Pengamanan Transmisi Hasil dan Data *Query* Basis Data dengan Algoritma Kriptografi RC4

Mohamad Firda Fauzan – NIM : 13504127

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : mf.fauzan@gmail.com

Abstrak

Makalah ini menjelaskan tentang pengamanan transmisi hasil dan data *query* basis data dengan algoritma kriptografi RC4. Pengamanan transmisi dilakukan dengan cara mengenkripsi/dekripsi hasil dan data *query* yang melewati jaringan antara komputer *server* yang merupakan *server* basis data dengan komputer *client*.

Implementasi pengamanan transmisi hasil dan data *query* ini dilakukan dengan membangun sebuah perangkat lunak berbasis web yang ditempatkan pada komputer *client* untuk mengakses basis data yang ada pada komputer *server*. Perangkat lunak dapat melakukan enkripsi dan dekripsi hasil dan data *query* yang melewati jaringan antara komputer *server* dan *client*. Perangkat lunak dibangun dengan menggunakan bahasa pemrograman *C#, HTML, JavaScript, SQL* dengan kakas *Microsoft Visual Studio 2005* sedangkan basis data menggunakan *Microsoft SQL Server 2005*.

Pengamanan transmisi hasil dan data *query* dapat dilakukan jika perangkat lunak berhasil mengenkripsi hasil dan data *query* selama data tersebut berada pada jaringan *server* dan *client* sehingga data yang berada di jaringan tidak dapat dimengerti oleh penyadap.

Makalah ini memberikan kesimpulan bahwa perangkat lunak yang diimplementasikan berhasil mengenkripsi data yang ditransmisikan antara *server* basis data dengan *server* web dengan algoritma kriptografi RC4 sehingga data tersebut dapat dijaga keamanannya dari penyadap.

Kata kunci : enkripsi, dekripsi, transmisi, *client*, *server*,kriptografi, RC4, *query*, basis data.

1. Latar Belakang

Sistem manajemen basis data adalah suatu kumpulan dari data yang saling terhubung dan suatu program yang dapat mengakses data tersebut[SIL02]. Kumpulan data tersebut kemudian lebih dikenal dengan istilah basis data. Basis data mengandung informasi yang sesuai dengan kebutuhan organisasi yang menggunakannya. Tujuan utama dari sistem manajemen basis data yaitu untuk menyediakan jalan untuk menyimpan dan mendapatkan kembali informasi pada basis data dengan nyaman dan efisien. Sistem basis data didesain untuk menangani jumlah data yang besar, memanajemen data baik struktur penyimpanan maupun mekanisme memanipulasi data. Selain itu basis data harus menjamin keamanan dari informasi yang disimpan, walaupun terjadi *crash* pada sistem dan akses ilegal.

Basis data telah menjadi suatu kebutuhan di beberapa organisasi dan perusahaan komersial pada saat ini. Basis data digunakan secara luas untuk berbagai bidang seperti bisnis, perbankan, pendidikan, kepegawaian, dan lain-lain.

Dengan kebutuhan basis data yang semakin kompleks maka timbul suatu kebutuhan keamanan data dari berbagai macam ancaman diantaranya pembacaan data, modifikasi data dan perusakan data oleh orang yang tidak berhak [SIL02]. Ada beberapa level keamanan pada basis data, diantaranya : keamanan sistem operasi, keamanan sistem manajemen basis data, keamanan jaringan, keamanan fisik, dan keamanan segi manusia[SIL02].

Untuk mengatasi masalah keamanan jaringan maka perlu dibuat suatu sistem yang dapat melakukan pengamanan data selama dalam jaringan, salah satu caranya yaitu mengimplementasikan kriptografi. Penerapan kriptografi untuk mengatasi keamanan transmisi hasil dan data *query* basis data dapat dilakukan dengan cara melakukan enkripsi data selama data tersebut berada dalam jaringan. Enkripsi data yang berupa hasil *query* basis data dilakukan sewaktu hasil *query* dari basis data tepat sebelum memasuki jaringan dan kembali didekripsi setelah sampai ditempat tujuan atau tempat pengakses yang aman. Secara teknis, penerapan kriptografi ini dilakukan dengan membuat modul pengenkripsi dan pendekripsi

pada sumber dan tujuan. Sumber data pada transmisi hasil *query* terdapat pada sistem manajemen basis data dan penerima berupa aplikasi pada *client*.

Implementasi kriptografi dilakukan dengan cara membuat modul kriptografi dijadikan *stored procedure* pada sistem manajemen basis data dan sebagai modul yang terintegrasi dengan aplikasi berbasis web pada *client*.

Pengamanan transmisi basis data memerlukan suatu proses yang cepat, karena itu algoritma kriptografi simetris adalah algoritma yang tepat diimplementasikan untuk kasus ini. Algoritma kunci simetris terbagi menjadi *block cipher* dan *stream cipher*, perbedaannya yaitu *block cipher* beroperasi dengan transformasi yang sama dengan blok besar dari plainteks data sedangkan *stream cipher* beroperasi dengan transformasi waktu pada tiap *byte* plainteks. Karena itu *stream cipher* memiliki kecepatan yang lebih cepat dan kebutuhan *hardware* yang lebih rendah dibandingkan dengan *block cipher*. RC4 merupakan algoritma *stream cipher* yang paling tepat dibandingkan dengan algoritma *stream cipher* lainnya untuk masalah transmisi hasil *query* basis data seperti ini. Hal itu dikarenakan RC4 memiliki proses enkripsinya yang cukup sederhana dan hanya melibatkan beberapa operasi saja per *byte*-nya.

Menurut hasil pengetesan kecepatan algoritma kriptografi *RC4* adalah 5380,035 *Kbytes*/detik pada *Pentium133* memori 16 *MB* pada *Windows 95* [BUD98]. Kecepatan dalam pengetesan ini adalah kecepatan enkripsi di memori, pada kenyataanya proses enkripsi *file* melibatkan banyak faktor lain seperti *interface IO*, tipe *hardisk*, dan lain-lain sehingga pada kenyataannya kecepatan enkripsi lebih lambat dari hasil tersebut, karena dipengaruhi faktor lain tersebut. Hasil pengetesan didapat dengan enkripsi 256 *byte* per blok sebanyak 20480 kali, atau setara dengan kurang lebih 5 *MB* data. Sebagai perbandingan, hasil pengetesan dengan algoritma *Blowfish* pada jenis komputer yang sama yaitu 2300 *KByte*/detik pada 8 *byte* per blok [BUD98]. Jadi pengetesan tersebut membuktikan bahwa *RC4* sebagai algoritma yang cepat dalam pemrosesan proses enkripsi dan dekripsi.

Diharapkan dengan mengimplementasikan algoritma kriptografi *RC4* dapat meningkatkan keamanan transmisi data dari ancaman penyadap tanpa mengurangi performa basis data secara signifikan.

2. Rumusan Masalah

Dari latar belakang yang telah diuraikan maka dapat dirumuskan beberapa permasalahan pada makalah ini yaitu :

1. Bagaimana mengintegrasikan algoritma kriptografi RC4 dengan sistem manajemen basis data yang tersedia dengan fasilitas yang tersedia pada sistem manajemen basis data tersebut.
2. Bagaimana merancang arsitektur sistem yang tepat sesuai dengan domain permasalahan tersebut

3. Tujuan

Dari permasalahan yang ada pada rumusan masalah maka makalah ini bertujuan :

1. Mempelajari cara untuk melakukan penggunaan algoritma kriptografi untuk pengamanan transmisi hasil dan data *query* basis data.
2. Mempelajari perancangan arsitektur sistem yang tepat berkaitan dengan domain permasalahan.
3. Mengimplementasikan algoritma kriptografi pada sistem yang berfungsi untuk mengamankan transmisi hasil dan data *query* basis data.

