

ALGORITMA RC4 DALAM PROTEKSI TRANSMISI DAN HASIL QUERY UNTUK ORDBMS POSTGRESQL

Yuri Ariyanto

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya
Jl. Arief Rachman Hakim 100 Surabaya
Email: yuri_bjn@yahoo.com

ABSTRAK: Pada penelitian ini dibahas mengenai bagaimana mengimplementasikan algoritma kriptografi RC4 dalam proteksi terhadap *query* dan hasil *query*, pengamanan dilakukan dengan cara melakukan enkripsi dan dekripsi selama keduanya berada di dalam jaringan. Pengimplementasian dari penelitian ini yaitu membangun sebuah software yang akan diletakkan di sisi client yang berfungsi mengakses database yang diletakkan di sisi server. Software yang dibangun memiliki fasilitas untuk mengenkripsi dan mendekripsi *query* dan hasil *query* yang dikirimkan dari client ke server dan juga sebaliknya. Dengan demikian transmisi *query* dan hasil *query* dapat terjamin keamanannya. Terjaminnya keamanan transmisi *query* dan hasil *query* dapat dikatakan berhasil jika software berhasil mengenkripsi *query* dan hasil *query* yang ditransmisikan sehingga apabila terjadi penyadapan terhadap keduanya, penyadap tidak akan mengerti isi data tersebut. Kesimpulan dari penelitian ini yaitu software yang dibangun berhasil mengenkripsi *query* dan hasil *query* yang ditransmisikan antara aplikasi client dan server database.

Kata kunci: enkripsi, transmisi, RC4, query, database.

ABSTRACT: In this research will be worked through about how cryptography RC4's algorithm implementation in protection to query result and of query, security by encryption and decryption up to both is in network. Implementation of this research which is build software in client that function access databases that is placed by the side of server. Software that building to have facility for encryption and decryption query result and of query that is sent from client goes to server and. transmission query result and of query can secure its security. Well guaranteed transmission security him of query result and of query can be told to succeed if success software can encryption query result and of query which transmission so that in the event of scanning to both, scanning will not understand data content. Conclusion of this research that is woke up software succeed encryption query and result of query which transmission between application of client and of server databases.

Keywords: encryption, transmission, RC4, query, databases.

PENDAHULUAN

Dengan adanya kebutuhan database yang semakin besar dan kompleks, secara otomatis akan diikuti dengan kebutuhan akan keamanan terhadap data yang tersimpan dari berbagai ancaman yang dapat berupa pengaksesan, perubahan serta perusakan data oleh pihak/*user* yang tidak mempunyai kewenangan. Terdapat beberapa level keamanan pada database, diantaranya: keamanan sistem operasi, keamanan sistem manajemen database, keamanan fisik dan keamanan dari segi *user/manusia*.

Untuk menambah tingkat keamanan dapat dilakukan dengan cara mengimplementasikan kriptografi untuk melindungi *query* yang dikirimkan dan hasil *query* yang akan diterima selama keduanya berada dalam jaringan komputer. Algoritma kriptografi yang digunakan adalah RC4.

Algoritma kriptografi Rivest Code 4 (RC4) merupakan salah satu algoritma kunci simetris yang dibuat oleh RSA Data Security Inc (RSADSI). Dipilihnya RC4 sebagai metode kriptografi untuk proteksi *query* dan hasil *query* berdasarkan pada beberapa hal, yaitu:

1. Pengamanan transmisi database memerlukan suatu proses yang cepat, maka algoritma kriptografi simetris adalah solusi yang tepat.
2. RC4 merupakan algoritma *stream cipher* yang paling tepat dibandingkan dengan algoritma *stream cipher* yang lain untuk masalah transmisi *query* database seperti ini. Hal ini dikarenakan RC4 memiliki proses enkripsi yang cukup sederhana dan hanya melibatkan beberapa operasi saja per *byte*-nya.

Menurut hasil pengujian, kecepatan algoritma kriptografi RC4 adalah 5380,035 Kbytes/detik pada Pentium1 33 memori 16 MB pada Windows 95.

