**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama : Gracius Cagar Gunawan**

**NRP : 5109100168**

**Dosen Wali : Arya Yudhi Wijaya, S. Kom., M. Kom.**

1. **JUDUL TUGAS AKHIR**

Implementasi Algoritma Rijndael dengan menggunakan kunci enkripsi yang berukuran melebihi 256 bit.

Implementation of Rijndael Algorithm using more than 256 bit cipher key.

1. **URAIAN SINGKAT**

Enkripsi adalah algoritma yang bertujuan untuk merahasiakan suatu informasi dengan cara mengubah sebuah informasi menjadi sebuah *string* yang tidak dapat dibaca secara langsung. Untuk dapat membaca hasil enkripsi, hasil enkripsi harus dikembalikan lagi ke bentuk sebelum dienkripsi. Proses enkripsi memerlukan masukan berupa informasi yang akan dienkripsi dan sebuah kunci enkripsi yang disebut juga dengan *cipher key*. Kunci enkripsi ini yang akan menentukan perubahan informasi masukan menjadi hasil enkripsi.

Banyak serangan yang dilakukan terhadap enkripsi. Serangan-serangan tersebut bertujuan agar dapat membaca informasi yang dienkripsi. Contoh-contoh serangan yang telah dilakukan adalah *brute* *force* *attack*, *dictionary* *attack*, dan *differential* *attack*.

Berbagai solusi telah diupayakan untuk mempertahankan kerahasiaan informasi dari serangan-serangan tersebut. Salah satu solusi yang cukup terkenal adalah *Data* *Encryption* *Standard* (DES). DES merupakan standarisasi enkripsi pada tahun 1999. DES cukup kebal terhadap *differential* *attack*, tetapi karena ukuran kunci enkripsi DES kecil, yaitu 56 bit, DES mudah dipecahkan menggunakan *brute* *force* *attack*.

Oleh karena itu, pada tahun 2001, dibuatlah standarisasi enkripsi baru, yaitu *Advanced* *Ecryption* *Standard* (AES). AES diimplementasikan menggunakan sebuah algoritma enkripsi yang dibuat oleh Joan Daemen dan Vincent Rijmen yang dinamakan Algoritma Rijndael. Algoritma tersebut menerima kunci enkripsi jauh lebeh besar dibandingkan dengan DES, yaitu 256 bit. Dengan kemampuan perangkat keras sekarang, hampir mustahil AES untuk dapat dipecahkan menggunakan serangan *brute* *force*.

Seiring berjalannya waktu, nasib AES akan sama dengan DES. Dalam beberapa tahun lagi, tidaklah mustahil lagi untuk memecahkan AES menggunakan serangan *brute* *force*. Oleh karena itu, pada tugas akhir ini, ditawarkan sebuah solusi yaitu implementasi Algoritma Rijndael yang dapat menerima kunci enkripsi yang berukuran melebihi 256 bit.

1. **PENDAHULUAN**
2. **LATAR BELAKANG**

Serangan-serangan telah dilakukan untuk memecahkan enkripsi sangat banyak dan beragam. Salah satu serangan yang paling terkenal adalah *brute* *force* *attack*. Pada tahun 1999, FIPS mengeluarkan standarisasi enkripsi yang disebut dengan DES [[1](#Fed99)]. DES cukup kebal terhadap serangan *differential* [[2](#How)], tetapi DES yang hanya menggunakan 56 bit kunci enkripsi mudah ditembus dengan serangan *brute* *force* karena komputer tercanggih sekarang dapat mencoba 255 kunci enkripsi yang berbeda setiap detiknya.

Oleh karena itu, pada tahun 2001, dibuatlah standarisasi enkripsi baru, yaitu *Advanced* *Ecryption* *Standard* (AES) [[3](#Fed01)]. AES diimplementasikan menggunakan sebuah algoritma enkripsi yang dibuat oleh Joan Daemen dan Vincent Rijmen yang dinamakan Algoritma Rijndael [[4](#Dae02)]. Algoritma tersebut menerima kunci enkripsi yang berukuran hingga 256 bit. Dengan kemampuan komputer sekarang, hampir mustahil AES untuk dapat dipecahkan menggunakan serangan *brute* *force* karena untuk memecahkan kunci enkripsi tersebut, waktu yang dibutuhkan adalah 273 detik.

Seiring berjalannya waktu, nasib AES akan sama dengan DES. Menurut Hukum Moore [[5](#Bow04)], pertumbuhan kecepatan komputer adalah sebanyak dua kali lipat setiap delapan belas bulan. Pada akhirnya, sekitar dua ratus tahun lagi, AES akan mudah dipecahkan menggunakan *brute* *force* *attack* [[6](#Han04)]. Oleh karena itu, pada tugas akhir ini, ditawarkan sebuah solusi yaitu implementasi Algoritma Rijndael yang dapat menerima kunci enkripsi yang berukuran melebihi 256 bit.