4. Batasan Masalah

Makalah menetapkan batasan-batasan masalah sebagai berikut :

1. Sistem ini hanya melakukan enkripsi dan dekripsi terhadap data selama dalam transmisi.
2. Sistem ini merupakan suatu bagian program yang berjalan pada satu sistem manajemen basis data tertentu.

5. Dasar Teori

5.1 Algoritma kriptografi RC4

Algoritma kriptografi Rivest Code 4 (RC4) merupakan salah satu algoritma kunci simetris dibuat oleh RSA Data Security Inc (RSADSI) yang berbentuk *stream chipper*. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA(merupakan singkatan dari tiga nama penemu: Rivest Shamir Adleman). RC4 menggunakan panjang kunci dari 1 sampai 256 *byte* yang digunakan untuk menginisialisasikan tabel sepanjang 256 *byte*. Tabel ini digunakan untuk generasi yang berikut dari *pseudo random* yang menggunakan XOR dengan plainteks untuk menghasilkan cipherteks. Masing-masing elemen dalam tabel saling ditukarkan minimal sekali.

RC4 merupakan salah satu jenis *stream cipher* sehingga RC4 memproses unit atau input data, pesan atau informasi pada satu saat. Unit atau data pada umumnya sebuah *byte* atau bahkan kadang kadang *bit* (*byte* dalam hal RC4)[BUD98] sehingga dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah input data, pesan atau informasi tertentu sebelum diproses, atau menambahkan *byte* tambahan untuk mengenkrip.

RC4 digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. Sampai saat ini diketahui tidak ada yang dapat memecahkan/membongkarnya, hanya saja versi ekspor 40 *bit*nya dapat dibongkar dengan cara "*brute force*" (mencoba semua kunci yang mungkin) [BUD98]. RC4 tidak dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas (*trade secret*).

Algoritma RC4 menggunakan dua buah *S-Box* yaitu *array* sepanjang 256 yang berisi permutasi dari bilangan 0 sampai 255, dan *S-Box* kedua, yang berisi permutasi merupakan fungsi dari kunci dengan panjang yang variabel.

Cara kerja algoritma RC4 yaitu inisialisasi *S-Box* pertama, S[0],S[1],...,S[255], dengan bilangan 0 sampai 255. Pertama isi secara berurutan S[0] = 0, S[1] = 1,...,S[255] = 255. Kenudian inisialisasi *array* lain (*S-Box* lain), misal *array* K dengan panjang 256. Isi *array* K dengan kunci yang diulangi sampai seluruh *array* K[0], K[1],...,K[255] terisi seluruhnya.

Proses inisialisasi *S-Box* (*Array* S)

```
for i = 0 to 255
S[i] = i
```

Proses inisialisasi *S-Box* (*Array* K)

```
Array Kunci // Array dengan
panjang kunci "length".
for i = 0 to 255
K[i] = Kunci[i mod length]
```

Kemudian lakukan langkah pengacakan S-Box dengan langkah sebagai berikut :

```
i = 0 ; j = 0
for i = 0 to 255
{
  j = (j + S[i] + K[i]) mod 256
  swap S[i] dan S[j]
}
```

Setelah itu, buat pseudo random byte dengan langkah sebagai :

```
i = ( i + 1 ) mod 256
j = ( j + S[i] ) mod 256
swap S[i] dan S[j]
t = (S[i] + S[j]) mod 256
K = S[t]
```

Byte K di-XOR-kan dengan plainteks untuk menghasilkan cipherteks atau di-XOR-kan dengan cipherteks untuk menghasilkan plainteks. Enkripsi sangat cepat kurang lebih 10 kali lebih cepat dari DES.

Berikut adalah implementasi algoritma RC4 dengan mode 4 byte (untuk lebih menyederhanakan).

Inisialisasi S-Box dengan panjang 4 byte, dengan S[0]=0, S[1]=1, S[2]=2 dan S[3]=3 sehingga array S menjadi :

0	1	2	3

Inisialisasi 4 byte kunci array, Ki. Misalkan kunci terdiri dari 2 byte yaitu byte 1 dan byte 7. Ulang kunci sampai memenuhi seluruh array K sehingga array K menjadi

1	7	1	7

Berikutnya mencampur operasi dimana kita akan menggunakan variabel i dan j ke index array S[i] dan K[i]. pertama kita beri nilai inisial untuk i dan j dengan 0. Operasi Pencampuran adalah pengulangan rumusan (j + S[i]+ K[i]) mod 4 yang diikuti dengan penukaran S[i] dengan S[j].

Untuk contoh ini, karena kita menggunakan array dengan panjang 4 byte maka algoritma menjadi :

```
for i = 0 to 4
j = (j + S[i] + K[i]) mod 4
swap S[i] dan S[j]
```

Dengan algoritma seperti di atas maka dengan nilai awal i = 0 sampai i = 3 akan menghasilkan array S seperti di bawah ini :

iterasi pertama :

```
i = 0, maka
j = (j + S[i] + K[i]) mod
  = (j + S[0] + K[0]) mod 4
  = (0 + 0 + 1 ) mod 4
  =  1
```

swap S[0] dan S[1] sehingga menghasilkan array S :

1	0	2	3

iterasi kedua :

```
i = 1, maka
j = (j + S[i] + K[i]) mod 4
  = (j + S[1] + K[1]) mod 4
  = (1 + 0 + 7 ) mod 4  =  0
```

Swap S[1] dan S[0] sehingga menghasilkan array S :

0	1	2	3

iterasi ketiga :

```
i = 2, maka
j = (j + S[i] + K[i]) mod 4
  = (j + S[2] + K[2]) mod 4
  = (0 + 2 + 1 ) mod 4
  =  3
```

Swap S[2] dan S[3] sehingga menghasilkan array S :

0	1	3	2

iterasi keempat :

```
i = 3, maka
j = (j + S[i] + K[i]) mod 4
  = (j + S[3] + K[3]) mod 4
  = (3 + 2 + 7 ) mod 4
  =  0
```

Swap S[3] dan S[0] sehingga menghasilkan array S :

2	1	3	0

Berikutnya adalah proses enkripsi yaitu meng-XOR-kan pseudo random byte dengan plainteks, misalnya plaintetks “HI”.

Plainteks terdiri dari dua karakter maka terjadi dua iterasi. Berikut iterasi 1 :

Inisialisasi i dan j dengan i = 0; j = 0.

```
i = 0;  j = 0
i = ( i + 1 ) mod 4
  = (0 + 1) mod 4
  = 1
```

dan

```
j = ( j + S[i] ) mod 4
  = ( 0 + 2 ) mod 4
  = 2
```

swap S[i] dan S[j] yaitu S[1] dan S[2] sehingga array S menjadi

2	3	1	0

```
t = (S[i] + S[j]) mod 4
  = (3 + 1) mod 4
  = 0
K = S[t] = S[0] = 2
```

Byte K di-XOR-kan dengan plainteks “H”. Kemudian iterasi 2 :

```
i = 1;  j = 2
i = ( i + 1 ) mod 4
  = (1 + 1) mod 4
  = 2
```

dan

```
j = ( j + S[i] ) mod 4
  = ( 2 + 2 ) mod 4 = 0
```

swap S[i] dan S[j] yaitu S[2] dan S[0] sehingga array S menjadi

1	3	2	0

```
t = (S[i] + S[j]) mod 4
  = (2 + 1) mod 4
  = 3
K = S[t] = S[3] = 2
```

Byte K di-XOR-kan dengan plainteks “**I**”.

