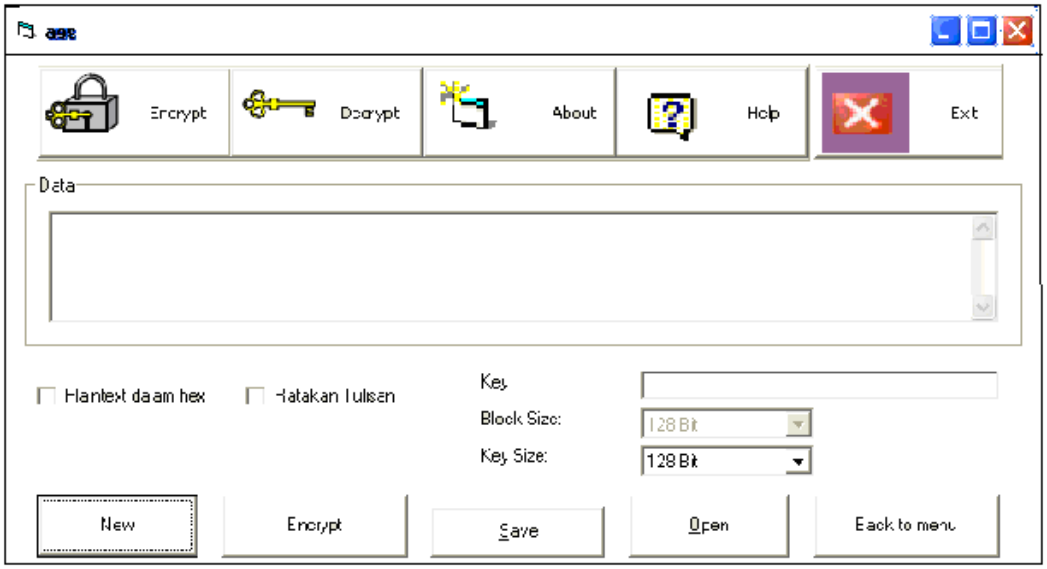
Gambar 3.3. Diagram Blok Rancangan Aplikasi AES



Gambar 3.4. Flowchart Proses Enkripsi dan Dekripsi AES

3.7. Rancangan Antarmuka

Rancangan ini digunakan untuk mendukung proses pembuatan program aplikasi kriptosistem dengan menggunakan algoritma AES. Rancangan antarmuka ini terdiri dari satu layar menu utama dapat dilihat ada Gambar 3.5, yang terdiri atas beberapa modul yaitu :



Gambar 3.5 Rancangan Antar muka

3.7.1. Rancangan Modul Enkripsi

Pada rancangan ini terdapat *frame message* untuk menampung pesan yang akan diubah dalam proses enkripsi. Pada modul ini terdapat pada *toolbar* - *toolbar* yang terletak pada bagian atas program aplikasi. *Toolbar* adalah daftar tombol pembantu yang dapat digunakan untuk mengaktifkan fungsi aplikasi. Tombol – tombol yang akan muncul bila anda menekan *toolbar Encrypt* adalah tombol New, tombol Proses encrypt, tombol Open, tombol Save, dan tombol Back to menu.

Jika anda menekan tombol new maka pada layar terlihat *frame message* yang kosong dan siap untuk diisi, bila *frame message* terdapat kata atau tulisan maka tombol new akan berubah menjadi clear. Tombol clear memiliki fungsi untuk membersihkan *frame message*. Pada tombol Proses encrypt memiliki fungsi untuk memulai pengacakan informasi, dan anda akan diminta untuk mengisikan kunci yang telah disepakati, setelah selesai mengisi kunci program aplikasi akan memproses pesan anda. Bila anda ingin menyimpan data atau informasi yang telah anda rahasiakan maka tekan tombol save. Setelah anda menyimpan data atau informasi anda dapat membuka file yang telah anda simpan sebelumnya dengan menekan tombol open.

3.7.2. Rancangan Modul Dekripsi

Pada rancangan ini terdapat *frame message*, seperti pada modul enkripsi *toolbar* ini memiliki beberapa tombol, antara lain adalah sebagai berikut : tombol New, tombol Open, tombol Save, tombol Proses Decrypt, dan tombol Back to menu.

