IMPLEMENTASI SISTEM ENKRIPSI PENGIRIM PESAN INSTAN JAVA DENGAN ALGORITMA *BLOWFISH*

Mohammad Gilang Kautzar HW – NIM : 13505101

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15101@students.if.itb.ac.id

Abstrak

Pengiriman pesan instan adalah suatu metode komunikasi yang bersifat real-time. Pengiriman pesan pada protokol YMSG yang digunakan oleh Yahoo! Messenger dilakukan dengan melakukan transaksi paket antara client dengan server. Pesan yang dienkapsulasikan ke dalam paket-paket tersebut tidak mengalami enkripsi, sehingga teks pesan dapat dibaca secara langsung. Untuk meningkatkan keamanan pesan, maka diimplementasikan suatu sistem enkripsi pada client (pengirim pesan instan).

Algoritma enkripsi yang digunakan pada perangkat lunak adalah Blowfish. Hal ini disesuaikan dengan sifat Blowfish yang tidak banyak memakan resource dalam mengenkripsi. Perangkat lunak dikembangkan dalam bahasa Java menggunakan API jYMSG, hal ini ditujukan agar perangkat lunak bersifat platform independent.

Pembangunan perangkat lunak dilakukan dalam beberapa tahapan, yaitu tinjauan pustaka, analisis masalah dan perangkat lunak, perancangan, implementasi, dan pengujian. Pengujian perangkat lunak dilakukan terhadap performa, proses enkripsi dan dekripsi, dan keamanan penyadapan. Hasil pengujian menunjukkan bahwa sistem enkripsi dapat digunakan tanpa menurunkan performa perangkat lunak secara signifikan. Dengan demikian dapat disimpulkan bahwa sistem enkripsi pada perangkat lunak dapat meningkatkan keamanan pesan dari percobaan penyadapan tanpa mengorbankan performa.

***Kata kunci**: pengirim pesan instan, Yahoo! Messenger, jYMSG, sistem enkripsi, algoritma Blowfish.*

1. Pendahuluan

Pengirim pesan instan (*instant messenger*) merupakan suatu perangkat lunak yang memfasilitasi pengiriman pesan singkat (*instant messaging*), suatu bentuk komunikasi secara langsung antara dua pihak atau lebih menggunakan teks yang diketik (*chatting*). Pengirim pesan instan akan mengirimkan teks melalui perangkat yang terhubung dengan suatu jaringan. Penggunaan teknologi ini memiliki suatu kelebihan dibandingkan surat elektronik (*e-mail*), yaitu komunikasi dapat terjalin secara langsung atau *real-time*. Hal tersebut merupakan salah satu penyebab pertumbuhan yang pesat pada jumlah penggunaan pengirim pesan instan untuk berkomunikasi.

Saat ini terdapat berbagai servis protokol yang mendukung penggunaan pengiriman pesan singkat, misalnya YMSG (*Yahoo! Messenger*), MSNP (*Windows Live Messenger*), dan IRC. Jumlah penggunaan protokol YMSG sebagai sarana pengiriman pesan singkat yang meningkat telah menimbulkan kekhawatiran mengenai keamanannya. Pada protokol YMSG, teks pesan yang dikirim melalui pengirim pesan instan dapat disadap dengan mudah karena tidak melalui proses enkripsi dalam perjalanannya [HUN03]. Oleh karena itu, suatu proses enkripsi diperlukan untuk mengamankan pesan teks yang dikirimkan.

Blowfish adalah sebuah algoritma enkripsi kunci simetris dengan panjang blok 64 bit yang dikembangkan untuk menggantikan DES [SCH94]. Ukuran kunci pada algoritma *Blowfish* berkisar dari 32 bit sampai 448 bit. Algoritma *Blowfish* memanfaatkan teknik pemanipulasian bit dan teknik pemutaran ulang dan pergiliran

kunci yang dilakukan sebanyak 16 kali. Algoritma utama terbagi menjadi dua upa-algoritma utama, yaitu bagian ekspansi kunci dan bagian enkripsi-dekripsi data.

Ekspansi kunci dilakukan dengan menerima masukan sebuah kunci. Keluaran yang dihasilkan adalah sebuah larik upa-kunci dengan total 4168 *byte*. Bagian enkripsi-dekripsi data dilakukan dengan memanfaatkan perulangan 16 kali jaringan Feistel. Setiap perulangan terdiri dari permutasi dengan masukan kunci, dan substitusi data.