Proses XOR pseudo random byte dengan plainteks, dapat dilihat pada tabel 1.

Tabel 1 Proses *XOR pseudo random byte* dengan plainteks pada enkripsi

	H	I
Plainteks	0 1 0 0 1 0 0 0	0 1 0 0 1 0 0 1
Pseodo random byte	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0
Cipherteks	0 1 0 0 1 0 1 0	0 1 0 0 1 0 1 1

Pesan dikirim dalam bentuk cipherteks sehingga setelah sampai di penerima pesan dapat kembali diubah menjadi plainteks dengan meng-XOR-kan dengan kunci yang sama.

Pemrosesan pesan setelah sampai pada penerima dapat dilihat pada tabel 2.

Tabel 2 Proses XOR pseudo random byte dengan cipherteks pada dekripsi

	H	I
Cipherteks	0 1 0 0 1 0 1 0	0 1 0 0 1 0 1 1
Pseodo Random *byte*	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0
Plainteks	0 1 0 0 1 0 0 0	0 1 0 0 1 0 0 1

Berdasarkan hasil pengujian performansi RC4 pada Intel E4500 2.2GHz memori 1GB, diperoleh hasil seperti terlihat pada tabel 3. Hasil pengetesan pada tabel 3 didapat dengan enkripsi 4 kbyte sebanyak 3200 kali, atau setara dengan 100 Mb data.

Tabel 3 Kecepatan Enkripsi RC4

Jml. Thread	Kecepatan Thread ke (dalam Mbps)					Total
	1	2	3	4	5	
1	91.42					91.42
2	87.67	86.48				174.15
3	58.71	61.53	58.71			178.95
4	41.29	56.63	48.48	50.39		196.79
5	36.99	37.42	43.53	41.02	36.36	195.32

5.2 Basis Data

Secara etimologi basis data terdiri dari dua kata yaitu basis dan data yang dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang dan sebagainya yang direkam

dalam bentuk angka, huruf, simbol , teks, gambar, atau kombinasinya [FAT99]. Menurut pengertian lain basis data adalah Sistem basis data adalah suatu sistem yang mengintegrasikan kumpulan dari data yang saling berhubungan satu dengan lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam-macam[DIC05].

5.2.1 *Structured Query Language* (*SQL*)

Suatu basis data mempunyai bahasa khusus yang diperlukan untuk melakukan interaksi dengan basis data itu sendiri. Bahasa basis data yang menjadi standar adalah *SQL* (bahasa *query* yang terstruktur). Basis data menyediakan *Data Definition Language* (*DDL*) yang menspesifikasiskan skema basis data dan *Data Manipulation Language* (*DML*) yang mengekspresikan *query* basis data dan meng-*update* basis data. Pada praktiknya *DDL* dan *DML* bukanlah dua bagian yang terpisah , tetapi *DDL* dan *DML* itu merupakan suatu bagian bentuk sederhana dari suatu basis data.

1. *Data Definition Language* (*DDL*)

   DDL digunakan untuk menspesifikasikan skema basis data, antara lain : membuat tabel baru, membuat indeks, mengubah struktur tabel dan sebagainya.

   Berikut contoh pernyataan dalam bahasa *SQL* untuk mendefinisikan tabel data_pelanggan

```
create table data_pelanggan
(no_pelanggan integer, nama
char(20), alamat char(20))
```

   Selain menspesifikasikan relasi, *DDL* juga menspesifikasikan informasi dari tiap relasi, seperti domain tipe, *primary key*, dan batasan lain sehingga sistem basis data harus selalu mengecek pada batasan tersebut sewaktu terjadi penambahan atau pengubahan data.

   SQL standar mendukung berbagai domain tipe, diantaranya yaitu *char*(n), *varchar*(n), *int*, *smallint*, *numeric*(p,d), *real*, *float*(n), *date*, *time* dan *timestamp*.

2. *Data Manipulation Language* (*DML*)

   Manipulasi data adalah :

   a. Mengambil informasi yang tersimpan dalam basis data.
   b. Menyisipkan informasi baru ke dalam basis data.
   c. Menghapus informasi dari basis data.
   d. Mengubah informasi yang tersimpan dalam basis data.

   DML adalah suatu bahasa yang dapat digunakan untuk mengakses dan memanipulasi data yang terorganisir oleh suatu model data[SIL02]. *DML* terbagi atas dua tipe, yaitu :

   1. *DML* prosedural, dimana pengguna harus menspesifikasikan data apa yang dibutuhkan dan bagaimana mendapatkan data tersebut
   2. *DML* deklaratif atau non-prosedural *DML*, dimana pengguna hanya menspesifikasikan data yang dibutuhkan tanpa menspesifikasikan bagaimana cara mendapatkan data itu. *DML* jenis ini adalah *DML* yang secara umum dikenal, contohnya adalah *SQL language.*

Query adalah suatu pernyataan permintaan untuk mengambil suatu informasi. Bagian dari *DML* yang dapat digunakan untuk mengambil informasi disebut *query language*. Meskipun secara teknis kata *query language* tidak sama dengan *DML* tetapi pada praktiknya kedua istilah ini sama.

6 Analisis Masalah

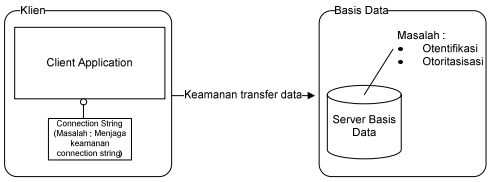
6.1 Masalah

Kebutuhan basis data semakin menjadi kebutuhan yang tidak terpisahkan dari suatu sistem perangkat lunak pada suatu organisasi. Karena itu basis data harus memiliki ketersediaan yang tinggi, yaitu harus dapat diakses kapanpun dan dari manapun. Untuk memenuhi kebutuhan dapat diakses dari manapun maka basis data harus dapat diakses secara *remote* melalui suatu jaringan komputer. Dengan cara ini pengakses basis data tidak harus berada pada lokasi yang secara fisik terintegrasi dengan lokasi data. Pengakses basis data dapat mengakses basis data selama terdapat jaringan yang menghubungkan lokasi basis data dengan lokasi pengakses dan tersedia antarmuka komunikasi dengan basis data.

Salah satu syarat untuk dapat mengakses basis data secara *remote* adalah dengan membuat sebuah aplikasi yang dapat menghubungkan pengakses basis data dengan basis data. Untuk selanjutnya aplikasi tersebut dapat disebut aplikasi *klien*. Aplikasi *klien* tentunya dapat terkoneksi dengan basis data dan dapat melakukan operasi basis data.

Aplikasi *klien* melakukan koneksi dengan basis data melalui media dan protokol tertentu.

Koneksi tersebut dapat dilakukan dengan media kabel maupun nir-kabel dengan protokol jaringan tertentu, misalnya protokol TCP/IP. Pengaksesan basis data, khususnya secara *remote* dapat menimbulkan masalah keamanan. Masalah keamanan akses data dapat dilihat pada gambar III-1.



Gambar 1 Masalah Keamanan Akses Data

Masalah keamanan akses basis data secara *remote* sesuai dengan gambar 1 dapat dikelompokkan menjadi 3 bagian, yaitu :

1. Masalah keamanan pada aplikasi klien

   Masalah keamanan pada klien meliputi masalah bagaimana mengamankan *string* koneksi, yang dapat menghubungkan aplikasi klien dengan basis data. Masalah lain pada aplikasi klien yaitu adanya *sql injection*, yaitu aksi untuk memasukkan data yang tidak dikenal ke dalam *sql language* pada *query*, sehingga eksekusi yang dilakukan *SQL* tidak sesuai dengan yang diinginkan.