Jika anda menekan tombol new maka pada layar terlihat *frame message* yang kosong dan siap untuk diisi, bila *frame message* terdapat kata atau tulisan maka tombol new akan berubah menjadi clear. Tombol clear memiliki fungsi untuk membersihkan *frame message*. Pada tombol Proses decrypt memiliki fungsi untuk mengembalikan pesan atau informasi yang telah mengalami pengacakan, dan anda akan diminta untuk mengisikan kunci yang telah disepakati, setelah selesai mengisi kunci program aplikasi akan memproses data anda. Jika kunci yang anda masukan

salah pesan yang ditampilkan akan tidak sesuai dengan yang anda inginkan, jika kunci tersebut benar maka pesan sebenarnya dapat dimengerti dan maksud pengirim dapat tersampaikan. Bila anda ingin menyimpan data atau informasi yang telah anda rahasiakan maka tekan tombol save. Setelah anda menyimpan data atau informasi anda dapat membuka file yang telah anda simpan sebelumnya dengan menekan tombol open.

3.7.3. Rancangan Modul About

Pada modul about, pengguna dapat melihat keterangan singkat mengenai program kriptosistem dan pembuatnya. Modul ini akan ditampilkan pada saat pengguna menekan *toolbar* about pada menu utama yang terletak pada bagian atas program aplikasi.

3.7.4. Rancangan Modul Help

Pada modul help, pengguna dapat melihat dan memilih keterangan mengenai cara menjalankan program dan keterangan mengenai fungsi –fungsi yang terdapat dalam program aplikasi kriptosistem tersebut. Modul ini akan ditampilkan bila pengguna menekan *toolbar* help pada menu utama yang terletak pada bagian atas program aplikasi.

3.8. Siklus Hidup Pengembangan Sistem

Metode Siklus Hidup Pengembangan Sistem atau yang sering disebut dengan SDLC*(System Development Life Cycle*) adalah suatu metode untuk merancang aplikasi perangkat lunak. Nama lain dari sistem ini adalah metode *waterfall*.

Perancangan dengan menggunakan metode SDLC dilakukan dalam enam tahap, antara lain:

1. Perencanaan (*System engineering*)

   Perencanaan adalah suatu kegiatan untuk menentukan program aplikasi yang akan dirancang dan dijalankan, dan siapa yang akan merancang program aplikasi tersebut.

2. Analisis

   Analisis adalah suatu kegiatan untuk menentukan topik dari permasalahan yang sedang dihadapi dan bagaimana cara pemecahan masalah tersebut.

3. Desain

   Desain adalah suatu kegiatan untuk menentukan konsep dasar rancangan dari suatu programyang akan dibuat sehingga diharapkan dengan desain yang baik, maka para penguna akan merasa nyaman dalam menggunakan program aplikasi tersebut

4. Pengkodean *(coding*)

   pengkodean adalah suatu kegiatan yang berguna untuk mengimplementasikan konsep dasar dari tahap sebelumnya yaitu tahap desain kedalam bahasa Pemrograman

5. Pengujian

Pengujian adalah suatu kegiatan untuk mencari kelemahan serta kesalahan yang terjadi pada program aplikasi dan kemudian memperbaiki kelemahan dan kesalahan tersebut. Terdapat beberapa metode pengujian yang digunakan untuk menguji fungsi – fungsi dari suatu program aplikasi. Salah satunya yaitu menggunakan metode *black box testing*.

Pengujian dengan metode *black box testing* tidak memperhatikan struktur internal / sifat dari program / modul, seperti pada *white box testing*, tetapi menguji apakah program sudah sesuai dengan spesifikasinya. Pengujian *black box testing* mengijinkan seseorang perancang aplikasi untuk memberikan sejumlah *input* lalu diproses oleh aplikasi yang dibuat sesuai dengan kebutuhan spesifikasinya. Pengujian ini dilakukan agar menghasilkan *output* atau tampilan yang benar sesuai dengan fungsi dari program tersebut. Metode *black box testing* berusaha menemukan kesalahan yang sesuai dengan beberapa kategori diantaranya : hilangnya suatu fungsi / penulisan program yang salah, kesalahan *interface* program, kesalahan pada saat *inisialisasi / validasi* , kesalahan pengakssan data, dan kesalahan performasi.