Algoritma *Blowfish* masih tergolong aman karena sampai saat ini belum ada metode kriptanalisis yang lebih efisien yang dapat digunakan untuk menyerang 16 putaran penuh algoritma *Blowfish* selain serangan *brute-force*. Sejauh ini tidak ada kelemahan yang berarti dari algoritma kecuali adanya kunci lemah (*weak key*), dimana dua entri dari kotak-S (*S-box*) mempunyai nilai yang sama [VAU95].

Algoritma *Blowfish* merupakan salah satu metode enkripsi blok (*block cipher*) yang tercepat [GAT03], kecuali apabila terjadi penggantian kunci, di mana setiap penggantian kunci memerlukan proses yang setara dengan enkripsi teks berukuran 4 *kilobytes*. Implementasi *Blowfish* tidak menggunakan *resource* memori banyak, hal ini menyebabkan algoritma tersebut banyak digunakan pada *embedded system*. Algoritma *Blowfish* juga merupakan metode enkripsi yang bebas paten. Oleh karena alasan-alasan tersebut algoritma *Blowfish* dapat dimplementasikan pada perangkat lunak pengirim pesan instan.

2. Rumusan dan Batasan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang dikaji dalam tugas akhir ini adalah:

1. Bagaimana membangun perangkat lunak pengirim pesan instan berprotokol YMSG dalam bahasa Java menggunakan API jYMSG.
2. Bagaimana mengimplementasikan algoritma *Blowfish* pada perangkat lunak pengirim pesan instan untuk mengenkripsi pesan yang dikirim.
3. Bagaimana menguji keamanan sistem enkripsi yang telah diimplementasikan pada perangkat lunak.

Ruang lingkup dan batasan masalah yang dikaji dalam tugas akhir ini adalah:

1. Dari seluruh fitur-fitur yang disediakan pada protokol YMSG, fitur yang akan diimplementasikan pada perangkat lunak hanya fitur *instant messaging* (pengiriman antara dua *user*).
2. Perangkat lunak tidak melakukan penanganan manajemen kunci berupa transmisi kunci.

3. Analisis Permasalahan

Permasalahan utama yang ada pada perangkat lunak adalah mengenai enkripsi dan dekripsi. Bagaimana melakukan enkripsi terhadap sebuah pesan teks sebelum dienkapsulasikan ke dalam paket YMSG dan dikirim ke *server* dan dekripsi terhadap paket dari *server* yang berisi *ciphertext* sehingga dapat ditampilkan ke *user* dalam *plaintext*. Bagian berikut ini berisi penjabaran penyelesaian masalahnya.

3. 1. Enkripsi

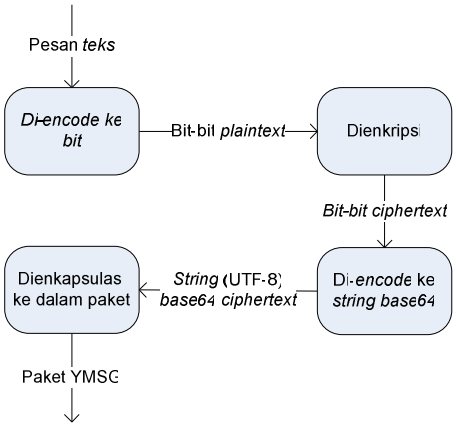
Enkripsi dilakukan sebelum pesan dikirim ke *server* untuk kemudian dilanjutkan ke *client*. Sehingga ketika pesan sedang dikirimkan melalui protokol YMSG, pesan tersebut sudah berupa karakter-karakter (*bytes*) acak. Dalam proses pengiriman pesan, teks dienkapsulasikan dalam sebuah paket yang kemudian dikirimkan melalui protokol YMSG.

Pada perangkat lunak pengirim pesan instan terdapat sebuah kelas yang bertugas menyusun paket-paket tersebut. Kelas tersebut memasukkan teks pesan yang akan dikirim ke dalam *body* paket. Agar pesan yang sedang dikirim sudah teracak dengan aman, maka sebelum teks dimasukkan ke dalam *body*, teks tersebut harus terlebih dahulu diproses oleh kelas lain yang mengimplementasikan *Blowfish*. Setelah mengalami proses enkripsi oleh method *encryptor* yang ada pada kelas tersebut, baru kemudian teks tersebut diteruskan ke kelas yang akan memasukkannya ke dalam paket yang selanjutnya dikirim ke *server*.