2. Masalah keamanan pada jaringan

   Masalah keamanan pada jaringan meliputi keamanan data yang melewati jaringan. Data yang melewati jaringan yang bersifat rahasia harus diamankan dari masalah ini. Diantara data tersebut yaitu data *login* yang didalamnya terdapat *username* dan *password*, data yang tersimpan pada basis data, baik data yang masuk seperti pada *insert query*, maupun data yang di-*retrieve*, seperti pada *select query*.

3. Masalah keamanan pada basis data

   Masalah keamanan pada basis data menyangkut masalah otentikasi pengguna. Hanya pengguna yang telah terotentikasi yang dapat masuk ke dalam basis data. Masalah lainnya adalah kewenangan, yaitu masalah tentang hak akses pengguna terhadap tabel yang berada dalam basis data.

Permasalahan yang menjadi fokus pada makalah ini, adalah masalah keamanan data yang melewati jaringan. Data yang melewati jaringan pada pengaksesan basis data diantaranya adalah data *query* dan hasil *query*. Data *query* adalah data yang terdapat pada sebuah pernyataan *query*, sedangkan hasil *query* adalah nilai kembalian dari basis data atas permintaan *query*. Baik data maupun hasil *query* merupakan sesuatu data yang harus diamankan dalam jaringan antara aplikasi klien dengan basis data.

Pentingnya keamanan data dan hasil *query* pada jaringan antara aplikasi klien dengan basis data dapat diilustrasikan dari contoh berikut. Misalnya seorang nasabah aplikasi perbankan mengecek saldo rekeningnya dengan aplikasi klien. Nasabah perbankan memasukkan *PIN* dalam aplikasi klien dan aplikasi klien mengirim data tersebut menjadi sebuah *query* untuk menampilkan daftar saldo. Keamanan data yang ditransfer baik *PIN* yang dimasukkan, ataupun *query* yang dikirim merupakan celah keamanan yang harus dijaga. Penyadap dapat mencuri *PIN* serta dapat mengintip struktur tabel dari *query* yang dikirim. Efek dari penyadapan ini jelas sangat berbahaya. Jika penyadap mengetahui *PIN*, penyadap dapat melakukan transaksi dengan bank dengan menggunakan *PIN* tersebut dan berlaku seolah-olah seperti nasabah pemilik *PIN* itu. Jika penyadap dapat mengetahui struktur tabel pada basis data, dikhawatirkan pengetahuan tersebut digunakan untuk membuat basis data palsu yang menipu aplikasi klien atau merusak basis data yang ada. Nilai kembalian *query* juga memiliki efek bahaya yang besar jika terjadi penyadapan. Misal data informasi detail karyawan sebuah perusahaan. Jika penyadap mengetahui data detail seorang karyawan, mungkin saja pengetahuan itu dimanfaatkan untuk tindakan penipuan. Dari ilustrasi di atas dapat memberikan gambaran, betapa pentingnya menjaga keamanan data dan hasil *query* yang ke basis data dan dari basis data.

Jadi permasalahan utama dari makalah ini yaitu bagaimana mengamankan transmisi data dan hasil *query*, serta masalah keamanan yang menyertai pengaksesan basis data secara *remote* seperti telah dijelaskan pada bagian sebelumnya.

5.3 Penanganan Masalah

Peningkatan ketersediaan basis data dapat dilakukan dengan membuat sebuah aplikasi klien yang dapat mengakses basis data secara *remote*. Salah satu aplikasi yang dapat melakukan pengaksesan basis data secara *remote* adalah aplikasi berbasis web. Dengan aplikasi berbasis web maka basis data dapat diakses dari tempat yang berjauhan secara fisik, dengan syarat tersedia jaringan yang

menghubungkan antara *server* web dengan *server* basis data.

Permasalahan keamanan merupakan isu utama dalam aplikasi yang mengakses basis data secara *remote*. Permasalahan keamanan yang telah dibahas pada sub bab sebelumnya dapat diselesaikan dengan cara :

1. Solusi keamanan level aplikasi
   Menjaga *string* koneksi dari pembacaan pihak luar yang menyusup dengan cara melakukan enkripsi *string* koneksi sehingga *string* koneksi tidak dapat dimengerti oleh penyusup. Untuk menjaga dari serangan *sql injection* dapat dilakukan dengan membuat *filter* pada level aplikasi terhadap masukkan dari pengguna.
2. Solusi keamanan level jaringan
   Solusi keamanan level jaringan dapat dilakukan dengan melakukan enkripsi data yang melewati jaringan.
3. Solusi keamanan level basis data
   Solusi keamanan pada level ini dapat dilakukan dengan melakukan otentikasi terhadap *user* yang telah terdaftar dan terpercaya. Masalah kewenangan dapat diatasi dengan memberikan hak akses yang dikurangi untuk *user* selain *Administrator*.

Permasalahan utama yang dibahas dalam makalah ini adalah adanya kemungkinan penyadapan data selama data tersebut berada dalam jaringan antara aplikasi klien dengan basis data. Karena alasan itu solusi pada level jaringan akan lebih dideskripsikan lebih detail lagi.

5.3.1 Penanganan Masalah Transmisi Data

Sesuai dengan solusi pada level jaringan yang telah disebutkan pada bagian sebelumnya, salah satu cara untuk melakukan pengamanan data selama dalam jaringan yaitu dengan melakukan enkripsi. Untuk aplikasi berbasis web, pengamanan jaringan ini dilakukan antara *server* web dengan *server* basis data.

Pengamanan data dengan enkripsi ini dilakukan untuk menjaga data agar data terjaga kerahasiaanya. Hal ini sesuai dengan layanan kriptografi yaitu menjaga isi pesan dari pihak manapun yang tidak berhak untuk membacanya. Enkripsi akan mempersulit penyadap, karena data hasil *sniffing* merupakan data yang terenkripsi yang tidak dimengerti oleh penyadap.

5.3.1.1 Implementasi Modul Enkripsi

Enkripsi yang dilakukan pada makalah ini adalah enkripsi data yang ditransmisikan antara *server* web dengan *server* basis data. Data yang ditransmisikan tersebut berupa data *query* dan hasil *retrieve query*. Enkripsi dilakukan untuk mengamankan data yang ditransmisikan. Karena itu dibutuhkan modul untuk melakukan enkripsi dan dekripsi data tersebut di pengirim maupun di penerima sehingga pada makalah ini modul tersebut ditempatkan pada *server* web dan modul lainnya berada pada *server* basis data. Data yang ditransmisikan memiliki dua arah yaitu dari *server* web ke *server* basis data dan sebaliknya dari *server* basis data ke *server* web maka masing-masing modul tersebut harus memiliki kemampuan untuk melakukan enkripsi maupun dekripsi.

Modul enkripsi yang ditempatkan pada *server* web dapat berupa modul program yang berjalan pada *server* web dengan bahasa pemrograman tertentu, sedangkan modul lainnya yang ditempatkan pada *server* basis data merupakan modul program dalam bahasa pemrograman tertentu yang dijalankan sebagai *stored procedure* basis data.

Algoritma kriptografi simetris, dimana kunci untuk enkripsi dan dekripsi adalah sama, menjadi pilihan dalam implementasi modul enkripsi dengan alasan kecepatan dan proses komputasi yang sederhana. Algoritma simetris terdiri atas *block cipher* dan *stream cipher*. Diantara dua pilihan tersebut, algoritma kriptografi *stream cipher* menjadi pilihan pada pengaksesan basis data secara *remote* ini. Hal itu dikarenakan *stream cipher* menghasilkan jumlah data yang sama antara sebelum dan setelah terenkripsi. Padahal ukuran data yang ditransmisikan memberikan pengaruh yang sangat signifikan terhadap kecepaan pengaksesan..