6. Pemeliharaan (maintenance)

Pemeliharaan adalah suatu kegiatan evaluasi terhadap program aplikasi yang berjalan seperti : kesalahan pengkodean dalam program, kerusakan program yang menyebabkan performa program menurun ataupun

memperbaharui program dengan fasilitas tambahan, supaya program dapat berjalan dengan baik dan performanya meningkat.

Rancangan SDLC untuk program aplikasi kriptosistem algoritma AES meliputi tahap Perencanaan, Analisis, desain, Pengkodean dan Pengujian.

Berikut ini adalah proses lengkapnya dari proses enkripsi sampai pendekripsian kembali.

```
input: cipher input
start: state at start of round[r]
s_box: state after s_box substitution
s_row: state after shift row transformation
m_col: state after mix column transformation
k_sch: key schedule value for round[r]
output: cipher output
iinput: inverse cipher input
istart: state at start of round[r]
is_box: state after inverse s_box substitution
is_row: state after inverse shift row transformation
ik_sch: key schedule value for round[r]
ik_add: state after key addition
ioutput: cipher output
PLAINTEXT: 3243f6a8885a308d313198a2e0370734 (pi * 2^124)
KEY: 2b7e151628aed2a6abf7158809cf4f3c ( e * 2^124)
CIPHER
R[ 0].input 3243f6a8885a308d313198a2e0370734
```

