

**ASPEK KEAMANAN
SISTEM *INSTANT MESSAGING*
PADA PROTOKOL JABBER**

Tugas Akhir
EC 7010 – Keamanan Sistem Lanjut

Oleh :
RITA YENIS
NIM : 23203123



**PROGRAM MAGISTER TEKNIK ELEKTRO
BIDANG KHUSUS TEKNOLOGI INFORMASI-DIKMENJUR
INSTITUT TEKNOLOGI BANDUNG
2004**

ABSTRAK

Instant Messaging (IM) saat ini mengalami perkembangan yang cukup pesat pada jaringan *user*, karena kemampuannya mengirimkan pesan secara singkat dan cepat antara pengguna telekomunikasi. IM menjadi perangkat yang sangat penting untuk industri di seluruh dunia. IM digunakan di dalam penjadwalan (*scheduling meeting*), pertukaran informasi bisnis dan informasi *client* dan lain-lain. IM telah dikembangkan pada sektor-sektor *private* atau antar *provider* seperti *American Online Instant Messenger* (AIM), MSN dan Yahoo. Pada tahun 1998 muncul protokol IM yang bersifat *open source* yang terkenal dengan protokol Jabber.

Jabber mulai dapat perhatian publik ketika didiskusikan antar *developer* pada *website* Slashdot pada bulan Januari 1999. Pada Mei 2000, protokol Jabber diluncurkan sebagai protokol yang bersifat *open source* berdasarkan referensi server dan saat ini tidak dapat saling dipertukarkan. Jabber menggunakan arsitektur *client-server*, bukan arsitektur *peer-to-peer* seperti yang digunakan pada sistem IM lainnya. Protokol Jabber menggunakan format pesan *Extensible Markup Language* (XML). Format dokumen XML menjadi bahasa generik yang digunakan pada berbagai aspek komunikasi, karena sifatnya yang berbasis teks, mudah dibaca oleh manusia, maka aplikasi yang berbasis XML mudah untuk *didebug* atau melewati *firewall*.

Protokol Jabber telah berkembang menjadi protokol yang sangat atraktif karena bersifat *open source* dan dapat diterima secara luas. Siapapun dapat membuat atau mengembangkan protokol Jabber secara fungsional tanpa memodifikasi protokol inti dengan hanya melakukan *maintainance* interoperabilitas dengan *client* IM lainnya seperti Yahoo dan MSN. Penggunaan teknologi IM Jabber semakin meningkat, oleh karena itu perlu ditingkatkan kebutuhan proteksi terhadap informasi. Dalam tugas ini akan dibahas sistem keamanan protokol Jabber pada sisi *client* dan kemampuan untuk membuat protokol Jabber yang sudah ada menjadi lebih aman melalui jaringan berdasarkan *library* kriptografi atau berdasarkan standar *library* yang *open source* sebagai tulang punggung untuk mengamankan *task-task* yang ada pada Jabber.

Key word : *instant messaging*, protokol Jabber, *library* kriptografi, *open source*

DAFTAR ISI

| | |
|---|-----|
| ABSTRAK | ii |
| DAFTAR ISI | iii |
| DAFTAR GAMBAR | iv |
| DAFTAR TABEL | v |
| | |
| BAB I PENDAHULUAN | 1 |
| | |
| BAB II PENGERTIAN UMUM JABBER | 2 |
| 2.1 Sekilas Tentang Protokol Jabber..... | 2 |
| 2.2 Tinjauan Teknologi Protokol Jabber..... | 2 |
| 2.3 Arsitektur Jabber..... | 4 |
| 2.3.1 Model <i>Client Server</i> | 4 |
| 2.3.2 Format Data XML..... | 5 |
| 2.3.3 Jaringan Terdistribusi..... | 6 |
| 2.3.4 Standar Berdasarkan Pengalamatan..... | 6 |
| 2.4 Komponen Utama Protokol Jabber..... | 7 |
| 2.4.1 Message..... | 7 |
| 2.4.2 Presence..... | 7 |
| 2.4.3 Info/Query..... | 8 |
| | |
| BAB III PILIHAN SISTEM KEAMANAN JABBER | 10 |
| 3.1 Stream Encryption..... | 10 |
| 3.1.1 SSL/TLS..... | 11 |
| 3.1.2 OpenPGP..... | 11 |
| 3.2 Stream Authentication..... | 12 |
| 3.2.1 SASL Authentication..... | 12 |
| 3.2.2 Dialback Authentication..... | 12 |
| 3.3 Bekerja pada Komunitas Jabber Lainnya..... | 13 |
| 3.3.1 Security Session..... | 13 |
| 3.3.2 Mekanisme <i>Key Transport</i> | 14 |
| 3.3.3 Message Protection..... | 15 |
| 3.4 XML Security..... | 15 |
| | |
| BAB IV IMPLEMENTASI DAN ANALISIS SISTEM KEAMANAN PADA PESAN JABBER | 17 |
| 4.1 Implementasi Pada Pesan Jabber..... | 17 |
| 4.1.1 <i>Tool</i> yang Dibutuhkan..... | 17 |
| 4.1.2 Modifikasi XML..... | 17 |
| 4.1.3 Kode Kriptografi..... | 18 |
| 4.1.4 Kode <i>Client</i> Jabber yang <i>Open Source</i> | 21 |
| 4.2 Analisis dan Pembahasan..... | 22 |
| | |
| BAB V KESIMPULAN | 25 |
| DAFTAR PUSTAKA | |

DAFTAR GAMBAR

| | | |
|------------|--|----|
| Gambar 2.1 | Aliran <i>client-server</i> Jabber | 4 |
| Gambar 2.2 | Aliran data pada protokol Jabber..... | 5 |
| Gambar 2.4 | Model <i>messaging</i> Jabber | 7 |
| Gambar 3.1 | <i>Security session</i> XML paket <i>generation</i> dan <i>receiving</i> antara <i>Initiator</i> dan <i>Responder</i> | 15 |
| Gambar 3.2 | Paket XML untuk <i>Key Transport</i> | 16 |
| Gambar 4.1 | Diagram alir protokol <i><message></i> | 19 |
| Gambar 4.2 | Algoritma <i>Key Exchange</i> Diffie-Helman | 20 |
| Gambar 4.3 | <i>Window</i> pesan asli Jabber | 23 |
| Gambar 4.4 | <i>Window</i> yang menampilkan pesan yang telah dimodifikasi..... | 24 |
| Gambar 4.5 | <i>Reply Message Window</i> yang asli dan telah dimodifikasi..... | 24 |

DAFTAR TABEL

| | | |
|-----------|---|----|
| Tabel 3.1 | <i>Properties</i> pada teknologi sistem keamanan yang ada saat ini dibandingkan dengan yang <i>support</i> Jabber | 11 |
| Tabel 4.1 | <i>Properties</i> pada teknologi sistem keamanan yang ada saat ini dibandingkan dengan yang <i>support</i> Jabber dan telah mengalami modifikasi .. | 24 |

BAB I

PENDAHULUAN

Kehadiran IM telah menjadi fenomena yang sangat besar di internet beberapa tahun kebelakang. Fenomena tersebut ditandai dengan besarnya jumlah pengguna IM yang mencapai angka 65 juta pengguna pada tahun 2003, dan diperkirakan akan terus tumbuh hingga 350 juta pada tahun 2005. Tingkat penetrasi penggunaan IM yang sangat tinggi tersebut terutama disebabkan oleh kemampuannya untuk memfasilitasi komunikasi secara cepat, serta mampu menimbulkan kesan ‘tanpa jarak’. Selain di akibatkan oleh kemampuan dasar tersebut, tingginya tingkat penetrasi penggunaan IM juga disebabkan oleh kemudahan penggunaannya serta tingkat kebutuhan yang cukup sederhana di sisi pengguna.

IM pada umumnya saat ini juga dilengkapi oleh *Contact List* dan manajemen *Presence*, yang memungkinkan pengguna untuk mengetahui status dari seluruh daftar teman (*buddy list*) yang sedang *on-line* atau *off-line*, tanpa perlu untuk menghubunginya. IM tidak hanya digunakan untuk keperluan informal saja, namun juga digunakan untuk mendukung kegiatan penelitian serta kegiatan perkantoran dan industri lainnya.

Saat ini terjadi fragmentasi pasar IM, beberapa protokol IM telah tersedia meskipun masing-masing protokol itu masih bekerja dengan cara yang sama. Tetapi kehadiran protokol Jabber memberikan perkembangan yang cukup besar dalam komunitas IM hal ini disebabkan karena protokol ini bersifat *open source*, memiliki fitur-fitur yang menarik serta memiliki kemampuan untuk dikembangkan (*extensibility*).

Agar IM bisa kompatibel dan sesuai dengan yang diharapkan sistem ini masih perlu ditingkatkan sistem keamanannya. Seperti yang telah diketahui, sistem IM saat ini sangat rentan terhadap virus yang ada di jaringan. Data sering ditransmisikan dalam bentuk *plaintext* melalui jaringan eksternal dan internet. *Password* sering dikirim dan disimpan tanpa dienkripsi atau menggunakan metode yang masih lemah terhadap serangan yang mungkin muncul secara mendadak sehingga informasi secara bebas diberikan oleh *user* dan tidak tahu lagi data yang mana yang perlu diperhitungkan kerahasiaannya. Komunikasi antar protokol sangat sedikit yang telah didokumentasikan sehingga semakin meningkat masalah kejahatan yang berhubungan dengan virus, *worm* dan *Trojan Horse*. Produk IM seperti AOL Instant Messenger sering menimbulkan persoalan bagi admin sistem, program dapat melewati *proxy* dan *firewall* dengan baik dan menempati *port-port* terbuka yang dapat dikonfigurasi dengan beragam *proxy server*.