RC4 merupakan pilihan yang tepat untuk masalah ini, dikarenakan RC4 memiliki proses enkripsinya yang cukup sederhana dan hanya melibatkan beberapa operasi saja per *byte*-nya. Menurut hasil pengetesan kecepatan algoritma kriptografi *RC4* adalah 5380,035 *Kbytes*/detik pada *Pentium133* memori 16MB pada *Windows 95* [BUD98]. Kecepatan dalam pengetesan ini adalah kecepatan enkripsi di memori, pada kenyataanya proses enkripsi *file* melibatkan banyak faktor lain seperti *interface IO*, tipe *hardisk*, dan lain-lain sehingga pada kenyataannya kecepatan enkripsi lebih lambat dari hasil tersebut, karena dipengaruhi faktor lain tersebut. Hasil pengetesan tersebut didapat dengan enkripsi 256 *byte* per blok sebanyak

20480 kali, atau setara dengan kurang lebih 5 *MB* data. Sebagai perbandingan, hasil pengetesan dengan algoritma *Blowfish* pada jenis komputer yang sama yaitu 2300 *KByte*/detik pada 8 *byte* per blok [BUD98]. Jadi pengetesan tersebut membuktikan bahwa RC4 sebagai algoritma yang cepat dalam pemrosesan proses enkripsi dan dekripsi.

Pengetesan performansi RC4 pada *Intel E4500 2.2GHz* memori 1GB menghasilkan kecepatan rata-rata 150-160 *Mbps*. Data tersebut diperoleh dengan cara mengenkripsi *file* sepanjang 4096 *byte* sebanyak 3200 kali atau setara dengan kurang lebih 100 *Mbit*. RC4 juga terbukti memproses enkripsi dengan cepat pada *stored procedure* basis data. Menurut pengetesan pada *Intel E4500 2.2GHz* memori 1GB, kecepatan proses enkripsi adalah 640 *Mbps*. Data tersebut diperoleh dengan cara mengenkripsi hasil *query* basis data dengan tipe *varchar* sepanjang 4096 *byte* sebanyak 3200 kali atau setara dengan kurang lebih 100 *Mbit* yang dilakukan dengan *stored procedure*. Dari data tes tersebut dapat disimpulkan proses enkripsi/dekripsi dengan RC4 tidak menjadi *bottleneck* dalam proses transmisi data pada jaringan komputer, karena kecepatan enkripsi baik pada komputer maupun pada *stored procedure* lebih cepat dari kecepatan jaringan yaitu 100 *Mbps*.

RC4 memiliki kelemahan terlalu tingginya kemungkinan terjadi tabel *S-Box* yang sama. Hal ini terjadi karena pada waktu pengisian *S-Box* yang kedua (*Array K*) menggunakan nilai yang berasal dari kunci yang diulang-ulang untuk mengisi seluruh *array K* sebanyak 256 *bytes*. Pengguna memasukkan kunci ‘a’ maupun pengguna memasukkan kunci ‘aaa’ akan menghasilkan *S-Box* (*Array K*) yang sama. Untuk menyelesaikan masalah itu maka kunci masukan dari pengguna sebelumnya dihitung nilai *hash*-nya, kemudian nilai *hash* tersebut dimasukkan untuk mengisi *S-Box* (*Array-K*). Pada makalah ini nilai *hash* dihitung dengan mengunakan *MD5* sehingga berapapun nilai kunci yang dimasukkan akan menghasilkan 16 *bytes* nilai *hash*, kemudian 16 *bytes* tersebut dilakukan permutasi untuk mengisi 256 *bytes S-Box*, sehingga setiap nilai kunci akan menghasilkan *S-Box* yang berbeda.

Kelemahan dari RC4 adalah dengan adanya kemungkinan dapat memecahkan kunci dengan cara melakukan identifikasi beberapa bagian dari cipherteks dengan beberapa bagian dari plainteks yang diketahui dengan meng-XOR-kan. Kelemahan RC4 tersebut dapat dimanfaatkan pada aplikasi ini, hal itu dikarenakan aplikasi ini melakukan enkripsi pada *query* basis data, padahal *query* basis data memiliki struktur yang hampir sama antara satu dengan lainnya. Karena itu mudah sekali untuk mendapatkan sebagian nilai kunci dengan cara meng-XOR-kan antara sebagian cipherteks dengan sebagian plainteks.

Untuk mengatasi masalah pengekstrakan kunci dari sebagian plainteks dan sebagian cipherteks dapat dilakukan dengan cara membuat *initialization vector* sehingga dengan plainteks yang sama dan kunci yang sama akan dihasilkan cipherteks yang berbeda. *Initialization vector* adalah sebuah nilai awal yang dibangkitkan secara acak dengan fungsi tertentu, yang digunakan untuk mengisi *S-Box* pada algoritma Kriptografi RC4. Nilai acak yang dibangkitkan tersebut berbeda setiap kali terjadi eksekusi *query* yang berbeda sehingga menghasilkan cipherteks yang berbeda-beda.

5.3.1.2 Manajemen Kunci Enkripsi dan Dekripsi

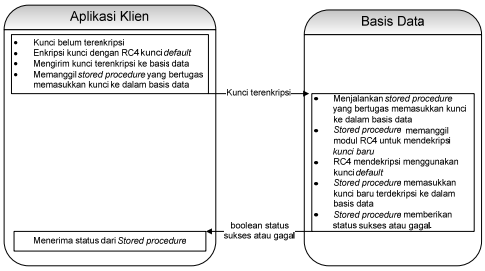
Modul enkripsi membutuhkan kunci untuk menjalankan kunci algoritma kriptografi dan harus memiliki kunci yang sama antara pengirim dan penerima sehingga dibutuhkan suatu manajemen kunci dari kedua modul tersebut. Manajemen kunci dilakukan untuk selalu menyamakan nilai kunci antara kedua modul. Jika ada perubahan kunci pada suatu modul maka kunci juga akan berubah untuk modul yang lain. Kunci pada modul enkripsi yang terletak pada *server* web dapat disimpan pada *file system*, sedangkan modul yang berada pada *server* basis data ditempatkan pada tabel basis data. Pengguna dapat melakukan perubahan kunci enkripsi. Pengguna melakukan perubahan kunci enkripsi pada modul yang berada pada *server* web. Setiap ada perubahan kunci pada modul yang terletak pada *server* web, modul tersebut secara otomatis akan mengirim *query* untuk meng-*update* kunci untuk modul yang terletak pada *server* basis data. Perintah perubahan kunci tersebut cukup menggunakan *query sql* biasa karena kunci untuk modul enkripsi pada basis data terletak pada tabel basis data. Untuk menjaga keamanan pendistribusian kunci, *query update* kunci tersebut dienkrip menggunakan algoritma RC4 dengan kunci lama.

Untuk menjaga integritas kunci baik yang berada pada *file system* maupun pada basis data maka jika terjadi perubahan kunci, sebelum melakukan perubahan kunci pada *file system*, aplikasi melakukan pengecekan terhadap keberhasilan *query* peng-*updatean* kunci pada basis data. Jika operasi *update* berhasil maka sistem melakukan perubahan pada *file system*,

jika gagal tidak melakukan operasi pada *file system* dan menampilkan pesan kegagalan operasi. Aplikasi juga menyediakan fasilitas untuk menghindari terjadinya perubahan kunci yang terdapat pada basis data, sedangkan pada *file system* tidak berubah. Cara menghindari hal itu dengan melakukan *set* kunci dengan nilai yang baru baik untuk kunci pada *file system* maupun pada basis data dengan kata lain dapat dikatakan me-*reset* kunci dengan nilai baru. Mekanisme *reset* kunci sama seperti mekanisme pembuatan kunci baru, yaitu menuliskan nilai kunci yang baru, kemudian nilai tersebut dikirim ke basis data dengan *query* yang terenkripsi dengan algoritma kriptografi RC4 dengan kunci *default*. Dengan cara itu maka integritas antara kunci yang satu dengan yang lain dapat terjaga.

