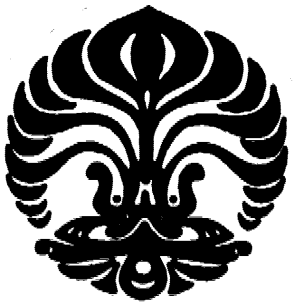
IMPLEMENTASI ALGORITMA DASAR RC4 *STREAM CIPHER* DAN PENGACAKAN *PLAINTEXT* DENGAN TEKNIK *DYNAMIC BLOCKING* PADA APLIKASI SISTEM INFORMASI KEGIATAN SKRIPSI DI DEPARTEMEN TEKNIK ELEKTRO

SKRIPSI

OLEH:

ENDANG FIANSYAH

04 04 03 0369



DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

Depok Juni 2008

IMPLEMENTASI ALGORITMA DASAR RC4 *STREAM CIPHER* DAN PENGACAKAN *PLAINTEXT* DENGAN TEKNIK *DYNAMIC BLOCKING* PADA APLIKASI SISTEM INFORMASI KEGIATAN SKRIPSI DI DEPARTEMEN TEKNIK ELEKTRO

SKRIPSI

OLEH:

ENDANG FIANSYAH

04 04 03 0369

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

Depok Juni 2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul**:**

IMPLEMENTASI ALGORITMA DASAR RC4 *STREAM CIPHER* DAN PENGACAKAN *PLAINTEXT* DENGAN TEKNIK *DYNAMIC BLOCKING* PADA APLIKASI SISTEM INFORMASI KEGIATAN SKRIPSI DI DEPARTEMEN TEKNIK ELEKTRO

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro, Departemen Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Jakarta, 17 Juli 2008

Endang Fiansyah

NPM 04 04 03 0318

LEMBAR PERSETUJUAN

Skripsi dengan judul:

IMPLEMENTASI ALGORITMA DASAR RC4 *STREAM CIPHER* DAN PENGACAKAN *PLAINTEXT* DENGAN TEKNIK *DYNAMIC BLOCKING* PADA APLIKASI SISTEM INFORMASI KEGIATAN SKRIPSI DI DEPARTEMEN TEKNIK ELEKTRO

Dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan pada sidang skripsi.

Depok, 17 Juli 2008

Dosen Pembimbing

Muhammad Salman, ST, MIT

NIP. 131 865 234

UCAPAN TERIMA KASIH

Selama pelaksanaan Skripsi ini penulis telah mendapatkan banyak bantuan serta dorongan yang berarti dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT, atas segara rahmat dan nikmat-Nya kepada penulis selama penulis melaksanakan skripsi dan dalam penulisan laporan skripsi ini.
2. Muhammad Salman, ST, MIT. sebagai dosen pembimbing atas segala bimbingan, saran, diskusi, dan penentuan judul skripsi ini.
3. Kedua orang tua, yang telah memberikan dukungan atas penyelesaian skripsi ini baik secara moral maupun finansial
4. Rekan-rekan di departemen teknik elektro khususnya di lab digital, TTPL, elektronik, telekomunikasi dan teman-teman lain yang namanya tidak dapat disebutkan satu persatu dan telah banyak memberikan dukungan serta semangat dalam penyelesaian skripsi ini.
5. Adik dan saudara-saudara lain yang juga telah memberikan banyak dukungan dan bantuan.

Endang Fiansyah
NPM 04 04 03 0369
Departemen Teknik Elektro

Dosen Pembimbing
Muhammad Salman, ST, MIT

IMPLEMENTASI ALGORITMA DASAR RC4 *STREAM CIPHER* DAN PENGACAKAN *PLAINTEXT* DENGAN TEKNIK *DYNAMIC BLOCKING* PADA APLIKASI SISTEM INFORMASI KEGIATAN SKRIPSI DI DEPARTEMEN TEKNIK ELEKTRO

ABSTRAK

Aplikasi yang memanfaatkan jaringan komputer memerlukan suatu mekanisme keamanan agar proses transaksi data pada jaringan komputer dapat berjalan dengan aman, dengan harapan data-data penting tidak mampu dibaca ketika proses transaksi berlangsung.

Pada skripsi ini akan dikembangkan aplikasi pengolah *database* pada pendataan kegiatan skripsi menggunakan algoritma enkripsi dasar RC4 *stream cipher dan* teknik tambahan *dynamic blocking* menggunakan bahasa pemrograman VB 6 dan *database access*.

Aplikasi yang dikembangkan memiliki kekurangan dan kelebihan berdasarkan pengujian. Kekurangannya ialah *client* menerima waktu tanggap yang lebih tinggi dibandingkan dengan aplikasi yang tidak menggunakan mekanisme keamanan enkripsi yaitu dengan rata-rata peningkatan waktu tanggap sebesar 8,84 detik atau 4,17% untuk proses *upload* dan 4,96 detik atau 200,95% untuk proses *download*. Kelebihannya, selain data yang diharapkan semakin aman dengan metode RC4 dan *dynamic blocking* juga karena semua proses enkripsi-dekripsi hanya dilakukan di *client,* dengan demikian tidak membebani *server* dan jaringan. Perbedaan waktu tanggap terbesar antara proses *download* terpisah dengan proses *download* bersama pada pengujian untuk 5000 *record* terhadap 3 *client* hanya 2,4 detik (3,9%). Kecepatan prosesor mempengaruhi kinerja dari aplikasi yang menggunakan teknik enkripsi-dekripsi. Hal ini terlihat pada penggunaan prosesor 2,4GHz dimana terdapat peningkatan kinerja sebesar 378,4% dibandingkan dari penggunaan prosesor 1,4GHz.

Keywords : RC4, *Dynamic Blocking*, Keamanan Data, *Database*

Endang Fiansyah
NPM 04 04 03 0369
Engineering Dept.

Supervisor
Muhammad Salman, ST, MIT Electrical

IMPLEMENTATION OF RC4 STREAM CIPHER ALGORITHM AND RANDOMIZE PLAINTEXT WITH DYNAMIC BLOCKING TECHNIQUE IN INFORMATION SYSTEM APPLICATION FOR RESEARCH DATABASE IN ELECTRICAL ENGINEERING DEPARTEMENT

ABSTRACT

Application that using computer network need security mechanism so data transaction process at computer network secure, and hope important data cannot be read while transaction process going on.

This research about to developed application database processing using rc4 stream cipher algorithm and dynamic blocking technique using VB 6 programming language and access database.

Developed application has minuses and pluses based on testing and analyzing. The minuses is client receive higher response time than the application that not using encryption security mechanism, with the average of raising response time 8,84 second or 4,17% for upload process and 4,96 second or 200,95% for download process. The pluses, data expect to be secure with RC4 method and dynamic blocking and also all encryption-decryption process only in client, so server and network will not more weighted by this encryption security process. The greatest differences of response time between separate download process and join download process on testing for 5000 records at 3 clients only 2.4 second (3.9%). On testing observed that processor speed influence application performance that using encryption-decryption technique. This thing seen at the using 2.4GHz processor, raising performance 378.4% comparison to the using 1,4GHz processor.

Keywords : RC4, Dynamic Blocking, Data Security, Database

Daftar Isi

Daftar Tabel

Daftar Gambar

Daftar Singkatan

Bps	bit per second
CPU	Central Processing Unit
DES	Data Encryption Standard
e-commerce	electronic commerce
e-business	electronic business
e-learning	electronik learning
IP	Internet Protocol
IPV6	Internet Protocol Version 6
IV	Initialization Vector
KSA	Key Scheduling Algoritm
LAN	Local Area Network
ODBC	Open Database Connectivity
OSI	Open System Interconnection
QOS	Quality of Service
RC4	*Ron Code* atau *Rivest's Cipher* 4
SQL	Structured Query Language
TKIP	Tempororal Key Integrity Protocol
WEP	Wired Eqivalent Privacy
WPA	Wi-Fi Protected Access

Daftar Istilah

Ciphertext	: Pesan hasil proses enkripsi dan merupakan hasil operasi kunci (opsional) dengan *plaintext* (pesan asli)
Cipher	: Alat berupa perangkat keras atau lunak untuk mengubah kode pesan dengan pengolahan operasi bit
Client	: Komputer yang mengakses atau meminta pelayanan data dan informasi dari komputer lain (misalkan *server*)
Database	: Tempat organisasi dan penyimpanan data dan informasi
Download	: Proses untuk mengambil data atau informasi dari pusat data
Field	: Kolom-kolom dalam suatu tabel *database*
Initialization Vector	: Nilai awal yang digunakan bersama dengan kunci enkripsi untuk menghasilkan kunci enkripsi yang berbeda
Internet	: Suatu sistem komunikasi global yang menghubungkan komputer-komputer dan jaringan-jaringan komputer diseluruh dunia
Key Stream	: Kunci-kunci enkripsi yang dioperasikan berurutan sesuai dengan indeksnya hanya pada operasi *stream cipher*.
Network	: Interkoneksi antar sistem komputer yang sama atau berbeda
Password	: Kata kunci untuk proses otorisasi pengguna
Plaintext	: Pesan asli sebelum terenkripsi yang umumnya memuat informasi yang akan dirahasiakan.
Programmer	: Seseorang yang membuat program atau perangkat lunak menggunakan bahasa pemrograman tertentu.
Record	: Catatan atau rekaman yang disimpan pada *database*
Server	: Komputer yang menyimpan atau melayani permintaan data dan informasi
Table	: Tempat berbentuk tabel untuk mengolah dan menyimpan *record* pada *database*.
Upload	: Proses untuk mengirimkan data atau informasi dari *client* ke

Server (sebagai komputer pusat data)

Username : Nama pengguna pada suatu sistem

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG MASALAH

Aplikasi dengan memanfaatkan jaringan informasi termasuk teknologi jaringan komputer dan telekomunikasi berkembang dengan pesat, bahkan berbagai kegiatan manusia terkini dibantu oleh teknologi jaringan komputer dan telekomunikasi, misalkan untuk bertukar dan berbagi informasi (*Web*, *Chating*), proses pembelajaran (*e-learning*, *sharing* ebook), bahkan berbisnis (*e-commerce*, *e-business*). Karena aplikasinya meluas tidak sedikit data-data penting yang ditransmisikan dijaringan komputer misalkan pada proses bisnis dengan saling menukarkan no kartu kredit, proses penyimpanan data penting pada *database* dan lain sebagainya. Pentingnya menjaga data tersebut dari pihak yang tidak memiliki wewenang hak akses dan olah data di sisi *client*, jaringan, *server* maupun *database*. Untuk itu perlu adanya keamanan baik pada aplikasi ataupun jaringan komputer agar data aman dari penyusup yang berniat atau tidak untuk membaca data penting dan rahasia.

Saat ini keamanan pada jaringan komputer terus berkembang, seiring dengan suatu pernyataan bahwa jaringan komputer tidak akan pernah aman hingga 100% dan terbukti hingga sekarang masih banyak kasus-kasus terkait dengan serangan terhadap data ataupun jaringan komputer dari *trafik monitoring and capturing* dan pembuatan virus yang umum terus dilakukan. Yang menjadi perhatian disini selain masalah keamanan ialah terdapat kemungkinan bahwa tambahan sistem keamanan pada jaringan komputer akan memberatkan disisi jaringan ataupun aplikasi yang menggunakannya seperti pada *server*. Untuk itu diperlukan mekanisme keamanan yang aman dan tetap memberikan kualitas layanan yang terbaik. Untuk itu

pada skripsi ini akan dikembangkan keamanan pada aplikasi di *client* untuk proses pengolahan *database* dalam aplikasi pendataan kegiatan penelitian skripsi dengan teknik enkripsi dasar RC4 yang terkenal memiliki keunggulan dalam kecepatan proses.

1.2 TUJUAN PENULISAN

Tujuan dari penulisan skripsi ini adalah untuk implementasi dan pengujian pengembangan algoritma enkripsi RC4 pada suatu aplikasi pengolahan *database* pada suatu sistem jaringan komputer.

Skripsi ini terutama merupakan tambahan untuk riset yang pernah dilakukan oleh Rahmat Sobari, Yoyok Andoyo, Noni Juliasari, ,Galuh Dian Maulana dalam tesis dengan judul "Pengamanan Data Sistem Biling Warnet dengan Menggunakan RC4 *Stream Cipher*". Pada tesis tersebut sebelumnya menggunakan teknik enkripsi RC4 dengan pengembangan teknik *blocking* statik dan penggunaan kunci tambahan yang acak pada aplikasi biling warnet.

Pada skripsi ini akan menggunakan teknik enkripsi RC4 dan *dynamic blocking* tanpa kunci tambahan yang acak serta pengujiannya terkait keamanan dan kualitas pelayanannya.

1.3 BATASAN MASALAH

Permasalahan yang dibahas pada skripsi ini terbatas pada implementasi keamanan transaksi pengolahan *database* di komputer *client* berbasis dasar algoritma RC4 dan teknik *blocking dynamic* pada suatu sistem jaringan komputer dan parameter yang mempengaruhi kualitas layanan aplikasi tersebut.

1.4 SISTEMATIKA PENULISAN

Laporan skripsi ini disusun dengan sistematika sebagai berikut:

1. **PENDAHULUAN**

   Bab pertama merupakan bagian yang akan menjabarkan latar belakang dan tujuan dari perancangan aplikasi dan penulisan skripsi ini.

2. **LANDASAN TEORI**

   Bab kedua merupakan bagian dari penulisan skripsi yang membahas mengenai landasan teori yang digunakan untuk penelitian dan analisa.

3. **PERANCANGAN SISTEM**

   Bab ketiga merupakan bab yang membahas tentang perancangan aplikasi dan jaringan uji, serta skenario perancangan dan metode pengukuran dan pengambilan data yang telah dilakukan.

4. **ANALISIS**

   Bab keempat ini akan dibahas mengenai analisis dari implementasi aplikasi pada jaringan LAN. Analisis tersebut merupakan analisis kinerja dari setiap skenario dan keamanan pada saat transaksi dari pengiriman hingga penyimpanan informasi pada *database*.

5. **KESIMPULAN**

   Bab kelima ini berisi kesimpulan dari semua pembahasan yang telah dilakukan pada penulisan skripsi ini.

BAB II

JARINGAN KOMPUTER DAN KRIPTOGRAFI

2.1 JARINGAN KOMPUTER DAN KONSEP KEAMANAN JARINGAN

2.1.1 Istilah Jaringan Komputer

Hingga saat ini manfaat jaringan komputer sangat luas dan terus meluas dari aplikasi untuk rumahan hingga aplikasi untuk bisnis. Jaringan komputer merupakan suatu himpunan interkoneksi sejumlah komputer yang *autonomous* oleh sebuah teknologi jaringan komputer. Dua atau lebih komputer dapat dikatakan terinterkoneksi bila komputer dapat saling bertukar informasi. Koneksi fisik antar komputer dapat menggunakan media yang berbeda dari kawat tembaga, serat optik, gelombang mikro ataupun satelit komunikasi[2].

Berdasarkan jarak dan protokol yang digunakan jaringan komputer dapat terbagi menjadi beberapa klasifikasi yaitu diantaranya diberikan pada tabel 1.1 berikut.

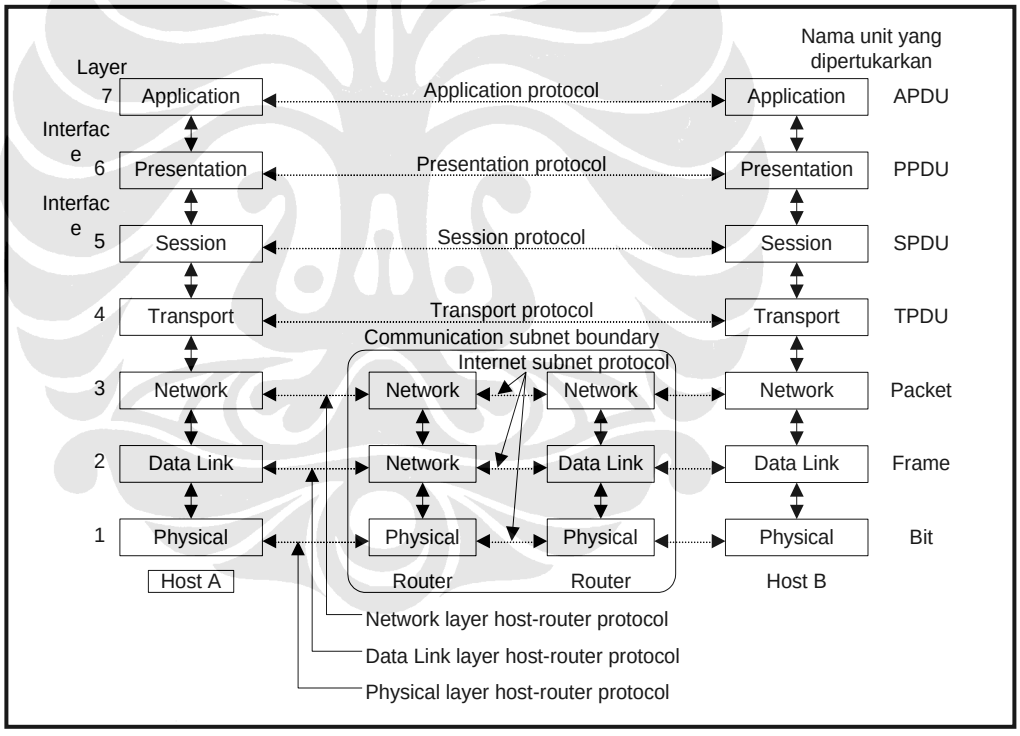
Tabel 1.1 Klasifiksi Jaringan Komputer Berdasarkan Jarak [1]

Jarak antar prosesor	Prosesor di tempat yang sama	Contoh	Jarak antar prosesor	Prosesor di tempat yang sama	Contoh
0,1 m	Papan rangkaian	*Data flow machine*	10 km	Kota	*Metropolitan Area Network*
1 m	Sistem	*Multicomputer*	100 km	Negara	*Wide Area Network*
10 m	Ruangan	*Local Area Network*	1.000 km	Benua	
100 m	Gedung		10.000 km	Planet	*The Internet*
1 km	Kampus				

2.1.2 OSI (*Open System Interconnection*)

2.1.2.1 Model Referansi OSI Sebagai Model Aliran Data dan Protokol Yang Digunakan

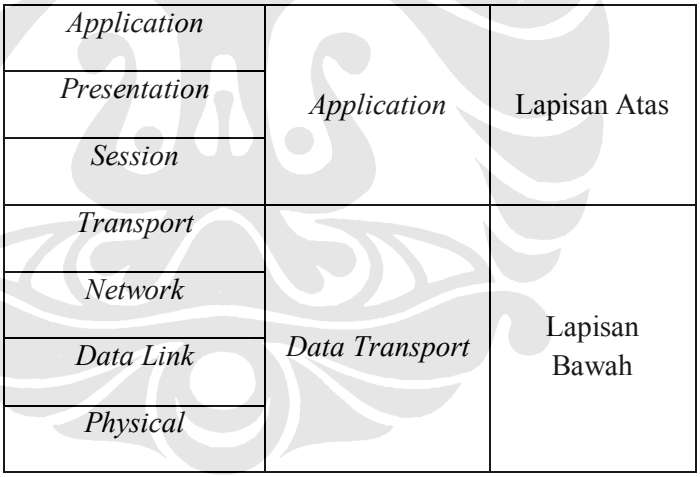
Model referensi OSI merupakan kerangka kerja yang menggambarkan bagaimana suatu aliran data dan pemrosesannya dari suatu aplikasi pada suatu komputer hingga aplikasi lain pada komputer yang berbeda. Model OSI terdiri atas tujuh lapisan dan memiliki tugas masing-masing dalam mengolah data yang ditransmisikan. 7 lapisan pada model referensi OSI seperti pada gambar 2.1 yaitu dari lapisan tertinggi dan yang lebih dekat kepada pengguna yaitu *application*, presentation, *session*, *transport*, *network*, *datalink* dan *phisical*.



Gambar 2.1 OSI Model [1]

Setiap lapisan memiliki tugasnya masing-masing dalam mengolah data dan memiliki standard atau protokol masing-masing sehingga dalam pengembangan jaringan komputer diharapkan sesuai standar OSI ini. Secara umum ke tujuh lapisan

OSI ini terbagi kedalam dua kategori yaitu lapisan atas dan lapisan bawah. Lapisan atas meliputi *application*, *presentation* dan *session*. Lapisan ini umumnya diaplikasikan pada perangkat lunak dan pada lapisan ini akan berhubungan langsung dengan *user*. Lapisan bawah secara umum digunakan untuk mengendalikan proses transport data. *Transport layer* secara umum digunakan untuk membangun suatu koneksi antar komputer sebelum mengirimkan data. *Network layer* secara umum digunakan untuk proses penjaluran sehingga diperoleh jalur terbaik saat data dikirimkan. *Transport layer* dan *network layer* pun diaplikasikan pada perangkat lunak. Lapisan bawah terakhir yaitu *data link* dan *physical layer*. Berbeda dengan lapisan lain *data link layer* diimplementasikan dengan perangkat lunak dan perangkat keras. Dan *phisical layer* diimplementasikan dengan perangkat keras. *Data link* dan *physical layer* secara umum digunakan untuk melakukan transmisi informasi hingga *pyisical* dan *data link layer* yang dituju.