Dalam tugas ini akan dibahas beberapa teknik sistem keamanan yang dikembangkan dan beberapa pilihan kebutuhan yang diperlukan untuk mengamankan pesan yang menggunakan protokol Jabber dengan memanfaatkan *library-library* kriptografi yang sudah tersedia pada sisi *client* Selain itu juga akan dibahas modifikasi-modifikasi yang mungkin dilakukan pada mekanisme pesan *client Jabber*.

BAB II PENGERTIAN UMUM JABBER

2.1 Sekilas Tentang Protokol Jabber

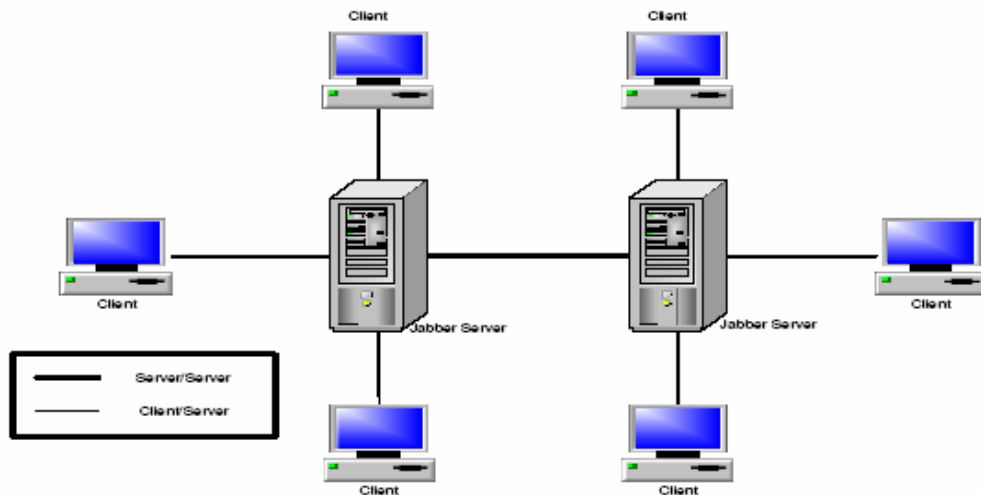
Jabber adalah sebuah protokol XML yang terbuka untuk pertukaran *message* dan *presence* yang *real-time* antara dua *user* di dalam jaringan Jabber. Banyak kegunaan teknologi Jabber, pada awalnya teknologi Jabber bersifat *asynchronous*, *platform* IM yang dapat digunakan secara luas dan jaringan IM berdasarkan fungsinya hampir sama dengan sistem IM yang resmi seperti *AOL Instant Messaging* (AIM) dan *Yahoo Instant Messaging* [1].

Sebagai usaha menjadikan Jabber sebagai protokol standar *Instant Messaging*, pada Juni 2000 komunitas Jabber telah mempublikasikan protokol tersebut sebagai *Request for Comment* (RFC) pada *Internet Engineering Task Force* (IETF) sebagai bagian dari standar *Instant Messaging and Presence Protocol* (IMPP), tetapi IMPP ini tidak berjalan sukses. Pada bulan Mei 2001, *Jabber Community* dan *Jabber Inc.* membuat *Jabber Software Foundation* untuk menyediakan asisten organisasi secara langsung (*direct organizational assistance*) dan asisten teknis secara tidak langsung terhadap komunitas Jabber.

Pada tahun 2002, *Internet Engineering Steering Group* (IESG) menyetujui formasi *Extensible Messaging and Presence Protocol Working Group* (XMPP) dengan *Internet Engineering Task Force* (IETF). Ruang lingkup *working group* adalah untuk mengeksplorasi dan dimana protokol tersebut digunakan, memodifikasi protokol yang sudah ada agar dapat memenuhi RFC 2799 seperti persyaratan yang ditentukan dalam spesifikasi *Common Presence and Instant Messaging* (CPIM). Fokus utama *working group* adalah membuat *XML stream* termasuk *stream* pada level *security* dan autentikasi, elemen data dan *namespace* yang dibutuhkan untuk mencapai dasar IM dan *Presence*^[2]. XMPP *working group* menerbitkan *XMPP Core Internet-Draft* sebagai dokumen yang menggambarkan fitur-fitur utama *Extensible Messaging* dan protokol *Presence*. Makalah XMPP ini memuat protokol Jabber yang bekerja pada sistem keamanan *client-server* dan *server-server*.

2.2 Tinjauan Teknologi Protokol Jabber

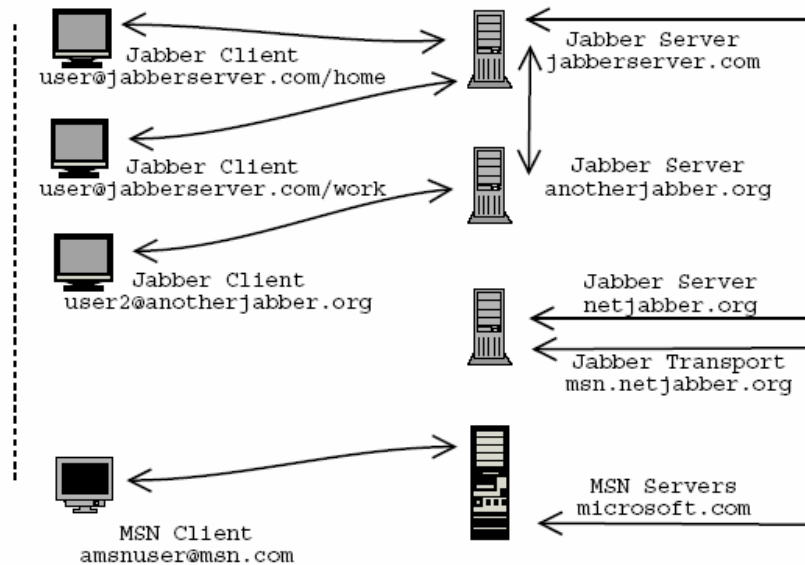
Jabber terkenal dengan arsitektur *client-server*nya, *client* Jabber dapat berkomunikasi dengan server Jabber pada *domain* Jabber mereka. *Domain* Jabber memiliki keuntungan yaitu kemampuannya dalam memisahkan zona komunikasi, yang ditangani oleh server Jabber yang berbeda, tidak seperti kebanyakan sistem IM lainnya yang menggunakan satu server terpusat untuk seluruh zona komunikasi^[3]. Gambar 2.1 menunjukkan *stream* Jabber *client-server*.



Gambar 2.1 Aliran *client-server* Jabber

XMPP merupakan protokol hasil formalisasi IETF dari *streaming* protokol standar XML, yang dikembangkan oleh *Jabber Community*. Protokol ini menghadirkan fitur lengkap untuk *Instant Messaging* dan *Presence* di atas *data transport layer* yang bersifat *dedicated*. Protokol ini telah stabil sejak tahun 1999. Jabber/XMPP adalah sebuah protokol yang telah didokumentasikan dengan baik dari seluruh protokol yang ada dan mudah untuk dipahami.

Teknologi dasar dari XMPP menyangkut proses negosiasi XML *stream* antara *client* dan server, dengan menggunakan *Simple Authentication and Security Layer* (SASL) dan *Transport Layer Security* (TLS) untuk mengamankan pengiriman datanya. Setelah melakukan autentikasi, selanjutnya pengguna dapat mengirimkan fragmen-fragmen XML sebagai hasil dari menjalankan fungsi-fungsi IM, seperti mengirimkan pesan, chat dengan teman, merubah *status presence*, mengatur *contact list*, bergabung dengan *chatroom*, dan lain-lain. *Server* kemudian akan mengirimkan *message* kepada *server* lain melalui XML *stream* yang telah melalui proses negosiasi, berhubungan dengan syarat-syarat *security* untuk kemudian mencapai lokasi responden pengguna. XMPP kompatibel dengan teknologi Jabber yang sudah ada, sehingga menjamin interoperabilitas dengan jaringan yang ada saat ini. Aliran data pada protokol Jabber dapat dilihat pada Gambar 2.2



Gambar 2.2 Aliran data pada protokol Jabber

Cara Jabber/XMPP bekerja sering digambarkan seperti sebuah *router XML* artinya jika pesan dikirim dalam bentuk paket XML dan *route*-nya (pesan tersebut akan dikirim ke lokasi yang berdasar *content*-nya). Jabber di desain serupa dengan HTTP dan email karena protokol ini relatif baru sampai saat ini Jabber memiliki sistem keamanan yang lebih baik.

Jabber merupakan sistem jaringan terdistribusi yang menggunakan konektivitas *Domain Name Service* (DNS), Jabber mempunyai sebuah fasilitas *dial-back* yang tidak sama dengan *email* untuk menempatkan alamat, artinya seseorang yang melakukan *spamming* pada sebuah server dengan jumlah data yang besar secara cepat. *Password* dapat disimpan dan di autentikasi dengan berbagai cara termasuk menggunakan PGP/SSL.