1. **RUMUSAN MASALAH**

Permasalahan yang dibahas dalam tugas akhir ini dapat diuraikan sebagai berikut:

1. Dapatkah Algoritma Rijndael diimplementasikan agar dapat menerima ukuran kunci enkripsi yang melebihi 256 bit?
2. Jika Algoritma Rijndael dapat menerima kunci enkripsi yang berukuran melebihi 256 bit, bagaimanakah implementasinya?
3. **RUANG LINGKUP MASALAH**

Permasalahan yang diselesaikan pada tugas akhir ini dibatasi ruang lingkupnya :

1. ukuran *cipher* *key* yang digunakan adalah 32 bit sampai dengan 64 kbit
2. ukuran masukan yang akan dienkripsi mencapai 32 kbit.
3. ukuran masukan dan kunci enkripsi selalu merupakan kelipatan 32 bit.
4. **TUJUAN PENELITIAN**

Tujuan dari tugas akhir ini adalah membuat sebuah perangkat lunak yang dapat melakukan enkripsi Rijndael menggunakan kunci enkripsi yang berukuran 32 bit sampai dengan 64 kbit.

1. **MANFAAT PENELITIAN**

Manfaat yang diberikan dari tugas akhir ini adalah keamanan suatu informasi dapat terjaga dari serangan *brute* *force*.

1. **TINJAUAN PUSTAKA**
2. C

C merupakan bahasa pemrograman yang dibuat oleh Dennis Ritchie pada tahun 1969. Bahasa C merupakan bahasa pemrograman yang bersifat terstruktur. Oleh karena itu, urutan eksekusi hasil kompilasi bahasa C sama dengan urutan baris-baris perintah yang tertulis pada berkasnya.

1. Dev-C

Dev-C adalah sebuah perangkat lunak yang berfungsi sebagai lingkungan kerja untuk pemrograman bahasa C yang menggunakan sistem operasi Windows. Dev-C menggunakan MinGW pada saat melakukan kompilasi.

1. Algoritma Rijndael

Algoritma Rijndael adalah algoritma yang terpilih sebagai AES. Sifat khusus dari algoritma ini adalah setiap bit keluaran dipengaruhi oleh setiap bit masukan dan kunci enkripsinya. Alur kerja Algoritma Rijndael terdiri dari empat bagian, yaitu:

1. *key* *expansion*

*Key* *expansion* adalah proses transformasi dari *cipher* *key* menjadi *expanded* *key*. Pada bagian ini, panjang cipher key diubah dari 4\*Nk menjadi Nb\*(Nr+1). Nk adalah panjang *cipher* *key* dalam satuan *word* (1 *word* = 32 bit). Nb adalah panjang *state* dalam satuan *word*. Selain itu, biasanya Nr = Nk + 6.

1. *add* *round* *key*

Proses ini merupakan proses XOR antara *state* dengan sebuah round key. *Round* *key* merupakan bagian dari *expanded* *key*. Pada sebuah *expanded* *key*, terdapat Nr+1 *round* *key* dengan panjang setiap *round* *key* sama. *Round* yang digunakan pada *add* *round* *key* pertama adalah *round* ke-0.

1. *round*

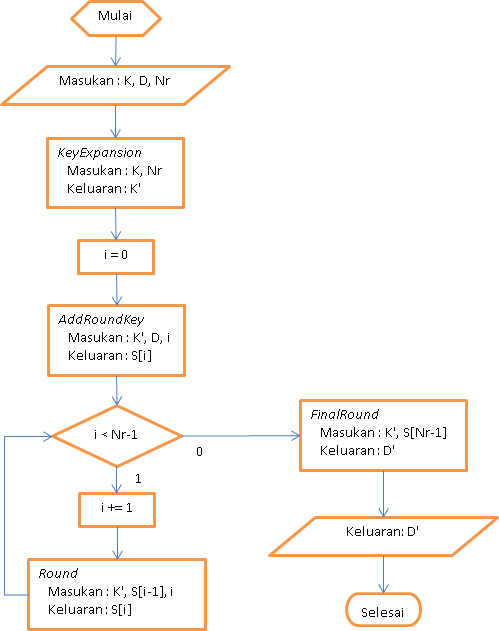
*Round* diulang sebanyak Nr kali. Pada setiap *round*, terjadi proses *byte* *sub*, *shift* *row*, *mix* *column*, dan *add* *round* *key*.

1. *final* *round*

Sama seperti *round*, hanya pada tahap ini *add* *round* *key* tidak dilakukan. Setelah *final* *round* selesai, maka enkripsi Rjindael selesai.