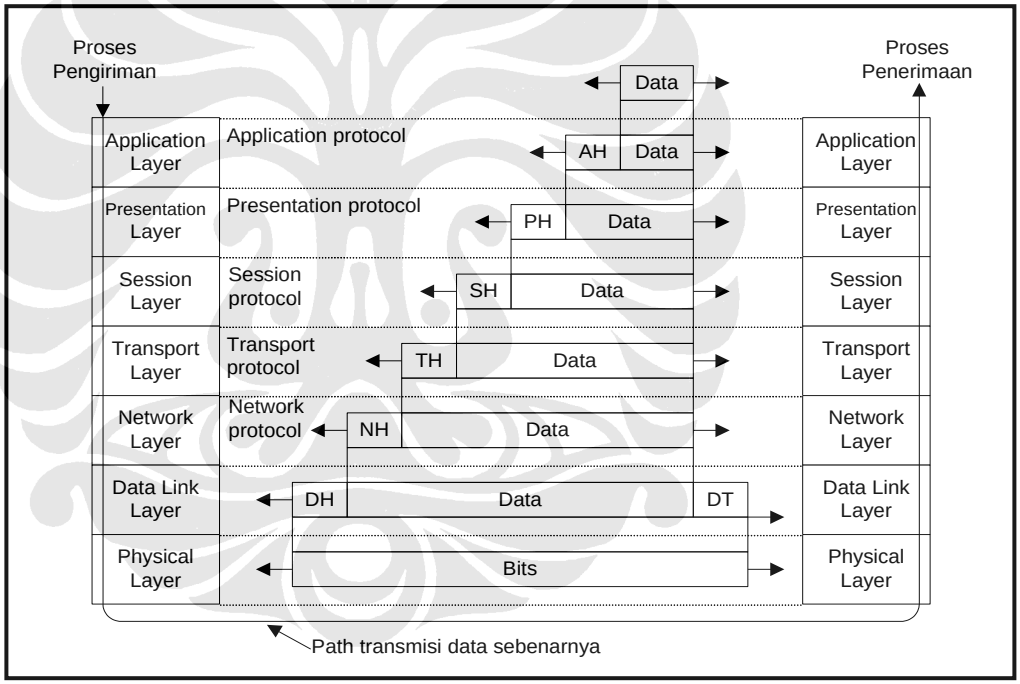
Gambar 2.2 Pembagian lapisan pada model OSI

Untuk menjalankan setiap komunikasi antar lapisan diperlukan protokol. Di dalam konteks jaringan data, protokol adalah suatu aturan dan kesepakatan untuk menentukan bagaimana komputer dapat saling bertukar informasi melewati suatu media jaringan. Setiap protokol yang ada dapat mengimplementasikan salah satu atau lebih dari lapisan-lapisan OSI. Misalkan group protokol LAN, WAN, protokol jaringan dan protokol *routing*.

2.1.2.2 Transmisi Data Pada Model OSI

Gambar 2.3 menjelaskan sebuah contoh tentang bagaimana data dapat ditransmisikan dengan menggunakan model OSI. Proses pengiriman memiliki data yang akan dikirimkan ke proses penerima. Setiap pemrosesan data pada setiap layer akan ditambah *header* untuk menunjukkan pengolahan dari setiap layer dan memiliki fungsi sesuai fungsi utama lapisan-lapisan tersebut.

Proses pemberian *header* ini berulang terus sampai data tersebut mencapai *physical layer*, dimana data akan ditransmisikan ke mesin lainnya. Pada mesin tersebut, semua *header* tadi dicopoti satu per satu sampai mencapai proses penerimaan.



Gambar 2.3 Gambaran transmisi data pada model OSI [1]

Walaupun bentuk pengolahan dan transmisi data secara vertikal dalam model OSI tetapi komunikasi berlangsung seperti secara horizontal. Setiap lapisan menambahkan *header* komunikasi dan akan dikirim ke lapisan yang sama. Misalkan dari *transport layer* ke *transport layer* lainnya.

2.1.3 Istilah Keamanan Jaringan

Dengan melihat bagaimana data dikirimkan, maka saat transmisi data ada kemungkinan bahwa informasi tidak sampai tujuan atau disadap oleh pengguna yang tidak berhak. Banyak gangguan yang dapat terjadi pada saat informasi dikirimkan baik itu disengaja ataupun tidak.

Keamanan jaringan didefinisikan sebagai sebuah perlindungan dari sumber daya daya terhadap upaya penyingkapan, modifikasi, utilisasi, pelarangan dan perusakan oleh pihak yang tidak diijinkan[1]. Dan dalam masalah keamanan suatu jaringan terdapat beberapa isu yang ditekankan untuk suatu sistem keamanan pada jaringan komputer. Beberapa isu tersebut adalah *confidentility*, *access control*, *data Integrity* dan *availability*.

Confidentility, yaitu bagaimanan agar pada jaringan komputer adanya jaminan pengriman data sampai pada tujuan yang sebenarnya tanpa adanya gangguan misalkan dari penyusup. Penyusup merupakan pengguna atau peralatan yang sebenarnya tidak memiliki hak akses data, informasi dan *resource* dari jaringan komputer.

Access control yaitu adanya perlindungan dan pembatasan akan akses ke jaringan komputer. Setiap pengguna seharusnya memiliki batasan masing-masing dalam mendapatkan dan mengolah sistem jaringan dan informasi.

Data integrity, yaitu adanya mekanisme untuk menjamin tidak adanya perusakan atau perubahan trafik saat informasi dikirimkan hingga diterima oleh sistem tujuan. Ada kalanya oleh penyusup data dirubah untuk kepentingannya seperti merubah *header* informasi sehingga tujuan dari komunikasi sebenarnya terganggu dan dialihkan atau merubah isi pesan sehingga informasi yang dikirimkan ke tujuan adalah palsu.

Availability, yaitu memastikan perangkat jaringan atau pengguna dapat mengakses jaringan komputer dan informasi kapan saja dibutuhkan. Setiap pengguna

menginginkan agar *resource* komputer dapat diakses kapan saja selama dibutuhkan. Terdapat banyak mekanisme serangan terkait dengan isu ini, seperti serangan DOS (*Denial Of Service*) yang umumnya membanjiri jaringan dengan informasi yang berbahaya sehingga informasi sebenarnya yang dikirimkan terganggu.

2.2 KUALITAS PELAYANAN JARINGAN

Kualitas layanan atau *Quality of Service* (QOS) adalah suatu mekanisme dalam jaringan yang memastikan suatu aplikasi atau layanan dapat beroperasi sesuai dengan yang diinginkan. Kualitas layanan berkaitan dengan segala hal yang dapat memastikan bahwa *response time* (waktu tanggap) dan *bandwidth* pada jaringan dapat diprediksi dan cocok untuk kebutuhan aplikasi atau layanan yang menggunakan jaringan itu.

2.2.1 *Response Time* (Waktu Tanggap)

Response time atau waktu tanggap merupakan waktu yang dibutuhkan suatu sistem untuk merespon dari masukan yang diberikan[4]. *Response time* juga merupakan waktu yang dibutuhkan suatu sistem untuk menyelesaikan suatu tugas yang diberikan berdasarkan masukan yang diberikan.

2.2.2 *Bandwidth*

Bandwidth adalah kapasitas sebuah sinyal atau teknologi tertentu untuk membawa informasi[5]. Pada jaringan komputer, *bandwidth* digunakan untuk menunjukan kapasitas suatu saluran komunikasi untuk membawa sinyal informasi. Semakin besar *bandwidth* yang tersedia, maka akan semakin besar data yang dapat ditransmisikan melalui jaringan tersebut dalam selang waktu tertentu. *Throughput* merupakan istilah yang digunakan untuk menyatakan tingkat aliran informasi dalam satuan *bit per second* (bps)[5]. Dibeberapa literatur populer, istilah *bandwidth*

seringkali disamakan dengan istilah *throughput*. *Throughput* ini dapat dihitung dengan Hukum Shannon seperti pada persamaan 2.1 ini.

Throughput (*bps*) = *Bandwidth* (Hz) x $\log_2$ [1+R] (2.1)

dengan R = *Signal Power* (*watt*) / *Noise Power* (*watt*). Melalui persamaan di atas dapat disimpulkan bahwa semakin besar *noise* yang timbul, kapasitas saluran untuk membawa informasi berkurang.

2.3 Teori Kriptografi

2.3.1 Kriptografi, Enkripsi dan Dekripsi

Kriptografi berasal dari dua suku kata yaitu kripto dan grafi. Kripto artinya menyembunyikan, sedangkan grafi artinya ilmu. Kriptografi (*cryptography*) adalah suatu ilmu yang mempelajari sistem penyandian untuk menjamin kerahasiaan dan keamanan data. Dan yang melakukannya disebut seorang kriptographer.

Enkripsi adalah suatu proses yang melakukan perubahan kode dari bisa dimengerti (yang disebut *plaintext*) menjadi kode yang tidak bisa dimengerti (yang disebut *ciphertext*). Sedangkan proses kebalikannya yaitu memperoleh kembali *plaintext* dari *ciphertext* disebut dekripsi[3]. Secara umum operasi enkripsi dan dekripsi secara matematis dapat digambarkan sebagai berikut :

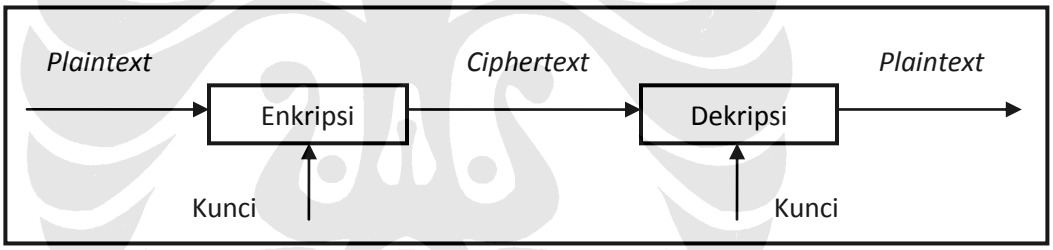
EK (M) = C (Proses Enkripsi)

DK (C) = M (Proses Dekripsi)

M merupakan pesan yang akan dirahasiakan atau disebut juga *plaintext*. E merupakan mekanisme enkripsi dan K adalah simbol kunci yang digunakan. Pembacaan rumus tersebut ialah proses enkripsi E dengan kunci K akan dilakukan pengolahan pesan M menjadi pesan C, dan C adalah *ciphertext* atau pesan hasil

enkripsi yang diharapkan berbeda dan tidak mampu menyingkap informasi aslinya dari pesan M.

Proses kebalikannya yaitu proses dekripsi. Pada proses dekripsi dengan kunci K, pesan C akan disandikan kembali menjadi pesan semula yaitu M sehingga diperoleh kembali pesan aslinya. Yang menjadi penting pada proses enkripsi-dekripsi disini ialah tidak hanya algoritma yang digunakan tetapi juga kunci enkripsi. Kunci merupakan kata kunci yang digunakan untuk proses penyandian. Setiap kunci yang berbeda dapat menghasilkan C dan M yang berbeda walau proses enkripsinya sama. Untuk itu walaupun algoritma enkripsi dan dekripsi diketahui misalkan oleh publik, kunci K harus dirahasiakan agar proses enkripsi dan dekripsi aman. **Gambar 2.4** memberikan contoh aliran proses enkripsi dan dekripsi.



Gambar 2.4 Algoritma Enkripsi dan Dekripsi sederhana

2.3.2 Teknik Enkripsi dan Dekripsi

Berikut ini adalah beberapa teknik enkripsi dasar yang digunakan untuk mengacak data.

2.3.2.1. Teknik Subtitusi

Pada teknik substitusi akan diganti suatu kode dengan kode yang lain, dengan cara melihat suatu tabel substitusi[3]. Pada gambar 2.5 diberikan contoh tabel substitusi.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	F	"	&	$	A	Q	1	@	W	G	Y	N	~	5	%	^	7	*	(	M	<	4	2	+	X

Gambar 2.5 Teknik Subtitusi

Tabel subtitusi diatas dibuat secara acak, misalkan huruf A akan diganti dengan huruf C, dan Huruf B akan diganti dengan huruf F dan seterusnya. Misalkan kata "ENKRIPSI" akan dienkripsi dengan teknik substitusi dengan tabel tersebut menjadi "$~G7@%*@". Untuk memperolah kata "ENKRIPSI" kembali maka dilakukan pula dengan melihat tabel substitusi tersebut.

2.3.2.2. Teknik *Blocking*

Teknik *blocking* yaitu membagi pesan atau *plaintext* kedalam blok-blok pesan yang kemudian akan dienkripsi secara independen[3]. Misalkan kalimat "TEKNIK BLOKING" akan dibagi kedalam blok-blok sebagai berikut :

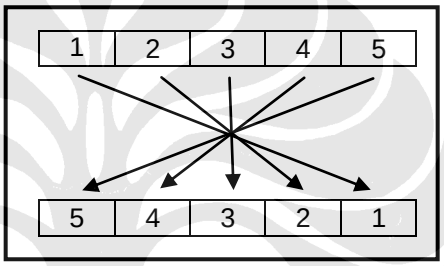
T	K	C
E		K
K	B	I
N	L	N
I	O	G

Gambar 2.6 Enkripsi dan Dekripsi dengan Teknik *Blocking*

Selanjutnya pesan dapat dienkripsi per blok, misalkan untuk proses pengacakan sederhana dengan pembacaan yang terbalik yaitu karena proses penyimpanan pada blok dilakukan per kolom, maka proses enkripsi akan dilakukan dengan cara pembacaan per baris dan menjadi "TILNEKOGK K NBI". Proses dekripsi akan dilakukan serupa dengan cara menaruh kalimat "TILNEKOGK K NBI" pada blok-blok pesan perbaris dan membaca dengan perkolom dan menjadi kalimat "TEKNIK BLOKING" kembali. Agar rumit dapat menggunakan kalkulasi tertentu untuk mengacak susunan blok seperti pada algoritma enkripsi RC4.

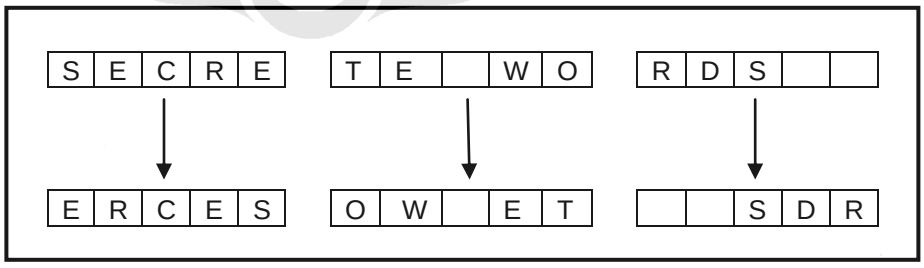
2.3.2.3. Teknik Permutasi

Teknik permutasi sering juga disebut transposisi. Teknik ini memindahkan atau merotasikan karakter dengan aturan tertentu. Prinsipnya adalah berlawanan dengan teknik substitusi. Dalam teknik substitusi, karakter berada pada posisi yang tetap tapi identitasnya yang diacak. Pada teknik permutasi, identitas karakternya tetap, namun posisinya yang diacak[3]. Sebelum dilakukan permutasi, umumnya *plaintext* terlebih dahulu dibagi menjadi blok-blok dengan panjang yang sama. Untuk contoh diatas, *plaintext* akan dibagi menjadi blok-blok yang terdiri dari 5 karakter, dengan aturan permutasi sebagai berikut :



Gambar 2.7 Contoh Teknik Permutasi

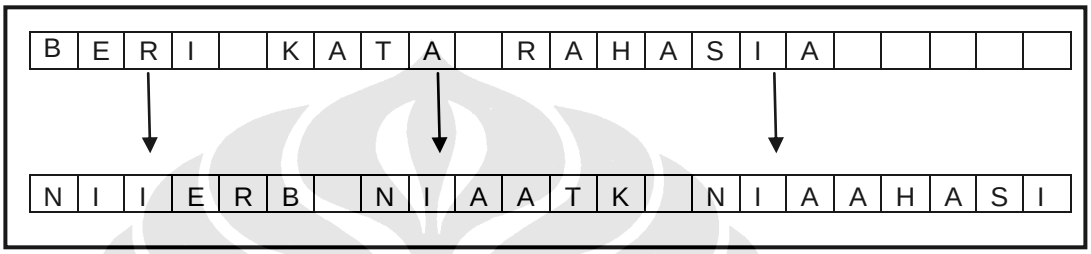
Misalkan kalimat "*SECRET WORDS*" dienkripsi menjadi "ERCESOW ET SDR" dengan cara membaginya kalimat kedalam 5 blok - 5 blok dan melakukan proses permutasi seperti pada **Gambar 2.8** dengan teknik pengacakan seperti pada **Gambar 2.7**. Proses dekripsi dilakukan hal yang serupa dengan membagi kedalam 5 blok – 5 blok pula dan membalik teknik permutasinya.



Gambar 2.8 Contoh Enkripsi dengan Permutasi

2.3.2.4. Teknik Ekspansi

Teknik ekspansi akan menambahkan beberapa *byte* kata kedalam *plaintext* dengan aturan tertentu. Proses penambahan beberapa *byte* kata ini diharapkan dapat menyembunyikan informasi dari *plaintext*[3]. Salah satu contoh penggunaan teknik ini adalah dengan menukar huruf awal dan akhir kata dan diberi kata diberi awalan "ni". Proses enkripsi dengan cara ekspansi terhadap *plaintext* terjadi sebagai berikut :

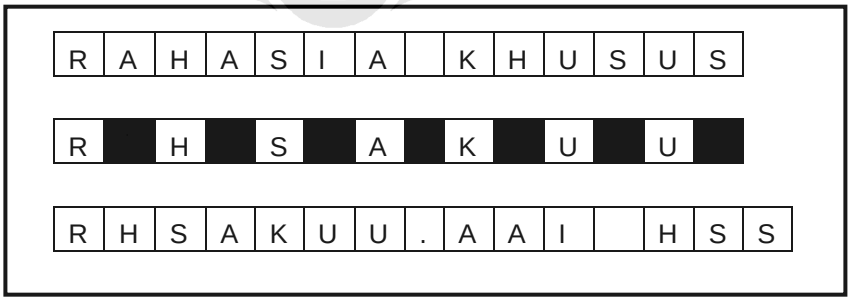


Gambar 2.9 Contoh Enkripsi dengan Ekspansi

Ciphertextnya adalah "IERBNI AATKIN AAHASIKN". Aturan ekspansi dapat dibuat lebih kompleks dan terkadang teknik ekspansi digabungkan dengan teknik lainnya.

2.3.2.5. Teknik Pemampatan (*Compaction*)

Mengurangi panjang pesan atau jumlah bloknya adalah cara lain untuk menyembunyikan isi pesan[3]. Misalkan untuk plaintext "RAHASIA KHUSUS", setiap kata ke dua akan dihilangkan dan disertakan pada akhir kalimat yang sebelumnya diberi tanda ".". Proses yang terjadi untuk *plaintext* tersebut adalah :



Gambar 2.10 Enkripsi dengan Pemampatan[3]

Aturan penghilangan karakter dan karakter khusus yang berfungsi sebagai pemisah menjadi dasar untuk proses dekripsi *ciphertext* menjadi *plaintext* kembali.

Dengan menggunakan kelima teknik dasar kriptografi diatas, dapat diciptakan teknik kriptografi yang amat banyak. Walaupun sekilas terlihat sederhana, kombinasi teknik dasar kriptografi dapat menghasilkan teknik kriptografi turunan yang cukup kompleks, dan beberapa teknik dasar kriptografi masih digunakan dalam kriptografi modern.

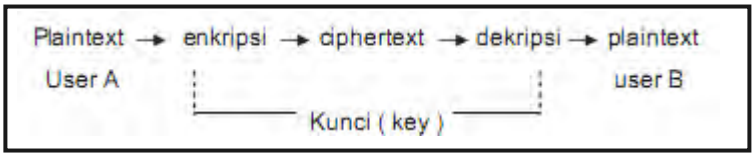
2.3.3 Kriptosistem

Kriptosistem (*Crytosystem* atau *cryptographic system*) adalah suatu fasilitas untuk mengkonversikan *plaintext* ke *ciphertext* dan sebaliknya. Dalam sistem ini, seperangkat parameter yang menentukan transformasi pen*cipher*an tertentu disebut suatu set kunci. Proses enkripsi dan dekripsi diatur oleh satu atau beberapa kunci kriptografi.

Suatu cryptosystem terdiri dari sebuah algoritma, seluruh kemungkinan *plaintext*, *ciphertext* dan kunci-kunci. Secara umum *cryptosystem* dapat digolongkan menjadi dua buah, yaitu :

2.3.3.1. *Symetric Cryptosystem*

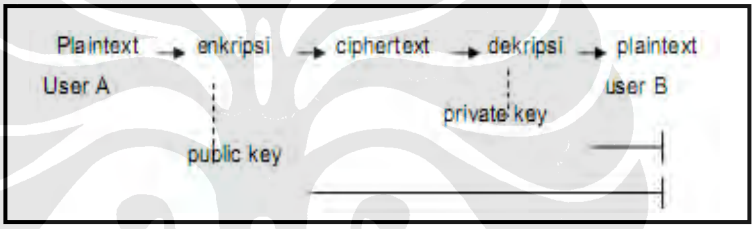
Dalam symmetric cryptosystem ini, kunci yang digunakan untuk proses enkripsi dan dekripsi pada prinsipnya sama tetapi satu buah kunci dapat pula diturunkan dari kunci yang lainnya, kunci – kunci ini harus dirahasiakan. Oleh sebab itu sistem ini sering disebut sebagai *secret key cipher system*. Contoh dari sistem ini adalah *Data Encryption Standard* (DES), *Blowfish*, IDEA.



Gambar 2.11 Symetric Cryptosystem

2.3.3.2. *Assymetric cryptosystem*

Dalam *assymetrick* kriptosistem ini digunakan dua buah kunci yang berbeda dalam proses enkripsi dan dekripsinya. Satu kunci yang disebut kunci publik (*public key*) yang dapat dipublikasikan, Sedangkan kunci yang lain yang disebut kunci privat (*private key*) yang harus dirahasiakan. Misalkan Bila A ingin mengirimkan pesan kepada B, A dapat menyandikan pesannya menggunakan kunci publik B, dan bila B ingin membaca pesan tersebut, ia perlu mendekripsikannya dengan kunci privatnya. Dengan demikian kedua belah pihak dapat menjamin asal pesan serta keaslian pesan tersebut, karena adanya mekanisme ini. Contoh dari sistem ini antara lain RSA *Scheme* dan Merkie Hellman Scheme.



Gambar 2.12 Assymmetrick Cryptosystem [3]

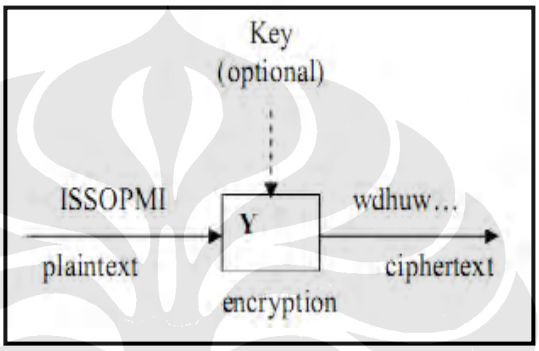
2.3.4 *Cipher* dalam Enkripsi dan Dekripsi

Secara umum dalam proses enkripsi dan dekripsi dikenal dua macam cipher berdasarkan cara kerja penyandiannya, yaitu :

2.3.4.1. *Stream Cipher*

Stream cipher adalah suatu sistem dimana proses enkripsi dan dekripsinya dilakukan dengan cara per bit atau *byte*. Pada sistem ini *key stream* (aliran bit kuncinya) dihasilkan oleh suatu pembangkit bit acak. *Key stream* ini dikenakan operasi XOR dengan aliran bit – bit dari *plaintext* untuk menghasilkan aliran bit-bit *ciphertext*. Pada proses dekripsi aliran bit *ciphertext* dikenakan operasi XOR dengan *key stream* yang identik untuk menghasilkan *plaintext*.