Saat ini tersedia banyak dokumentasi tentang komunikasi Jabber/XMPP dan protokol yang hanya sekali untuk didokumentasi secara keseluruhan. Jabber *support* terhadap sejumlah skema autentikasi dari algoritma *Hashing plaintext* dan standard SASL. Dengan menggunakan Jabber, komunikasi *client* ke *server* melalui SSL dan beberapa *client* menggunakan PGP berdasarkan *software* enkripsi. Sistem Jabber dapat juga terhubung ke sistem lainnya dengan sesuatu yang disebut *transport* yang berdasarkan *client emulation* dan dapat dijalankan pada *server* Jabber berdasarkan interoperabilitas antar protokol.

Ditinjau dari sistem keamanan, pada protokol Jabber terjadi *client bugs* semacam *buffer overflow* yang berpengaruh pada versi khusus dari aplikasi yang secara langsung tidak dipengaruhi oleh virus atau *hacker*.

2.3 Arsitektur Jabber

2.3.1 Model Client-Server

Jabber menggunakan arsitektur *client-server*, bukan arsitektur langsung *peer-to-peer* seperti yang digunakan oleh sistem *messaging* lainnya. Akibatnya, seluruh data Jabber dikirim dari satu *client* ke *client* lainnya harus melewati minimal satu *server* Jabber. *Client* Jabber terhubung pada sebuah *server* Jabber pada TCP melalui port 5222. Koneksi ini selalu *on* untuk *session client* yang berjalan pada *server*, artinya *client* tidak dapat

mengumpulkan pesan sebagai sebuah *email client*. Sebuah pesan diharapkan tersedia pada *client* dan dengan segera diharapkan *client messenger* sepanjang *client* masih terhubung. Server akan dapat menjajaki (*tracking*) apakah *client* masih *online* atau tidak, dan ketika *client* dalam kondisi *off-line* akan menyimpan beberapa pesan yang telah dikirim kepada *client* untuk menyediakan kapan dia akan terhubung lagi.

Kekhasan yang dimiliki oleh protokol Jabber antara lain *modular server* dan *simple client* yang penjelasannya sebagai berikut :

1. Modular server

Server Jabber memiliki tiga peranan utama yaitu :

- Menangani koneksi *client* dan berkomunikasi secara langsung dengan *client* Jabber
- Berkomunikasi dengan server Jabber yang lain
- Mengkoordinasikan beragam komponen *server* yang diasosiasikan dengan *server*

Server Jabber di desain modular, dengan paket kode internal yang khusus sehingga dapat menangani fungsionalitasnya seperti registrasi, autentikasi, *present*, *contact list*, penyimpanan pesan yang berstatus *off-line* dan sebagainya. Selain itu *server* Jabber dapat dikembangkan dengan komponen eksternal yang memungkinkan *administrator server* untuk mensuplemen *server* pusat dengan layanan tambahan semacam gerbang untuk sistem *messaging* lainnya.

2. Simple client

Satu kriteria desain sistem Jabber bahwa ia harus memiliki kemampuan untuk mendukung *client* yang sederhana misalnya koneksi telnet pada *port* yang benar. Dalam hal ini tentu saja arsitektur Jabber memberikan sedikit batasan pada *client*.

Task-task pada *client* Jabber harus dapat mengenal dan melengkapi :

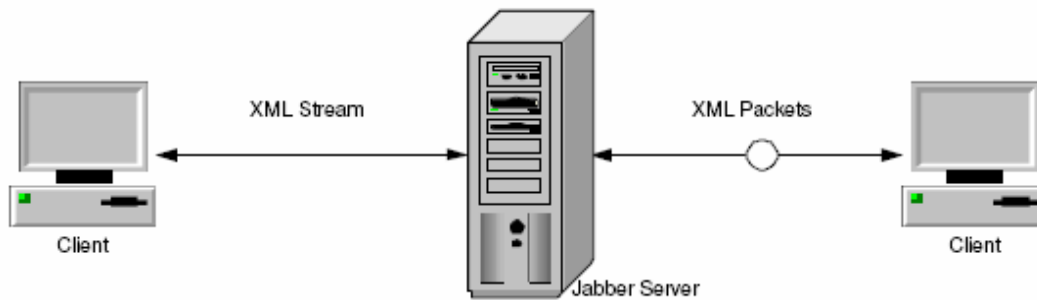
- Komunikasi dengan *server* Jabber melalui soket TCP
- Melakukan *parsing* dan interpretasi XML dengan format yang baik melalui XML stream
- Memahami tipe data utama Jabber (*message*, *presence* dan *iq*)

Keuntungan di dalam Jabber adalah dapat memindahkan kompleksitas dari *client* ke *server*. Secara praktis, banyak fungsi yang *low-level* pada *client* seperti proses *parsing* XML dan memahami tipe *data core* Jabber yang ditangani oleh *library-library client Jabber*, memungkinkan *client developer* untuk fokus pada *user interface*^[3].

2.3.2 Format Data XML

Format data XML adalah bagian integral arsitektur Jabber karena sepenuhnya penting sehingga arsitektur secara fundamental dapat dikembangkan dan mampu diekspresikan dengan bentuk data yang terstruktur. Gambar 2.3 menunjukkan model *messaging* Jabber yang digabungkan dengan 4 elemen utama yaitu :

- paket XML memuat data yang di *marked-up*,
- XML *stream* yang digunakan untuk transportasi paket XML ,
- *client* dan *server* Jabber yang dapat saling dipertukarkan.



Gambar 2.3 Model *messaging* Jabber

Dengan terhubungnya *client* pada *server*, berarti membuka satu jalur ke *XML stream* dari *client* ke *server*, dan *server* merespon dengan satu jalur *XML stream* dari *server* ke *client*. Selanjutnya masing-masing *session* melibatkan dua *XML stream*. Seluruh komunikasi antara *client* dan *server* terjadi pada *stream* ini, contohnya adalah sebagai berikut :

```
<message from='jolie@jabber.com/home' to='aim@rhybox.com/work' >
  <body>Hello, I need to ask you a question !?</body>
</message>
```

Ketika banyak *snippet* Jabber yang hanya sangat sederhana, format XML Jabber dapat juga ditingkatkan melalui *namespace* XML yang telah diatur oleh *Jabber Software Foundation* dan *namespace* disesuaikan untuk aplikasi yang khusus. Hal ini yang membuat Jabber menjadi *platform* yang *powerful* untuk memilih struktur data yang digunakan termasuk *XML Remote Procedure Calls* (XML-RPC), *Resource Description Framework Site Summary* (RSFSS) dan *Scalable Vector Graphics* (SVG)^[1].

2.3.3 Jaringan Terdistribusi

Jaringan terdistribusi dalam hal ini bagaimana sebuah *server* Jabber dapat berkomunikasi dengan *server* Jabber lainnya dan dapat diakses melalui internet. Masing-masing *user* terhubung pada *home server*, yang menerima informasi untuk mereka, selanjutnya *server* akan mentransfer data untuk kepemilikan *user*. Maka suatu domain dapat jalan pada *server* Jabber. Masing-masing fungsi *server* bebas terhadap yang lainnya, dan di-*maintain* sendiri di dalam daftar *user*. *User* khusus diasosiasikan dengan *server* yang khusus pula, dan alamat Jabber memiliki bentuk yang sama dengan alamat *email*. Jaringan yang terdistribusi ini menghasilkan sesuatu yang fleksibel, jaringan yang mampu terkontrol pada server yang memiliki skala yang lebih tinggi dibandingkan monolitik, tetapi dengan syarat bahwa layanan yang terpusat hanya dapat jalan pada vendor IM yang resmi:^[1].

2.3.4 Standar Berdasarkan Pengalaman

Ada beberapa perbedaan entitas pada Jabber untuk dapat berkomunikasi dengan yang lainnya. Entitas ini dapat direpresentasikan berupa *transport*, *groupchat room* atau *single Jabber user*. *Jabber ID* telah digunakan secara internal dan eksternal untuk menyatakan

kepemilikan atau *routing* informasi. Masing-masing *Jabber ID* memuat sekelompok order elemen. *JID* dibentuk dalam suatu *domain*, *node* dan *resource* dalam format [node@domain[/resource]]^[4], contohnya : rita@jabber.com/home

2.4 Komponen Utama Protokol Jabber

Ada tiga komponen utama pada protokol Jabber yang diandalkan dengan mekanisme *messaging* :

2.4.1 Message

Protokol *message* pada kenyataannya adalah protokol yang paling sederhana dalam Jabber. Banyak *traffic* di dalam jaringan Jabber yang termasuk dalam protokol *message*. Message terdiri dari 4 atribut, dan *zero* atau beberapa *child element*. Attribute dalam *message* adalah :

- to** : jenis yang diharapkan oleh pihak penerima pesan
- from** : jenis pesan yang dikirim
- id** : sebuah identifier unik yang bersifat opsional dengan tujuan dapat menjejaki *message*
- type** : sebuah spesifikasi opsional dari konteks percakapan sebuah *message*

Sedangkan atribut pada elemen anak antara lain ;

- body** : isi tekstual dari *message*, secara normal termasuk tetapi tidak dibutuhkan.
- subject** : subjek dari *message*
- thread** : string acak yang di-*generated* oleh pengirim, digunakan untuk *tracking* sebuah *thread conversation*.
- error** : deskripsi pesan kesalahan