Pada *Yahoo! Messenger*, teks pesan yang diketik direpresentasikan dalam karakter UTF-8 [YAH07]. Karena algoritma *Blowfish* adalah metode enkripsi yang bekerja dalam bit-bit, maka karakter-karakter *string* UTF-8 tersebut perlu diubah ke dalam bit. Setelah pengacakan dilakukan, keluaran yang masih dalam bentuk bit

tersebut perlu diubah ke dalam *string* kembali. Hal ini disebabkan oleh struktur data pada paket YMSG yang hanya dapat mengirimkan teks yang direpresentasikan dalam UTF-8.

Namun muncul permasalahan lain, tidak ada jaminan bahwa seluruh keluaran bit-bit hasil pengacakan dapat dipetakan ke dalam karakter-karakter UTF-8. Ada kemungkinan bahwa bit-bit *ciphertext* yang dikeluarkan merupakan bit-bit yang *invalid* dalam UTF-8. Hal ini dapat menyebabkan ada data (bit-bit) yang hilang bahkan sebelum pesan dikirimkan. Oleh karena itu, agar bit-bit tersebut tetap dapat dipetakan dalam karakter-karakter UTF-8, maka bit-bit tersebut akan di-*encode* ke dalam format *base64*. Dengan demikian, teks (*string base64*) tersebut dapat dienkapsulasikan ke dalam *body* suatu paket YMSG dan kemudian dikirimkan tanpa ada masalah. Tabel skema proses enkripsi ini dapat dilihat pada gambar 2-1.

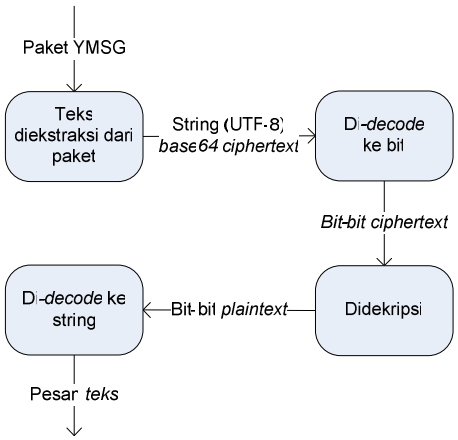


Gambar 2-1 Proses enkripsi dan pengiriman pesan dari *client* ke *server*

3. 2. Dekripsi

Pada algoritma *Blowfish*, operasi-operasi yang dilakukan pada proses dekripsi sama dengan pada proses enkripsi dengan pengecualian urutan larik-P digunakan secara terbalik dari belakang.

Untuk mendapatkan teks pesan, paket yang diterima oleh *client* dari *server* harus diekstraksi terlebih dahulu. Hasil ekstraksi yang dilakukan oleh kelas pembangun paket adalah *string base64 ciphertext* yang perlu didekripsi. Oleh karena itu, *string base64* tersebut terlebih dahulu di-*decode* ke dalam bit-bit *ciphertext*, kemudian bit-bit itu didekripsi oleh *method decryptor* yang ada pada kelas implementasi *Blowfish*. Keluaran dari dekripsi tadi adalah bit-bit yang dapat diubah ke dalam bentuk karakter-karakter UTF-8. Karakter-karakter inilah yang selanjutnya ditampilkan oleh *client* dan dibaca oleh *user* sebagai teks pesan. Skema proses dekripsi ini dapat dilihat pada gambar 2-2.



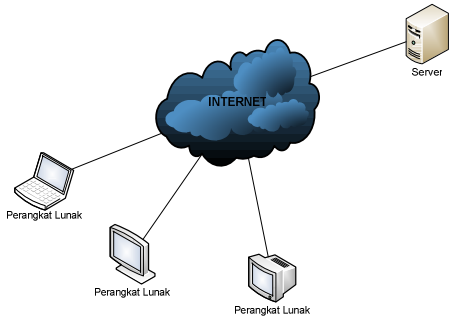
Gambar 2-1 Proses dekripsi dan penerimaan pesan dari *server* ke *client*

4. Deskripsi Umum Perangkat Lunak

Perangkat lunak pengirim yang dibangun adalah sebuah *client* dari *server Yahoo! Messenger*. Dengan kata lain, arsitektur perangkat lunak ini adalah *client-server* di mana komunikasi yang berlangsung antara *client* dan *server* tersebut dilakukan melalui protokol YMSG yang ada pada jaringan internet.