Keamanan dari sitem ini tergantung dari pembangkit kunci, jika pembangkit kunci menghasilkan aliran bit-bit 0 maka *ciphertext* yang dihasilkan akan sama dengan *plaintext*, sehingga seluruh operasi akan menjadi tidak berguna oleh karena itu diperlukan sebuah pembangkit kunci yang dapat menghasilkan aliran bit-bit kunci yang acak dan tidak berulang. Semakin acak aliran kunci yang dihasilkan oleh pembangkit kunci, maka *ciphertext* akan semakin sulit dipecahkan.

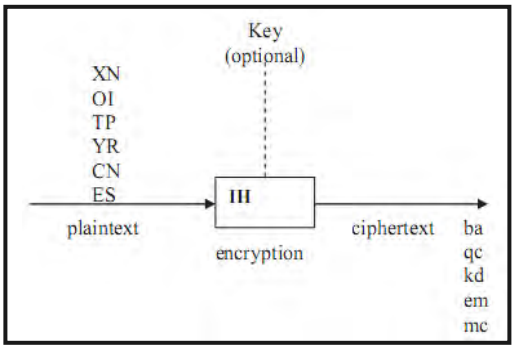


Gambar 2.13 Sistem *Stream* Cipher [3]

2.3.4.2. *Block Cipher*

Sistem *block cipher* mengkodekan data dengan cara membagi *plaintext* menjadi perblok dengan ukuran yang sama dan tetap. Kemudian setiap bloknya dienkripsi atau didekripsi sekaligus. Cara ini bekerja lebih cepat karena *plaintext* dibagi atas beberapa blok. Biasanya proses enkripsi dan dekripsi dilakukan dalam ukuran blok tertentu.

Transposisi merupakan contoh dari penggunaan block *cipher*. Pada transposisi kolumnar dengan menggunakan matriks, pesan diterjemahkan sebagai satu blok. Ukuran blok yang dibutuhkan tidak memiliki kesamaan dengan ukuran sebuah karakter. Block *cipher* bekerja pada blok *plaintext* dan menghasilkan blok-blok *ciphertext*.



Gambar 2.14 Sistem *Block Cipher* [3]

2.4 RC4

2.4.1 RC4 *Stream Cipher*

RC4 merupakan salah satu jenis *stream cipher* yang didesain oleh Ron Rivest di laboratorium RSA (*RSA Data Security inc*) pada tahun 1987. RC4 sendiri merupakan kepanjangan dari Ron *Code* atau *Rivest's Cipher*. RC4 *stream cipher* ini merupakan teknik enkripsi yang dapat dijalankan dengan panjang kunci yang variabel dan beroperasi dengan orientasi *byte*.

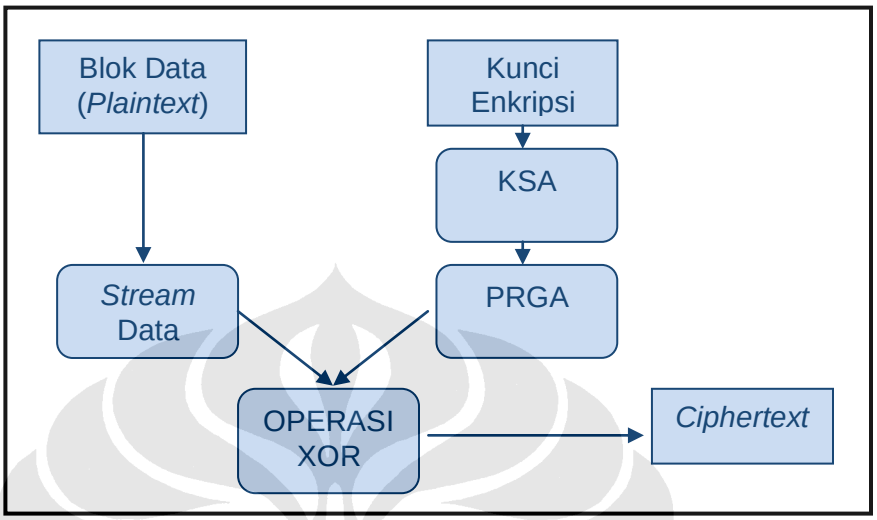
Algoritma yang sesungguhnya tidak dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas sampai sekarang. Namun pada bulan September 1994 ada seseorang yang telah mengirimkan sebuah *source code* yang diyakini sebagai RC4 ke *mailinglist Cyperpunks* dan keberadaanya pun langsung tersebar. Karena algorima yang dipublikasikan ini sangat identik dengan implementasi RC4 pada produk resmi.

2.4.2 Algoritma RC4 *Stream Cipher*

2.4.2.1 Algoritma Enkripsi RC4 *Stream Cipher*

RC4 memiliki sebuah S-box, S0,S1, ...S255 yang berisi permutasi dari bilangan 0 sampai 255. Dalam algoritma enkripsi metode ini akan membangkitkan pseudorandom *byte* dari *key* yang akan dikenakan operasi XOR terhadap *plaintext*

untuk menghasilkan *ciphertext*. Berikut ini akan diberikan diagram proses dari proses enkripsi data :



Gambar 2.15 Rangkaian Proses Enkripsi RC4 *Stream Cipher*

Secara garis besar algoritma dari metode RC4 *Stream Cipher* ini terbagi menjadi dua bagian, yaitu : *key setup* atau *Key Scheduling Algorithm* (KSA) dan *stream generation* atau Pseudo Random Generation Algorithm (PRGA) dan proses XOR dengan *stream* data. Berikut ini akan dijelaskan bagian-bagian dari algoritma RC4 *Stream Cipher* tersebut.

1. *Key Setup* / *Key Scheduling Algorithm* (KSA)

Pada bagian ini, terdapat tiga tahapan proses didalamnya, yaitu:

1 1. Inisialisasi S-Box

Pada tahapan ini, S-Box akan diisi dengan nilai sesuai indeksnya untuk mendapatkan S-Box awal. Algoritmanya adalah sebagai berikut:

a. untuk i=0 hingga i=255 lakukan

b. isikan S dengan nilai i

c. Tambahkan i dengan 1, kembali ke 2

Dari algorima di atas akan didapat urutan nilai S-Box sebagai berikut:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	240	241	242	243	245	246	247	248	249	250	251	252	253	254	255

Gambar 2.16 Contoh Hasil dari inisialisasi Sbox dengan memasukkan data urut pada setiap blok[3]

1. 2. Menyimpan kunci dalam *Key Byte Array*

Pada tahapan ini, kunci (*key*) yang akan kita gunakan untuk mengenkripsi atau dekripsi akan dimasukkan ke dalam array berukuran 256 secara berulang sampai seluruh array terisi. Algoritmanya adalah sebagai berikut:

a. isi j dengan 1

b. untuk i=0 hingga i=255 lakukan

c. jika j > panjang kunci maka

d. j diisi dengan nilai 1

e. akhir jika

f. isi K ke i dengan nilai ASCII karakter kunci ke j

g. nilai j dinaikkan 1

h. tambahkan i dengan 1, kembali ke 2

Dari algoritma tersebut akan didapat urutan array *key* misalkan sebagai berikut untuk kunci dengan panjang 8 Karakter dengan urutan karakter dalam ASCII " 109 97 104 98 98 97 104 " :

109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104
109	97	104	97	98	98	97	104	109	97	104	97	98	98	97	104

Gambar 2.17 Contoh Hasil penyimpanan kunci pada *key byte array* [3]

1.3. Permutasi pada S-Box

Pada tahapan ini, akan dibangkitkan sebuah nilai yang akan dijadikan aturan untuk permutasi pada S-Box. Algoritmanya adalah sebagai berikut :

a. isi nilai j dengan 0

b. untuk i=0 hingga i=255 lakukan

c. isi nilai j dengan hasil operasi (j+S(i)+K(i)) mod 256

d. Tukar nilai S(i) dan S(j)

e. Tambahkan i dengan 1, kembali ke 2

Dari algoritma tersebut akan diperoleh nilai S-Box yang telah mengalami proses transposisi sehingga urutannya diacak misalkan untuk kunci pada contoh 1.2 sebagai berikut:

109	26	57	76	157	245	52	49	181	31	45	103	84	240	73	192
61	235	28	98	3	205	160	128	130	59	22	38	92	122	137	30
247	121	4	135	156	148	231	170	228	16	68	208	159	53	210	20
176	168	220	214	136	19	87	42	120	219	27	169	183	116	200	96
204	110	64	216	89	222	44	126	105	213	164	188	23	47	194	33
209	65	162	142	107	195	119	80	166	238	237	72	246	102	243	241
35	40	77	215	141	100	74	115	146	56	189	163	113	55	67	229
236	185	83	46	18	140	118	50	233	248	99	8	172	203	54	132
13	124	152	252	151	153	147	139	179	190	82	154	224	161	86	0
34	149	29	225	78	14	48	158	25	104	150	184	218	187	95	193
79	43	155	223	143	232	177	251	66	114	202	10	9	88	6	226
239	250	178	71	106	51	11	60	125	242	17	90	197	91	249	173
94	15	201	101	108	234	227	41	75	117	165	174	230	138	207	62
70	244	129	145	182	131	112	7	171	253	111	198	134	180	167	211
39	255	2	1	32	85	254	21	63	217	212	175	12	93	123	191
36	186	37	133	144	24	69	81	206	5	196	199	97	221	58	127

Gambar 2.18 Contoh Hasil pengacakan Sbox berdasarkan *key* yang digunakan. [3]

2. *Stream* Generation

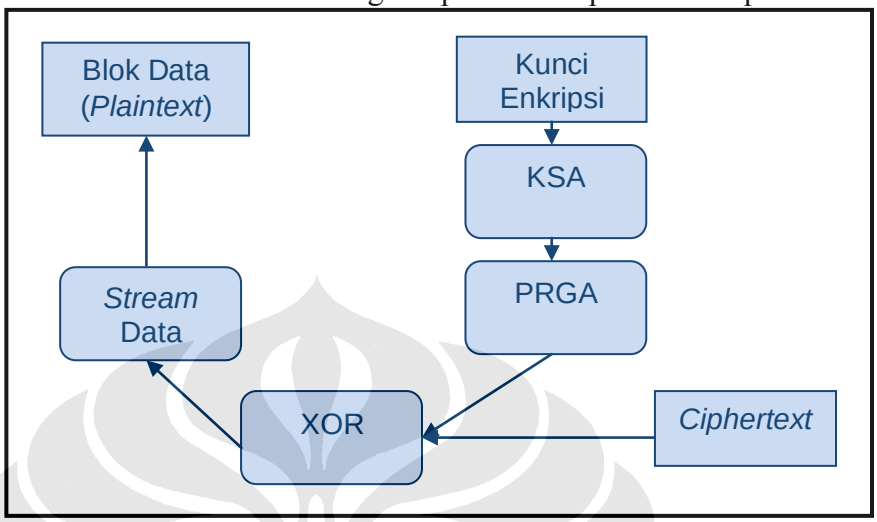
Pada tahapan ini akan dihasilkan nilai *pseudorandom* yang akan dikenakan operasi XOR untuk menghasilkan *ciphertext* ataupun sebaliknya yaitu untuk menghasilkan *plaintext*. Algoritmanya adalah sebagai berikut:

1. isi indeks i dan j dengan nilai 0
2. untuk i=0 hingga i=panjang *plaintext*
3. isi nilai i dengan hasil operasi (i+1) mod 256
4. isi nilai j dengan hasil operasi (j+S(i)) mod 256
5. Tukar nilai S(i) dan S(j)
6. Isi nilai t dengan hasil operasi (S(i)+(S(j) mod 256))mod 256
7. isi nilai y dengan nilai S(t)
8. nilai y dikenakan operasi XOR terhadap *plaintext*
9. Tambahkan i dengan 1, kembali ke 2.

Dengan demikian akan dihasilkan misalkan *ciphertext* dengan hasil XOR antar *stream key* dari Sbox dan *plaintext* secara berurutan.

2.4.2.2 Algoritma dekripsi RC4 *Stream Cipher*

Berikut ini akan diberikan diagram proses dari proses dekripsi data :



Gambar 2.19 Rangkaian Proses dekripsi RC4 *Stream* Cipher

Algoritma dekripsi RC4 *Stream Cipher* mirip dengan algoritma enkripsinya, perbedaannya hanyalah pada saat *stream generation*, yaitu untuk menghasilkan *plaintext* semula, maka *ciphertext* nya yang akan dikenakan operasi XOR terhadap *pseudorandom byte*nya. Algoritma *key setup* pada proses dekripsi sama dengan algoritma enkripsinya yang dari proses inisialisai S-Box, penyimpanan kunci kedalam *key byte array* hingga proses permutasi S-Box berdasarkan *key byte array* nya. Untuk itu proses dekripsi dan enkripsi akan menghasilkan *key stream* yang sama. Perbedaannya hanya pada *stream generationnya*, yaitu yang dioperasikan bersama *key stream* adalah *ciphertext* untuk menghasilkan kembali *plaintext*.

Algoritmanya adalah sebagai berikut:

1. isi indeks i dan j dengan nilai 0
2. untuk i=0 hingga i=panjang *ciphertext* (panjang *ciphertext=plaintext*)
3. isi nilai i dengan hasil operasi (i+1) mod 256
4. isi nilai j dengan hasil operasi (j+S(i)) mod 256
5. Tukar nilai S(i) dan S(j)

6. Isi nilai t dengan hasil operasi (S(i)+(S(j) mod 256))mod 256

7. isi nilai y dengan nilai S(t)

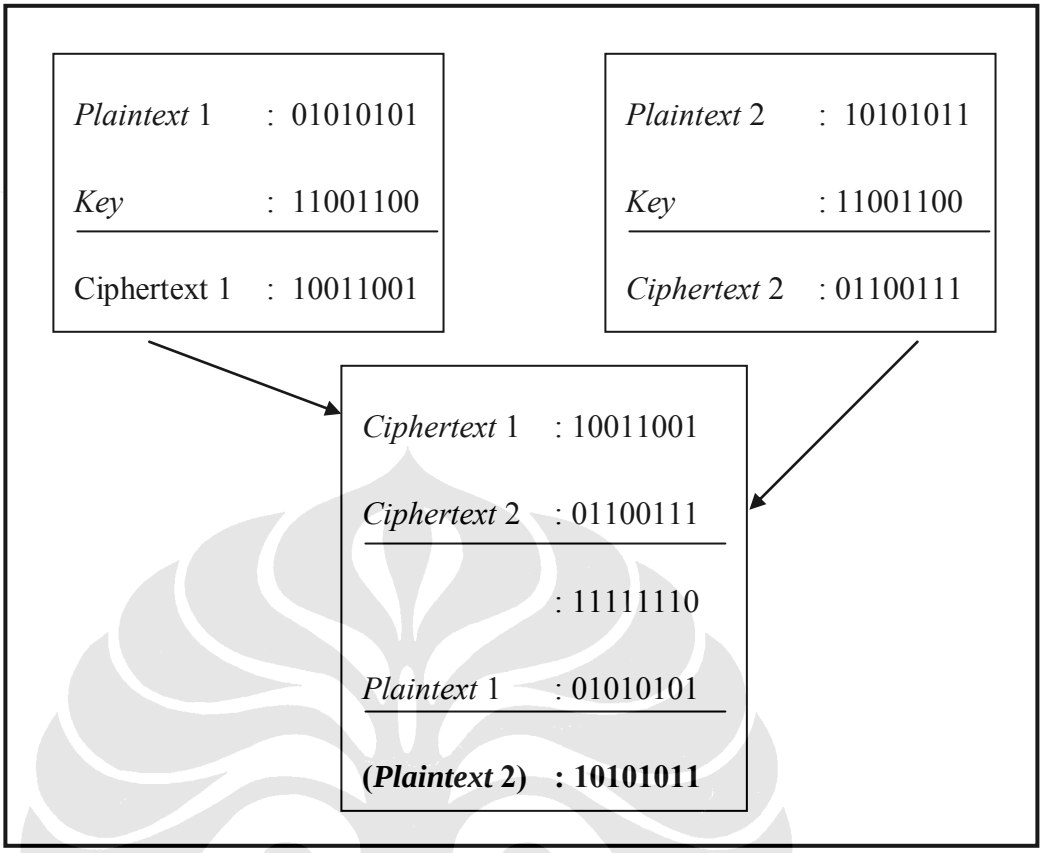
8. nilai y dikenakan operasi XOR terhadap *ciphertext*

9. Tambahkan i dengan 1, kembali ke 2.

2.4.2.3 Permasalahan Pada Algoritma RC4 *Stream* Cipher

Permasalah yang dapat terjadi pada penerapan algoritma RC4 *stream* cipher salah satunya ialah sebagai berikut :

Enkripsi RC4 adalah operasi XOR antara data *byte*s dan *pseudorandom byte stream* yang dihasilkan dari kunci. Penyerang akan mungkin untuk menentukan beberapa *byte* pesan asli dengan meng-XOR dua set cipher *byte*, bila beberapa *plaintext* diketahui (atau mudah ditebak). Misalkan diasumsikan A berhasil menyadap dua buah *ciphertext* berbeda menggunakan kunci yang sama. A kemudian meng-XOR-kan kedua *ciphertext* yang berhasil disadapnya. Jika A berhasil mengetahui *plaintext* dari salah satu pesan terenkripsi tersebut maka A akan dengan mudah menemukan *plaintext* yang lain tanpa mengetahui rangkaian kuncinya. Misalkan dengan contoh mekanisme pada **Gambar 2.20.**



Gambar 2.20 Contoh Hasil Operasi XOR (1 *byte* data) sehingga mendapatkan data rahasia (*plaintext* 2)

Selain itu dengan kunci yang sama akan dihasilkan *stream key* yang sama, hal ini akan mempermudah analisan *cipher*, dimana jika telah diperoleh beberapa *ciphertext* dan *plaintext*nya maka dengan mudah akan diperoleh *byte stream key* dari algoritma ini, dengan perumusan XOR sebagai berikut :

K1 = P1 XOR C1

K2 = P2 XOR C2

K3 = P3 XOR C3

Dan seterusnya

Hingga

$\mathbf{K_{256}}$ **= P$_{256}$ XOR C$_{256}$** (jika *plaintext* digunakan hingga 256 karakter (maksimum))

P adalah *plaintext* dan C adalah *ciphertext*, dan K adalah *key stream* dan nomor tersebut dari 1 hingga 256 adalah urutan *stream*nya (256 sesuai dengan maksimum jumlah box pada Sbox). Maka jika Suatu P dan C diketahui beserta urutan pemprosesannya (dari 1 hingga 256) maka *key stream* (dengan urutan tersebut) dapat segera ditemukan. Bahkan dengan mudahnya *plaintext* yang lain akan diperoleh dengan operasi ini. Dan akan sangat berbahaya jika *plaintext* tersebut ialah data sensitif seperti nomor kartu kredit atau lain sebagainya.

Salah satu pemecahan masalah algoritma RC4 ini ialah dengan cara mengganti *key stream* setiap kali proses enkripsi dilakukan. Dan cara yang cukup populer ialah dengan menambahkan beberapa *byte* tambahan dari kunci yang sebenarnya. *Byte* tambahan ini biasanya disebut *Initializatin Vector* (IV) yang digunakan sebagai nilai awal kunci dimana kunci yang akan dimasukkan pada "*Key Byte Array"* adalah IV + Kunci. Setiap kali enkripsi dilakukan nilai dari IV ini berubah-ubah sehingga otomatis akan menghasilkan *key stream* yang berbeda-beda. Cara ini cukup populer dan saat ini digunakan pada protokol enkripsi dan otentifiksi WEP dan WPA yang digunakan sebagai keamanan enkripsi wireless LAN. Akan tetapi WEP tidak digunakan kembali karena ditemukan korelasi IV dengan kunci yang digunakan. Sehingga walaupun nilai IV berubah-ubah, terdapat metode statistik untuk menemukan berapa persen hubungan IV yang digunakan dengan kemungkinan kunci yang digunakan, dan dengan bantuan metode serangan *bruteforce attack*, kunci dapat segera ditemukan.

Walaupun demikian pada protocol enkripsi dan otentifikasi WPA kelemahan IV tersebut diperbaiki dengan suatu metode baru yang disebut *Tempororal Key Integrity Protocol* (TKIP). Salah satu metode TKIP ialah dengan proses *mixing* dari IV dan kunci untuk memutuskan korelasi antara IV dan kunci yang digunakan dan membuat *key stream* yang unik setiap waktu. Saat ini metoda ini cukup aman dengan syarat yaitu menggunakan kunci yang tidak mudah ditebak seperti kunci yang tidak terdapat pada kamus atau kata yang mudah ditebak lainnya. Akan tetapi untuk aplikasi yang dikembangkan pada skripsi ini tidak akan digunakan metode IV karena

memiliki beberapa kelemahan dan tidak cocok untuk diterapkan pada metode pengolahan *database*. Kelemahannya ialah:

a. Nilai IV akan digunakan sebagai nilai publik dimana akan ditransmisikan bersama dengan *ciphertext*. IV akan diekstrak pada akhir transmisi dan dioperasikan dengan kunci yang digunakan untuk menghasilkan *key stream* yang digunakan juga pada proses enkripsi. Kelemahan untuk operasi pengolah *database* ialah, nilai IV ini harus disimpan pada *database* sehingga akan memperbesar memori dari *database*.

b. Kelemahan yang kedua yaitu nilai IV disimpan pada *database* dan *key stream* yang digunakan akan berbeda-beda pula berdasarkan nilai IV nya. Sehingga untuk melakukan proses pencarian IV harus diekstrak diprogram aplikasi (*client*) dan proses pencarian dilakukan pada program aplikasi. Hal ini akan memperlambat proses pencarian. Selain itu dengan kunci yang acak sebenarnya sulit untuk menentukan kunci sebenarnya yang digunakan pada proses enkripsi kecuali adanya metode IV (atau kode tertentu untuk menghasilkan *key stream* yang sama untuk proses dekripsi).

c. Kelemahan yang ketiga yaitu dengan penambahan nilai IV publik pada *ciphertext* maka hasil *ciphertext* akan lebih besar dari *plaintext* yang seharusnya ditransmisikan, sehingga terdapat kemungkinan bahwa akan memperlambat proses transmisi pada jaringan.