R[0].k_sch 2b7e151628aed2a6abf7158809cf4f3c

R[1].start 193de3bea0f4e22b9ac68d2ae9f84808

R[1].s_box d42711aee0bf98f1b8b45de51e415230

R[1].s_row d4bf5d30e0b452aeb84111f11e2798e5

R[1].m_col 046681e5e0cb199a48f8d37a2806264c

R[1].k_sch a0fafe1788542cb123a339392a6c7605

R[2].start a49c7ff2689f352b6b5bea43026a5049

R[2].s_box 49ded28945db96f17f39871a7702533b

R[2].s_row 49db873b453953897f02d2f177de961a

R[2].m_col 584dcaf11b4b5aacdbe7caa81b6bb0e5

R[2].k_sch f2c295f27a96b9435935807a7359f67f

R[3].start aa8f5f0361dde3ef82d24ad26832469a

R[3].s_box ac73cf7befc111df13b5d6b545235ab8

R[3].s_row acc1d6b8efb55a7b1323cfdf457311b5

R[3].m_col 75ec0993200b633353c0cf7cbb25d0dc

R[3].k_sch 3d80477d4716fe3e1e237e446d7a883b

R[4].start 486c4eee671d9d0d4de3b138d65f58e7

R[4].s_box 52502f2885a45ed7e311c807f6cf6a94

R[4].s_row 52a4c89485116a28e3cf2fd7f6505e07

R[4].m_col 0fd6daa9603138bf6fc0106b5eb31301

R[4].k_sch ef44a541a8525b7fb671253bdb0bad00

R[5].start e0927fe8c86363c0d9b1355085b8be01

R[5].s_box e14fd29be8fbfbba35c89653976cae7c

R[5].s_row e1fb967ce8c8ae9b356cd2ba974ffb53

R[5].m_col 25d1a9adbd11d168b63a338e4c4cc0b0

R[5].k_sch d4d1c6f87c839d87caf2b8bc11f915bc

R[6].start f1006f55c1924cef7cc88b325db5d50c

R[6].s_box a163a8fc784f29df10e83d234cd503fe

R[6].s_row a14f3dfe78e803fc10d5a8df4c632923

R[6].m_col 4b868d6d2c4a8980339df4e837d218d8

R[6].k_sch 6d88a37a110b3efddbf98641ca0093fd

R[7].start 260e2e173d41b77de86472a9fdd28b25

R[7].s_box f7ab31f02783a9ff9b4340d354b53d3f

R[7].s_row f783403f27433df09bb531ff54aba9d3

R[7].m_col 1415b5bf461615ec274656d7342ad843

R[7].k_sch 4e54f70e5f5fc9f384a64fb24ea6dc4f

R[8].start 5a4142b11949dc1fa3e019657a8c040c

R[8].s_box be832cc8d43b86c00ae1d44dda64f2fe

R[8].s_row be3bd4fed4e1f2c80a642cc0da83864d

R[8].m_col 00512fd1b1c889ff54766dcdfa1b99ea

R[8].k_sch ead27321b58dbad2312bf5607f8d292f

R[9].start ea835cf00445332d655d98ad8596b0c5

R[9].s_box 87ec4a8cf26ec3d84d4c46959790e7a6

R[9].s_row 876e46a6f24ce78c4d904ad897ecc395

R[9].m_col 473794ed40d4e4a5a3703aa64c9f42bc

R[9].k_sch ac7766f319fadc2128d12941575c006e

R[10].start eb40f21e592e38848ba113e71bc342d2

R[10].s_box e9098972cb31075f3d327d94af2e2cb5

R[10].s_row e9317db5cb322c723d2e895faf090794

R[10].k_sch d014f9a8c9ee2589e13f0cc8b6630ca6

R[10].output 3925841d02dc09fbdc118597196a0b32

INVERSE CIPHER

```
R[ 0].iinput 3925841d02dc09fbdc118597196a0b32
R[ 0].ik_sch d014f9a8c9ee2589e13f0cc8b6630ca6
R[ 1].istart e9317db5cb322c723d2e895faf090794
R[ 1].is_row e9098972cb31075f3d327d94af2e2cb5
R[ 1].is_box eb40f21e592e38848ba113e71bc342d2
R[ 1].ik_sch ac7766f319fadc2128d12941575c006e
R[ 1].ik_add 473794ed40d4e4a5a3703aa64c9f42bc
R[ 2].istart 876e46a6f24ce78c4d904ad897ecc395
R[ 2].is_row 87ec4a8cf26ec3d84d4c46959790e7a6
R[ 2].is_box ea835cf00445332d655d98ad8596b0c5
R[ 2].ik_sch ead27321b58dbad2312bf5607f8d292f
R[ 2].ik_add 00512fd1b1c889ff54766dcdfa1b99ea
R[ 3].istart be3bd4fed4e1f2c80a642cc0da83864d
R[ 3].is_row be832cc8d43b86c00ae1d44dda64f2fe
R[ 3].is_box 5a4142b11949dc1fa3e019657a8c040c
R[ 3].ik_sch 4e54f70e5f5fc9f384a64fb24ea6dc4f
R[ 3].ik_add 1415b5bf461615ec274656d7342ad843
R[ 4].istart f783403f27433df09bb531ff54aba9d3
R[ 4].is_row f7ab31f02783a9ff9b4340d354b53d3f
R[ 4].is_box 260e2e173d41b77de86472a9fdd28b25
R[ 4].ik_sch 6d88a37a110b3efddbf98641ca0093fd
R[ 4].ik_add 4b868d6d2c4a8980339df4e837d218d8
R[ 5].istart a14f3dfe78e803fc10d5a8df4c632923
R[ 5].is_row a163a8fc784f29df10e83d234cd503fe
R[ 5].is_box f1006f55c1924cef7cc88b325db5d50c
R[ 5].ik_sch d4d1c6f87c839d87caf2b8bc11f915bc
```