Versi protokol Jabber XMPP saat ini menggunakan standar yang merepresentasikan seluruh atribut dan elemen anak yang ada pada *message protocol*^[5]. Contoh paket *message* ditunjukkan seperti berikut :

```
<message to='romeo@montague.net'
from="Juliet@capulet.com/balcony"
type='chat'>
  <subject xml : lang='en'>
    Greeting !
  </subject>
  <body xml : lang='en'>
    Hello!!
  </body>
<thread>e0f92794b9683a38</thread>
</message>
```

2.4.2 Presence

Protokol ini bertanggung jawab terhadap *subscription*, persetujuan, dan *update* informasi *presence* dalam komunitas Jabber. Atribut diasosiasikan dengan dengan protokol ini sama seperti pada protokol *message*, *presence* memiliki memiliki tipe atribut yang memiliki 7 *state* sebagai berikut ;

1. **unavailable** : *client* tidak lama tersedia untuk berkomunikasi
2. **subscribe** : pengirim mengirimkan *request* untuk *subscribe* terhadap *presence*

- 3.subscribed** : penerima
- 3.subscribed** : pengirim yang telah diizinkan terhadap recipient untuk menerima *presence* mereka.
- 4.unsubscribe** : *subscription request* yang telah ditolak atau *subscription* yang telah di *cancel* sebelumnya.
- 5. probe** : *request* dari *client* yang *presence* saat ini
- 6.error** : pesan kesalahan yang berlangsung berdasarkan pemrosesan atau menyediakan paket *presence* yang telah dikirim sebelumnya.

Paket *presence* akan bernilai 0 atau 1 untuk masing-masing elemen anak berikut :

- show** : menggambarkan status yang tersedia dari entities atau *resource* yang spesifik. show memiliki empat nilai yang digunakan :
 - away : *temporarily away*
 - chat : bebas untuk *chat*
 - xa : *extended away*
 - dnd : *do not disturb*
- status** : merupakan deskripsi bahasa natural yang bersifat opsional yang mendeskripsikan status yang tersedia.
- priority** : bilangan *integer* bukan negatif yang menampilkan level prioritas pada *resource* yang terkoneksi, dengan 0 sebagai prioritas terendah.
- error** : deskripsi pesan kesalahan

Contoh paket *presence* adalah sebagai berikut :

```
<presence>
  from='juliet@capulet.com/balcony'
  to='romeo@montague.net/orchard'>
<show> away </show>
<status> be right back </status>
<priority> 0 </priority>
</presence>
```

Paket *presence* dapat memuat *namespace* yang sesuai dengan elemen anak yang tidak mengganggu struktur *namespace* dan tag yang tersedia^[5].

2.4.3 Info/Query

Protokol IQ adalah protokol Jabber yang terakhir dan yang paling peduli dibandingkan *message* dan protokol *presence*. IQ adalah protokol *request-response* yang umum sehingga di desain secara mudah untuk dikembangkan seperti HTTP yang merupakan medium *request-respon*. *Content* data dari *request* dan *respon* yang ditentukan dengan deklarasi *namespace* elemen anak secara langsung dari elemen IQ. Atribut diasosiasikan dengan protokol IQ memiliki atribut yang sama dengan protokol *message* dan *presence* kecuali jika protokol tersebut memiliki tipe atribut yang berbeda. Tipe atribut pada protokol *info/query* memiliki 4 nilai yang dapat digunakan ;

1. **get** : informasi *request*
2. **set** : menyediakan data yang dibutuhkan
3. **result** : respon terhadap *get* dan *set request* yang sukses
4. **error** : kesalahan yang terjadi dalam pemrosesan dan layanan *get* dan *set request*

Berikut adalah contoh paket IQ yang memblokir *incoming message* dari JID yang khusus:

```
<iq type='set' id='msg1'>
<query xmlns='jabber:iq:privacy'>
<list name='message-jid-example'>
  <item type='jid' value='tybalt@capulet.com'
        action='deny' order='3'>
    <message/>
  </item>
</list>
</query>
</iq>
```

Protokol IQ ini sangat penting jika kita ingin membangun *server* berdasarkan kebijakan keamanan sistem yang harus dipenuhi oleh *client*. Jika sistem keamanan *client* telah terpenuhi maka harus mendukung pula terhadap sistem keamanan pada sisi *server*.

BAB III PILIHAN SISTEM KEAMANAN JABBER

Sistem keamanan Jabber saat ini dapat memproteksi data pesan yang secara umum tidak memenuhi untuk beberapa tahap *deployment*, sistem keamanan Jabber ini diterapkan pada lingkungan yang cukup luas, perusahaan komersial atau perwakilan pemerintahan. Fitur sistem keamanan saat ini harus dapat menjawab masalah yang berhubungan dengan skalabilitas, penggunaan, dan fitur-fitur yang mendukung terhadap teknologi yang digunakan meskipun masih kurang pada beberapa standarisasi yang belum diterapkan^[3].

Tabel 3.1 menggambarkan fitur-fitur sistem keamanan sebelum ditambahkan *library* kriptografi pada pesan Jabber jika dibandingkan dengan *properties* yang ada pada sistem keamanan komputer saat ini.

Tabel 3.1 *Properties* pada teknologi sistem keamanan yang ada saat ini dibandingkan dengan yang *support* Jabber

| <i>Security Property</i> | Deskripsi | <i>Existing Technology</i> | <i>Jabber Support</i> |
|--------------------------|--|------------------------------------|--|
| Autentikasi | Memastikan sebuah entitas siapa yang telah mengklaim | JAASI, Kerebos, Microsoft Passport | <i>Jabber Authentication Protocol</i> |
| Autorisasi | Menentukan apakah entitas telah mendapat izin untuk mengakses data atau mengontrol sumber daya | JAAS, <i>access control list</i> | Autorisasi biner. User yang tidak diautentikasi dan dijamin benar, dan autentikasi oleh user yang lainnya. |
| Integrity | Memastikan bahwa data tidak dirusak | <i>Message digest</i> | Tidak <i>support</i> |
| Non-repudiation | Memastikan bahwa <i>author</i> data akan selalu dapat diautentikasi | <i>Digital signature</i> | Tidak <i>support</i> |
| Confidentiality | Memastikan bahwa data hanya dapat dibaca oleh entitas yang resmi | Enkripsi | Dibatasi untuk kerahasiaan <i>client-server</i> menggunakan SSL. |

Ada beberapa pilihan sistem keamanan yang tersedia dan telah didokumentasikan di dalam komunitas Jabber, meskipun banyak informasi yang diajukan masih dalam tahap pengembangan karena sampai saat ini belum ada kebijakan final mengenai sistem keamanan untuk protokol Jabber. Diantara pilihan-pilihan sistem keamanan tersebut adalah:

3.1 Stream Encryption

XMPP merupakan sebuah metode untuk mengamankan *stream* dari kerusakan atau pembicaraan yang didengar oleh pihak lain (*eavesdropping*). Metode enkripsi ini menggunakan protokol *Transport Layer Security* (TLS) yang merupakan kelanjutan dari

STARTTLS. Identifier *namespace* STARTTLS XML tersedia akan dapat digunakan antara suatu entitas awal dan entitas penerima. Sebagai contohnya adalah *stream* dari *client* ke *server* atau dari satu *server* terhadap yang lainnya)^[5].

3.1.1 SSL/TLS

Sebelum menggunakan SSL/TLS, *client* dapat mulai dengan membahas STARTTLS dan memantau respon *server* apakah mendukung TLS atau tidak. Contoh berikut menunjukkan aliran data untuk mengamankan *client* menggunakan STARTTLS, ada beberapa langkah yang harus dilalui antara lain :

- a. *Client* mengawali *stream* terhadap *server*
- b. *Server* memberikan respon dengan mengirim sebuah *tag stream* kepada *client*
- c. *Server* mengirim STARTTLS *extension* kepada *client* dengan mekanisme autentikasi dan fitur *stream* lainnya
- d. *Client* mengirim perintah STARTTLS ke *server*
- e. *Server* menginformasikan kepada *client* untuk *running*
- f. *Server* menginformasikan kepada *client* bahwa negosiasi TLS telah gagal dan *client* tersambung dengan *stream authentication*
- g. *Client* dan *server* melengkapi negosiasi melalui TCP
- h. *Client* memulai *stream* baru pada *server*
- i. *Server* merespon dengan mengirim sebuah *stream header* kepada *client* sepanjang adanya negosiasi *stream*
- j. *Client* melanjutkan *stream authentication*^[5]

Langkah-langkah diatas mengilustrasikan negosiasi *client* terhadap *server*, oleh karena itu administrator akan memilih menggunakan TLS antara dua domain dengan tujuan mengamankan komunikasi *server* ke *server*.

3.1.2 OpenPGP

XMPP *working group outline* menggunakan solusi OpenPGP yang digunakan saat ini dengan tidak ada modifikasi aktual di dalam draft internet mereka dengan judul *End-To-End Object Encryption*. XMPP *working group* menggambarkan enkripsi objek sebagai mekanisme *key exchange* yang dilakukan dengan menggunakan *key server* OpenPGP. Demonstrasi berikut ini menggunakan OpenPGP menggunakan sistem Jabber.