Dengan beberapa kelemahan ini maka metode IV tidak digunakan pada aplikasi yang dikembangkan pada skripsi ini, tetapi dengan metode pengacakan *plaintext* dengan teknik *dynamic blocking* untuk mengacak *plaintext*.

2.5 Aplikasi Pendukung

2.5.1 *Visual Basic* 6

Visual Basic merupakan lingkungan pem-programan dari Microsoft dimana *programmer* dapat menggunakan GUI (*Graphical User Interface*) untuk memilih dan memodifikasi kode program yang ditulis dalam bahasa pemrograman BASIC. *Visual basic* 6 dikembangkan pada pertengahan tahun 1998 dan banyak dilakukan perbaikan dibandingkan versi sebelumnya salah satunya ialah kemampuan untuk membentuk aplikasi berbasis web.

2.5.2 *Microsoft Access*

Access adalah perangkat lunak *database* yang disediakan oleh *microsoft* dan umumnya dipaket pada Microsoft Office. *Access* baik digunakan dari aplikasi kecil hingga aplikasi sedang. *Access* dapat berhubungan dengan *database* lain seperti *MySQL* dan *SQL server* dengan koneksi ODBC.

2.5.3 *Wireshark*

Wireshark merupakan perangkat lunak yang digunakan untuk menangkap trafik pada jaringan dan melakukan analisa trafik jaringan seperti *throughput*, *latency* dan lain sebagainya.

BAB III

Perancangan Sistem

3.1 KONSEP APLIKASI

Aplikasi dirancang menggunakan bahasa pemrograman *Visual Basic* (VB) dengan *database access*. Koneksi pada jaringan LAN dilakukan menggunakan koneksi ODBC. Dengan koneksi ini aplikasi *database* seperti *MySql* dan *SQL server* dapat terkoneksi pada aplikasi ini.

Aplikasi yang dibangun yaitu aplikasi pengolah *database* pada aplikasi perangkat lunak sistem kearsipan kegiatan penelitian skripsi. Aplikasi akan mengolah *database* dengan melakukan proses enkripsi data. Proses enkripsi menggunakan algoritma RC4 dengan pengembangan teknik *dynamic blocking* yang berubah berdasarkan panjang kunci yang digunakan.

3.1.1 Perancangan Sistem Enkripsi – Dekripsi dan Algoritma *Dynamic Blocking*

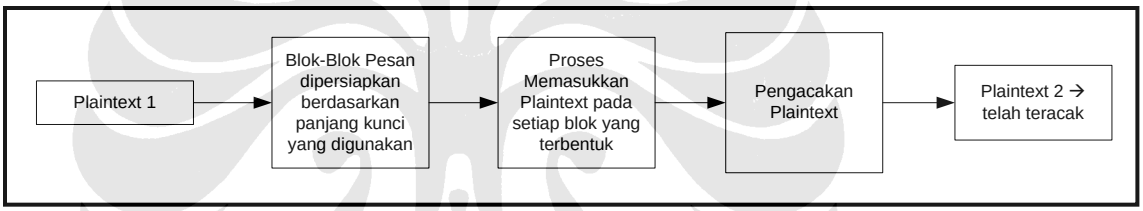
Perancangan proses enkripsi dan dekripsi menggunakan algoritma dasar RC4. Proses enkripsi dan dekripsi dilakukan hanya pada aplikasi di komputer *client* dan dilakukan pada hampir setiap *field* pada setiap *record.* Untuk perancangan *database* dibahas pada **sub bab 3.1.3**.

Modifikasi algoritma RC4 yaitu adanya proses pengacakan *plaintext* menggunakan metode *dynamic blocking* yaitu sebelum adanya operasi XOR antara *plaintext* dan *key stream*-nya (kunci aliran-nya). Teknik enkripsi *dynamic blocking* merupakan teknik enkripsi dengan teknik enkripsi dasar *blocking*, hanya saja blok-

blok pesan bersifat dinamik atau dapat berubah sesuai dengan kunci yang digunakan, atau lebih tepatnya panjang kunci yang digunakan pada skripsi ini. Dengan sistem ini jika panjang kunci dan pesan yang digunakan berubah maka pengacakan *plaintext* juga akan berubah.

Jika menggunakan teknik pengacakan *blocking* biasa yang bersifat statis (misalkan panjang baris selalu sama yaitu 5 atau 6) ialah pengguna dapat lebih mudah menebak susunan *plaintext* yang asli berdasarkan panjang *ciphertext*-nya jika algoritma enkripsi telah diketahui, tetapi jika susunan pengacakan dapat berbeda-beda bergantung dari kunci yang digunakan maka diharapkan pengacakan tidak diketahui selama kunci dan panjang yang digunakan tidak diketahui.

Berikut diagram proses pengacakan *plaintext* dan penjelasannya:



Gambar 3.1 Diagram proses pengacakan *plaintext* dengan teknik *dynamic blocking*

Plaintext dimasukkan pada blok data yang bersifat dinamik bergantung dari panjang kunci yang digunakan. Blok-blok data yang dinamik bergantung pada jumlah baris dari blok data tersebut, dan jumlah baris itulah yang bervariasi bergantung pada panjang kunci yang digunakan dengan rumus sebagai berikut:

$$J = (P - (P \bmod 5)) / 5 + 1 \qquad (3.1)$$

Jumlah baris pada blok-blok data = J

Panjang Kunci = P

Artinya setiap panjang kunci kelipatan 5 maka baris bertambah 1. Dan untuk panjang minimal kunci adalah 5 maka jumlah baris pada blok data sebanyak 2 baris. Yang lain misalkan panjang kunci yang digunakan adalah 25 karakter maka jumlah baris pada blok data yaitu 6 baris. Teknik ini digunakan agar *plaintext* diacak secara

acak menggunakan kunci yang bersifat dinamik. Selain jumlah baris, kolom juga bervariasi bergantung pada panjang *plaintext* yang akan dienkripsi.

Berikutnya setelah blok data terbentuk maka *plaintext* akan dimasukan berurut perkolom dimulai dari kolom pertama dan proses enkripsinya akan dibaca kebalikannya perbaris pada posisi awal baris terakhir. Berikut blok-blok data dengan menggunakan panjang kunci 25 karakter, misalkan kata ”CONTOH DATA ENKRIP” akan dienkripsi menjadi ”H POAITTRNAKODNC E”). Dan susunan ini dapat berubah pula jika panjang kunci yang digunakan berubah.

C		E
O	D	N
N	A	K
T	T	R
O	A	I
H		P

Gambar 3.2 Contoh penerapan pengacakan teks dengan teknik *blocking*

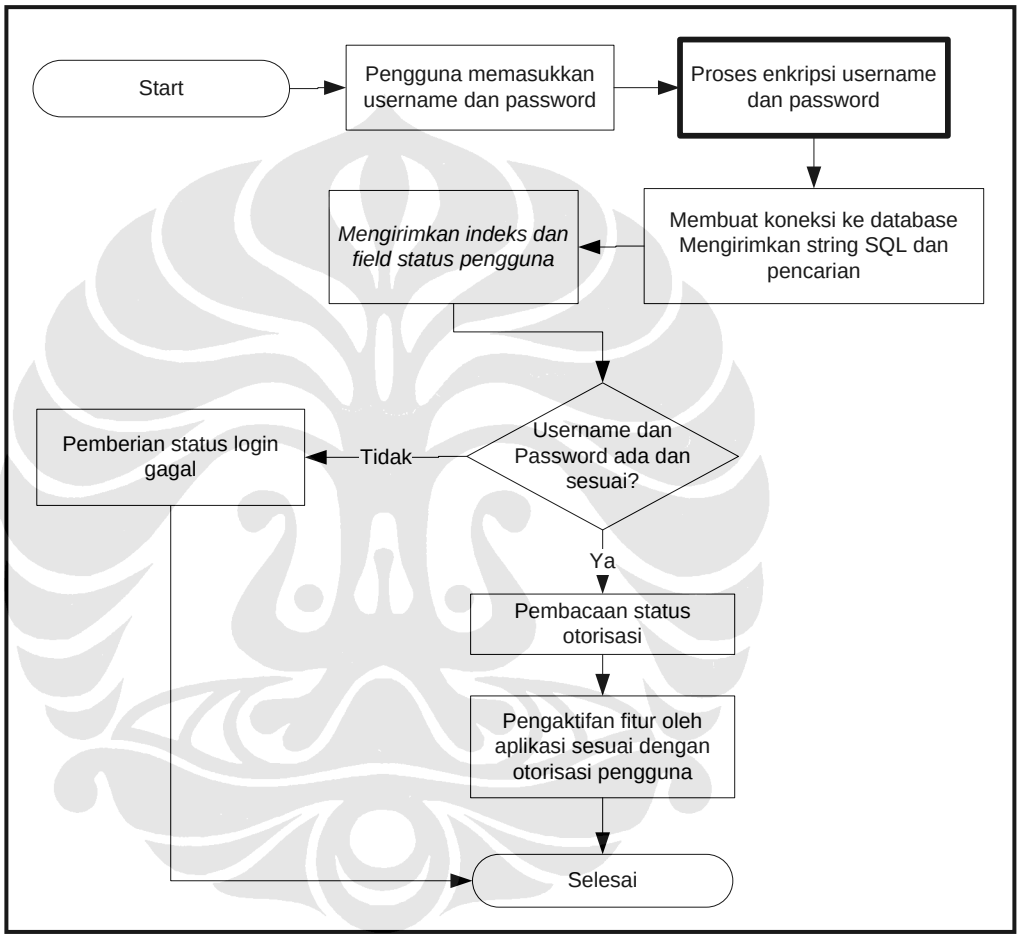
Pengubahan susunan ini diharapkan dapat memutuskan hubungan *plaintext* terhadap *ciphertext* yang menjadi kelemahan enkripsi *stream cipher* menggunakan operasi standard yaitu XOR seperti yang dijelaskan pada bab 2 mengenai kelemahan RC4. Teknik susunan pengacakan juga diharapkan tidak diketahui selama panjang kunci yang digunakan tidak diketahui.

3.1.2 Perancangan Sistem

Perancangan sistem digunakan untuk memenuhi sebagian besar kebutuhan aplikasi dan terdapat 3 proses utama yang dirancang untuk aplikasi ”sistem informasi kegiatan skripsi” ini yaitu proses otentifikasi (*login*), *upload* data dan *download* data. Ketiga proses ini selanjutnya dapat dikembangkan menjadi berbagai fitur yang ada pada aplikasi tersebut. Berikut penjelasan dan *flowchart* masing-masing proses.

a. Proses Otentifikasi

Proses otentifikasi merupakan proses untuk menseleksi dan memberikan hak akses kepada pengguna yang akan menggunakan aplikasi ini. Proses otentifikasi harus dilalui oleh pengguna yang akan menggunakan sistem ini. **Gambar 3.1** memberikan *flowchart login* pada sistem.



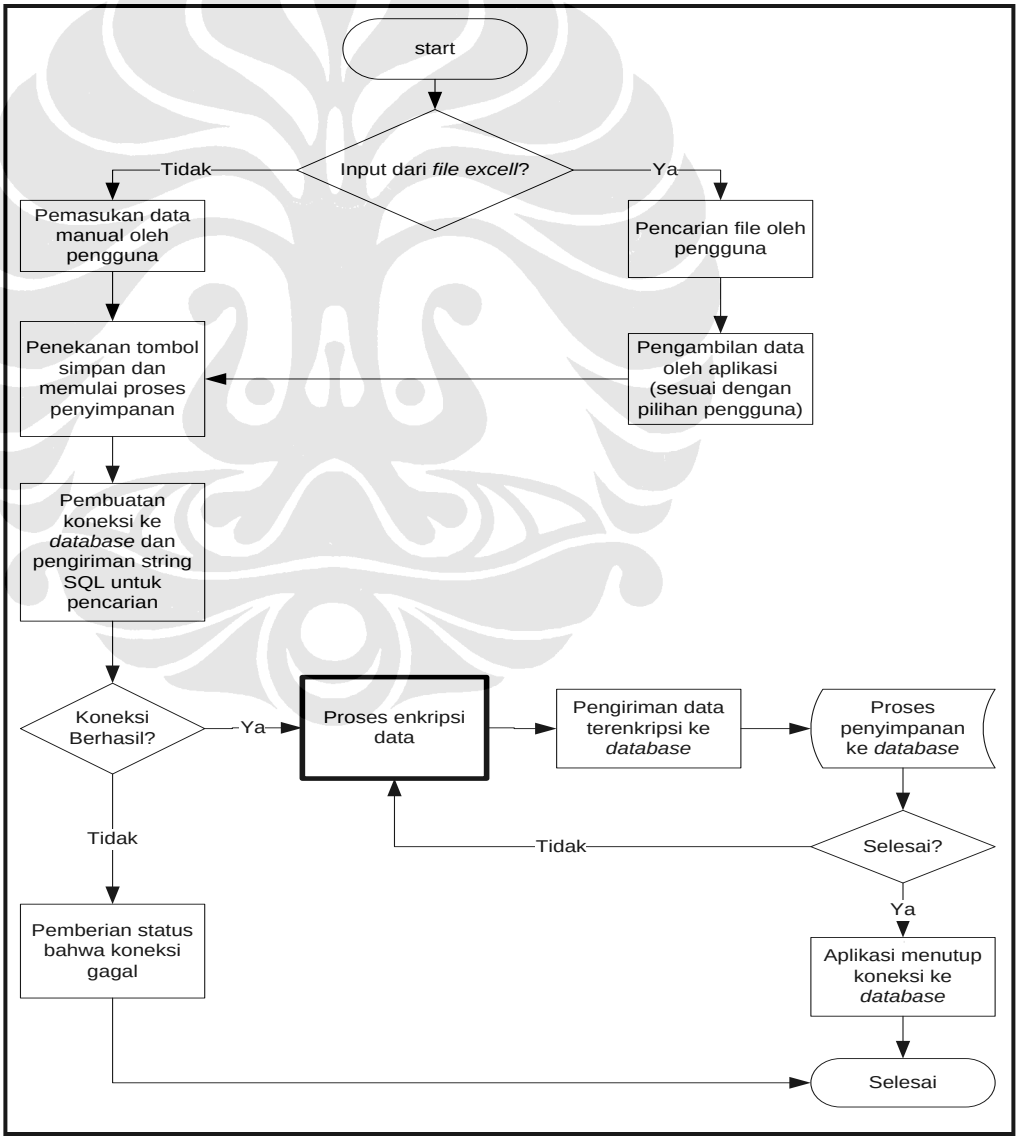
Gambar 3.3 Gambar *flowchart login*

Seperti digambarkan pada algoritma proses otentifikasi pada gambar 3.1 terdapat proses pengaktifan fitur berdasarkan otorisasi pengguna yaitu *administrator*, *editor* dan *viewer*. Peran dari *administrator* yaitu untuk melakukan pengolahan nama-nama departemen, fakultas, penjurusan, dosen, dan mahasiswa serta untuk mengolah *username*, *password*, hak akses pengguna dan olah kunci yang digunakan untuk proses enkripsi dan dekripsi data. Peran dari *editor* yaitu melakukan pencarian dan

pengolahan *database* meliputi menambah, mengubah bahkan menghapus isi data. Sedangkan peran dari *viewer* yaitu hanya untuk melakukan proses pencarian isi dari *database*.

b. Proses *Upload* Data

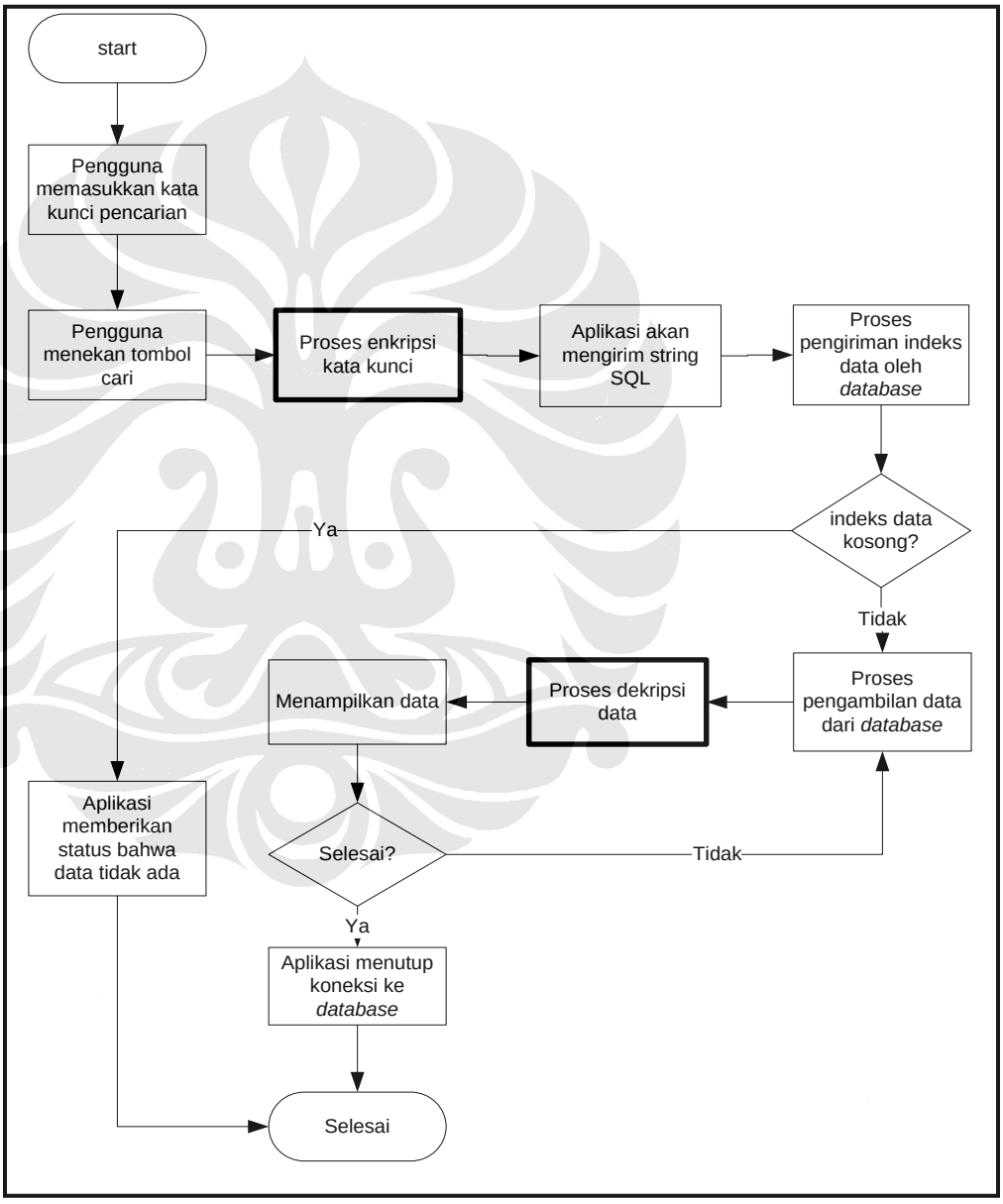
Proses *upload* ini berfungsi secara umum untuk memasukkan data ke *database*, baik itu dimasukkan satu persatu secara manual oleh pengguna maupun secara otomatis dengan cara mengambil dari *database* lain seperti file *excel*. **Gambar 3.2** memberikan *flowchart* proses *upload* pada system.



Gambar 3.4 Gambar proses *upload* data pada sistem aplikasi

a. Proses *Download* data

Proses *download* ini berfungsi secara umum untuk mengambil data dari *database*. **Gambar 3.2** memberikan *flowchart* proses *download* pada sistem.



Gambar 3.5 Gambar proses *download* data pada sistem aplikasi

Proses ini dapat dilakukan secara umum oleh semua pengguna hanya saja berbeda fitur. Untuk *editor*, *download* data banyak digunakan untuk proses pencarian dan pengolahan data kegiatan skripsi, untuk administrator akan banyak digunakan untuk mencari dan mengolah nama-nama departemen, fakultas, mahasiswa hingga dosen, dan untuk *viewer* digunakan umumnya hanya untuk pencarian data.

3.1.3 Perancangan dan Struktur *Database*

Database digunakan sebagai tempat simpan seluruh data yang digunakan pada aplikasi. *Database* dapat menggunakan *database* seperti *MySQL* hingga *SQL Server* dengan koneksi utama ODBC dengan *database access*. *Database* utama yang digunakan pada aplikasi ini diberi nama "Skripsi_Server", dimana aplikasi akan membaca *database* ini ketika pertama kali *login*.

Terdapat 8 Tabel yang digunakan dalam aplikasi ini yaitu tabel *login*,tabel data, tabel departemen, tabel fakultas, tabel dosen, tabel penjurusan, tabel mahasiswa dan tabelkunci. Terdapat 3 tabel utama yang digunakan disini yaitu:

a. Tabel *Login*

   Tabel *login* diberi nama tbl*Login* pada *database* "Skripsi_Server". Berikut *field-field* yang ada pada tabel *Login*.

Tabel 3.1 Tabel *Login*

Field	**Tipe Data**	**Keterangan**
Nomor	Text	Digunakan sebagai *primary key*
UserName	Text	*Id* pengguna yang digunakan saat *login*
Password	Text	*Password* yang digunakan saat *login*
FullName	Text	Nama Lengkap dari pengguna
Admin	Text	Berisi 1/0 untuk menunjukkan hak pengguna sebagai admin atau tidak
Editor	Text	Berisi 1/0 untuk menunjukkan hak pengguna sebagai *editor* (pengolah data) atau tidak
Viewer	Text	Berisi 1/0 untuk menunjukkan hak pengguna sebagai pangamat (pencari) data atau tidak
LastModified	Date/Time	Untuk mengetahui status *record* kapan *record* dicatat atau berubah
ModifiedBy	Text	Untuk mengetahui status *record* siapa yang mencatat atau merubah data

b. Tabel Data

Tabel data diberi nama tbData. Tabel data digunakan untuk menyimpan data-data utama yaitu data kegiatan skripsi pada aplikasi ini. Berikut *field-field* yang ada pada tabel *Login*.