R[5].ik_add 25d1a9adbd11d168b63a338e4c4cc0b0

R[6].istart e1fb967ce8c8ae9b356cd2ba974ffb53

R[6].is_row e14fd29be8fbfbba35c89653976cae7c

R[6].is_box e0927fe8c86363c0d9b1355085b8be01

R[6].ik_sch ef44a541a8525b7fb671253bdb0bad00

R[6].ik_add 0fd6daa9603138bf6fc0106b5eb31301

R[7].istart 52a4c89485116a28e3cf2fd7f6505e07

R[7].is_row 52502f2885a45ed7e311c807f6cf6a94

R[7].is_box 486c4eee671d9d0d4de3b138d65f58e7

R[7].ik_sch 3d80477d4716fe3e1e237e446d7a883b

R[7].ik_add 75ec0993200b633353c0cf7cbb25d0dc

R[8].istart acc1d6b8efb55a7b1323cfdf457311b5

R[8].is_row ac73cf7befc111df13b5d6b545235ab8

R[8].is_box aa8f5f0361dde3ef82d24ad26832469a

R[8].ik_sch f2c295f27a96b9435935807a7359f67f

R[8].ik_add 584dcaf11b4b5aacdbe7caa81b6bb0e5

R[9].istart 49db873b453953897f02d2f177de961a

R[9].is_row 49ded28945db96f17f39871a7702533b

R[9].is_box a49c7ff2689f352b6b5bea43026a5049

R[9].ik_sch a0fafe1788542cb123a339392a6c7605

R[9].ik_add 046681e5e0cb199a48f8d37a2806264c

R[10].istart d4bf5d30e0b452aeb84111f11e2798e5

R[10].is_row d42711aee0bf98f1b8b45de51e415230

R[10].is_box 193de3bea0f4e22b9ac68d2ae9f84808

R[10].ik_sch 2b7e151628aed2a6abf7158809cf4f3c

R[10].ioutput 3243f6a8885a308d313198a2e0370734

BAB 4

HASIL DAN PEMBAHASAN

4.1. Cara Pengujian

Setelah program aplikasi ini melewati proses tahap pengkodean, maka tahap selanjutnya adalah tahap pengujian. Pengujian terhadap program ini dilakukan dengan tujuan untuk mengetahui apakah program berjalan dan berfumgsi sesuai dengan spesifikasi rancangan atau tidak.

Metode yang digunakan adalah metode pengujian *black box*, yaitu metode pengujian yang hanya memberikan input pada program aplikasi.*Input* tersebut lalu diproses dan akan menghasilkan *output* yang menentukan kesesuaian program dengan spesifikasi rancangan dan kebutuhan fungsional yang diignginkan pengguna. Bila dari input yang diberikan menghasilkan *output* yang sesuai dengan spesifikasi rancangan, maka program aplikasi sudah benar dan tidak perlu dilakukan perbaikan. Namun bila dari *input* yang diberikan menghasilkan *output* yang tidak sesuai dengan spesifikasi rancangan, maka pada program aplikasi masih terdapat kesalahan dan perlu dilakukan perbaikan. Perbaikan ini dilakukan hingga program aplikasi menghasilkan *output* yang sesuai dengan spesifikasi rancangan dan kebutuhan fungsional pengguna.

4.2. Hasil Pengujian

Proses pengujian aplikasi dilakukan pada setiap modul, hasil pengujian dapat dilihat pada tabel 4.1.

Tabel 4.1.

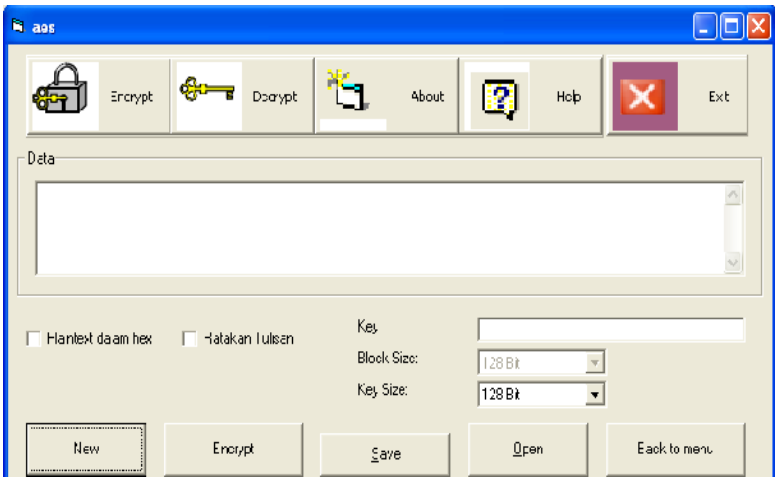
Perbandingan Antara Rancangan Dengan Hasil Pengujian

No.	Modul Yang Diuji	Spesifikasi Rancangan	Hasil Pengujian	Banyaknya Pengujian
1.	Enkripsi	Berfungsi untuk meng-acak pesan menjadi sesuatu yang rahasia	Fungsi sudah sesuai dengan spesifikasi rancangan.	5 kali
2.	Dekripsi	Berfungsi untuk meng-embalikan pesan yang teracak kebentuk semula	Fungsi sudah sesuai dengan spesifikasi rancangan.	5 kali
3.	About	Berfungsi untuk mem beri keterangan tentang pembuat dan program	Fungsi sudah sesuai dengan spesifikasi rancangan.	3 kali
4.	Help	Berfungsi untuk me-mandu dalam menjalan - kan program.	Fungsi sudah sesuai dengan spesifikasi rancangan.	3 kali

Modul yang diuji yaitu :