Kecepatan dalam testing ini adalah kecepatan enkripsi di memori, pada kenyataannya 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 testing didapat dengan enkripsi 256 byte per blok sebanyak 20480 kali, atau setara dengan kurang lebih 5 MB data. Sebagai perbandingan, hasil testing dengan algoritma Blowfish pada jenis komputer yang sama yaitu 2300 KByte/detik pada 8 byte per blok [1, 2]. Jadi testing tersebut membuktikan bahwa RC4 sebagai algoritma yang cepat dalam pemrosesan proses enkripsi dan dekripsi.

Dengan penggunaan algoritma RC4, diharapkan dapat meningkatkan tingkat keamanan transmisi data tanpa mengurangi performansi database secara signifikan.

METODE

Algoritma RC4

Algoritma kriptografi Rivest Code 4 (RC4) merupakan salah satu algoritma kunci simetris dibuat oleh RSA Data Security Inc (RSADSI) yang berbentuk stream chipper [3]. Algoritma ini ditemukan pada tahun 1978 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) 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.

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, $S[0], S[1], \dots, S[255]$, dengan bilangan 0 sampai 255. Pertama isi secara berurutan $S[0] = 0$,

$S[1] = 1, \dots, S[255] = 255$. Kemudian inisialisasi array lain (S-Box lain), misal array K dengan panjang 256. Isi array K dengan kunci yang diulangi sampai seluruh array $K[0], K[1], \dots, K[255]$ 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

$K[i] = \text{Kunci}[i \bmod \text{length}]$

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 berikut:

$i = (i + 1) \bmod 256$

$j = (j + S[i]) \bmod 256$

swap $S[i]$ dan $S[j]$

$t = (S[i] + S[j]) \bmod 256$

$K = S[t]$

Byte K di-XOR-kan dengan plainteks untuk menghasilkan cipherteks atau di-XOR-kan dengan cipherteks untuk menghasilkan plainteks. Enkripsi ini 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, K_i . 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 diinisialisasi i dan j dengan 0. Operasi Pencampuran adalah pengulangan rumusan $(j + S[i] + K[i]) \bmod 4$ yang diikuti dengan penukaran $S[i]$ dengan $S[j]$. Untuk contoh ini, karena kita mengguna-