Tabel 3.2 Tabel Data

Field	**Tipe Data**	**Keterangan**
Nomor	Text	Digunakan sebagai *primary key*
Fakultas	Text	Fakultas Mahasiswa
Departemen	Text	Departemen Mahasiswa
Penjurusan	Text	Penjurusan Mahasiswa
Tema	Text	Tema Skripsi
Judul	Text	Judul Skripsi
Bulan	Text	Bulan pelaksanaan akhir skripsi
Tahun		Tahun pelaksanaan skripsi
Mahasiswa	Text	Nama Mahasiswa pelaksana skripsi
DosenPbb	Text	Nama Dosen Pembimbing Mahasiswa
Grade	Text	Nilai akhir dari Skripsi
LokasiSimpan1	Text	Catatan lokasi 1 simpan buku atau media lain skripsi
LokasiSimpan2	Text	Catatan lokasi 2 simpan buku atau media lain skripsi
LastUpdate	Date/Time	Untuk mengetahui status *record* kapan *record* dicatat atau berubah
ModifiedBy	Text	Untuk mengetahui status *record* siapa yang mencatat atau merubah data

c. Tabel Kunci

Tabel kunci diberi nama tblKunci pada *database* Skripsi_*Server*. Tabel kunci ini digunakan untuk manejeman kunci yang hanya dapat diolah oleh administrator. Berikut *field-field* yang ada pada tabel kunci.

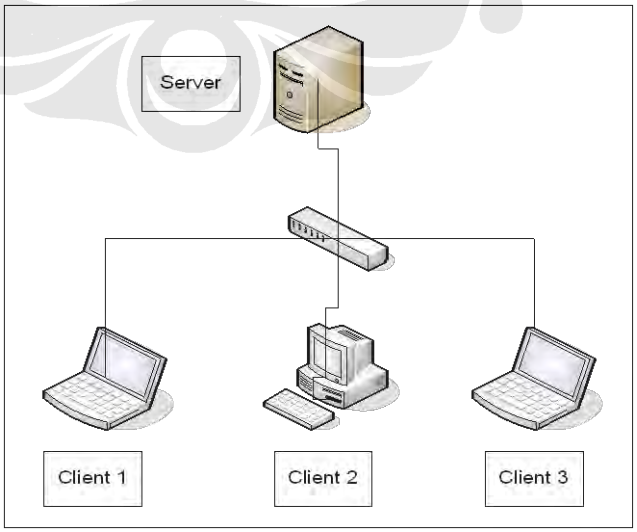
Tabel 3.3 Tabel Kunci

Field	**Tipe Data**	**Keterangan**
ID	*AutoNumber*	Digunakan sebagai primary *key* dan *autonumber*
Kunci	Text	Data yang digunakan sebagai kunci enkripsi

3.2 PERANCANGAN JARINGAN UJI

Jaringan dirancang dengan jaringan LAN sederhana dimana terdapat 4 komputer. 1 komputer berperan sebagai *server* dan 3 komputer lainnya berperan sebagai *client*. Berikut rancang bangun jaringan tersebut :

\



Gambar 3.6 Topologi jaringan yang digunakan

Spesifikasi komputer yang digunakan diberikan pada tabel berikut ini :

Tabel 3.5 Tabel spesifikasi komputer *server* dan *client* pada jaringan uji

Spesifikasi	***Server***	***Client* 1**	***Client* 2**	***Client* 3**
Processor	Intel(R) Pentium(R) 4 CPU 2,4 Ghz (2 CPUs)	Pentium Dual CPU 1.6 Ghz	Intel(R) Pentium(R) 4 CPU 2,4 Ghz	Intel(R) Celeron(R) CPU 1.4 Ghz
Memori RAM	512 MB	2 GB	512MB	502 MB
`VGA	RADEON 9200 SE FAMILY 128 MB	VGA VIA Chrome 9 HC IGP Family 256 MB	NVIDIA GeForce4 MX 440with AGP8X 64 MB	Mobile Intel(R) 965 Express Chipset FAMILY 128 MB
Sistem Operasi	Microsoft Windows XP Profesional	Microsoft Windows XP Profesional	Microsoft Windows XP Profesional	Microsoft Windows XP Profesional

Secara umum yang diperhatikan pada pengujian adalah kecepatan prosesor dari masing-masing komputer, terutama *client* dimana kecepatan prosesor *client* 1 lebih besar dari kecepatan prosesor *client* 2 dan 3.

Dan konfigurasi dari masing-masing komputer tersebut ialah :

Tabel 3.5 Tabel konfigurasi komputer *server* dan *client* pada jaringan uji

Konfigurasi	***Server***	***Client* 1**	***Client* 2**	***Client* 3**
IP *address*	152.118.101.69	152.118.101.67	152.118.101.68	152.118.101.70
NetMask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

3.3 SKENARIO PENGUJIAN

Salah satu pengujian yang dilakukan pada aplikasi ini ialah keamanan data, dan kualitas pelayanan jaringan seperti waktu tanggap (*response time*). Terdapat beberapa skenario utama yang dilakukan yang diantaranya:

3.3.1 Skenario Pengujian 1

Pengujian pertama yaitu pengujian proses *upload* dan *download* pada komputer lokal artinya *database* ditaruh pada komputer lokal untuk menguji kinerja olah data berupa waktu tanggap dengan metode keamanan enkripsi data yang dibandingkan terhadap aplikasi tanpa enkripsi. Tujuan dari pengujian ini yaitu untuk membandingkan kecepatan olah data enkripsi dan tanpa enkripsi.

3.3.2 Skenario Pengujian 2

Pengujian kedua yaitu pengujian proses *upload* dan *download* pada jaringan uji hanya oleh 1 *client*. Pengujian yang dilakukan tidak hanya menggunakan parameter waktu tanggap tetapi juga untuk menguji keamanan enkripsi data dan besar paket yang dikirimkan antara menggunakan enkripsi dan tanpa menggunakan enkripsi.

3.3.3 Skenario Pengujian 3

Pengujian ketiga yaitu pengujian proses *download* oleh 3 *client* yang berbeda kecepatan prosesor secara terpisah (tidak bersamaan). Tujuan dari pengujian ini yaitu membandingkan kinerja dekripsi (*download*) data terhadap kecepatan prosesor komputer yang digunakan.

3.3.3 Skenario Pengujian 4

Pengujian terakhir yaitu pengujian proses *download* bersama oleh 3 *client* yang berbeda kecepatan prosesor. Skenario ini menggunakan 3 *client* bersamaan untuk men-*download record* pada *database*. Tujuan dari pengujian ini yaitu melihat kinerja dekripsi data pada saat beberapa *client* secara bersamaan men-*download* data.

BAB IV

UJI COBA DAN ANALISA PENGUJIAN APLIKASI

4.1 UJI COBA APLIKASI

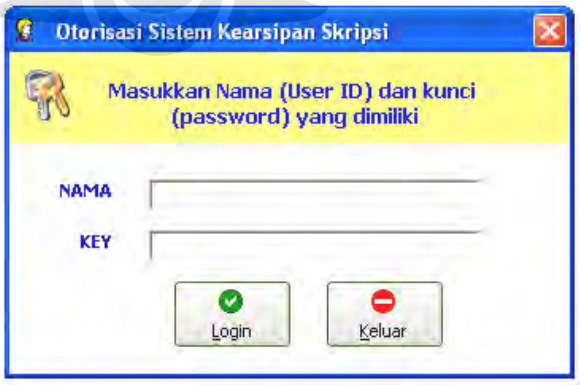
Pada pengujian ini akan dicoba berbagai fitur utama yang dikembangkan pada aplikasi ini. Fitur-fitur yang dicoba pada aplikasi ini adalah. Berikut tampilan-tampilan utama fitur-fitur tersebut.

4.1.1 Uji Coba Fitur Aplikasi

Pada pengujian fitur aplikasi akan diuji beberapa fitur utama yang ada pada aplikasi yaitu fitur *Login*, *Upload* Data dan Edit (Pengolahan) Data.

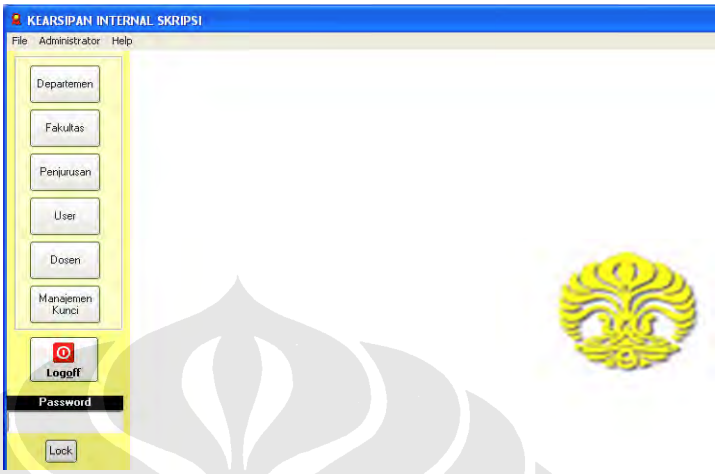
4.1.1.1 Fitur *Login*

Saat program pertama kali dijalankan telah tampil menu *login* untuk pengguna agar memasukkan *username* dan *password* ke sistem, sehingga sistem dapat menentukan hak akses fitur yang dapat digunakan. Pada **Gambar 4.1** merupakan tampilan menu *login* sistem.

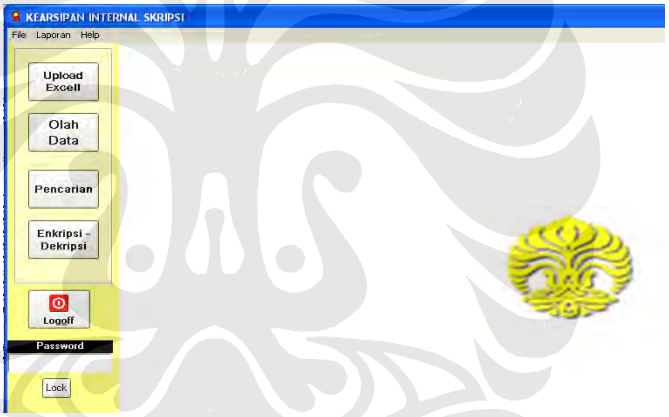


Gambar 4.1 Tampilan Utama Saat *Login*

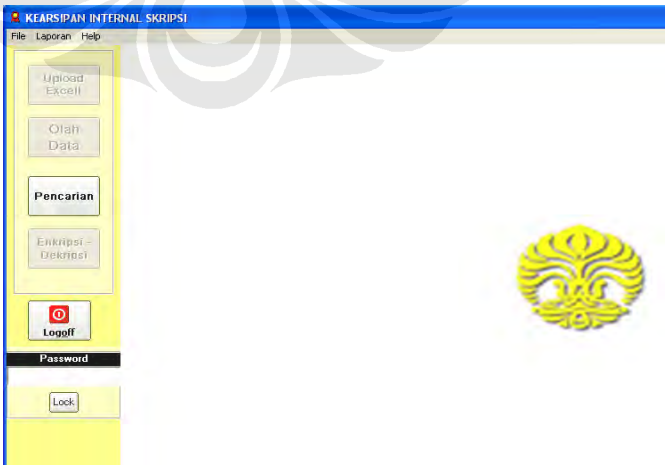
Setelah dimasukkan *username* dan *password* ke aplikasi maka telah tampil beberapa fitur yang berbeda sesuai dengan hak akses fitur yang dapat digunakan.



Gambar 4.2 Tampilan utama untuk *administrator*



Gambar 4.3 Tampilan utama untuk *editor*



Gambar 4.4 Tampilan utama untuk *viewer*

4.1.1.2 Fitur *Upload* Data

Upload data hanya dapat dilakukan oleh *editor*. Pada menu *editor* ditekan tombol "*upload excell*" dan telah tampil menu *upload* dan berhasil meng*upload* beberapa data dari file excel. Beberapa contoh file yang berhasil ditransmisikan dibahas pada **subbab 4.1.6** tentang uji coba keamanan transaksi.

4.1.1.3 Fitur *Download* Data

Download data hanya dapat dilakukan oleh *editor* dan *viewer*. Pada menu *editor* atau *viewer* ditekan tombol "pencarian" dan telah tampil menu pencarian dan telah berhasil mengambil beberapa data dari *database* sesuai dengan kata kunci yang digunakan. Contoh file yang berhasil ditransmisikan dari *database* ke *client* dibahas pada **subbab 4.1.6** tentang uji coba keamanan transaksi.

4.1.2 Uji Coba Keamanan Transaksi

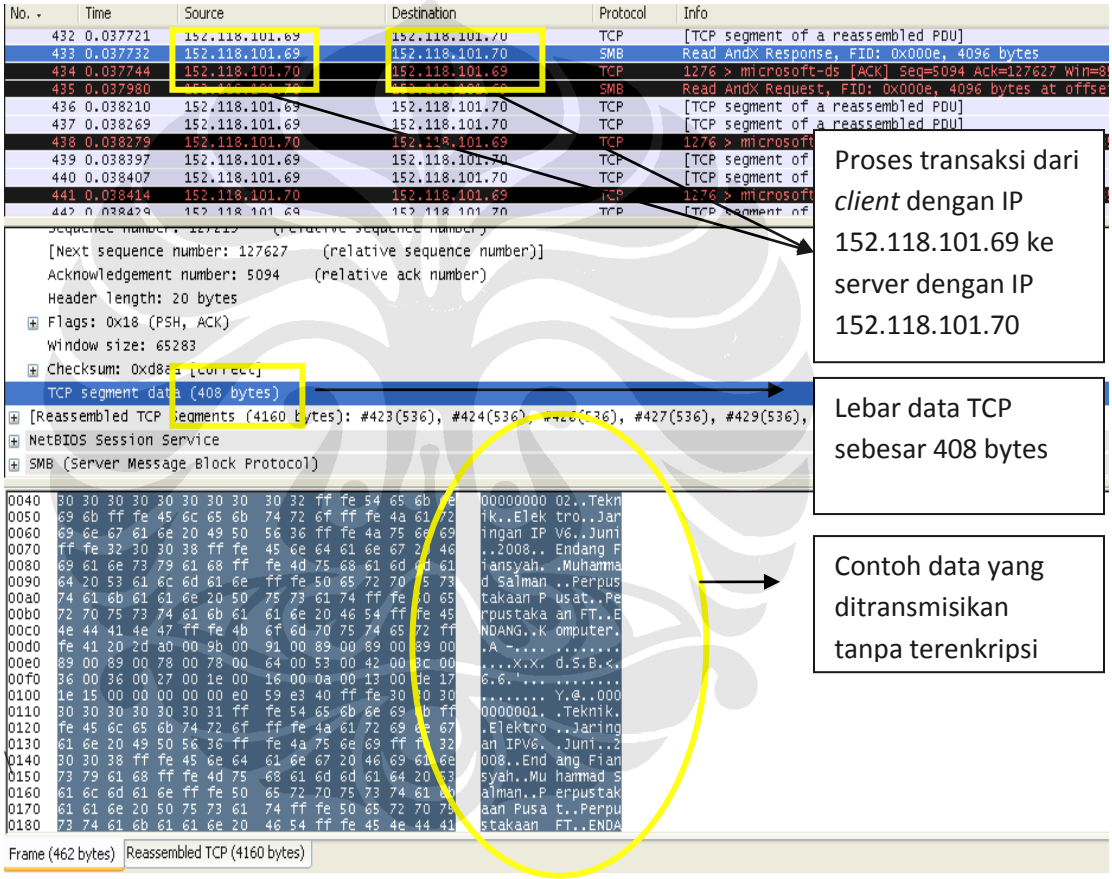
Setelah bagaimana aplikasi tersebut berjalan dengan baik pada modul utama maka berikutnya akan diuji bagaimana aplikasi ini diterapkan pada jaringan saat proses transaksi *upload* dan *download* dilaksanakan. Proses *upload* dan *download* merupakan kegiatan utama pada aplikasi ini dimana proses *upload* secara umum mengartikan bahwa aplikasi akan mengirimkan data berupa *record* ke *database* dan proses *download* merupakan proses dimana aplikasi meminta (*request*) data berupa *record* ke *database* dan *database* mengirimkannya ke aplikasi berdasarkan SQL yang diberikan oleh aplikasi.

Proses *upload* menggambarkan bagaimana enkripsi dilakukan, karena pada saat proses *upload* data harus terenkripsi terlebih dahulu sebelum dikirimkan ke *database*, sedangkan proses *download* menggambarkan bagaimanana dekripsi dilakukan, karena pada saat proses *download* data harus didekripsi terlebih dahulu sebelum aplikasi menampilkan dalam bentuk tabel. Berikut diberikan beberapa contoh proses yang terjadi menggunakan perangkat lunak *Wireshark* untuk

menganalisa hasil paket-paket yang dikirimkan dari sisi *client* maupun *server* pada jaringan uji.

4.1.2.1 Proses *Upload*

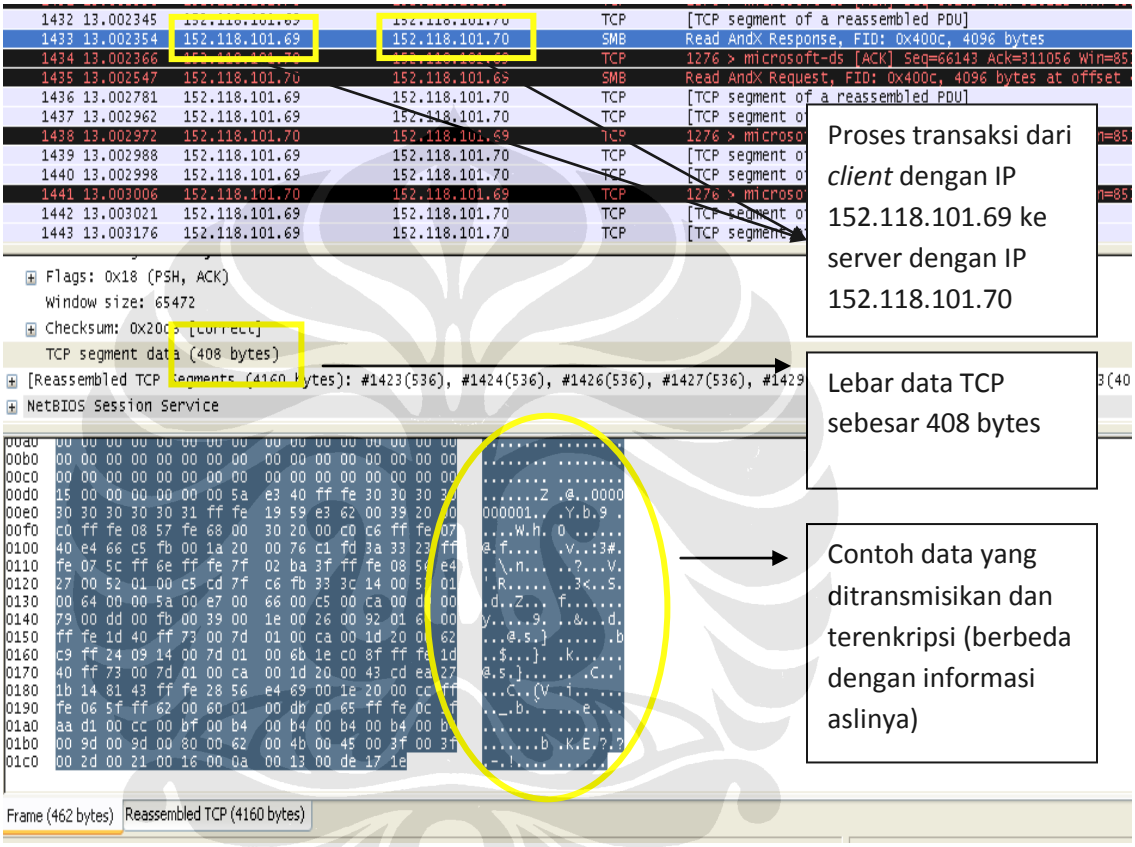
Berikut gambaran hasil tangkap paket oleh *wireshark* pada sisi *client* dan *server* serta akan dibandingkan juga paket yang menggunakan teknik enkripsi dengan paket yang tidak menggunakan teknik enkripsi.



Gambar 4.5 Salah satu contoh hasil tangkap oleh *wireshark* di *client* saat proses *upload* dilakukan oleh aplikasi tanpa enkripsi

Pada gambar tersebut diperlihatkan bagaimana data pada no urut ke 000000000002 dan 000000000001 dikirimkan dari sisi *client* ke *server*. Diperlihatkan bagaimana aplikasi tanpa enkripsi memberikan data yang dapat diamati dan ditangkap dan memberikan informasi asli yang dibawa oleh satu *client* ke *server*.

Teramati bahwa pada data ke 000000000002 dikirimkan data Teknik Elektro, Januari, IPV6,Juni 2008, dan seterusnya. Artinya tanpa proses enkripsi dijaringan informasi yang ditransmisikan dapat terlihat. Selanjutnya berikut gambar hasil penangkapan paket untuk aplikasi yang menggunakan teknik enkripsi.

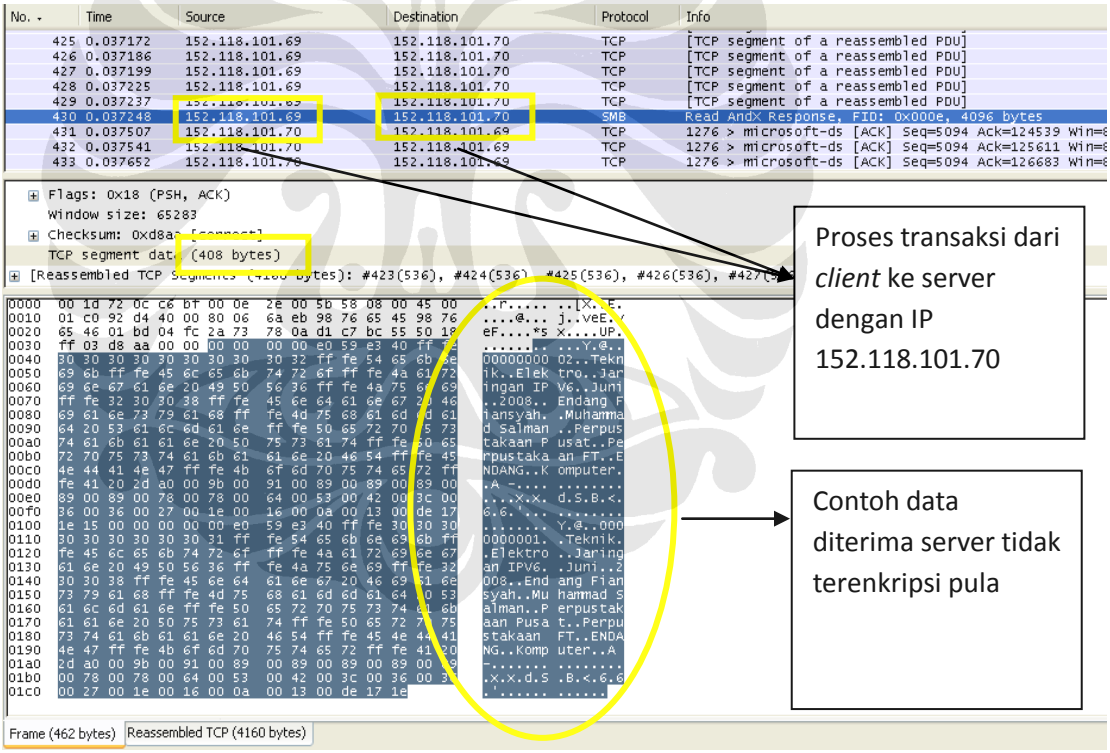


Gambar 4.6 Salah satu contoh hasil tangkap oleh *wireshark* di *client* saat proses *upload* dilakukan oleh aplikasi dengan enkripsi

Pada gambar tersebut diperlihatkan bagaimana *Wireshark* menangkap paket yang dikirimkan oleh *client* ke *server*. Pengiriman data menggunakan data yang sama dengan aplikasi tanpa enkripsi. Teramati bahwa data berisi karakter-karakter yang berbeda dengan data aslinya. Karakter-karakter ini merupakan hasil enkripsi kata atau kalimat Teknik Elektro, Januari, IPV6,Juni 2008, dan seterusnya.