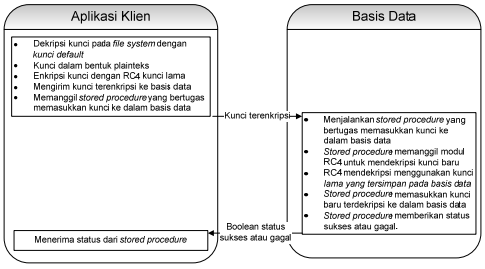
5.3.1.3 Protokol Transmisi Data

Protokol transmisi inisialisasi kunci antara aplikasi klien dan basis data dapat dilihat pada gambar 2.



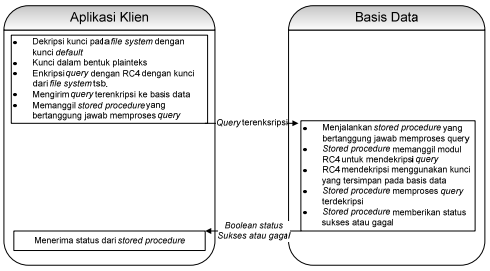
Gambar 2 Protokol inisialisasi kunci enkripsi

Protokol transmisi mengganti kunci antara aplikasi klien dan basis data dapat dilihat pada gambar 3.



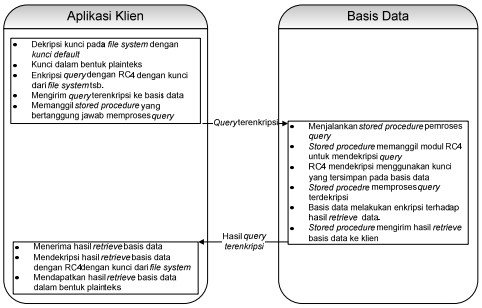
Gambar 3 Protokol mengganti kunci enkripsi

Protokol transmisi eksekusi *query* yang tidak me-*retrieve* nilai,seperti *insert, edit* dan *delete*, antara aplikasi klien dan basis data dapat dilihat pada gambar 4.



Gambar 4 Protokol eksekusi *query* yang tidak me-*retrieve* data

Protokol transmisi eksekusi *query* yang me-*retrieve* nilai, seperti perintah *select*, antara aplikasi klien dan basis data dapat dilihat pada gambar 5.



Gambar 5 Protokol eksekusi *query* yang mer-*retrieve* data

5.3.2 Penanganan Masalah Otentikasi

Keamanan di level aplikasi dapat ditingkatkan dengan membuat otentikasi pengguna pada level aplikasi. Karena itu, aplikasi klien memiliki pengguna yang terbagi menjadi *Administrator* dan pengguna biasa. Dimana *Administrator* memiliki hak untuk menambah dan menghapus *user account* pengguna biasa. Pengguna biasa hanya memiliki hak untuk mengubah *password*. *Username* dan *password* untuk otentikasi lokal ini disimpan pada *file system* dalam bentuk menyimpan nilai *hash* dari *username* dan *password*. Dengan adanya otentikasi pengguna ini maka hanya pengguna yang terdaftar yang dapat menjalankan aplikasi ini.

5.3.3 Penanganan Masalah Integritas Data

Keamanan untuk menjaga integritas basis data dilakukan dengan membuat batasan dalam manipulasi data. Karena itu aplikasi klien membutuhkan pengetahuan tentang *key* yang ada pada tabel dalam basis data, *key* diantaranya *primary key* dan *foreign key*. Jika pengguna melakukan manipulasi terhadap tabel dalam suatu basis data maka aplikasi akan melakukan pengecekan terhadap batasan tabel tersebut. Batasan tabel tersebut berkaitan dengan relasi tabel tersebut dengan tabel yang

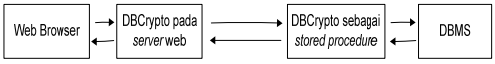
lain. Contoh, jika pengguna ingin melakukan penghapusan terhadap suatu *record* pada tabel tertentu, sebelum melakukan operasi penghapusan, aplikasi akan melakukan pengecekan terhadap *primary key* apakah *key* tersebut menjadi *foreign key* di tabel yang lain. Jika suatu *primary key* menjadi *foreign key* di tabel yang lain maka aplikasi akan memberikan pesan kesalahan. Hal yang sama juga berlaku ketika pengguna memasukkan suatu record dalam suatu tabel yang mengandung *foreign key*. Sebelum dilakukan operasi, aplikasi mengecek dahulu, apakah *foreign key* yang dimasukkan ada sebagai *primary key* di tabel yang lain. Jika ada maka operasi dilanjutkan dan jika tidak maka akan memunculkan pesan kesalahan. Dengan cara seperti itu maka integritas dari suatu basis data dapat dijaga.

6. Analisis dan Perancangan Perangkat Lunak

6.1 Analisis Perangkat Lunak

Perangkat lunak yang dibangun merupakan perangkat lunak yang digunakan untuk melakukan pengamanan data yang ditransmisikan dari *client* ke *server* atau sebaliknya dengan cara mengenkripsi data tersebut dengan algoritma kriptografi RC4. *Client* dalam makalah ini adalah pihak yang mengirim *query* yaitu *server* web dan *Server* adalah pihak yang menerima dan memproses *query* yaitu *server* basis data.

Perangkat lunak yang dibangun dinamakan **DBCrypto**. Perangkat lunak yang akan dibangun merupakan suatu modul program yang terletak pada *server* web dan pada *server* basis data. Modul program yang berada pada *server* web akan dapat diakses melalui *web browser*, sedangkan modul program yang berada pada *server* basis data adalah suatu *stored procedure* yang akan dipanggil oleh modul program yang terdapat pada *server* web. Skema umum perangkat lunak dapat dilihat pada gambar 6.



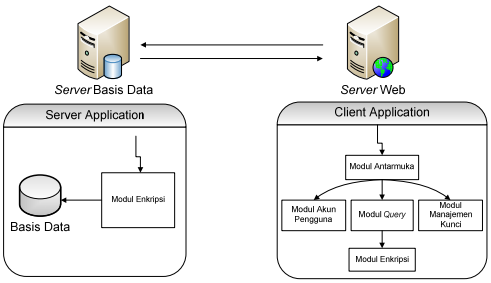
Gambar 6 Skema umum perangkat lunak

Hasil akhir dari aplikasi ini adalah aplikasi berbasis web. Aplikasi berbasis web dipilih karena pertimbangan ketersediaan dan kemudahan. Selain itu diharapkan pengguna dapat dengan cepat mempelajari dan memanfaatkan aplikasi ini hanya dengan menggunakan web browser yang biasa digunakan.