Payload yang telah dienkripsi memuat apakah yang akan menjadi *main element* <body> jika pesan tidak dienkripsi, dengan sebuah *message ID* untuk membantu mencegah penyerangan kembali antara pasangan informasi yang dibungkus di dalam elemen <payload> pada *scope namespace* 'http://jabber.org/protocol/e2e#payload'

Payload yang dienkripsi memuat sebuah elemen <id>. CDATA dari elemen <id> yang dibangun berdasarkan algoritma berikut ini :

- 1) gabungkan seluruh JID sender (user@host/resource) dengan seluruh JID *recipient*
- 2) gabungkan karakter string JID dengan berdasarkan ISO-8601 *timestamp* termasuk tahun, bulan, hari, menit, detik dan UTC offset yang memenuhi format yyyy-mm-dd Thh:mm:ss-hh:mm;
- 3) *hash* menghasilkan string berdasarkan algoritma SHA1
- 4) konversikan *output* SHA1 yang berupa *lowercase hexadecimal*.

Keseluruhan elemen `<payload/>` akan dienkripsi berdasarkan algoritma OpenPGP menggunakan KeyID sender. Hasil *cipher text* akan tersedia sebagai CDATA pada elemen `</x>` pada *scope namespace* `'http://jabber.org/protocol/e2e'`. Pesan yang secara opsional merupakan sebuah elemen anak `</body>` yang tidak dienkripsi dimana informasi CDATA *recipient* merupakan *message body* yang tidak dienkripsi.

3.2 Stream Authentication

XMPP menggunakan dua macam metode untuk memperkuat autentikasi pada level XML stream. Ketika sebuah entitas telah siap dengan yang lainnya, maka hubungan antar entitas seolah-olah telah dibangun ketika *user* melakukan registrasi dengan *server* atau seorang *administrator* sebuah *server* terhadap *server* lainnya yang terpercaya, metode yang tersedia untuk autentikasi stream antara 2 entitas menggunakan XMPP yang telah beradaptasi dengan algoritma *Simple Authentication and Security Layer* (SASL).

Ketika tidak ada hubungan yang dapat dipercaya antara 2 server, beberapa level akan dibangun berdasarkan apa yang sudah ada pada *Domain Name Server* (DNS), metode autentikasi digunakan dalam kasus ini adalah pada server dipasang protokol yang merupakan XMPP asli. Jika SASL digunakan untuk autentikasi *server* ke *server*, *server* harus tidak digunakan *dialback*^[5]. SASL dan metode autentikasi *dialback* digambarkan dalam sesi berikut ini

3.2.1 SASL Authentication

SASL menyediakan metode umum untuk menambahkan autentikasi yang mendukung koneksi berbasis protokol. XMPP menggunakan sebuah profil *namespace* XML yang umum dan *namespace identifier* untuk protokol ini.

Langkah-langkah autentikasi SASL pada protokol Jabber adalah sebagai berikut :

1. *Client* mengawali *stream* pada *server* :
2. *Server* merespon dengan sebuah *tag stream* yang telah dikirim ke *client*
3. *Server* menginformasikan pada *client* bahwa mekanisme autentikasi telah tersedia
4. *Client* memilih sebuah mekanisme autentikasi (*initial respon*)
5. *Server* mengirim *challenge* yang di-encodekan dengan *base64* ke *client*
challenge didekodekan
6. *Client* merespon *challenge*, respon didekodekan
7. *Server* mengirim *challenge* yang lain ke *client* dan selanjutnya *challenge* dikodekan
8. *Client* memberi *respon* terhadap *challenge*
9. *Server* menginformasikan kepada *client* apakah autentikasi sukses atau gagal
10. *Client* mengawali sebuah *stream* baru ke *server*
11. *Server* memberi respon dengan mengirim *stream header* ke *client*, dengan *stream* yang telah siap diautentikasi

3.2.2 Dialback Authentication

Di dalam XMPP termasuk sebuah metode level protokol untuk membuktikan bahwa koneksi antara 2 server dapat dipercaya (minimal seperti DNS yang dapat dipercaya). Metode ini disebut *dialback* dan hanya dapat digunakan dengan XML stream yang dideklarasikan berdasarkan *namespace* jabber:server. Tujuan dari protokol *dialback*

adalah membuat *spoofing* protokol menjadi lebih sulit, dan selanjutnya akan lebih sulit pula untuk memalsukan bait-bait pada *tag XML*. *Dialback* tidak diharapkan sebagai mekanisme untuk mengamankan atau mengenkripsi *stream* antar server, tetapi hanya untuk membantu mencegah terjadinya *spoofing* pada *server* dan mengirim data-data yang salah yang ada pada *server*. Bagaimanapun, domain membutuhkan sistem keamanan yang lebih kokoh (*robust*) yang seharusnya tidak diperhitungkan hanya pada pilihan autentikasi ini saja^[5].

3.3. Bekerja pada Komunitas Jabber Lainnya

Sebuah *draft* yang belum lengkap tentang sistem keamanan protokol Jabber telah diajukan pada *Jabber Software Foundation* untuk mengamankan percakapan melalui Jabber.

Protokol yang diajukan mempunyai syarat-syarat sebagai berikut :

- Harus merupakan ekstensi opsional dari protokol Jabber yang sudah ada
- Harus lebih transparan terhadap *server* Jabber yang sudah ada
- Harus mempunyai fungsi yang mengagumkan dalam kasus dimana beberapa anggota komunitas yang tidak *running* pada sebuah *user agent* yang mendukung protokol yang ada
- Harus menciptakan penggunaan XML yang lebih baik
- Harus dapat mencegah algoritma yang dibebani (*encumbered algorithm*)
- Harus dapat diimplementasikan dengan menggunakan *toolkit* kriptografi yang tersedia
- Seharusnya tidak lagi membutuhkan *public key infrastructure* (PKI)

Draft yang diajukan memiliki empat *session* sistem keamanan utama yang dibahas berikut ini :

3.3.1. Security Session

Security session diidentifikasi dengan tiga *tuple* yang terdiri dari :

1. **Initiator** : merupakan JID dari *user* yang telah diawali pada *session*
2. **Responder** : merupakan JID *user* yang telah merespon berdasarkan *request initiator*
3. **SessionId** : label yang dibangkitkan oleh *initiator*

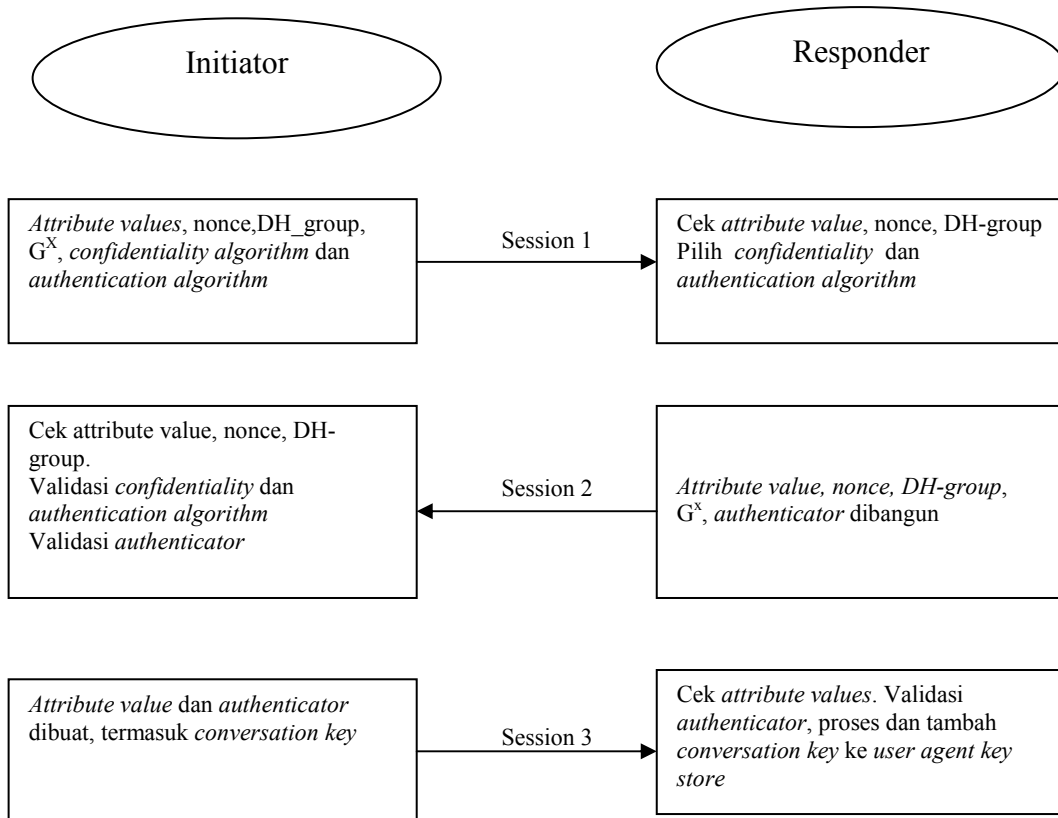
Security session untuk digunakan sebagai *transport conversation key* antara partisipan yang sedang melakukan percakapan. *Security session* dinegosiasi dengan menggunakan sebuah autentikasi pertukaran kunci *Diffie-Hellman* .

Dua tujuan utama dari pertukaran kunci ini adalah :

- untuk membentuk mutual autentikasi
- untuk menyetujui bahwa sesuatu yang sifatnya rahasia hanya diketahui oleh masing-masing *user*.