kan 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 4
= (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 plainteks "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 "T".
```

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 | 01001000 | 01001001 |
| Pseudo random byte | 00000010 | 00000010 |
| Cipherteks | 01001010 | 01001011 |

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

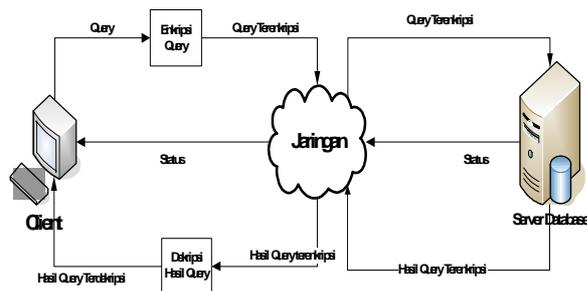
| | H | I |
|--------------------|----------|----------|
| Plainteks | 01001010 | 01001011 |
| Pseudo random byte | 00000010 | 00000010 |
| Cipherteks | 01001000 | 01001001 |

PERANCANGAN SISTEM UNTUK APLIKASI PERCOBAAN

Rancangan Aplikasi Percobaan Secara Umum

Pada dasarnya software ini memiliki fungsi utama untuk melakukan proses enkripsi dan dekripsi data yang dikirimkan antara aplikasi *client* dan *server* database dan juga sebaliknya. Enkripsi dilakukan untuk *query* dan hasil *query* saja, dan algoritma yang digunakan adalah RC4. Software ini menerima inputan data berupa *string query*, *user* dapat menginputkan *query* sederhana seperti *insert*, *update*, *delete*. kemudian *query* ini dienkripsi oleh fungsi yang terdapat pada aplikasi *client* sebelum *query* tersebut dikirim ke *server* database. Data yang terenkripsi dikirim ke *server* database, kemudian setelah sampai di *server* database, data tersebut kembali didekripsi oleh *function/stored procedure* yang berada pada *server* database. Hasil *query* akan dikirim kembali ke aplikasi *client* dengan cara yang sama, yaitu data dienkripsi sebelum ditransmisikan dan kembali didekripsi sesampainya di tujuan.

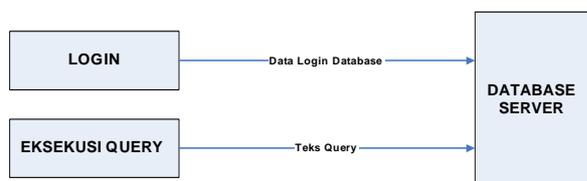
Gambaran rancangan aplikasi secara umum dapat dilihat pada Gambar 1.



Gambar 1. Gambaran Aplikasi secara umum

Perancangan Aplikasi Client

Pada aplikasi client terdapat dua proses yaitu login dan eksekusi query. Proses login yaitu login ke database server (Gambar 2). Proses eksekusi query dibedakan menjadi dua yaitu query yang me-retrieve data dan query yang tidak me-retrieve data.



Gambar 2 Blok Diagram Aplikasi Client

Perancangan modul pada Server Database

Pada server database akan dibuat dua *function* yaitu *function* enkripsi RC4 dan *function* *sql_runner*.

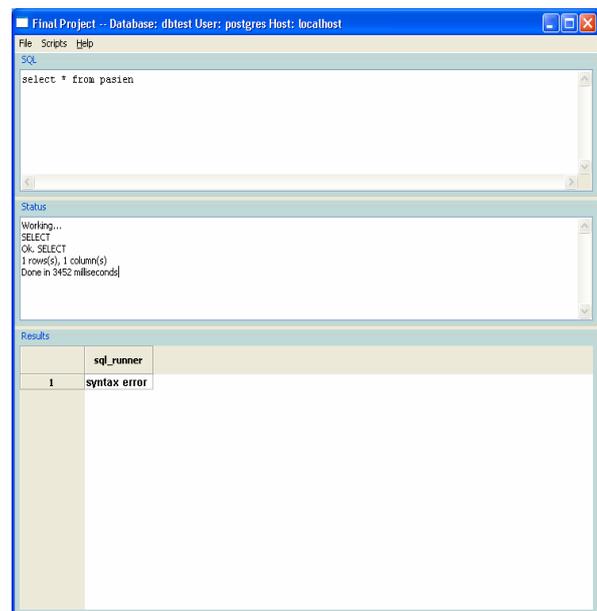
Kedua fungsi itu akan dibuat pada database yang akan diakses oleh aplikasi client. *Function* enkripsi RC4 berfungsi sebagai dekriptor untuk mendekripsi teks *query* sebelum dijalankan oleh *function* *sql_runner*. Di samping itu juga sebagai enkriptor untuk hasil *query* yang akan dikirimkan kepada aplikasi *client*.

IMPLEMENTASI APLIKASI UNTUK PERCOBAAN

Aplikasi Client

Pada halaman *query editor* terdapat 3 panel, yaitu panel SQL, status dan result. Pada panel SQL terdapat text box multiline yang berguna bagi user untuk input sintaks *query* yang nantinya akan dieksekusi. Panel kedua yaitu panel status, pada panel ini terdapat text box *multiline* yang berfungsi untuk menampilkan status eksekusi *query*, apakah berhasil atau gagal. Serta dapat menampilkan *error* dari proses eksekusi *query*, dan juga untuk menampilkan lama proses eksekusi *query*.

Sedangkan pada panel ke tiga, terdapat grid yang berfungsi untuk menampilkan hasil eksekusi *query*. Tampilan aplikasi *client* dapat dilihat pada Gambar 3.



Gambar 3. Halaman Query Editor

Modul Enkripsi dan Dekripsi di Server Database

Implementasi fungsi enkripsi RC4