Perangkat lunak adalah sebuah pengirim pesan instan yang mehubungkan seorang *user* dengan *server* yang disediakan oleh *Yahoo!*. Setelah terhubung dengan *server* melalui perangkat lunak yang bertindak sebagai *client*, *user* dapat mengirim pesan teks ke *user* lain yang juga terhubung ke *server* melalui *client*. Apabila kedua *user* menggunakan perangkat lunak yang sama sebagai *client*, maka mode enkripsi dapat digunakan. Mode enkripsi adalah mode di mana seluruh pertukaran pesan yang terjadi melalui proses enkripsi. Dengan kata lain, sebelum pesan teks dikirim oleh seorang *user* ke *user* lain, maka bit-bit teks pesan tersebut dienkripsi terlebih dahulu. Dengan demikian, apabila terjadi penyadapan pesan ketika pesan sedang dikirim, maka yang terbaca hanyalah teks acak. Perangkat lunak ini menggunakan metode enkripsi *Blowfish* untuk mengacak bit-bit teks. Ilustrasi arsitektur perangkat lunak dapat dilihat pada gambar 4-1.

Perangkat lunak ini menggunakan API jYMSG. API ini menyediakan berbagai macam fitur-fitur dan servis *Yahoo! Messenger* yang dapat diimplementasikan dan digunakan oleh *client*. Dengan menggunakan API tersebut, perangkat lunak dapat terhubung dengan *server*.



Gambar 4-1 Arsitektur Perangkat Lunak

5. Implementasi Perangkat Lunak

Perangkat lunak pengirim pesan instan dibangun pada lingkungan perangkat keras dengan spesifikasi sebagai berikut:

1. Prosesor Intel Core 2 Duo 6420 2,13 GHz,
2. Memori 3,25 GB RAM DDR2,
3. 2 buah *hard disk Sea Gate* 250 GB.

Sedangkan spesifikasi perangkat lunak yang digunakan untuk mengembangkan perangkat lunak adalah sebagai berikut:

1. Sistem operasi *Microsoft Windows XP Professional Service Pack*,
2. *Sun Java SDK Standard Edition* 1.6.0,
3. Notepad++ 5.3.1.

Perangkat lunak juga memiliki batasan-batasan implementasi, yaitu:

1. Perangkat lunak hanya mengimplementasikan servis pengiriman pesan (*instant messaging*) tanpa mengimplementasikan fitur-fitur lainnya seperti *conferencing*, *room chatting*, *smilies*, dan lain-lain.
2. Perangkat lunak tidak melakukan tranmisi kunci melalui suatu jalur yang aman (*secure channel*).

6. Pengujian

Pengujian dilakukan pada perangkat lunak dilakukan secara bertahap. Adapun tujuan pengujiannya adalah sebagai berikut:

1. Menguji pengaruh sistem enkripsi terhadap performa dan kinerja perangkat lunak. Performa diukur dari penggunaan waktu dan memori perangkat lunak.
2. Menguji kebenaran proses enkripsi dan dekripsi teks pesan pada perangkat lunak.
3. Menguji apakah proses penyadapan akan menghasilkan teks acak (*ciphertext*) apabila mode enkripsi diaktifkan.

Pengujian dilakukan pada lingkungan yang sama dengan implementasi perangkat lunak.

6.1. Kasus Uji

Pengujian terbagi menjadi tiga tahap, yaitu pengujian performa, enkripsi, dan penyadapan. Kasus uji pada pengujian performa dilakukan dengan menganalisis konsumsi waktu dan memori ketika perangkat lunak dijalankan. Untuk menganalisis konsumsi waktu digunakan kode `System.nanoTime()`. Sementara untuk mengetahui penggunaan memori diselipkan kode `Runtime.getRuntime().freeMemory()`. Sementara itu pengujian enkripsi dan dekripsi secara umum dilakukan dengan dua cara, yaitu pengujian pengiriman dan penerimaan teks. Pengujian yang terakhir ditujukan untuk memastikan bahwa sistem enkripsi pada perangkat lunak dapat digunakan untuk mempersulit proses penyadapan. Untuk itu pengujian tersebut dilakukan dengan mencoba menyadap pesan yang dikirim melalui suatu perangkat lunak yang dapat menangkap paket-paket YMSG.