Jika sepasang *user* menyetujui *secret*-nya dapat dibagi, maka masing-masing *user* diturunkan *material key* dari *secret* tersebut, *material key* ini digunakan untuk mengamankan transportasi antar *conversation key*, yang secara aktual digunakan untuk

memproteksi *data conversation*. *Protocol Data Unit (PDU)* digunakan untuk membandingkan elemen pertukaran *transport* dengan elemen protokol Jabber yang sudah ada. Tiga *session PDU* untuk melengkapi *task-task* ini dapat dilihat pada Gambar 3.1, dan nilai atributnya berupa *version*, *Initiator*, *initiator JID*, *responder JID*, *sessionId*, dan *hmac value*.



Gambar 3.1 *Security session XML* paket *generation* dan *receiving* antara *Initiator* dan *Responder*

3.3.2. Mekanisme *Key Transport*

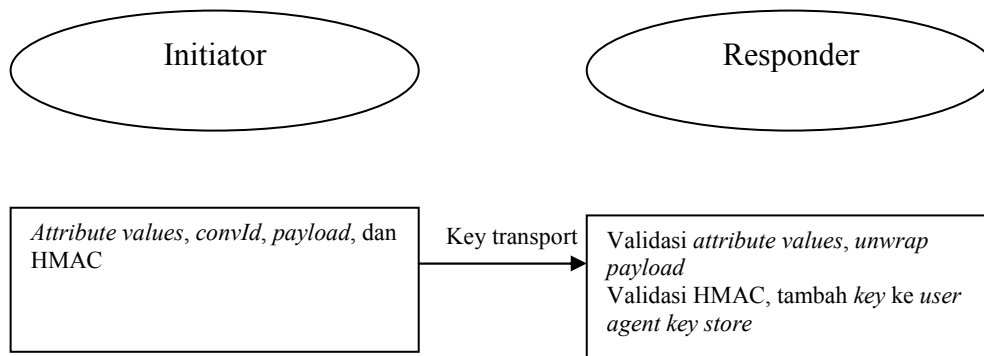
Conversation key ditransportasikan menggunakan kunci simetris yang dibungkus dengan fitur enkripsi XML yang digandeng dengan *key Transport PDU*. Contoh mekanisme transpor kunci adalah sebagai berikut:

```

<!ELEMENT keyTransport
  (convId, payload, hmac) >
<!ATTLIST keyTransport
  version CDATA #REQUIRED
  initiator CDATA #REQUIRED
  responder CDATA #REQUIRED
  sessionId CDATA #REQUIRED >
<!ELEMENT convId (#PCDATA)* >
<!-- These are actually instances of xenc:EncryptedKey -->
<!ELEMENT payload
  (confKey, hmacKey) >
  
```

```
<!ELEMENT hmac (#PCDATA)* > <!ATTLIST hmac
    encoding (base64 | hex) #REQUIRED >
```

Pada Gambar 3.2 adalah paket XML pada *Key Transport*, adapun nilai-nilai atributnya berupa *version*, *initiator JID*, *responder JID*, dan *session ID*.



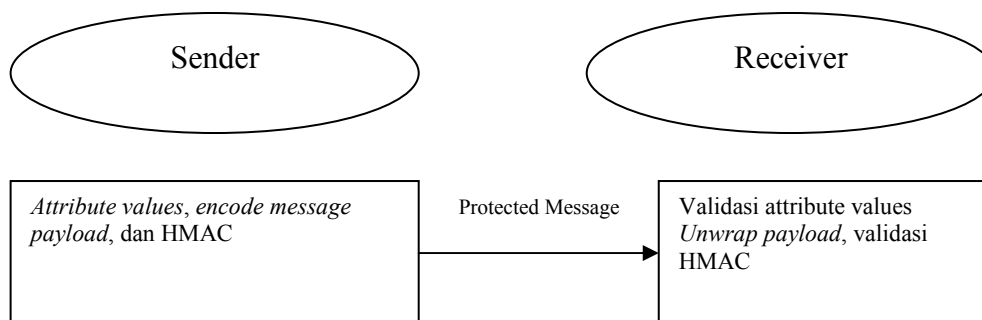
Gambar 3.2. Paket XML untuk *Key Transport*

3.3.3 Message Protection

Sebuah pesan yang telah diproteksi didefinisikan sebagai sebuah pesan tradisional Jabber yang *content* dari *body*-nya di kembangkan untuk memasukkan transport secara kriptografi adalah *message body* yang telah diproteksi. Ada 2 fitur kunci yang digunakan pada proteksi pesan yaitu :

- *Usual body element* : memuat beberapa teks yang bersifat berubah-ubah
- Message yang memuat elemen Jabber $\langle x \rangle$: yang mendefinisikan sistem keamanan Jabber, *message*, *namespace*, elemen ini merupakan *transport message* yang telah diproteksi

Mekanisme ini memiliki keuntungan yaitu mengizinkan integrasi yang transparan dengan *server* dan *client* Jabber yang telah ada^[7]. Nilai atribut pada *Message Protection* adalah *version*, *from*, *to*, *conId*, dan *seqNum*. Pada Gambar 3.3 aliran paket XML untuk *message protection*.



Gambar 3.3 Paket XML untuk *Message Protection*

3.4 XML Security

XML digunakan sebagai teknologi dasar untuk mendesain dokumen yang terstruktur berdasarkan pertimbangan bahwa XML merupakan standar yang terbuka dan telah diterima secara luas untuk mendukung transaksi berbasis Internet^[8]. Dengan karakteristik berbasis teks dan bersifat terbuka tersebut, maka kelemahan pada XML ini dapat juga menjadi sumber kerentanan pada aplikasi dengan membuka peluang adanya gangguan atau serangan terhadap keamanan dokumen.

Salah satu keuntungan protokol Jabber adalah kemampuan untuk menambah sistem keamanan yang cocok tanpa mengubah Jabber yang standar, protokol atau *server* Jabber. Standar enkripsi XML yang diajukan oleh *World Wide Web Consortium*(W3C) dapat digunakan pada Jabber tanpa secara aktual memodifikasi atau mengubah standar-standar yang sudah ada pada Jabber. Beragam komponen sistem keamanan XML pada beragam proses standarisasi telah dilakukan, diantaranya *XML Digital Signature* dan *XML Encryption* yang menyediakan autentikasi dan kerahasiaan yang cocok untuk kriptografi XML^[9].

BAB IV IMPLEMENTASI DAN ANALISIS SISTEM KEAMANAN PADA PESAN JABBER

Dalam tugas ini akan ditampilkan bagaimana bekerja dengan *client* Jabber dengan menggunakan *key exchange Diffie-Helman* dan algoritma DES untuk *key exchange* dan enkripsi atau dekripsi informasi IM. Karena Jabber merupakan protokol yang bersifat *open source*, pencarian *client* dapat secara mudah dilakukan dengan internet. Disini yang akan dijelaskan hal-hal apa saja yang dibutuhkan untuk membangun atau menggunakan sistem keamanan Jabber jika *client* telah tersedia dan memodifikasi kode-kode untuk menambahkan *key exchange* dan melakukan enkripsi.

Untuk membuat *messenger* yang aman dibutuhkan beberapa persyaratan untuk bekerja dengan *client* Jabber secara aman dibatasi dengan beberapa hal berikut:

- Pengetahuan dasar tentang XML
- Melakukan akses terhadap *library-library* kriptografi yang ada
- Siap bekerja dengan *client* Jabber yang bersifat *open source*

4.1 Implementasi pada Pesan Jabber

4.1.1 Tool yang Dibutuhkan

Implementasi keamanan Jabber pada sisi client ini menggunakan bahasa pemrograman Java, hal ini karena Java merupakan bahasa mesin yang independen karena kode-kodenya berjalan pada *Java Virtual Machine (JVM)*. *Library* kriptografi yang digunakan adalah *Java Cryptographic Extension (JCE)* dari *Sun Microsystem*, yang sudah tersedia pada J2SE 1.4.2. *Library-library* JCE ini akan dipakai pada *Java Jabber Client (JJC)*.

4.1.2 Modifikasi XML

Pada dasarnya kita tidak perlu lagi untuk menemukan struktur dokumen XML untuk proses selanjutnya. Struktur pesan dalam *tag* XML sudah cukup untuk melengkapi *task-task* yang dimodifikasi dalam *client* Jabber. Seluruh *task-task* dalam bahasan ini menggunakan *tag* `<message>` karena dibatasi hanya untuk mengamankan *message* dalam protocol Jabber, sedangkan *tag* `<body>` merupakan *tag* yang bersifat opsional. Desain paket *message* memiliki 3 *tag* tambahan yaitu ;

1. `<keyexchange>` : *tag* ini digunakan untuk menginisialisasi mekanisme pertukaran kunci dengan penerima, yang memiliki format :

```
<message from='node@host' to='receiving-ID'>
<keyexchange>
MIIDKTCcApKgAwIBAgIBDDANBgkqhkiG9w0
BAQQFADCB0TELMakGA1UEBhMCWkExFTATBg
NVBAGTDFd1c3R1cm4gQ2FwZTESMBAGA1UEB
xMJQ2FwZSBUb3duMR0wGAYDVQQKEkxFTUaGF3
</keyexchange>
</message>
```