Fungsi RC4 yang berada pada *server* berguna untuk mendekripsi *query* yang dikirimkan oleh aplikasi *client* yang dalam keadaan terenkripsi, dan juga berguna untuk mengenkripsi hasil *query* sebelum dikirimkan kembali kepada *client*.

Evaluasi

Ujicoba Performa

Ujicoba berikut bertujuan untuk mengetahui kemampuan sistem dalam melakukan proses enkripsi dan dekripsi. Pengujian dilakukan dengan menghitung waktu yang dibutuhkan dari mulai pengiriman *query* hingga hasil *query* diterima kembali oleh client.

Ujicoba Tanpa Menggunakan Fungsi Enkripsi dan Dekripsi

Uji coba ini dilakukan untuk mengetahui seberapa cepat pemrosesan *query* tanpa dilakukan pengimplementasian fungsi enkripsi RC4 pada sisi *client* dan *server* database.

Tabel 3. Hasil Pengujian Tanpa Menggunakan Fungsi Enkripsi dan Dekripsi

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|-------------|------------------|
| 1 | 100 | 5730 Bytes | 30 milliseconds |
| 2 | 200 | 12468 Bytes | 63 milliseconds |
| 3 | 300 | 18692 Bytes | 77 milliseconds |
| 4 | 400 | 25363 Bytes | 92 milliseconds |
| 5 | 500 | 31249 Bytes | 125 milliseconds |
| 6 | 600 | 38127 Bytes | 141 milliseconds |
| 7 | 700 | 44183 Bytes | 155 milliseconds |
| 8 | 800 | 50236 Bytes | 187 milliseconds |

Dapat dilihat pada Tabel 3 bahwa semakin panjang kunci yang dibentuk maka semakin lama waktu yang diperlukan untuk pemrosesan *query*.

Ujicoba Menggunakan Fungsi Enkripsi dan Dekripsi

Uji coba ini dilakukan untuk mengetahui seberapa cepat pemrosesan *query* dengan melakukan pengimplementasian fungsi enkripsi RC4 pada sisi client dan server database. Ujicoba ini dilakukan dengan menggunakan panjang kunci yang berbeda-beda, hal ini dilakukan untuk mengetahui apakah panjang kunci mempengaruhi secara signifikan terhadap performa enkripsi dan dekripsi.

Uji coba yang pertama yaitu menggunakan modul enkripsi yang dibangun dengan bahasa pemrograman C, dengan panjang kunci enkripsi sepanjang 8, 20, dan 30 karakter. Hasil pengujian dapat dilihat pada Tabel 4, 5, dan 6.

Dari ketiga tabel dapat disimpulkan bahwa panjang kunci dapat mempengaruhi kecepatan dari enkripsi dan dekripsi tapi dengan selisih waktu yang sangat kecil, sehingga perbedaan panjang kunci tidak

mempengaruhi performa dari fungsi enkripsi dan dekripsi secara keseluruhan.

Tabel 4. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 8 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27741 Bytes | 156 milliseconds |
| 2 | 200 | 56909 Bytes | 328 milliseconds |
| 3 | 300 | 85451 Bytes | 483 milliseconds |
| 4 | 400 | 114603 Bytes | 657 milliseconds |
| 5 | 500 | 142711 Bytes | 811 milliseconds |
| 6 | 600 | 172121 Bytes | 983 milliseconds |
| 7 | 700 | 200327 Bytes | 1157 milliseconds |
| 8 | 800 | 228659 Bytes | 1328 milliseconds |

Tabel 5. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 20 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27741 Bytes | 125 milliseconds |
| 2 | 200 | 56909 Bytes | 297 milliseconds |
| 3 | 300 | 85451 Bytes | 483 milliseconds |
| 4 | 400 | 114603 Bytes | 655 milliseconds |
| 5 | 500 | 142711 Bytes | 796 milliseconds |
| 6 | 600 | 172121 Bytes | 985 milliseconds |
| 7 | 700 | 200327 Bytes | 1156 milliseconds |
| 8 | 800 | 228659 Bytes | 1282 milliseconds |

Tabel 6. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 30 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27741 Bytes | 155 milliseconds |