Pada dasarnya perangkat lunak ini memiliki fungsi utama untuk melakukan enkripsi atau dekripsi data yang ditransmisikan dari *server* web ke *server* basis data atau sebaliknya. Enkripsi dilakukan baik untuk *string query* maupun hasil *query* dengan algoritma kriptografi RC4. Perangkat lunak ini menerima masukan dari pengguna melalui antarmuka yang disediakan. Pengguna dapat memasukkan *query* sederhana, seperti *select*, *insert*, *edit* dan *delete*, tanpa perlu melalui penulisan bahasa *query*, tetapi cukup dengan melakukan interaksi dengan antarmuka yang telah disediakan. Karena itu, perangkat lunak ini juga mengolah masukan dari pengguna tersebut menjadi sebuah *query* tertentu, kemudian *query* ini dienkripsi oleh modul program yang terdapat pada *server* web sebelum *query* tersebut dikirim ke *server* basis data. Data yang terenkripsi dikirim ke *server* basis data, kemudian setelah sampai di *server* basis data, data tersebut kembali didekripsi oleh modul program yang berada pada *server* basis data. Hasil *query* akan dikirim kembali ke aplikasi yang berada pada *server* web dengan cara yang sama, yaitu data dienkripsi sebelum ditransmisikan dan kembali didekripsi sesampainya di tujuan.

6.2 Perancangan Perangkat Lunak

Perangkat lunak yang dibuat pada makalah ini merupakan perangkat lunak berarsitektur *client-server*. Dimana aplikasi yang berada pada *server* web bertindak sebagai *client* dan aplikasi yang berada pada *server* basis data bertindak sebagai *server*. Arsitektur perangkat lunak dapat dilihat pada gambar 7.



Gambar 7 Arsitektur perangkat lunak

Berdasarkan gambar 7 maka arsitektur perangkat lunak dapat dibagi menjadi tiga bagian proses utama yaitu :

1. Proses Manajemen Akun

   Proses ini merupakan proses yang berhubungan dengan akun pengguna, seperti membuat akun pengguna, otentikasi *username* dan *password* yang dimasukkan

oleh pengguna dan pengubahan *password* untuk tiap-tiap akun pengguna.

Proses pembuatan akun adalah proses menambahkan *username* dan *password* baru pada daftar pengguna sehingga dihasilkan akun pengguna baru.

Proses otentikasi ini berupa proses pencocokan nilai *hash* dari *username* dan *password* yang dimasukkan dengan nilai *hash* yang tersimpan pada *file system*. Proses ini juga mengatur kewenangan pengguna, dimana pengguna dengan status *Administrator* memiliki hak yang lebih dibandingkan dengan pengguna biasa. Hak tersebut seperti hak membuat dan menghapus *user account* untuk pengguna lain, dan pengubahan kunci enkripsi/dekripsi.

2. Proses *Query*

   Proses ini merupakan proses yang berhubungan dengan eksekusi sebuah *query*. Perintah tersebut berupa :

   1. Perintah mendapatkan daftar basis data.
   2. Perintah mendapatkan daftar tabel.
   3. Perintah mendapatkan struktur tabel.
   4. Perintah mendapatkan data yang tersimpan dalam tabel.
   5. Perintah untuk memanipulasi isi tabel yaitu menambah, mengubah dan menghapus.

   Proses *query* ini membutuhkan proses enkripsi dan dekripsi untuk mengamankan data selama dalam transmisi dari *server* web ke *server* basis data dan sebaliknya. Karena itu terdapat modul enkripsi yang terletak pada *server* web dan *server* basis data yang bertanggung jawab untuk menangani masalah enkripsi dan dekripsi data selama transmisi.

3. Proses Manajemen Kunci

   Proses ini merupakan proses untuk menginisialisasi, meng-*update* dan menjaga integritas kunci untuk enkripsi dekripsi yang terdapat pada *server* web dan *server* basis data.

7. Implementasi Perangkat Lunak

7.1 Batasan Implementasi

Hal-hal yang menjadi batasan implementasi adalah :

1. Perangkat lunak hanya diimplementasikan pada satu macam basis data relasional yaitu *Microsoft SQL Server 2005*.
2. Hasil perancangan diimplementasikan pada jaringan komputer lokal dengan satu komputer sebagai *server* basis data dan satu komputer sebagai *server* aplikasi web.

7.2 Lingkungan Implementasi

Lingkungan implementasi perangkat lunak terdiri atas :

1. Perangkat Keras

   Perangkat keras yang digunakan dalam pengembangan perangkat lunak pada makalah ini antara lain :

   1. Prosesor *Intel Pentium* 4 2.4 *GHz*
   2. Memori RAM 768 *MB*.
   3. Kapasitas *harddisk* 120 GB.

2. Perangkat Lunak

   Perangkat lunak yang digunakan dalam pengembangan perangkat lunak pada tugas akhir ini antara lain :

   1. Sistem Operasi : *Microsoft Windows XP*.
   2. Server basis data : *Microsoft SQL Server 2005*.
   3. Server Web : *ASP .NET Development Server*
   4. Bahasa Pemrograman : *C#, HTML, JavaScript, SQL*.
   5. Kakas pengembangan modul : *Microsoft Visual Studio 2005*
   6. Browser : *Internet Explorer*.

8. Pengujian

8.1 Tujuan Pengujian

Pengujian perangkat lunak pada makalah ini bertujuan untuk menguji kemampuan perangkat lunak yaitu apakah perangkat lunak mampu:

1. Mengamankan transmisi data antara *server* basis data dengan *server* web dengan cara mengenkripsi dan dekripsi data *query* dan hasil *query* basis data pada *server* web dan *server* basis data.
2. Melakukan otentikasi *username* dan *password* untuk dapat masuk ke aplikasi dan menambah atau menghapus akun pengguna.
3. Menampilkan basis data beserta tabel dan isi tabel pada basis data.
4. Menerjemahkan perintah berbasis GUI dari pengguna menjadi *query* basis data.
5. Mengganti kunci enkripsi dan dekripsi.

8.2 Lingkungan Pengujian

Lingkungan pengujian perangkat lunak terdiri atas :

1. Komputer Klien

   Pada pengujian ini komputer klien merupakan komputer dimana perangkat lunak terinstalasi sebagai aplikasi berbasis web. Komputer ini berfungsi sebagai *server* web. Berikut spesifikasi komputer klien :

   1. Prosesor *Intel Core 2 T5300 1.73GHz*
   2. Memori *RAM* 2 *GB*.
   3. Kapasitas *harddisk* 120 GB.
   4. Antarmuka Jaringan : *Realtek RTL8168/8111 PCI-E Gigabit Ethernet NIC* dan kabel LAN sepanjang 10 M.
   5. Sistem Operasi : *Microsoft Windows XP*.
   6. *Server* Web : *IIS 5.1*
   7. Browser : *Internet Explorer*.

2. Komputer *Server*

   Pada pengujian ini komputer *server* merupakan komputer dimana basis data terinstalasi. Komputer ini berfungsi sebagai *server* basis data. Berikut spesifikasi komputer *server* :

   1. Prosesor *Intel Pentium* 4 2.4 *GHz*
   2. Memori *RAM* 768 *MB*.
   3. Kapasitas *harddisk* 120 *GB*.
   4. Antarmuka Jaringan : *VIA Rhine II Fast Ethernet Adapter*.
   5. Sistem Operasi : *Microsoft Windows XP*.
   6. *Server* Basis Data: *Microsoft SQL Server 2005*.
   7. Kakas Pengujian : *Wireshark* (kakas untuk melihat *traffic* data pada antarmuka jaringan)

8.3 Evaluasi Pengujian

Berdasarkan kasus uji dan hasil pengujian yang dilakukan selama implementasi perangkat lunak maka dapat diperoleh hasil evaluasi pengujian sebagai berikut :