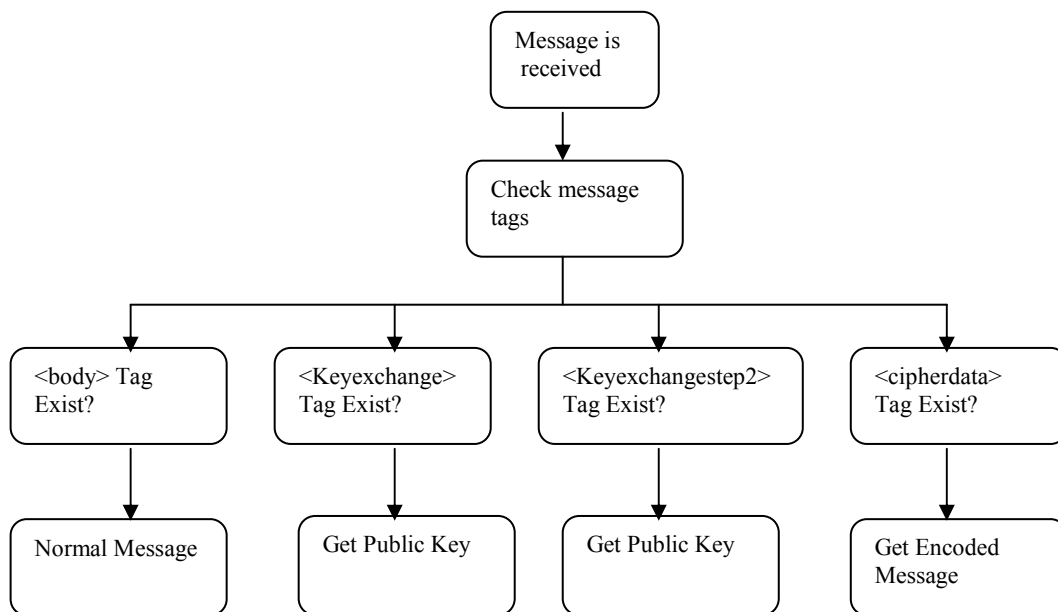
2. **<keyexchangestep2>** : langkah kedua ini ada dalam *key agreement protocol* yang terdiri dari *public key party* yang lain, yang menghasilkan paket **<keyexchange>** diatas. Tag ini mempunyai format sebagai berikut :

```
<message from='node@host' to='receiving-ID'>
<keyexchangestep2>
MIIDKkjasdkhjhHHMmsjhYhsj767wJiG9w0
oksoKKKKjfd8734j2YHSKAWWRffsdvTATBg
</keyexchangestep2>
</message>
```

3. **<cipherdata>** : merupakan tempat pesan-pesan yang telah dienkripsi. *Tag* ini hanya digunakan untuk menyimpan data yang telah dienkripsi dan akan memberikan sinyal dengan *decrypt method* pada sisi *receiver*, dengan formasi sebagai berikut :

```
<message from='node@host' to='receiving-ID'>
<cipherdata>
JJksjfkds763ajsdhfu4ThrY
</cipherdata>
</message>
```

Diagram berikut menunjukkan 4 tipe pesan yang dapat digunakan :



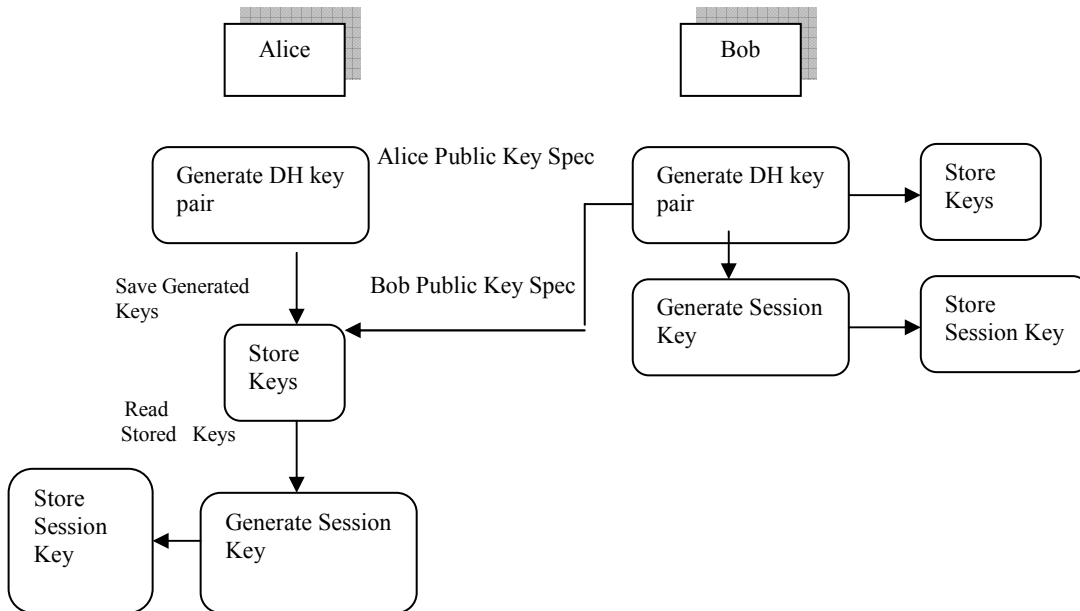
Gambar 4.1. Diagram alir protokol **<message>**

4.1.3 Kode Kriptografi

Library kriptografi yang dibahas dalam makalah ini menggunakan *Java Cryptography Extention (JCE)* yang disediakan oleh *Sun Microsystem*. JCE berisi set paket-paket yang menyediakan *framework* dan implementasi untuk enkripsi, *key generation*, *key agreement*, dan algoritma *Message Authentication Code (MAC)*^[1].

Kegunaan *library* kriptografi yang ditinjau dalam makalah ini dibatasi pada algoritma pertukaran kunci Diffie-Helman dan algoritma DES. *Library* JCE dibundel dengan J2SE

dan J2EE dan kedua *tool* ini tidak dibutuhkan memodifikasi instalasi yang mendukung library JCE. Gambar 4.2 menunjukkan algoritma *key exchange* Diffie-Helman antara 2 *user* :



Gambar 4.2 Diagram alir algoritma *key exchange* Diffie-Hellamn

Kode-kode program yang digunakan untuk melengkapi *key exchange* dan *task-task* yang di enkripsi, dimana *task* tersebut dibagi atas *class-class* untuk proses enkripsi dan kunci yang digunakan.

4.1.3.1 Diffie-Helman *Key Exchange*

Ada 3 *class* yang diperlukan untuk membuat kunci dan menukarkannya :

1. DHKeysCreation.class

Merupakan langkah awal dalam membuat dan menukarkan kunci □rotoc Diffie-Hellman dengan *user* Jabber yang lainnya. Fragmentasi kode dibuat untuk membangkitkan sepasang kunci Diffie-Hellman untuk *user* seperti berikut ini:

```
System.out.println("Creating Diffie-Hellman parameters...");
AlgorithmParameterGenerator paramGen
= AlgorithmParameterGenerator.getInstance("DH");
paramGen.init(512);

AlgorithmParameters params = paramGen.generateParameters();
dhSkipParamSpec = (DHParameterSpec)params.getParameterSpec
(DHParameterSpec.class);
KeyPair Kpair = KpairGen.generateKeyPair();
System.out.println("Initialization ...");

KeyAgreement KeyAgree = KeyAgreement.getInstance("DH");
```

```

KeyAgree.init(Kpair.getPrivate());
// Encodes the public key
byte[] PubKeyEnc = Kpair.getPublic().getEncoded();

```

2. DHKeysCreationStep2.Class

Class ini akan dipanggil ketika kunci publik yang diterima dari pengirim yang telah diinisialisasi *key exchange* dan akan diekstrak material yang ada pada kunci publik, selanjutnya mulai membangkitkan *public key* dan *session key*. Prosedur ini tidak jauh berbeda dengan pembuatan kunci, kecuali dalam hal penyimpanan dan pembacaan kunci publik yang diterima.

Berikut contoh kode program dalam langkah ini :

```

{ ...
String directoryPath =
System.getProperty("user.home")+"//jabberkeys//"+to;
FileInputStream publicKeyFileInputStream = new
FileInputStream(directoryPath+"//"+to+"PublicKey.DH");
byte[]enryptedPublicKey = new
byte[publicKeyFileInputStream.available()];

publicKeyFileInputStream.read(enryptedPublicKey);
publicKeyFileInputStream.close();
KeyFactory kf = KeyFactory.getInstance("DH");
X509EncodedKeySpec x509KeySpec = new
X509EncodedKeySpec(enryptedPublicKey);
PublicKey alicePubKey = kf.generatePublic(x509KeySpec);
...
bobKeyAgree.doPhase(alicePubKey, true);
SecretKey bobDesKey = bobKeyAgree.generateSecret("DES");
ObjectOutputStream keyOut = new ObjectOutputStream(
new FileOutputStream(directoryPath+"//MyDesKey.des"));
keyOut.writeObject(bobDesKey);
keyOut.close();
return encodedData;
}

```

3. DHKeyAgreementPhase1.class

Class ini dieksekusi ketika ada balasan yang diterima dari *receiver end-point* yang memegang kunci publik, karena *user* telah siap dengan kunci publik yang tersedia, maka proses akan membangkitkan *session key* secara langsung dan selanjutnya akan ditransormasikan ke kunci DES yang disimpan dan digunakan jika perlu.

```

\\Generate the session key.
aliceKeyAgree.doPhase(bobPubKey, true);
byte[] aliceSharedSecret = aliceKeyAgree.generateSecret();

\\Use the session key for DES.
aliceKeyAgree.doPhase(bobPubKey, true);
SecretKey aliceDesKey = aliceKeyAgree.generateSecret("DES");

\\Store the DES key.
ObjectOutputStream keyOut = new ObjectOutputStream(
new FileOutputStream(directoryPath+"//MyDesKey.des"));
keyOut.writeObject(aliceDesKey);
keyOut.close();
...