6.2. Pengujian Performa

Pengujian performa dilakukan dengan membandingkan penggunaan waktu dan memori ketika perangkat lunak dijalankan.

6.2.1. Pelaksanaan dan Hasil Pengujian Waktu

Pengujian dilakukan dengan memasukkan fungsi `System.nanoTime()` pada *source code*. Fungsi ini akan menghasilkan angka berisi waktu dalam satuan nanodetik (10^{-9} detik). Berikut ini adalah tabel 6-1 yang berisi hasil pengujian tersebut.

Tabel 6-1 Hasil pengujian penggunaan waktu perangkat lunak

No	Kondisi	Waktu yang digunakan (nanodetik)	Persentase peningkatan
1	Pengiriman pesan tanpa enkripsi	802576	37%
2	Pengiriman pesan dengan enkripsi	1102722	
3	Penerimaan pesan tanpa dekripsi	118310	107%
4	Penerimaan pesan dengan dekripsi	245435	

Dari hasil pengujian dapat terlihat persentase kenaikan waktu yang digunakan ketika mode enkripsi atau dekripsi diaktifkan. Ketika mode enkripsi diaktifkan, maka waktu yang dipergunakan untuk mengirim suatu pesan bertambah sebanyak 37% (300146 nanodetik). Dan ketika perangkat lunak menerima pesan, waktu pemrosesan bertambah sebanyak 107% (127125 nanodetik). Walaupun keberadaan mode enkripsi telah terbukti akan menyebabkan *delay* terhadap performa perangkat lunak, namun hal ini tidak akan terlalu mengganggu karena nilai tersebut tidak terlalu berarti dalam komunikasi melalui *instant messenger*.

6.2.2. Pelaksanaan dan Hasil Pengujian Memori

Pengujian dilakukan dengan menyisipkan fungsi `Runtime.getRuntime().freeMemory()` pada *source code* perangkat lunak. Hasil pengujian dapat dilihat pada tabel 6-2. Pada pengujian di mana enkripsi tidak digunakan, maka paket kelas yang berkaitan dengan enkripsi dihapus dari perangkat lunak.

Tabel 6-2 Hasil pengujian memori perangkat lunak

No	Kondisi	*Free Memory* (*bytes*)	*Total Memory* (*bytes*)	Rasio
1	Setelah mengirim pesan tanpa enkripsi	3723336	51773344	7,1916 %
2	Setelah mengirim pesan dengan enkripsi	3920040	51773344	7,5715 %
3	Setelah menerima pesan tanpa dekripsi	3626872	51773344	7,0053 %
4	Setelah menerima pesan dengan dekripsi	4118304	51773344	7,9545 %

Pada tabel terlihat bahwa, ketika perangkat lunak mengirimkan pesan, maka penggunaan memori lebih besar mode enkripsi diaktifkan. Hal yang sama terjadi ketika perangkat lunak menerima pesan. Penggunaan memori lebih tersebut disebabkan oleh alokasi memori yang digunakan untuk instansiasi objek yang berhubungan dengan kelas enkripsi. Oleh karena itu penggunaan memori menjadi lebih sedikit ketika kelas enkripsi tersebut tidak digunakan pada perangkat lunak. Meskipun penggunaan memori meningkat, namun hal ini tidak mengurangi performa perangkat lunak karena peningkatan rasio masih kurang dari 1%. Dengan demikian dapat disimpulkan bahwa proses enkripsi maupun dekripsi bukanlah proses yang akan menurunkan performa perangkat lunak.

6.3. Pengujian Enkripsi-Dekripsi

Pengujian Enkripsi-Dekripsi terbagi menjadi empat bagian. Masing-masing bagian akan dijelaskan berikut ini.

Pada pengujian pertama dilakukan percobaan dengan mengirimkan pesan antar dua buah *client* perangkat lunak tanpa mode enkripsi. Dengan kata lain, pengujian ini hanya memeriksa apakah pesan dapat terkirim dengan baik. Tabel pengujiannya dapat dilihat pada tabel 6-3.

Tabel 6-3 Pengiriman pesan tanpa mode enkripsi dan dekripsi

No	Pesan Pengirim	Pesan Penerima
1	Ini teks pengujian	Ini teks pengujian

Pada pengujian kedua ini pengujian dilakukan dari perangkat lunak dengan mode enkripsi aktif namun ditujukan ke *client* lain yang tidak memiliki mode dekripsi atau tidak mengaktifkannya. Tabel pengujian terdapat pada tabel 6-4.