1. Perangkat lunak dapat mengamankan transmisi *query* dan hasil *query* antara *server* basis data dengan *server* web dengan cara mengenkripsi data *query* dan hasil *query*. Hal itu dibuktikan pada pengujian *use case query* dimana data yang melewati jaringan merupakan data yang terenkripsi. Enkripsi ini menghasilkan cipherteks yang berbeda walaupun berasal dari plainteks yang sama dan kunci yang sama. Hal itu dikarenakan enkripsi menggunakan *initialization vector* yang nilainya akan berbeda untuk setiap eksekusi *query*.
2. Perangkat lunak mengharuskan pengguna yang dapat menggunakan perangkat lunak merupakan pengguna yang terotentikasi. Jika pengguna tidak terotentikasi maka pengguna tidak dapat menggunakan perangkat lunak walaupun pengguna langsung menuju alamat halaman yang akan digunakan.
3. Perangkat lunak dapat menampilkan isi tabel basis data dengan mengkonversi semua tipe data menjadi bentuk *string*. Keluaran tampilan tabel dapat berbeda untuk mesin basis data yang berbeda. Contoh keluaran tipe tanggal pada umumnya dalam format mm/dd/yyyy hh:mm:ss AM/PM(jam dalam format 12 jam), tetapi pada kasus lain ditampilkan format tanggal dalam dd/mm/yyyy hh:mm:ss (jam dalam format 24 jam). Keluaran tipe *binary* dan *varbinary* ditampilkan dalam bentuk *string* bertuliskan system.byte[].Untuk tipe bit keluaran yang ditampilkan dalam bentuk *string* "*True*" atau "False"
4. Keluaran tipe *binary* dan *varbinary* dalam bentuk *string* bertuliskan system.byte[] dan tanggal dengan format dd/mm/yyyy hh:mm:ss (jam dalam format 24 jam) memiliki permasalahan ketika baris dengan tipe data tersebut dilakukan perubahan dan pengguna tidak melakukan perubahan kolom pada tipe tersebut maka akan terjadi kegagalan *query*. Untuk mengatasinya pengguna harus melakukan perubahan format pada textbox yang tersedia sesuai dengan kompatibilitas data tersebut. Karena itu untuk tipe data *binary* dan *varbinary* pengguna harus memasukkan nilai berupa bilangan integer ataupun heksadesimal, sedangkan untuk tipe data tanggal harus memasukkan tanggal dengan format mm/dd/yyyy atau mm/dd/yyyy hh:mm:ss AM/PM(jam dalam format 12 jam). Cara lain yaitu dengan mengosongkan data pada textbox dengan asumsi data bersangkutan kosong nilainya dan memiliki nilai NULL pada basis data.
5. Perangkat lunak dapat melakukan operasi *select*, *insert*, *edit* dan *delete* dengan cara melakukan pengisian textbox dan klik link perintah pada antarmuka perangkat lunak.
6. Perangkat lunak mampu melakukan operasi *query* basis data untuk semua tipe data selain *image*. Hal itu dikarenakan pemrosesan *image* tidak dapat dilakukan dengan *query* dalam bentuk *string*.
7. Perangkat lunak hanya mampu melaksanakan operasi *edit* dan *delete* jika tabel mengandung *primary key*. Hal itu dikarenakan *query* untuk *edit* dan *delete* menggunakan klausa *where* pada atribut *primary key*.

8. Perangkat lunak hanya mampu melaksanakan operasi *query*, jika data masukkan pengguna kompatibel dengan tipe data kolom bersangkutan.
9. Perangkat lunak mampu melakukan pembuatan kunci enkripsi dekripsi dan pengubahan kunci kecuali untuk pembuatan dan penggantian kunci dengan *string* yang menghasilkan nilai hash yang mengandung karakter yang tidak bisa disimpan oleh basis data sehingga tidak semua *string* dapat digunakan untuk dijadikan kunci enkripsi dekripsi.
10. Pembuatan dan pengubahan kunci enkripsi dekripsi mendistribusikan kunci dari aplikasi klien ke basis data dengan mengirimkan kunci dalam bentuk cipherteks yang merupakan hasil enkripsi algoritma kriptografi RC4 dengan kunci lama untuk perubahan kunci dan kunci *default* untuk pembuatan kunci. Karena itu distribusi pembuatan kunci memiliki risiko dapat dipecahkan nilai kunci jika pengguna mengetahui nilai kunci *default* yang tersimpan pada aplikasi.

9. Penutup

9.1 Kesimpulan

Makalah ini yang membahas pengamanan transmisi hasil dan data *query* basis data dengan algoritma kriptografi RC4 memberikan kesimpulan :

1. Perangkat lunak yang dihasilkan berhasil mengenkripsi data *query* dan hasil *query* selama dalam transmisi antara *server* basis data *Microsoft SQL Server 2005* dengan *server* web.
2. Arsitektur perangkat lunak yang tepat untuk permasalahan transmisi data dan hasil *query* basis data ini adalah arsitektur *client-server*. Arsitektur *client-server* dipilih karena pada permasalahan ini basis data diakses secara remote dari tempat yang berbeda dari lokasi basis data berada. Oleh karena itu modul perangkat lunak diimplementasikan baik pada sisi *client* maupun *server*.
3. Perangkat lunak yang dihasilkan berhasil melakukan otentikasi dan otorisasi pada pengguna, melakukan manajemen pengguna dan manajemen kunci enkripsi dekripsi, serta dapat melakukan operasi pengolahan basis data, seperti operasi *select*, *insert*, *edit* dan *delete* pada tabel yang berada pada basis data yang ditampilkan oleh perangkat lunak.
4. Perangkat lunak berhasil melakukan operasi *insert* dan *edit* pada tabel jika nilai masukan tabel sesuai dengan kompatibilitas tipe data tiap kolom pada suatu tabel. Operasi *insert* dan *edit* akan gagal jika nilai masukan tidak kompatibel dengan tipe data kolom pada suatu tabel.

9.2 Saran

Beberapa saran untuk pengembangan perangkat lunak dalam makalah ini atau pun penelitian yang dapat dilakukan lebih lanjut adalah sebagai berikut :

1. Perangkat lunak ini dapat dikembangkan lebih lanjut sehinga tidak hanya dapat melakukan operasi *select*, *insert*, *edit* dan *delete*.
2. Perangkat lunak dalam makalah ini dapat dikembangkan lebih lanjut sehingga dapat diaplikasikan pada sistem basis data lain selain *Microsoft SQL Server*.
3. Perangkat lunak dapat dikembangkan lebih lanjut untuk menangani pengamanan transmisi data dan hasil *query* pada sistem basis data terdistribusi.

10 Daftar Referensi

[BUD98] B. Sukmawan , *RC4 Stream Cipher*, 1998. http://www.bimacipta.com/rc4.htm , *diakses 23 April 2008 20.40 WIB*

[DIC05] D. Ekklesia , *Studi dan Implementasi Pengamanan Basis Data dengan Teknik Kriptografi RC4*, Tugas Akhir Departemen Teknik Informatika ITB , 2005.

[FAT99] Fathansyah, *Basis Data*, Informatika, Bandung, 1999

[MFF08] M.F.Fauzan, *Pengamanan Transmisi Hasil dan Data Query basis Data dengan Algoritma Kriptografi* , Tugas Akhir Program Studi Teknik Informatika STEI ITB, 2008.

[MEI02] J.D. Meier, Alex Mackman, Michael Dunner, and Srinath Vasireddy, *Secure Communication & Data Acces Security*, Microsoft Corporation, 2002.

[MUN06] R. Munir, *Diktat Kuliah IF5054 Kriptografi*, Institut teknologi Bandung , 2006.

[SCH96] B. Schneier, *Aplied Cryptography 2rd*, John Wiley,& Sons, 1996.

[SIL02] A. Silberschatz, H. F.Korth, S.Sudarshan, *Database System Concepts*, 4rd edition, McGraw-Hill, 1999