4.2.1. Pengujian Modul Enkripsi

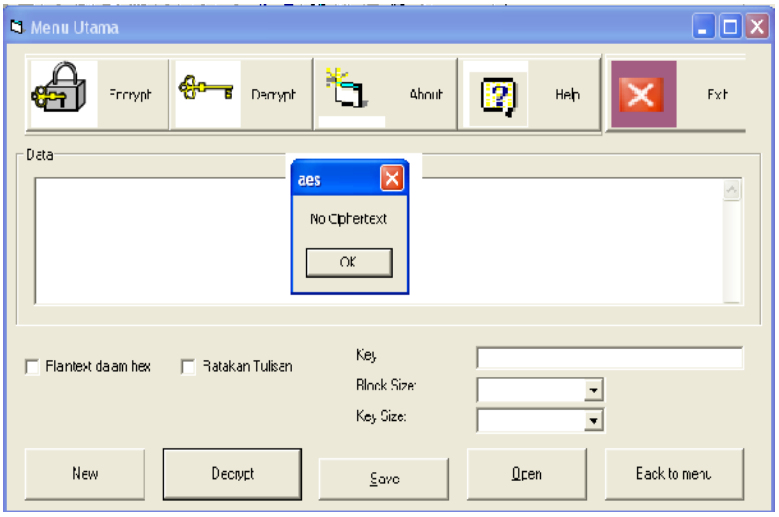
Modul enkripsi yang berfungsi sebagai pengacak pesan atau informasi sehingga menjadi bentuk yang tidak dapat terbaca oleh orang lain dan menjadi suatu pesan yang rahasia. Fungsi modul enkripsi pada program ini berjalan sesuai dengan spesfikasi rancangan. Tampilan modul enkripsi dapat dilihat pada Gambar 4.1



Gambar 4.1. Modul Encrypt

4.2.2. Pengujian Modul Dekripsi

Modul Dekripsi yang berfungsi untuk menerjemahkan pesan yang telah diacak sehingga dapat dibaca oleh si penerima pesan atau oleh pihak – pihak yang berhak menerima pesan tersebut. Fungsi modul dekripsi pada program ini berjalan sesuai dengan spesfikasi rancangan. Tampilan modul dekripsi dapat dilihat pada Gambar 4.2.



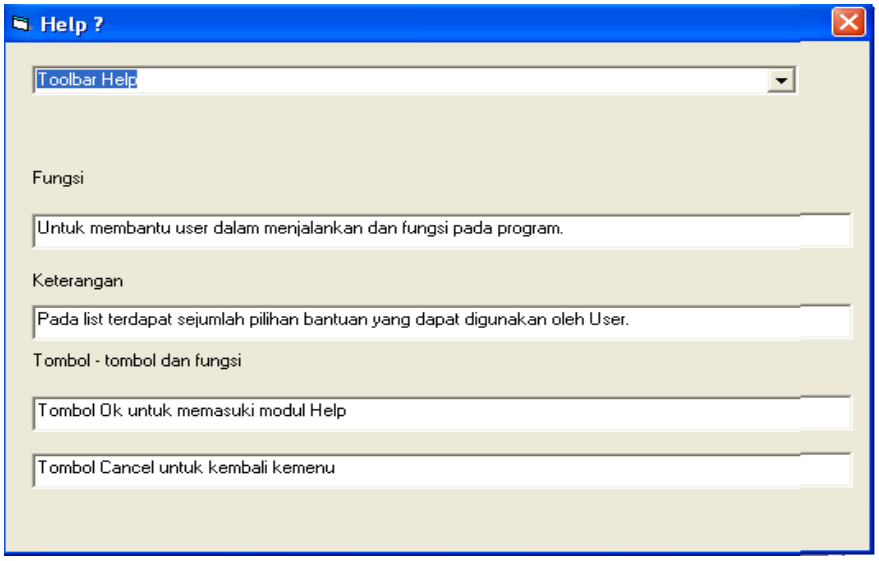
Gambar 4.2 Modul Decrypt

4.2.3. Pengujian Modul About

Modul ini menampilkan penjelasan singkat mengenai program aplikasi dari pembuat program aplikasi ini. Fungsi modul About pada program aplikasi ini berjalan sesuai dengan spesfikasi rancangan. Modul ini berjalan dengan baik,