Tabel 6-4 Pengiriman pesan tengan enkripsi tetapi tanpa dekripsi

No	Pesan Pengirim	Kunci Enkripsi	Pesan Penerima
1	Ini teks pengujian	kuncirahasia	qef2h1egQ7wxhh+dbv9MBQ79CH8k3zh4

Pengujian berikutnya dilakukan dengan melakukan pengiriman teks pesan dari *client* yang tidak mengaktifkan atau memiliki mode enkripsi namun diterima oleh *client* yang mode enkripsinya aktif. Pesan yang dikirim akan tetap melalui proses dekripsi. Apabila pesan yang dikirim tidak dapat didekripsi, maka akan keluar *input exception*. Namun, apabila pesan yang dikirim dapat didekripsi, maka keluaran yang diterima adalah *string* hasil dekripsinya. Tabel pengujian terdapat pada tabel 6-5.

Tabel 6-5 Pengiriman pesan tanpa enkripsi tetapi dengan dekripsi

No	Pesan Pengirim	Pesan Penerima	Kunci Dekripsi
1	Ini teks pengujian	-	kuncirahasia
2	qef2h1egQ7wxhh+dbv9MBQ79CH8k3zh4	Ini teks pengujian	kuncirahasia

Pengujian pada bagian keempat ini dilakukan dengan mengaktifkan mode enkripsi-dekripsi pada kedua pihak, baik pengirim maupun penerima. Pada pengujian ini juga dilakukan percobaan apabila kunci yang digunakan untuk mengenkripsi dan mendekripsi berbeda. Tabel pengujian dapat dilihat pada tabel 6-6.

Tabel 6-6 Pengiriman pesan dengan mode enkripsi dan dekripsi

No	Pesan Pengirim	Kunci Enkripsi	Pesan Penerima	Kunci Dekripsi
1	Ini teks pengujian	kuncirahasia	Ini teks pengujian	kuncirahasia
2	Ini teks pengujian	kuncirahasia	0V!çDØs-cXÒæÜH]:Œ™□•	kunciberbeda

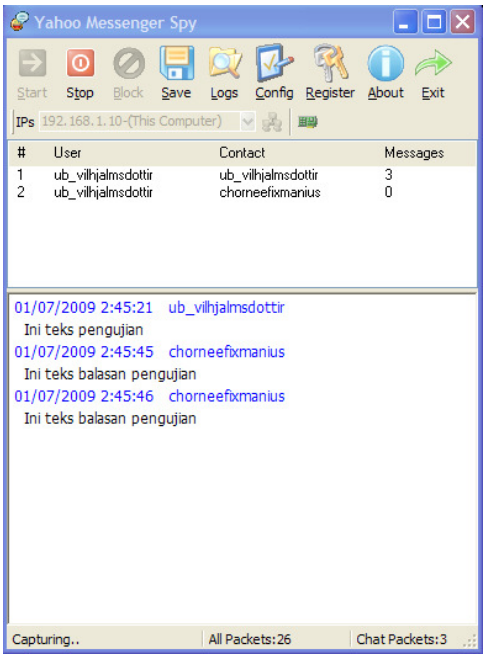
Pada hasil pengujian pengiriman pesan dan penerimaan pesan, terlihat bahwa perangkat lunak dapat dengan berhasil mentransmisikan pesan yang telah dienkapsulasikan ke dalam suatu paket YMSG melalui *server* yang selanjutnya diterima oleh *client* lain. Dapat terlihat pula bahwa mode enkripsi pada perangkat lunak berjalan baik. Ketika pesan yang terenkripsi diterima oleh *client* yang tidak dapat mendekripsi maka isi pesan tidak dapat dimengerti. Namun dengan begitu diasumsikan bahwa apabila terjadi penyadapan, maka pesan yang didapat hanyalah pesan acak yang memerlukan kunci tertentu agar dapat dibaca. Pesan terenkripsi juga tidak dapat dibaca apabila kunci yang digunakan untuk mendekripsi tidak sama dengan yang dipakai untuk mendekripsi. Jadi dapat disimpulkan bahwa pengiriman pesan akan berlangsung dengan baik apabila kedua *client* yang berinteraksi sama-sama mengaktifkan mode enkripsi-dekripsinya dan kunci yang digunakan sama.