| 2 | 200 | 56909 Bytes | 280 milliseconds |
| 3 | 300 | 85451 Bytes | 483 milliseconds |
| 4 | 400 | 114603 Bytes | 657 milliseconds |
| 5 | 500 | 142711 Bytes | 828 milliseconds |
| 6 | 600 | 172121 Bytes | 983 milliseconds |
| 7 | 700 | 200327 Bytes | 1141 milliseconds |
| 8 | 800 | 228659 Bytes | 1328 milliseconds |

Uji coba yang kedua yaitu menggunakan modul enkripsi yang dibangun dengan bahasa pemrograman python, dengan panjang kunci enkripsi sepanjang 8, 20 dan 30 karakter. Hal ini dilakukan untuk mengetahui apakah ada perbedaan waktu yang signifikan dengan menggunakan modul yang dibangun dengan bahasa pemrograman C, dikarenakan postgresql dibangun dengan bahasa C. Hasil pengujian dapat dilihat pada Tabel 7, Tabel 8, dan Tabel 9.

Tabel 7. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 8 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27742 Bytes | 203 milliseconds |
| 2 | 200 | 56910 Bytes | 437 milliseconds |
| 3 | 300 | 85452 Bytes | 641 milliseconds |
| 4 | 400 | 114604 Bytes | 891 milliseconds |
| 5 | 500 | 142712 Bytes | 1108 milliseconds |
| 6 | 600 | 172122 Bytes | 1328 milliseconds |
| 7 | 700 | 200328 Bytes | 1546 milliseconds |
| 8 | 800 | 228660 Bytes | 1608 milliseconds |

Tabel 8. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 20 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27742 Bytes | 202 milliseconds |
| 2 | 200 | 56910 Bytes | 437 milliseconds |
| 3 | 300 | 85452 Bytes | 656 milliseconds |
| 4 | 400 | 114604 Bytes | 922 milliseconds |
| 5 | 500 | 142712 Bytes | 1093 milliseconds |
| 6 | 600 | 172122 Bytes | 1344 milliseconds |
| 7 | 700 | 200328 Bytes | 1547 milliseconds |
| 8 | 800 | 228660 Bytes | 1750 milliseconds |

Tabel 9. Hasil Uji Coba Enkripsi dan Hasil Enkripsi Menggunakan kunci dengan panjang 30 karakter

| Uji Coba Ke | Banyaknya record | Ukuran Data | Waktu |
|-------------|------------------|--------------|-------------------|
| 1 | 100 | 27742 Bytes | 203 milliseconds |
| 2 | 200 | 56910 Bytes | 437 milliseconds |
| 3 | 300 | 85452 Bytes | 657 milliseconds |
| 4 | 400 | 114604 Bytes | 891 milliseconds |
| 5 | 500 | 142712 Bytes | 1093 milliseconds |
| 6 | 600 | 172122 Bytes | 1233 milliseconds |
| 7 | 700 | 200328 Bytes | 1562 milliseconds |
| 8 | 800 | 228610 Bytes | 1766 milliseconds |

KESIMPULAN

Berdasarkan hasil ujicoba yang dilakukan selama implementasi software dapat diperoleh hasil sebagai berikut:

1. Software berhasil mengamankan transmisi *query* dan hasil *query* selama keduanya berada di jaringan dengan cara mengenkripsi keduanya menggunakan algoritma RC4.
2. Software pada sisi client dapat melakukan operasi *insert*, *update*, *delete* dengan cara menginputkan *query* pada text box yang disediakan.
3. Penerapan algoritma untuk mengenkripsi *query* dan hasil *query* secara langsung berpengaruh terhadap waktu eksekusi dari *query*, hal ini dapat dilihat pada hasil uji coba.

DAFTAR PUSTAKA

1. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, "HANDBOOK of APPLIED CRYPTOGRAPHY", Asiacypt/Auscrypt Proceedings, 1997.
2. Stallings, W., 2005, *Cryptography and Network Security Principles and Practices*, 4th Edition", Prentice Hall.
3. Sukmawan, B., 1998, *RC4 Stream Cipher*, <http://www.bimacipta.com/rc4.htm>, diakses 23 April 2009.
4. Fathansyah, 1999, *Basis Data*, Informatika, Bandung.
5. Munir, R., 2006, *Kriptografi*, Informatika, Bandung.
6. PostgreSQL 8.3.7 Documentation
7. Python 2.5 Documentation