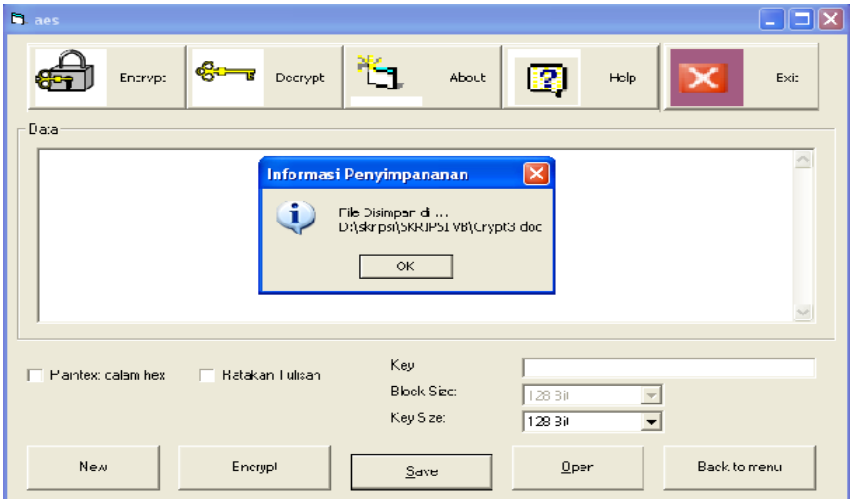
4.2.4. Pengujian Modul Help

Modul Help dapat digunakan sebagai panduan untuk menjalankan program aplikasi ini. Fungsi modul ini berjalan sesuai dengan spesfikasi rancangan. Modul ini berjalan dengan baik, dapat dilihat pada Gambar 4.4

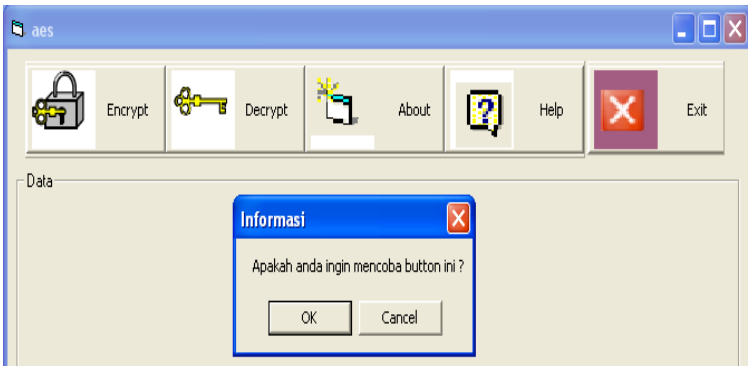


Gambar 4.4. Modul Help

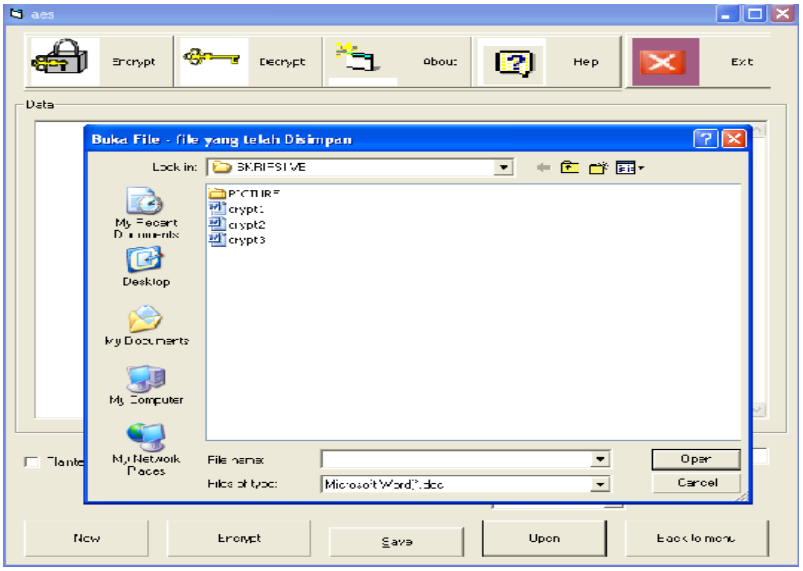
Adapun Keterangan-keterangan lain hasil pengujian file dilihat pada gambar di bawah ini :