6.4. Pengujian Penyadapan

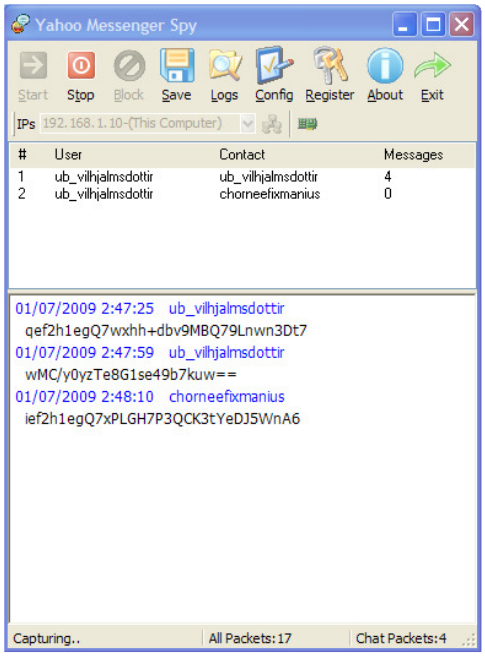
Percobaan penyadapan ini dilakukan dengan menggunakan sebuah perangkat lunak *shareware* yang bernama *IMMonitor Yahoo Messenger Spy* 2.0. Perangkat lunak ini dikembangkan oleh *immonitor* untuk menyadap sebuah percakapan yang dilakukan melalui servis *Yahoo! Messenger*. Penyadapan dapat dilakukan selama *client* yang disadap masih berada dalam satu jaringan LAN dengan penyadap.

Pada pengujian ini, pesan dicoba dikirimkan oleh perangkat lunak pengirim pesan instan ketika aplikasi *IMMonitor* sedang berjalan. Pada percobaan pertama, pengirim pesan instan akan mengirimkan pesan tanpa mode enkripsi. Dengan kata lain, percakapan dapat dibaca dengan mudah oleh penyadap. *Screenshot* dapat dilihat pada gambar 6-1. Pada gambar terlihat

bahwa pesan yang dikirim antara dua *user* dapat dibaca secara langsung.



Gambar 6-1 Penyadapan ketika mode enkripsi tidak aktif



Gambar 6-2 Penyadapan ketika mode enkripsi aktif

Pada proses pengujian berikutnya, sistem enkripsi pada perangkat lunak pengirim pesan instan diaktifkan. Pesan yang sama dicoba dikirimkan namun dengan kali ini dalam kondisi terenkripsi. Ketika pesan sedang ditransmisikan ke *server*, perangkat lunak penyadap tetap dapat menangkap isi teks pesan yang dikirim, namun teks pesan tersebut adalah teks pesan yang telah dienkripsi. *Screenshot* pengujian tersebut dapat dilihat pada gambar 6-2.

Seperti yang telah dilihat pada hasil pengujiannya, ketika mode enkripsi pada perangkat lunak pengirim pesan instan sedang tidak aktif, maka pesan dapat disadap dan dibaca dengan mudah. Namun, apabila mode enkripsi pada perangkat lunak diaktifkan, maka pesan yang ditangkap penyadap hanyalah *ciphertext* yang perlu didekripsi terlebih dahulu untuk dibaca isinya. Dengan demikian dapat dikatakan bahwa mode enkripsi pada perangkat lunak akan mempersulit proses penyadapan yang sebelumnya dapat dilakukan dengan mudah.

7. Kesimpulan

Ada beberapa kesimpulan yang didapatkan sejak pengerjaan tugas akhir sampai selesai. Berikut ini akan dijabarkan masing-masing kesimpulan tersebut:

1. Pengirim pesan instan yang menggunakan servis-servis *Yahoo! Messenger* dapat dibangun dengan menggunakan suatu API yang disebut dengan jYMSG. Pada API tersebut telah tersedia kelas-kelas yang vital untuk mempermudah implementasi protokol YMSG tersebut, misalnya kelas pembangun paket, koneksi, mekanisme *login*, dan lain-lain.
2. Implementasi algoritma enkripsi *Blowfish* dalam pengirim pesan instan akan meningkatkan keamanan yang sebelumnya tidak ada pada proses pengiriman pesan dalam protokol YMSG. Dengan sistem enkripsi tersebut, penyadapan yang dilakukan ketika pesan sedang ditransmisikan menjadi semakin sulit karena pesan yang ditangkap berupa *ciphertext*. Perangkat lunak juga bebas dari resiko serangan *replay attack* dan serangan lainnya karena pada setiap paket YMSG terdapat *session* yang dapat diubah setiap waktu oleh *server*.
3. Implementasi algoritma enkripsi *Blowfish* tidak menurunkan performa perangkat lunak dengan berarti. Hal ini disebabkan pengaruh enkripsi yang tergolong kecil terhadap penggunaan memori dan waktu pemrosesan perangkat lunak. Dengan mengaktifkan mode enkripsi, maka keamanan teks pesan yang dikirim dapat dijamin tanpa