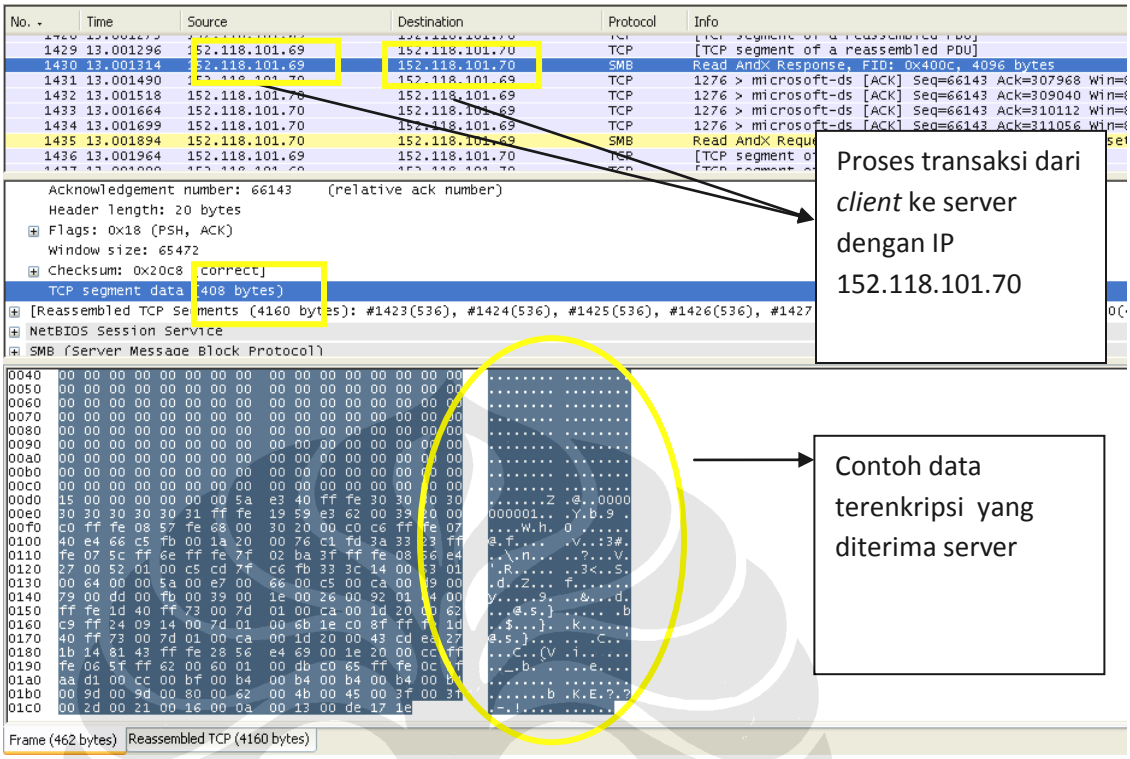
Hal lain yang diperhatikan yaitu yaitu lebar data yang dikirimkan (gambar yang diberi tanda kotak) memperlihatkan bahwa lebar data yang dikirimkan dengan teknik enkripsi sama dengan lebar data yang dikirimkan tanpa enkripsi yaitu sebesar 408 *byte*s. Hal ini menunjukkan bahwa teknik enkripsi pada aplikasi ini tidak menambah apapun kepada data yang dikirimkan, tetapi hanya mengubah kode pesan kebentuk lain yang diharapkan tidak mampu dimengerti informasi aslinya. Keuntungan dari lebar data yang sama antara aplikasi tanpa enkripsi dengan aplikasi dengan teknik enkripsi yaitu aplikasi dengan enkripsi tidak lebih memberatkan *server* dan jaringan dari segi *bandwidth* dan memori.

Berikut gambar hasil tangkap paket saat proses *upload* tersebut terjadi dari sisi *server*.



Gambar 4.7 Salah satu contoh hasil tangkap oleh *wireshark* di *server* saat proses *upload* dilakukan oleh aplikasi tanpa enkripsi

Teramati bahwa data dikirimkan tidak terenkripsi sama seperti apa yang dikirimkan oleh *client*.



Gambar 4.8 Salah satu contoh hasil tangkap oleh *wireshark* di *client* saat proses *upload* dilakukan oleh aplikasi dengan enkripsi

Teramati bahwa data dikirimkan terenkripsi sama seperti apa yang dikirimkan oleh *client*. Berikut contoh gambar isi dari *database* pada tabel "tblData" untuk aplikasi yang menggunakan teknik keamanan enkripsi dan aplikasi yang tidak menggunakan teknik keamanan enkripsi sama sekali.

Nomor	Fakultas	Departemen	Penjurusan	Tema	Judul	Bulan	Tahun	Mahasiswa	DosenPbb	Grade	LokasiSimpan1
0000000001	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000002	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000003	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000004	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000005	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000006	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000007	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000008	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000009	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F
0000000010	Teknik	Elektro	Komputer	Jaringan IPV6		Juni	2008	Endang Fiansy:	Muhammad Sa	A -	Perpustakaan F

(a)

Nomor	Fakultas	Departemen	Penjurusan	Tema	Judul	Bulan	Tahun	Mahasiswa	DosenPbb	Grade	LokasiSimpan1
0000000001	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000002	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000003	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000004	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000005	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000006	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000007	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000008	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000009	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$
0000000010	□Ÿâb‹À	□Wþh‰ÀÆ	□_ÿbŠÛÀe	□@áfÂû,vÁý:3#		□\ÿn	□□º?	□Vä'ŒÃÍ□Æü3		□□ª	□@ÿsŽÊ"bÉÿ$

(b)

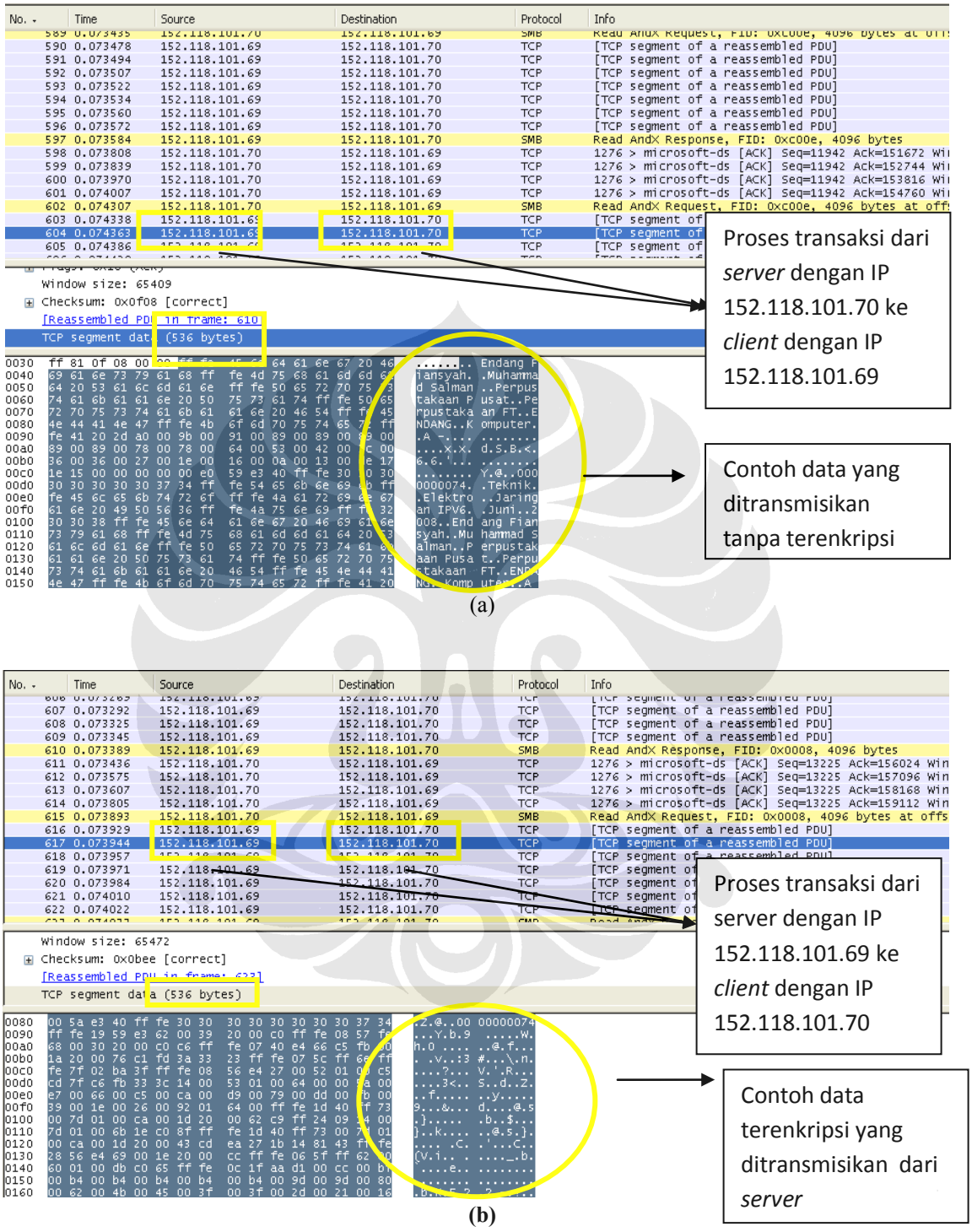
Gambar 4.9 Salah satu contoh hasil simpan data pada *database* pada aplikasi tanpa enkripsi (a) dan aplikasi dengan teknik enkripsi (b)

Pada **Gambar 4.13** tersebut diperlihatkan bagaimana perbedaan data yang disimpan dengan aplikasi yang menggunakan teknik enkripsi (Gambar 4.13 (a) dan aplikasi yang tidak menggunakan teknik enkripsi (Gambar 4.13 (b).

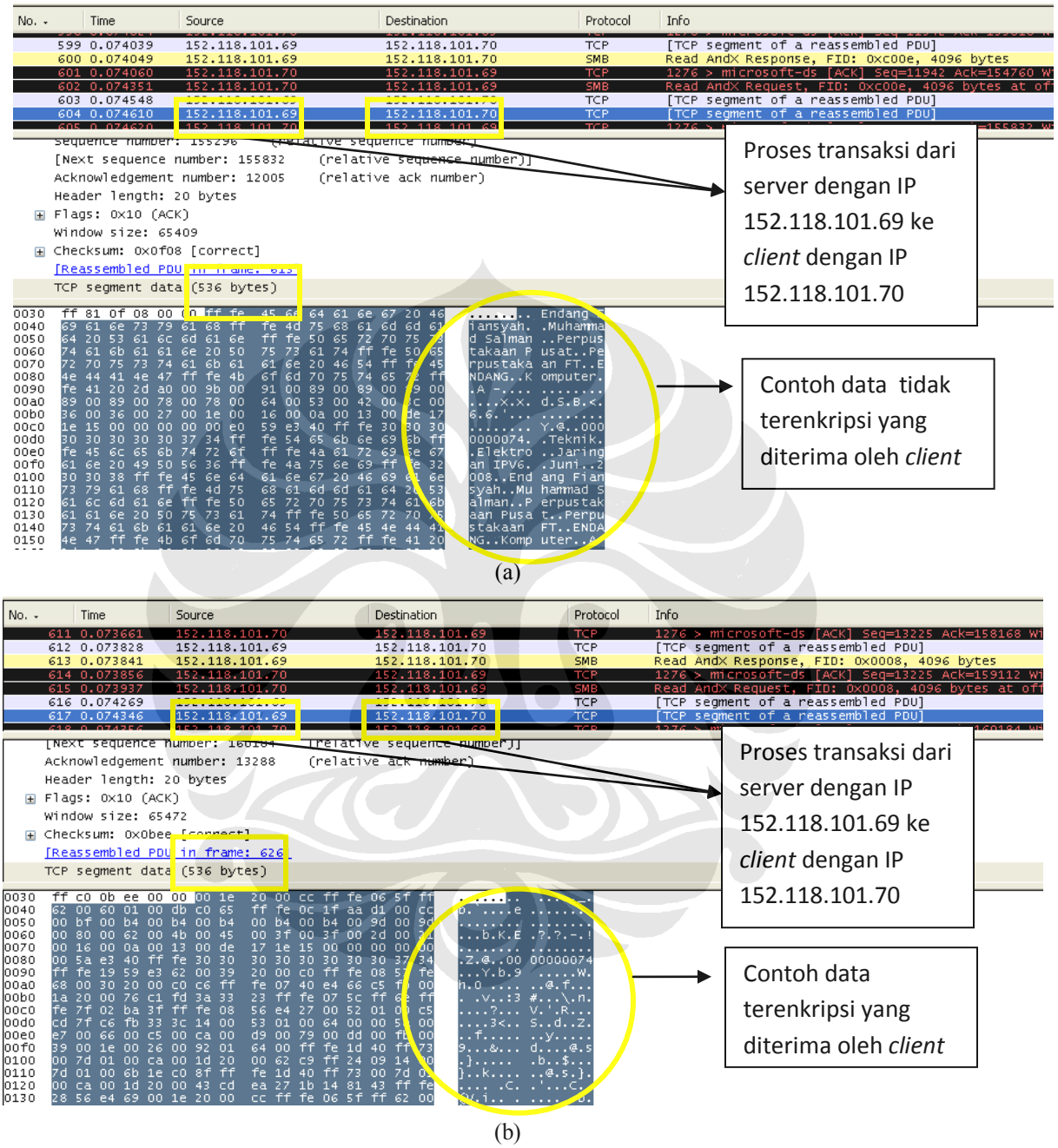
Dari gambaran tersebut memperlihatkan bagaimana proses *upload* di sisi *client*, *server* dan *database* yang menggunakan teknik enkripsi akan mentransmisikan data dengan kondisi terenkripsi, tetapi tidak membuat data tersebut semakin lebar (tidak menambah jumlah memori pada sisi jaringan, *server* dan *database*).

4.1.2.2 Proses *Download*

Proses *download* pun untuk aplikasi yang menggunakan teknik enkripsi dan mengirimkan data dari *server* ke *client* akan mengirimkan data yang tetap terenkripsi. Data yang sebenarnya akan didekripsi pada sisi aplikasi *client*. Berikut beberapa contoh hasil tangkap paket pada sisi *client* dan *server*.



Gambar 4.10Salah satu contoh hasil tangkap oleh *wireshark* di *server* saat proses *download* dilakukan oleh aplikasi tanpa enkripsi (a) dan aplikasi dengan enkripsi (b)



Gambar 4.11 Salah satu contoh hasil tangkap oleh *wireshark* di *client* saat proses *download* dilakukan oleh aplikasi tanpa enkripsi (a) dan aplikasi dengan enkripsi (b)

Dari gambar penangkapan paket oleh perangkat lunak *wireshark* dari sisi *server* dan juga *client* menunjukkan perbedaan karakter yang dikirimkan antara aplikasi yang tidak menggunakan metode keamanan enkripsi dan aplikasi yang menggunakan metode keamanan enkripsi. Hal yang serupa untuk proses otentifikasi dimana seluruh data yang ditransmisikan akan terenkripsi terlebih dahulu sebelum disimpan ke *database*, sehingga diharapkan proses otentifikasi pun berjalan aman (dengan tidak mengetahui username dan password yang digunakan karena terenkripsi).

Dengan metode enkripsi ini diharapkan data yang diolah dapat ditransaksikan dengan aman. Dengan asumsi bahwa kelemahan dari RC4 yaitu terdapat relasi langsung antara kunci *stream* yang digunakan dengan *plaintext* yang diolah dan *ciphertext* yang dihasilkan seperti dijelaskan pada **sub bab 2.4.2.3**. Apabila proses pengacakan *plaintext* dilakukan maka hubungan tersebut diharapkan dapat terputus, karena walaupun diketahui suatu *plaintext* dan *ciphertext*nya maka kunci *stream* yang dihasilkan tidak lah menunjukkan bahwa

$\mathbf{K_1 = P_1\ XOR\ C_1}$

$\mathbf{K_2 = P_2\ XOR\ C_2}$

$\mathbf{K_3 = P_3\ XOR\ C_3}$ dst

Dimana K_1,K_2 hingga K_{256} adalah urutan *stream* kunci dari Sbox. Karena bisa jadi sebenarnya pengolahannya ialah

$\mathbf{K_1 = P_3\ XOR\ C_1}$

$\mathbf{K_2 = P_1\ XOR\ C_2}$

$\mathbf{K_3 = P_2\ XOR\ C_2}$ .....karena *Plaintext* diacak terlebih dahulu. Dengan asumsi penyerang tidak mengetahui proses pengacakannya terkecuali mengetahui kunci enkripsinya dan panjang kunci yang digunakan maka kunci *stream* yang

sebenarnya juga tidak diketahui. Terlebih lagi dimana pengacakan tidak statis dan bergantung dari kunci yang digunakan, maka *plaintext* akan diacak dan tidak diketahui kecuali kunci enkripsi benar-benar ditemukan.

Untuk menemukan *key stream* yang misalkan terdiri dari 10 karakter saja maka terdapat kombinasi yang luar biasa yaitu misalkan karakter yang mungkin hanya dari a-z, A-Z, dan angka 0-9 (62 kombinasi) maka untuk 10 karakter saja terdapat kemungkinan kunci $62^{\wedge 10}$ atau sama dengan $8{,}4 \times 10^{17}$. Jika menggunakan metode bruteforce dengan menggunakan prosesor 3 GHz maka membutuhkan waktu

$$8{,}4 \times 10^{17} / 3 \times 10^{9} = 2{,}79 \times 10^{8} \text{ detik atau } 8{,}99 \text{ tahun.}$$

Jika menggunakan karakter yang lebih dari 10, maka kemungkinan dan waktu proses untuk mendapatkannya akan semakin besar. Diharapkan proses enkripsi dan dekripsi ini memberikan keamanan untuk data yang ditransmisikan.

4.2 ANALISIS PENGUJIAN APLIKASI

Pengujian aplikasi dilakukan pada 4 skenario yaitu :

1. Proses *upload* dan *download* pada *database* lokal (untuk *server*).
2. Proses *upload* dan *download* di jaringan (*client* - *server*).
3. Proses *dowload* oleh client 1, 2 dan 3 secara terpisah di jaringan.
4. Proses *download* oleh 3 *client* secara bersamaan di jaringan.

Setiap skenario menggunakan jumlah data yang bervariasi dan diharapkan mendapatkan gambaran secara umum mengenai kualitas dari aplikasi dengan keamanan enkripsi ini. Setiap data rata-rata yang disajikan merupakan rata-rata dari 5 hingga 10 kali pengambilan data. Berikut analisa masing-masing skenario tersebut.

4.2.1 Analisis Skenario 1

4.2.1.1 Analisis *upload* oleh *client* 1 pada *database* lokal

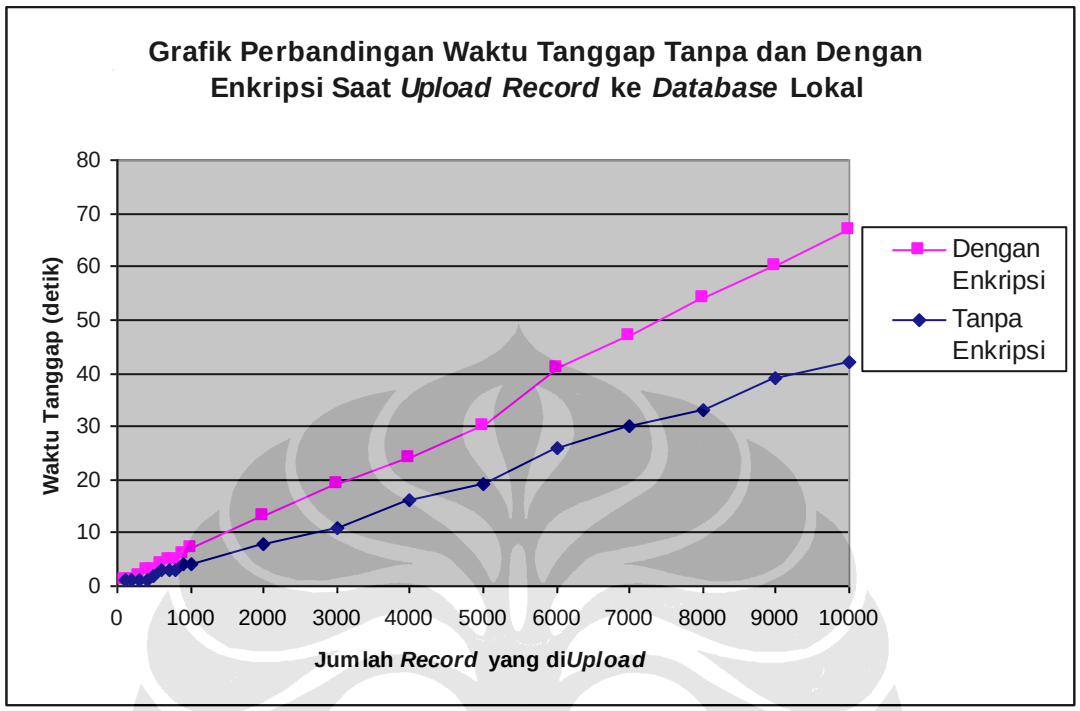
Pada pengujian pertama ini dilakukan transaksi pada *database* lokal, dan berikut data hasil pengujian tersebut.