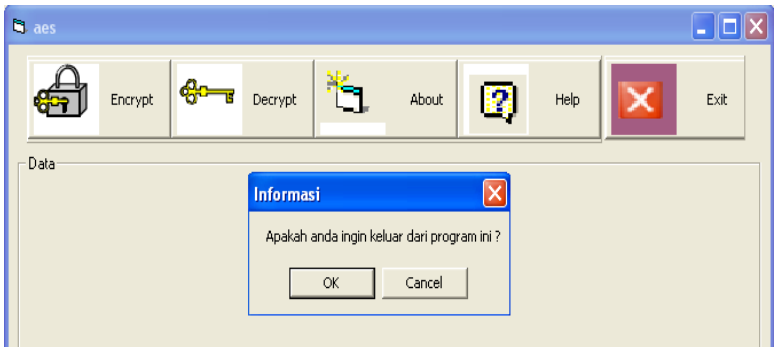
Gambar 4.5. Penyimpanan File



Gambar 4.6. Tampilan Setiap menekan Tombol Toolbar



Gambar 4.7. Membuka File



Gambar 4.8. Modul keluar dari program

4.3. Pembahasan

Dari hasil pengujian yang diperoleh, maka dilakukan pembahasan terhadap hasil pengujian dari berbagai modul tersebut . modul yang dibahas pada program aplikasi ini adalah :

4.3.1. Modul Enkripsi

Modul ini pada umumnya digunakan oleh oleh komputer pengirim. Proses pengiriman informasi dimulai dari memasukan pesan yang ingin dikirimkan dan dirahasiakan (misalnya : HERI SANTOSO 0119116701 ILMU KOMPUTER UINSU). Setelah pesan itu dimasukan, dilakukan proses enkripsi dengan mengisi kunci dan menekan tombol encrypt untuk memproses informasi.

Untuk mengenkripsi pesan, pengguna harus memasukkan *key* yang telah disepakati oleh kedua belah pihak, baik dari pihak pengirim pesan maupun pihak penerima pesan. Kemudian setelah kunci ditetapkan maka pesan yang akan dikirim disertakan kunci yang telah disepakati dan tnggal menekan tombol encrypt maka pesan akan berupa *ciphertext*.

4.3.2. Modul Dekripsi

Modul ini pada umumnya digunakan oleh komputer penerima. Ketika pesan diterima, pesan yang berupa *ciphertext* akan diubah menjadi *plaintext* sehingga pembaca dapat membaca pesan yang dikirim oleh komputer pengirim.

Untuk menjalan proses dekripsi, pengguna harus memasukan *key* yang telah disepakati dan menekan tombol decrypt. Kemudian pesan asli akan dapat dibaca oleh sipenerima pesan. Jika *key* yang dimasukan salah atau tidak sesaui dengan kesepakatan, maka pesan tidak akan dapat diterima dengan baik dan maksud si-pengirim tidak akan sampai kepada pihak penerima pesan.

4.3.3. Modul About

Modul ini yang menampilkan penjelasan singkat mengenai program aplikasi ini . Modul ini berjalan dengan baik.

4.3.4. Modul Help

Modul Help ini digunakan sebagai panduan untuk menjalankan program aplikasi ini. Modul ini juga memberikan informasi dan manual pemakaian program aplikasi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil perancangan dan pembuatan program aplikasi kriptosistem menggunakan algoritma *Advanced Encryption Standar*d (AES) ini, dapat diambil kesimpulan sebagai berikut :

1. Spesifikasi program aplikasi ini dapat dijalankan sesuai dengan spesifikasi teknis yang dirancang.
2. Program aplikasi kriptosistem ini akan membatasi orang yang tidak berhak atas informasi atau data yang dimiliki oleh si-pengirim untuk dibaca karena pesan sudah dienkripsi.
3. Program aplikasi kriptosistem ini menjaga kerahasiaan pesan atau informasi file-file yang ada dalam sebuah komputer.

5.2. Saran

Saran – saran yang berguna untuk pengembangan lebih lanjut terhadap program aplikasi ini adalah sebagai berikut:

1. Input untuk proses enkripsi tidak hanya dilakukan untuk format berbentuk text atau angka saja, tetapi bisa juga digunakan untuk mengenkripsi data yang berupa gambar (*image*), suara, video dan lain sebagainya.
2. Pengguna juga dapat memilih format penyimpanan data tidak hanya pada microsoft word (*.doc) saja, tetapi format lain juga dapat digunakan.