mempengaruhi performa perangkat lunak secara signifikan.

4. Transmisi data hasil enkripsi pada perangkat lunak dilakukan dalam format *encoding base64* agar pengiriman data dapat dilakukan dalam format teks. Namun dengan demikian ukuran paket yang dikirimkan akan menjadi lebih besar ketika mode enkripsi berada dalam mode aktif. *Ciphertext* menjadi lebih panjang daripada *plaintext* karena *base64* bekerja pada basis 64, sementara satu buah karakter pada UTF-8 berukuran minimal 1 *byte* (basis 256).

7. Saran

Saran-saran yang diberikan terkait dengan tugas akhir ini adalah sebagai berikut:

1. Pada perangkat lunak tugas akhir ini hanya diimplementasikan servis *instant messaging*. Sementara itu ada banyak sekali servis yang disediakan oleh *Yahoo! Messenger*, misalnya *conference*, *chatting*, dan lain-lain. Pengembangan perangkat lunak selanjutnya diharapkan agar dapat mendukung fitur-fitur tersebut.
2. Pada perangakat lunak dapat diimplementasikan suatu *secure channel* sehingga pertukaran kunci dapat dilakukan antara kedua pihak dengan aman. Metode pertukaran kunci ini bisa juga menggunakan perpaduan antara algoritma simetris dan asimetris (algoritma *hybrid*). Dengan demikian setiap *user* dan *contacts* pada perangkat lunak *client* memiliki sepasang kunci publik dan privat yang berbeda satu sama lain. Kemudian kunci publik dan privat tersebut digunakan untuk bertukar kunci enkripsi (*shared symmetric key*).
3. Perangkat lunak ini dikembangkan pada *platform* J2SE (*Java Platform*, *Standard Edition*). Pengembangan selanjutnya dapat mengimplementasikan pengirim pesan instan terenkripsi untuk *mobile devices* dalam J2ME (*Java 2 Platform*, *Micro Edition*).

DAFTAR REFERENSI

[GAT03] Gatliff, Bill. (2003). *Encrypting Data with the Blowfish Algorithm*. EE Times-India.

[HUN03] Hung, Celia, dan Nathan Miller. (2003). *Analysis of Instant Messenger Programs*. Oregon State University.

[MEN96] Menezes, Alfred J., P. van Oorschot, dan S. Vanstone. (1996). *Handbook of Applied Cryptography*. CRC Press.

[MUN03] Munir, Rinaldi. (2003). *Kriptografi, Diktat Kuliah IF5054 Kriptografi Program Studi Teknik Informatika*. Institut Teknologi Bandung.

[RYA08] Ryan, Mark Dermot. (2008). *Symmetric Key Cryptography* (diakses tanggal 21 Juni 2009).

http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/symmetric-key.html

[SCH94] Schneier, Bruce. (1994). *Fast Software Encryption, Cambridge Security Workshop Proceedings*. Springer-Verlag.

[VAU95] Vaudenay, Serge. (1995). *On the Weak Keys of Blowfish*. Springer-Verlag.

[YAH07] *Yahoo! Coders Cookbook Tutorial*. (2007) (diakses tanggal 19 Juni 2009).

http://ycoderscookbook.com/

DAFTAR PUSTAKA

[FEB07] Setiadi, Febrian. (2007). *Implementasi Yahoo! Messenger Mobile Client Dengan Menggunakan J2ME*. Institut Teknologi Bandung.

[MUN03] Munir, Rinaldi. (2003). *Kriptografi, Diktat Kuliah IF5054 Kriptografi Program Studi Teknik Informatika*. Institut Teknologi Bandung.

[YAH07] *Yahoo! Coders Cookbook Tutorial*. (2007) (diakses tanggal 19 Juni 2009).

http://ycoderscookbook.com/