Tabel 4.1 Tabel data skenario pengujian 1 untuk proses *upload* data

Jumlah *Record* yang di *Upload*	Rata-rata Waktu Tanggap (detik)		
	Dengan Enkripsi	Tanpa Enkripsi	Selisih Waktu
100	1	1	0
200	1	1	0
300	2	1	1
400	3	1	2
500	3	2	1
600	4	2,7	1,3
700	5	2,9	2,1
800	5,8	3	2,8
900	6	3,8	2,2
1000	7	4,1	2,9

Jumlah *Record* yang di *Upload*	Rata-rata Waktu Tanggap (detik)		
	Dengan Enkripsi	Tanpa Enkripsi	Selisih Waktu
1000	7	4,1	2,9
2000	12,9	7,9	5
3000	19,2	11,1	8,1
4000	24	16,1	7,9
5000	30,4	19	11,4
6000	41,2	26,3	14,9
7000	47,6	30,8	16,8
8000	54	33	21
9000	60	39,8	20,2
10000	67	42,4	24,6

Berikut grafik dari data tersebut.



Gambar 4.12 Gambar grafik pengujian skenario 1 untuk proses *upload* data

Dari tabel tersebut diperlihatkan bagaimana untuk proses *upload* dari 100 hingga 1000 *record* maksimum perbedaan waktu tanggap jika menggunakan teknik enkripsi yang dikembangkan yaitu sebesar 2,9 detik. Sedangkan untuk jumlah *record* yang semakin tinggi selisih waktu akan semakin membesar, dan mulai dari jumlah *record* melebihi 5000 selisih waktu melebihi 10 detik dan selisih tertinggi ditunjukkan pada saat jumlah pengujian *record* tertinggi (10.000 *record*) yaitu sebesar 24,6 detik.

Pada aplikasi ini seperti yang dijelaskan pada bab 3, untuk proses pengolahan data khususnya pada proses *upload* ini terdapat 15 *field* yang dikirimkan dari aplikasi ke *database* untuk setiap *record* dan pada pengolahan ini 11 *field* yang dienkripsi. Untuk itu setiap *record* harus mengalami 11 x proses enkripsi. Hal ini mengakibatkan waktu proses semakin lama seiring dengan peningkatan jumlah *record* yang di

upload. Pada pengujian ini perbedaan waktu tanggap kurang dari 10 detik terjadi saat jumlah *record* yang diolah kurang dari 5000 *record*.

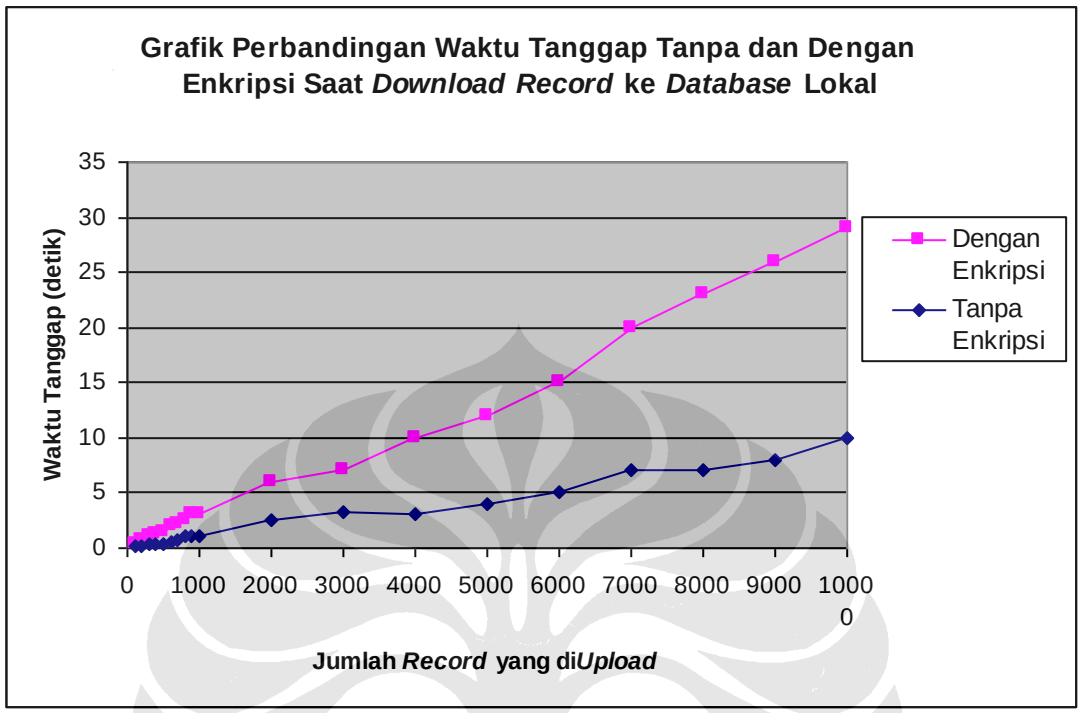
Selanjutnya akan dianalisa bagaimana kinerja proses *download* atau dekripsi pada aplikasi.

4.2.1.2 Analisis *download* oleh *client* 1 pada *database* lokal

Tabel 4.2 Tabel data skenario pengujian 1 untuk proses *download* data

Jumlah *Record*	**Rata-rata Waktu Tanggap (detik)**			**Jumlah *Record***	**Rata-rata Waktu Tanggap (detik)**		
	Dengan Enkripsi	**Tanpa Enkripsi**	**Selisih Waktu**		**Dengan Enkripsi**	**Tanpa Enkripsi**	**Selisih Waktu**
100	0,3	0,1	0,2	1000	3,1	1	2
200	0,7	0,1	0,6	2000	6	2	4
300	1	0,3	0,7	3000	7	3	4
400	1,1	0,3	0,8	4000	10,2	3,4	6,8
500	1,5	0,4	1,1	5000	12	4	8
600	2	0,5	1,5	6000	15,2	5	10,2
700	2,2	0,9	1,3	7000	20,1	7	13,1
800	2,5	1	1,5	8000	22,9	7,2	15,7
900	3	1	2	9000	26,2	8	18,2
1000	3,1	1	2,1	10000	29,8	10	19,8

Berikut grafik dari data tersebut.



Gambar 4.13 Gambar grafik pengujian skenario 1 untuk proses *download* data

Dari tabel tersebut diperlihatkan bagaimana untuk proses *download* dari 100 hingga 1000 *record* maksimum perbedaan waktu tanggap jika menggunakan teknik enkripsi yang dikembangkan yaitu sebesar 2,1 detik. Sedangkan untuk jumlah *record* yang semakin tinggi selisih waktu akan semakin membesar, dan mulai dari jumlah *record* melebihi 6000 selisih waktu melebihi 10 detik dan selisih tertinggi ditunjukkan pada saat jumlah pengujian *record* tertinggi yaitu sebesar 19,8 detik.

Pada aplikasi ini seperti yang dijelaskan pada bab 3, untuk proses pengolahan data khususnya pada proses *download* ini terdapat 15 *field* yang dikirimkan dari *server* ke *database*. Pada pengolahan *database* ini 11 *field* yang dienkripsi oleh karena itu pada aplikasi akan melakukan proses dekripsi sebanyak 11 kali untuk pengolahan 1 *record*. Hal ini mengakibatkan waktu proses semakin lama seiring dengan peningkatan jumlah *record* yang di di*download*. Pada pengujian ini

perbedaan waktu kurang dari 10 detik terjadi saat jumlah *record* yang diolah kurang dari 6000 *record*.

Yang menjadi perhatian lain adalah mengapa proses *download* lebih cepat dibandingkan proses *upload*. Secara sederhana hal ini terjadi karena pada proses *upload* terjadi mekanisme penyimpanan data ke memori dan mekanisme penyimpanan ke memori akan membutuhkan waktu lebih lama dibandingkan waktu untuk hanya mengambil data tersebut. Selain itu seluruh proses pencarian menggunakan SQL (*structured query language*) dan proses pencarian menggunakan metode *indexing* yang akan membuat proses pencarian lebih cepat.

Selanjutnya akan diuji fenomena ini pada jaringan uji dan bagaimana pengaruh jaringan uji terhadap kinerja proses enkripsi dan dekripsi.

4.2.2 Analisis Skenario 2

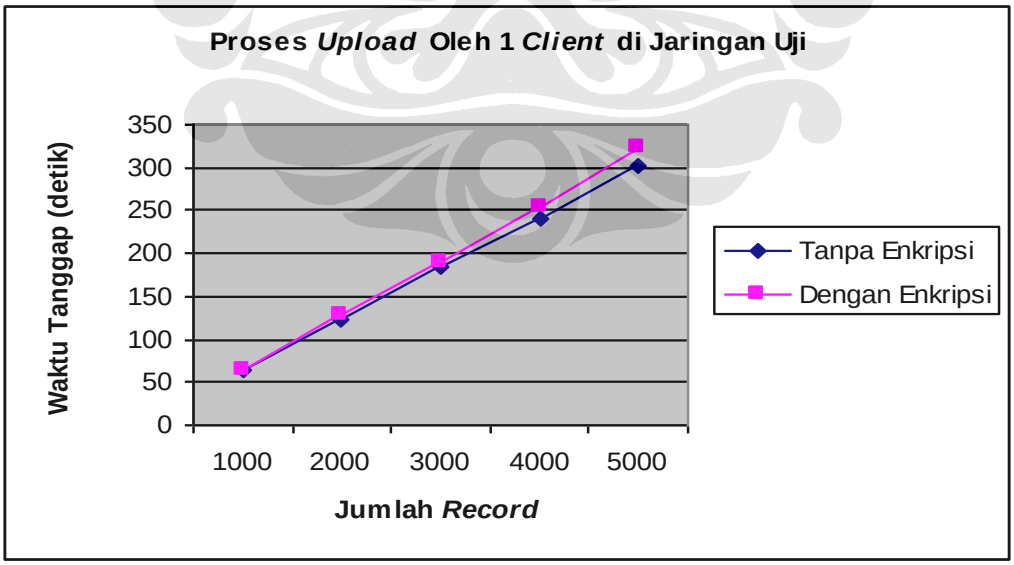
4.2.2.1 Analisis *upload* oleh 1 *client* pada jaringan uji

Pada pengujian kedua ini dilakukan transaksi pada *database* dengan aplikasi pada komputer *client* dan *database* terletak pada dengan komputer (*server*) yang dipisahkan oleh sebuah *switch*. Dan berikut data hasil pengujian tersebut.

Tabel 4.3 Tabel data skenario pengujian 2 untuk proses *upload*

Jumlah *Record*	**Rata-rata Waktu Tanggap (detik)**			**Persentase peningkatan waktu Tanggap (%)**
	Dengan Enkripsi	**Tanpa Enkripsi**	**Selisih Waktu**	
1000	64,6	62,5	2,1	3,36
2000	128	124	4	3,22
3000	188,4	183,7	4,7	2,56
4000	252,5	242	10,5	4,34
5000	323	302	21	6,95
Rata-rata			8,46	4,08

Berikut grafik dari data tersebut



Gambar 4.14 Gambar grafik pengujian skenario 2 untuk proses *upload* data

Dari tabel data dan grafik tersebut dapat diambil kesimpulan bahwa juga terjadi peningkatan waktu tanggap seiring dengan peningkatan jumlah data yang diolah. Dari tabel tersebut juga diperlihatkan bagaimana untuk proses *upload* dari 1000 hingga 5000 *record*. Perbedaan waktu tanggap melebihi 10 detik terjadi saat jumlah *record* kurang dari 4000 *record* dan perbedaan waktu tanggap maksimum untuk pengujian ini yaitu saat jumlah pengujian *record* 5000 yaitu sebesar 21 detik.

Dengan mengamati tabel dan grafik menunjukkan bahwa waktu tanggap untuk melakukan proses enkripsi akan lebih lama, akan tetapi walaupun perbedaan waktu tanggap hingga 21 detik (untuk *upload* 5000 *record*) persentase peningkatan waktu tanggap maksimum hanya 6,95 % dan rata-rata peningkatan waktu tanggap sebesar 8,46 detik (4,08%). Hal ini dikatakan wajar karena adanya mekanisme enkripsi pada setiap *record* yang akan dilakukan pengiriman dan penyimpanan data ke *database*.

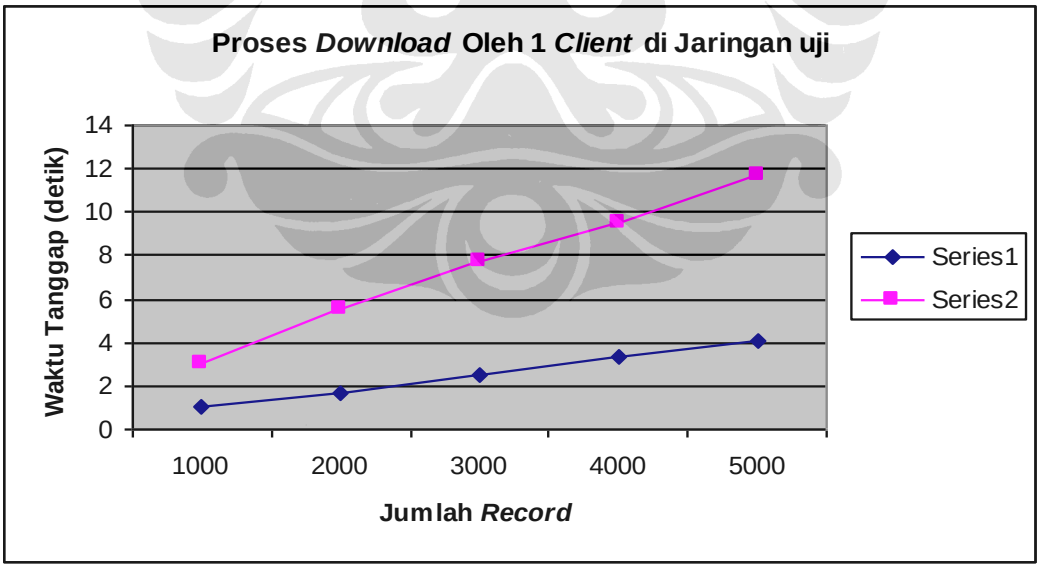
Perbedaan yang mencolok adalah waktu yang dibutuhkan untuk melakukan proses *upload* ke *database* ke komputer terpisah (oleh *switch*) membutuhkan waktu yang lebih lama dibandingkan waktu yang diperlukan untuk *upload* ke *database* pada komputer lokal (skenario 1). Hal ini dapat terjadi akibat lamanya waktu proses penyimpanan di komputer *server* dan diiringi oleh waktu tunda yang dilakukan untuk proses enkripsi data.

4.2.2.1 Analisis *download* oleh 1 *client* pada jaringan uji

Tabel 4.4 Tabel data skenario pengujian 2 untuk proses *download*

Jumlah *Record*	Rata-rata Waktu Tanggap (detik)			Persentase peningkatan waktu tanggap (%)
	Dengan Enkripsi	Tanpa Enkripsi	Selisih Waktu	
1000	3	1	2	200
2000	5,5	1,7	3,8	223,53
3000	7,7	2,5	5,2	208
4000	9,5	3,3	6,2	187,83
5000	11,7	4,1	7,6	185,36
Rata-rata			4,96	200,95

Berikut grafik dari data tersebut :



Gambar 4.15 Gambar grafik pengujian skenario 2 untuk proses *download* data

Dari tabel tersebut diperlihatkan bagaimana untuk proses *download* dari 1000 hingga 5000 *record*, perbedaan waktu tanggap melebihi 10 detik tidak terlihat pada pengujian ini. Akan tetapi yang menjadi kelemahannya ialah persentase peningkatan waktu tanggap rata-rata mencapai 200,95%. Seperti pada saat jumlah proses *download* sebanyak 3000 *record*, perbedaan waktu sebesar 5,2 detik dan merupakan 200,8% dari waktu tanggap untuk aplikasi yang tidak menggunakan enkripsi sama sekali (2,5 detik). Hal ini menunjukkan bahwa pada saat proses *download* perbedaan waktu tanggap tidak terlalu mencolok (tidak melebihi 10 detik) tetapi persentase peningkatannya sangat tinggi hingga persentase tertinggi untuk 2000 data yaitu sebesar 223,53%.

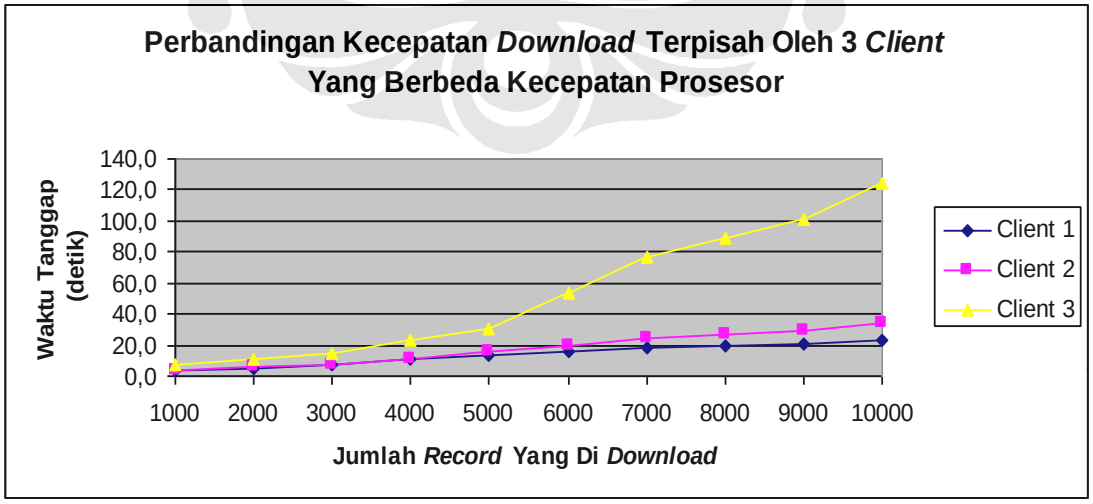
4.2.3 Analisis Skenario 3

Pada pengujian ketiga ini dilakukan transaksi seperti pada pengujian yang kedua, hanya saja jumlah *client* menjadi 3 dan setiap *client* memiliki kecepatan prosesor yang berbeda-beda, dimana kecepatan prosesor *client* 1 adalah 1,6 GHz dualcore, *client* 2 2,4 GHZ dan *client* 3 1,4 GHz.

Tabel 4.5 Tabel data skenario pengujian 3 untuk proses *download*

Jumlah *record*	**Rata-Rata Waktu Tanggap** (detik)								
	Client 1			*Client* 2			*Client* 3		
	Enkripsi	Tanpa	Selisih	Enkripsi	Tanpa	Selisih	Enkripsi	Tanpa	Selisih
1000	3,5	1,1	2,5	3,9	1,2	2,7	7,6	2,0	5,7
2000	5,3	1,6	3,7	5,8	1,7	4,1	11,4	2,9	8,5
3000	7,0	2,1	4,9	7,7	2,3	5,4	15,2	3,9	11,3
4000	10,5	3,2	7,4	11,6	3,5	8,1	22,8	5,9	17,0
5000	14,0	4,2	9,8	15,4	4,6	10,8	30,4	7,8	22,6
6000	16,3	5,4	10,9	20,1	5,8	14,3	53,8	9,9	44,0
7000	18,5	6,6	11,9	24,7	6,9	17,8	77,2	11,9	65,3
8000	19,6	7,2	12,4	27,0	7,5	19,6	88,9	12,9	76,0
9000	20,8	7,8	13,0	29,4	8,1	21,3	100,6	14,0	86,7
10000	23,0	9,0	14,0	34,0	9,2	24,8	124,0	16,0	108,0

Berikut Grafik dari data pada skenario 3 :



Gambar 4.16 Gambar grafik pengujian skenario 3 untuk proses *download* data

Diingatkan kembali bahwa kecepatan prosesor untuk *client* 1 (1,6 GHz dualcore) lebih besar dibandingkan kecepatan prosesor untuk *client* 2 (2,4 GHZ) dan lebih besar dibandingkan prosesor untuk *client* 3 yang hanya 1,4 GHz. Dan pada tabel data dan grafik juga diperlihatkan hal senada. Artinya waktu pengolahan untuk prosesor yang lebih cepat akan membutuhkan waktu untuk proses *download* lebih cepat pula dibandingkan dengan dengan waktu pengolahan oleh prosesor yang lebih lambat.

Teramati bahwa untuk *client* 1 dengan kecepatan prosesor 1,6 GHz (*dualcore*) perbedaan waktu tanggap tertinggi (saat jumlah men-*download* 10.000 *record*) hanya 14 detik (155,6%), untuk *client* 2 dengan kecepatan prosesor 2,4 GHz perbedaan waktu tanggap tertinggi (saat jumlah men-*download* 10.000 *record*) yaitu 24 detik (296,,6%), dan perbedaan waktu tanggap yang tidak lebih dari 10 detik pada *client* 2 terjadi saat *download* kurang dari 5000 *record*. Pada *client* 3 yang lebih lambat perbedaan waktu tanggap kurang dari 10 detik hanya terjadi saat *download* kurang dari 3000 *record*, dan perbedaan waktu tanggap tertinggi (saat jumlah men-*download* 10.000 *record*) yaitu sebesar 108 detik (675%). Selain itu untuk penggunaan prosesor yang lebih tinggi terjadi peningkatan kinerja, misalkan untuk penggunaan prosesor 2,4GHz maka peningkatan kinerja sebesar 378,4% (675% - 296,6%).