1. **METODOLOGI**

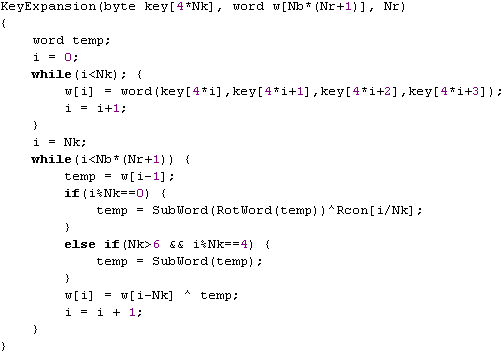
Alur kerja Algoritma Rijndael dapat dilihat pada Gambar 1.



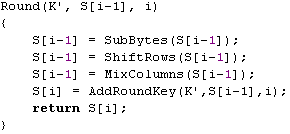
Gambar 1- Diagram alur Algoritma Rijndael

Keterangan Gambar 1:

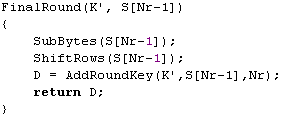
1. K adalah kunci enkripsi
2. D adalah informasi yang akan dienkripsi
3. Nr adalah banyaknya *round* yang akan dilakukan
4. K’ adalah *expanded* *key*
5. S adalah *state*, S[i] adalah *state* ke-i
6. D’ adalah hasil enkripsi
7. *Pseudocode* dari masing-masing proses *KeyExpansion*, *Round*, dan *FinalRound* dapat dilihat pada Gambar 2, Gambar 3, dan Gambar 4.
8. Proses *AddRoundKey* merupakan proses XOR antara setiap bit pada *state* masukan dengan setiap bit pada *expanded* *key* masukan.



Gambar 2 - *Pseudocode* *KeyExpansion*



Gambar 3 - *Pseudocode* *Round*



Gambar 4 - *Pseudocode* *FinalRound*

Langkah-langkah pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi literatur

Tahap ini adalah tahap pengumpulan informasi yang diperlukan untuk perancangan algoritma enkripsi ini. Informasi-informasi tersebut dapat diperoleh melalui *paper*.

1. Implementasi

Implementasi dari algoritma Rijndael dilakukan pada tahap ini. Hal-hal yang akan diimplementasi dapat dilihat pada Gambar 1, Gambar 2, Gambar 3, dan Gambar 4. Implementasi dilakukan dalam bahasa pemrograman C dengan menggunakan perangkat lunak Dev-C.

1. Uji coba dan evaluasi

Pada tahap ini, dilakukan uji coba terhadap hasil implementasi yang dilakukan pada tahap sebelumnya. Uji coba enkripsi menggunakan tes data yang terdapat pada situs spoj.com. Uji coba dikatakan berhasil apabila situs tersebut menerima solusi enkripsi yang dikirimkan.

Data tes yang terdapat pada situs spoj. com memiliki bentuk sebagai berikut:

* + - * 1. baris pertama berisi |K|, yaitu besar kunci enkripsi
        2. baris kedua berisi K, kunci enkripsinya
        3. baris ketiga berisi |D|, yaitu ukuran informasi yang akan dienkripsi
        4. baris keempat berisi D, informasi yang akan dienkripsi.

1. **JADWAL PEMBUATAN TUGAS AKHIR**

Tugas akhir ini diharapkan dapat dikerjakan sesuai jadwal berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Tahapan | 2013 | | | |
| Maret | April | Mei | Juni |
| 1 | Studi Literatur |  |  |  |  |
| 2 | Implementasi |  |  |  |  |
| 3 | Uji coba dan evaluasi |  |  |  |  |
| 4 | Penyusunan Buku Tugas Akhir |  |  |  |  |

1. **DAFTAR PUSTAKA**

x

|  |  |
| --- | --- |
| [1] | Federal Information Processing Standards, DATA ENCRYPTION STANDARD (DES), 1999. |
| [2] | Howard M. Heys, A Tutorial on Linear and Differential Cryptanalysis, 2002. |
| [3] | Federal Information Processing Standards, ADVANCED ENCRYPTION STANDARD (AES), 2001. |
| [4] | Joan Daemen and Vincent Rijmen, *The Design of Rijndael : AES - The Advanced Encryption Standard*. Verlag, Germany: Springer, 2002. |
| [5] | Murrae Bowden, Moore’s Law and the Technology S-Curve, 2004. |
| [6] | Vincent Rijmen, Aleksandra Sowa Hans Dobbertin, *Advanced Encryption Standard - AES*, 4th ed. Bonn, Germany, 2004. |

x