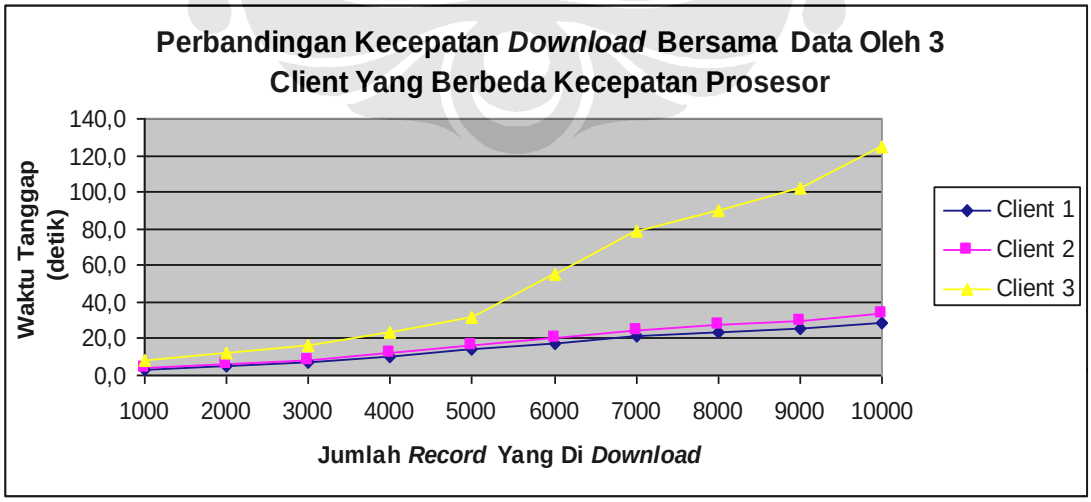
4.2.1 Analisis Skenario 4

Pada pengujian keempat ini dilakukan transaksi seperti pada pengujian yang ketiga, hanya saja 3 *client* mengolah secara bersamaan.

Tabel 4.6 Tabel data skenario pengujian 4 untuk proses *download*

Jumlah *record*	Rata-Rata *Waktu Tanggap* (detik)								
	Client 1			*Client* 2			*Client* 3		
	Enkripsi	Tanpa	Selisih	Enkripsi	Tanpa	Selisih	Enkripsi	Tanpa	Selisih
1000	3,4	1,2	2,2	4,0	1,3	2,8	2,0	8,0	6,0
2000	5,2	1,7	3,5	6,0	1,9	4,1	3,0	12,0	9,0
3000	7,0	2,3	4,7	8,0	2,5	5,5	4,0	16,0	12,0
4000	10,3	3,7	6,7	12,0	3,8	8,3	6,0	24,0	18,0
5000	14	5	9,0	16	5	11,0	8	32	24,0
6000	17,8	6,0	11,8	20,5	6,6	13,9	10,0	55,3	45,3
7000	21,5	7,0	14,5	25,0	7,2	17,8	12,0	78,5	66,5
8000	23,4	7,5	15,9	27,3	7,7	19,6	13,0	90,1	77,1
9000	25,3	8,0	17,3	29,5	8,1	21,4	14,0	101,8	87,8
10000	29	9	20,0	34	9,3	24,7	16	125	109,0

Berikut Grafik dari data pada skenario 4 :



Gambar 4.17 Gambar grafik pengujian skenario 4 untuk proses *download* data

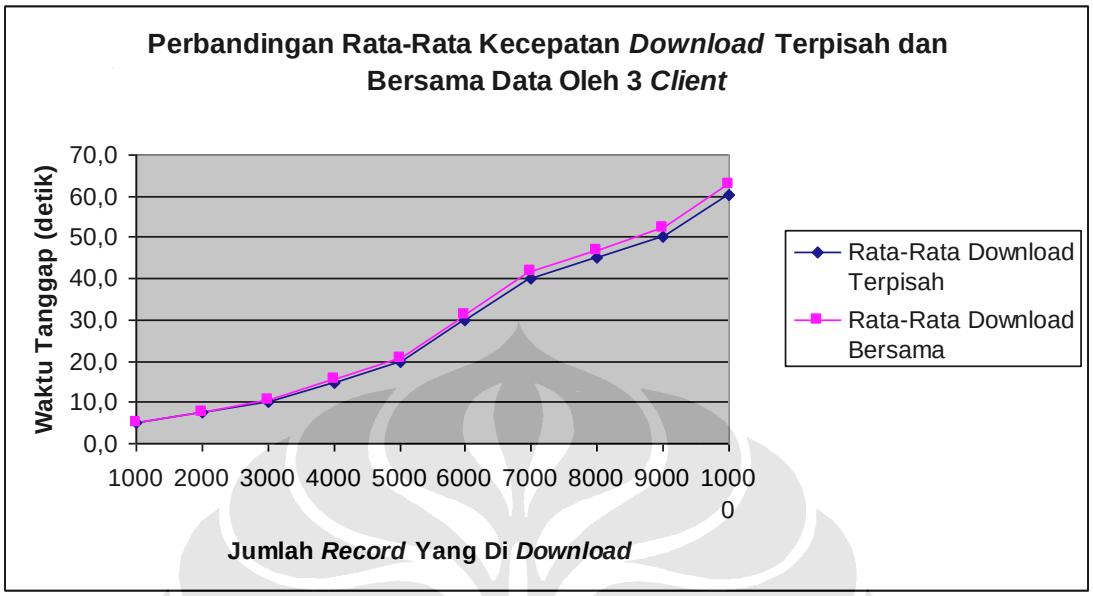
Kesimpulan yang senada diberikan pada skenario 4 ini dimana, kecepatan prosesor yang lebih tinggi akan membuat kecepatan olah dan perbedaan waktu tanggap terhadap aplikasi tanpa enkripsi semakin kecil. *Client* 3 menunjukkan waktu tanggap yang lebih tinggi dari yang lainnya, dan *client* 2 menunjukkan waktu tanggap yang lebih tinggi dari *client* 1 sesuai dengan kecepatan prosesornya.

Dari hasil pengujian pada skenario 3 dan 4 akan diperoleh kesimpulan sebagai berikut dengan mengambil nilai rata-rata waktu tanggap 3 *client* yaitu sebagai berikut:

Tabel 4.7 Tabel rata-rata data skenario pengujian 3 dan 4

Jumlah *record*	Waktu Tanggap (detik)			**Persentase peningkatan waktu tanggap**
	Rata-Rata 3 *client* (terpisah)	Rata-Rata 3 *client* (bersamaan)	Selisih Waktu Tanggap	
1000	5,0	5,1	0,1	2,0
2000	7,5	7,7	0,2	2,7
3000	10,0	10,3	0,3	3,0
4000	15,0	15,4	0,4	2,7
5000	19,9	20,7	0,8	4,0
6000	30,0	31,2	1,2	4,0
7000	40,1	41,7	1,7	3,9
8000	45,2	46,9	1,7	3,8
9000	50,2	52,2	2,2	3,9
10000	60,3	62,7	2,7	3,9

Berikut grafik hasil pengamatan berdasarkan skenario 3 dan 4



Gambar 4.18 Gambar grafik gabungan pengujian skenario 3 dan 4 untuk proses *download* data

Teramati bahwa waktu proses untuk melakukan proses *download* jika dilakukan secara bersamaan akan membutuhkan total waktu tanggap yang lebih lama dibandingkan dengan proses *download* secara terpisah. Hal ini wajar karena proses *download* bersamaan membutuhkan waktu untuk *server* untuk membagi tugasnya kepada setiap *client*. Hanya saja yang menarik peningkatan karena proses *download* bersama untuk pengujian 3 *client* secara bersamaan tidak terlalu tinggi. Hal ini dapat saja terjadi karena secara bersamaan proses enkripsi dan proses dekripsi tetap dilakukan pada aplikasi, sehingga tidak akan lebih memberatkan *server* dan jaringan. Dibandingkan dengan aplikasi tanpa enkripsi maka aplikasi dengan teknik enkripsi sama-sama tidak memberatkan *server* karena pengaruh pemrosesan (enkripsi-dekripsi) yang dilakukan oleh *client*, justru *server* harus hanya menunggu *client* untuk meminta data berikutnya.

Pada pengujian ini hingga 3 *client* aplikasi dengan teknik enkripsi akan memberikan penurunan kinerja yang rendah saat *client* melakukan proses *download* secara bersamaan dibandingkan saat *client* melakukan proses *download* secara

terpisah yaitu hingga *download* 10.000 *record* secara bersamaan perbedaaan waktu dengan secara terpisah hanya 2,7 detik (3,9%).

BAB V

KESIMPULAN

Berdasarkan hasil pengujian dan analisa yang telah dilakukan maka dapat diambil beberapa kesimpulan yang diantaranya ialah:

1. Keamanan data pada aplikasi yang menggunakan teknik enkripsi untuk melakukan pengolahan dan transaksi dengan *database* akan dapat lebih terjaga karena data terenkripsi tidak dapat dibaca kecuali dengan menemukan kunci enkripsinya.

2. Proses enkripsi dan dekripsi dapat meningkatkan waktu tanggap pada pengolahan *upload* dan *download* data. Rata-rata peningkatan waktu tanggap pada proses *upload* dijaringan uji sebesar 8,46 detik atau 4,08% dan rata-rata peningkatan waktu tanggap pada proses *download* sebesar 4,96 detik atau 200,95%.

3. Pengaruh enkripsi lebih terasa pada saat proses *download,* hal ini terlihat dari rata-rata persentase peningkatan waktu tanggap yang lebih tinggi dibandingkan proses *upload* yaitu pada proses *upload* sebesar 4,08% sedangkan pada proses *download* mencapai 200,95%.

4. Kecepatan prosesor mempengaruhi kinerja dari penggunaan teknik enkripsi dan dekripsi. Kecepatan prosesor yang lebih tinggi menunjukkan waktu kinerja aplikasi yang menggunakan teknik enkripsi menjadi lebih baik. Perbedaan waktu tanggap tertinggi antara aplikasi yang tidak menggunakan teknik enkripsi dan aplikasi yang menggunakan teknik enkripsi pada proses *download* 10.000 *record* oleh *client* dengan kecepatan prosesor 1,4 GHz sebesar 108 detik (675%). Pada prosesor yang lebih tinggi yaitu untuk

prosesor 2,4 Ghz *dual core* perbedaan waktu tanggap menjadi 24,8 detik (296,6%) artinya terjadi peningkatan kinerja sebesar 378,4%.

5. Pengolahan *database* secara simultan masih memiliki kinerja yang baik, dilihat dari proses *download* yang dilakukan secara bersamaan oleh 3 *client*. Perbedaan waktu tanggap pada aplikasi yang menggunakan teknik enkripsi secara terpisah dan bersamaan untuk *download* hingga 10.000 *record* adalah sebesar 2,1 detik atau peningkatan waktu tanggap sebesar 3,9%. Hal ini terjadi karena proses enkripsi dan dekripsi dilakukan pada sisi *client* dan ukuran data *plaintext* sama dan ukuran data *ciphertext* dengan demikian tidak memberatkan *server* dan jaringan.

DAFTAR ACUAN

[1] Sarosa,Moechammad.,Anggori,Sigit.,"Jaringan Komputer,Data Link, Network & Issue". 2000

[2] S. Tanenbaum, Andrew., "*Computer* Networks,Fourth Edition". Bab 1.4 , Prentice Hall,2003

[3] Sobari,Rahmat., Andoyo,Yoyok., Juliasari,Noni.., Maulana,Galuh Dian., "Pengamanan Data Sistem Biling Warnet dengan Menggunakan RC4 *Stream* Cipher". Tesis. Jakarta. 2005

[4] _____.,"Definitions & Terminology - Including Pertinent Links". www.angelfire.com/pa/baconbacon/page4.html diakses tanggal 1 Juli 2008

[5] _____.,"Glosary - IT Knowledge". http://www.mysouthwest.com.au/Business/SmallBusinessIT/Glossary?printversion=true diakses tanggal 1 Juli 2008

DAFTAR PUSTAKA

Balena,Francesco., " *Programming Microsoft Visual Basic* 6.0". Visual Basic Programmer's Journal. 1999.

Fogie,Seth., Perikari,Cyrus., "Maximum Wireless Security". Sam Publishing. 2002

Haribowo,Yudi., "Pengamanan Situs dengan Enkripsi Head dan Body HTML Menggunakan Algoritma RC4". Skripsi. 2008

Helmi,Sofwan., "Seni Kriptografi dan Algoritma Enkripsi". http://helmishare.blogspot.com/2007/12/seni-kriptografi-dan-algoritma-enkripsi.html , diakses tanggal 1 Juli 2008

Perera,David., "*Security is job* 1 – *Data breaches force agencies to consider securitya moving target*". *Federal Computer Week*.26 November 2007.21, 38.pg S4-S7. ProQuest Computing

Salomon,David.,"Foundations of *Computer* Security". Springer. 2006

Sarosa,Moechammad.,Anggori,Sigit.,"Jaringan Komputer,Data Link, Network & Issue".2000

Savill,John.,Robichaux Paul.,"Exchange Ideas-Tips, News, and community resource for messaging admins". Exchange & Outlook Administrator.Jul 2006. 9, 7. ProQuest Computing. pg. 13

Sawyer, Dan., "Automate Your Security Audits-Storedprocedure tracks is commonly monitored checkpoints". Artikel. SQL *Server Magazine*. January 2007

Sobari,Rahmat., Andoyo,Yoyok., Juliasari,Noni., Maulana,Galuh Dian., "Pengamanan Data Sistem Biling Warnet dengan Menggunakan RC4 *Stream Cipher*". Tesis. Jakarta. 2005

S. Tanenbaum, Andrew., "*Computer Networks*". *Fourth Edition* Prentice Hall, 2003

Sukmawan,Budi.," Keamanan Data Dan Metode Enkripsi". http://students.ukdw.ac.id/~22033120/enskripsi.html diakses tanggal 1 Juli 2008

Sukmawan,Budi.,"RC4 ".http://www.bimacipta.com/rc4.htm diakses tanggal 28 Juni 2008

Sukmawan,Budi.,"Keamanan Data dan Metode Sukmawan,Budi., "Enkripsi". http://www.pusatartikel.com/article/komputer/hacking/keamanan-data-dan-metode-enskripsi.html diakses tanggal 1 Juli 2008

Natalia., Yessi., “Konsep, Perancangan Dan Analisa Perbandingan Performa Aplikasi *Video Streaming* Pada *Single Hop* dan *Multi Hop AD Hoc Network*”. Skripsi,hal 19. Juni 2005.

_____.,”Algoritma Kriptografi Modern”. http://www.mann.web.id/?pilih=news&aksi=lihat&id=8 diakses tanggal 1 Juli 2008

_____.,“Enkripsi”. http://id.wikipedia.org/wiki/Enkripsi, diakses tanggal 1 Juli 2008.

_____.,”FunctionX Tutorial : SQL Tutorials”. www.functionx.com/sql, diakses tanggal 28 Juni 2008.

_____.,“FunctionX Tutorial : VB Tutorials”. www.functionx.com/vb6 , diakses tanggal 28 Juni 2008.

_____.,”GLOSSARY / Q & A”.www.mobidick.biz/glossary.html diakses tanggal 1 Juli 2008

_____.,”RC4”. http://en.wikipedia.org/wiki/RC4 , diakses tanggal 1 Juli 2008

_____.,”SQL Server Magazine – The Smart Guide to Building World-Class Applications”.www.sqlmag.com diakses tanggal 1 Juli 2008

Lampiran-1: Transkrip Data Primer Skenario 1

***Upload* Tanpa Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*									
	100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
1	1	1	1	1	2	3	3	3	4	4
2	1	1	1	1	2	3	3	3	4	4
3	1	1	1	1	2	3	3	3	4	4
4	1	1	1	1	2	2	3	3	3	4
5	1	1	1	1	2	3	3	3	4	4
6	1	1	1	1	2	3	3	3	4	4
7	1	1	1	1	2	2	2	3	4	5
8	1	1	1	1	2	2	3	3	3	4
9	1	1	1	1	2	3	3	3	4	4
10	1	1	1	1	2	3	3	3	4	4
Rata-Rata	**1**	**1**	**1**	**1**	**2**	**2,7**	**2,9**	**3**	**3,8**	**4,1**

No Urut Pengambilan Data	Jumlah *Record*									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
1	4	8	11	16	19	26	30	33	39	42
2	4	8	11	16	19	26	30	33	39	42
3	4	8	11	16	19	27	30	33	39	42
4	4	7	11	17	19	26	30	33	39	43
5	4	8	12	16	18	26	30	33	41	43
6	4	8	11	16	19	26	33	33	41	43
7	5	8	11	16	20	26	31	33	42	42
8	4	8	11	16	19	28	32	33	40	42
9	4	8	11	16	19	26	32	33	39	43
10	4	8	11	16	19	26	30	33	39	42
Rata-Rata	**4,1**	**7,9**	**11,1**	**16,1**	**19**	**26,3**	**30,8**	**33**	**39,8**	**42,4**

***Upload* Dengan Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*									
	100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
1	1	1	2	3	3	4	5	5	6	7
2	1	1	2	3	3	4	5	6	6	7
3	1	1	2	3	3	4	5	6	6	7
4	1	1	2	3	3	4	5	6	6	7
5	1	1	2	3	3	4	5	6	6	7
6	1	1	2	3	3	4	5	6	6	7
7	1	1	2	3	3	4	5	6	6	7
8	1	1	2	3	3	4	5	6	6	7
9	1	1	2	3	3	4	5	6	6	7
10	1	1	2	3	3	4	5	5	6	7
Rata-Rata	**1**	**1**	**2**	**3**	**3**	**4**	**5**	**5,8**	**6**	**7**

No Urut Pengambilan Data	Jumlah *Record*									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
1	7	12	19	24	30	41	47	54	60	67
2	7	13	19	24	30	41	47	54	60	67
3	7	13	19	24	30	41	47	54	60	67
4	7	13	19	24	32	41	47	54	60	67
5	7	13	19	24	30	41	49	54	60	67
6	7	13	19	24	32	41	48	54	60	67
7	7	13	19	24	30	42	48	54	60	68
8	7	13	21	24	30	41	49	54	60	67
9	7	14	19	24	30	42	47	54	60	67
10	7	13	19	24	30	41	47	54	60	67
Rata-Rata	**7**	**12,9**	**19,2**	**24**	**30,4**	**41,2**	**47,6**	**54**	**60**	**67**

***Download* Tanpa Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*									
	100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
1	0	0	0	0	0	0	1	1	1	1
2	0	0	1	1	0	0	1	1	1	1
3	0	0	0	0	1	1	1	1	1	1
4	0	0	1	0	1	0	1	1	1	1
5	0	0	0	1	0	1	1	1	1	1
6	0	0	0	0	1	0	0	1	1	1
7	0	0	0	0	1	1	1	1	1	1
8	1	1	1	1	0	1	1	1	1	1
9	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	1	1	1	1
Rata-Rata	**0,1**	**0,1**	**0,3**	**0,3**	**0,4**	**0,5**	**0,9**	**1**	**1**	**1**

No Urut Pengambilan Data	Jumlah *Record*									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
1	1	2	3	3	4	5	7	7	8	10
2	1	2	3	3	4	5	7	7	8	10
3	1	2	3	3	4	5	7	7	8	10
4	1	2	3	3	4	5	7	7	8	10
5	1	2	3	3	4	5	7	7	8	10
6	1	2	3	3	4	5	7	7	8	10
7	1	2	3	4	4	5	7	7	8	10
8	1	2	3	4	4	5	7	7	8	10
9	1	2	3	4	4	5	7	8	8	10
10	1	2	3	4	4	5	7	8	8	10
Rata-Rata	**1**	**2**	**3**	**3,4**	**4**	**5**	**7**	**7,2**	**8**	**10**

***Download* Dengan Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*									
	100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
1	0	1	1	1	1	2	2	3	3	3
2	1	1	1	1	2	2	2	3	3	3
3	0	0	1	1	2	2	2	2	3	3
4	0	1	1	1	1	2	2	2	3	3
5	0	1	1	1	2	2	2	2	3	3
6	1	0	1	1	2	2	2	3	3	4
7	0	1	1	1	2	2	2	3	3	3
8	0	1	1	1	1	2	2	3	3	3
9	1	1	1	1	1	2	3	2	3	3
10	0	0	1	2	1	2	3	2	3	2
Rata-Rata	**0,3**	**0,7**	**1**	**1,1**	**1,5**	**2**	**2,2**	**2,5**	**3**	**3,1**

No Urut Pengambilan Data	Jumlah *Record*									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
1	3	6	7	10	12	15	20	23	26	29
2	3	6	7	10	12	15	20	23	26	29
3	3	6	7	10	12	15	20	23	26	29
4	3	6	7	11	12	15	20	23	26	29
5	2	6	7	10	12	15	20	22	27	29
6	4	6	7	10	12	16	20	23	26	29
7	2	6	7	10	12	16	20	23	26	29
8	3	6	7	11	12	15	21	23	27	31
9	3	6	7	10	12	15	20	23	26	32
10	2	6	7	10	12	15	20	23	26	32
Rata-Rata	**3,1**	**6,2**	**7**	**10,2**	**12**	**15,2**	**20,1**	**22,9**	**26,2**	**29,8**

Lampiran-2: Transkrip Data Primer Skenario 2

***Upload* Dengan Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*				
	1000	**2000**	**3000**	**4000**	**5000**
1	64	128	191	251	323
2	64	128	189	253	323
3	66	128	188	251	323
4	64	128	188	252	323
5	64	128	188	251	323
6	64	128	188	251	323
7	64	128	188	253	323
8	64	128	188	251	323
9	66	128	188	256	323
10	66	128	188	256	323
Rata-Rata	**64,6**	**128**	**188,4**	**252,5**	**323**

***Upload* Tanpa Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*				
	1000	**2000**	**3000**	**4000**	**5000**
1	62	124	183	245	302
2	62	124	184	239	302
3	63	124	184	242	301
4	63	124	184	244	301
5	63	124	183	243	301
6	62	124	183	243	301
7	62	124	184	239	303
8	63	124	184	240	303
9	62	124	184	242	303
10	63	124	184	243	303
Rata-Rata	**62,5**	**124**	**183,7**	**242**	**302**

***Download* Dengan Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*				
	1000	**2000**	**3000**	**4000**	**5000**
1	3	6	8	10	12
2	3	6	7	9	12
3	3	6	7	10	11
4	3	6	7	9	11
5	3	5	7	9	12
6	3	6	9	9	12
7	3	5	8	11	12
8	3	5	7	9	11
9	3	5	9	9	12
10	3	5	8	10	12
Rata-Rata	**3**	**5,5**	**7,7**	**9,5**	**11,7**

***Download* Tanpa Enkripsi**

No Urut Pengambilan Data	Jumlah *Record*				
	1000	**2000**	**3000**	**4000**	**5000**
1	1	2	2	3	4
2	1	2	2	2	4
3	1	2	2	3	4
4	1	1	3	3	4
5	1	1	3	4	4
6	1	2	3	4	4
7	1	2	3	4	4
8	1	2	3	4	4
9	1	1	2	3	4
10	1	2	2	3	5
Rata-rata	**1**	**1,7**	**2,5**	**3,3**	**4,1**