```

4.1.3.2 Enkripsi/Dekripsi DES

Di dalam sistem keamanan *instant messaging* dengan protokol *Jabber* ini hanya satu *class* Java yang dibangkitkan dengan metode enkripsi dan dekripsi DES, yaitu:

1.DesEncryption.class

Dalam class ini seluruh enkripsi dan dekripsi ditempatkan setelah 2 orang *user* menyetujui *session key*. Kunci DES disimpan lebih awal sebelum dibaca dengan *class* Java dan selanjutnya digunakan untuk melakukan enkripsi dan dekripsi berdasarkan argumen yang dilewatkan. Contoh fragmen metode enkripsinya adalah sebagai berikut:

```
{ ...
String directoryPath =
System.getProperty("user.home")+"//jabberkeys//"+to;
ObjectInputStream keyIn = new ObjectInputStream(

new FileInputStream(directoryPath+"//MyDesKey.des"));
Key key = (Key) keyIn.readObject();
keyIn.close();
Cipher bobCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
bobCipher.init(Cipher.ENCRYPT_MODE, key);
byte[] cleartext = msg.getBytes();
byte[] ciphertext = bobCipher.doFinal(cleartext);
encodedData = encoder.encode(ciphertext);
return encodedData;
.... }
```

4.1.4 Kode *Client* Jabber yang *Open Source*

Pesan yang dipertukarkan antara user dan client yang diharapkan adalah bagaimana membangkitkan dan menukarkan kunci serta bagaimana membangun komunikasi yang aman antara dua *user*. Dalam *open source code Jabber client*, integrasi *crypto-code* telah siap digunakan dan tersedia pada kode *Java Jabber Client* yang terbuka dan mudah untuk dipahami bagaimana kode-kode tersebut diterapkan pada protokol Jabber.

Pada umumnya modifikasi *source code* dibuat pada *JabberMessageCenter.java*, kode berikut menampilkan untuk mengecek tag *<cipherdata>* dan jika telah ada, akan dimulai mendekripsi pesan :

```
if (msg.cipherdata != null){
try{
String s = new String(JabberUtil.deXML(msg.cipherdata));
final String nick = jr.getNickname(msg.from);
DesEncryption cipher = new DesEncryption();
String cipherMessage = new String(cipher.decrypt(s,nick));
..... }
}
```

Pada kode tersebut, *msg.cipherdata* ada node XML yang memiliki nama *cipherdata* dan akan dikirim pada *class deXML* yang mendapatkan sebuah nilai pada *node* ini yang akan disimpan pada string (*s*) yang selanjutnya akan ditransfer pada *class DesEncryption* menggunakan metode dekripsi untuk mendekripsi *cipher message*. Selain pada *source code*, modifikasi terhadap GUI bersifat *mandatory* untuk menampilkan *user* yang tersedia atau sedang *online* serta mendukung untuk pertukaran kunci dan enkripsi.

4.2 Analisis dan Pembahasan

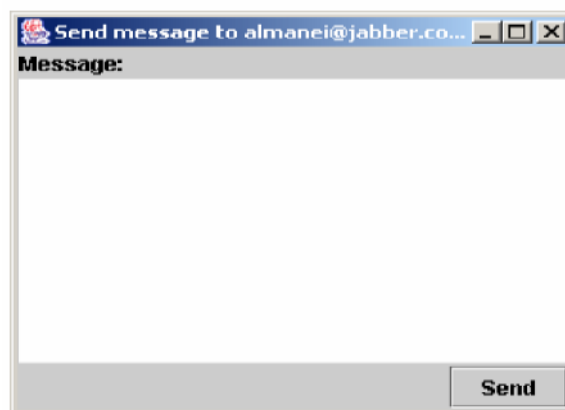
Pembahasan sistem keamanan IM pada protokol Jabber ini menggunakan *library* kriptografi *Java Cryptographic Extention* (JCE) yang bekerja pada *Java Jabber Client* (JJC). Dengan menggunakan *library* kriptografi ini berguna bagi para *developer* untuk memulai desain fungsi-fungsi keamanan protokol Jabber dan memberikan kontribusi bagi komunitas Jabber menghasilkan standar untuk mengamankan lingkungan Jabber yang selalu mengalami perkembangan.

Sistem keamanan yang ditawarkan ini diadopsi dari user yang regular dan peduli terhadap administrator *server* karena seluruh data pesan yang dienkripsi tidak mudah digunakan tetapi memiliki kemungkinan untuk di-*crack* karena menggunakan algoritma DES.

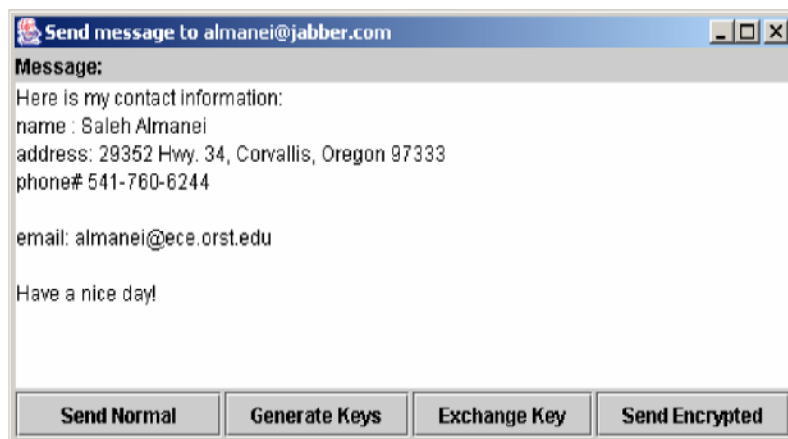
Ada beberapa kerentanan yang akan muncul untuk mengklarifikasi kekuatan sistem keamanan dari keamanan *Java Jabber Security* (JJS). Serangan *intruder-in-the-middle* memungkinkan dalam pertukaran kunci Diffie-Helman, dimana intruder dapat mengacak-acak dua *user*^[12]. Selain itu algoritma DES merupakan algoritma yang sudah kuno dan sudah mulai digantikan dengan *Advance Encryption Standar* (AES). Hal ini memungkinkan untuk memecah algoritma DES yang menjadi persoalan di kemudian hari dan dengan teknologi tingkat lanjut ini akan ditempatkan sebagai pengganti algoritma DES yang aman di masa depan^[13]. Model sistem keamanan yang ditawarkan dalam tugas ini dapat secara mudah dikembangkan untuk mendukung algoritma AES yang baru dan menjadikan algoritma DES sebagai algoritma pilihan atau menjadi algoritma yang telah dimodifikasi yang dikenal dengan *Triple DES*.

4.2.1 Analisis dari sisi *User Interface*

Sistem keamanan yang ditawarkan disini lebih menekankan kepada modifikasi, dimana adanya JJC yang baru yang berbeda dari desain aslinya yang terlihat pada gambar berikut ini. Gambar 4.3 terlihat desain asli pesan yang dikirim, sedangkan pada Gambar 4.4 pesan yang telah dimodifikasi terlihat adanya *Window* yang memperlihatkan adanya pilihan berupa *Send Normal*, *Generate Keys*, *Exchange Keys*, dan *Send Encrypted*.

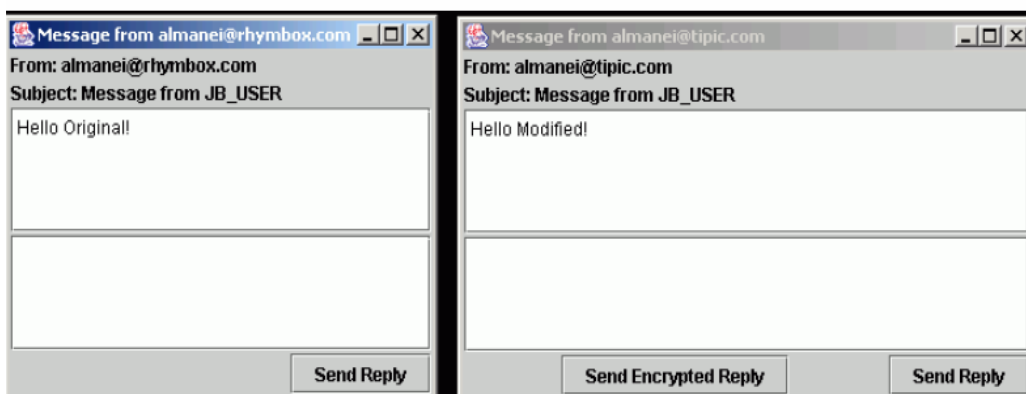


Gambar 4.3 *Window* pesan asli Jabber



Gambar 4.4 *Window* yang menampilkan pesan yang telah dimodifikasi

Selain itu beberapa modifikasi lain yang dibuat dengan menggunakan library kriptografi JJC ini adalah adanya *window* pesan jawaban (*Reply Message Window*) yang telah dimodifikasi dari aslinya dengan adanya tambahan fungsi *Send Encrypted Reply* yang ditampilkan pada Gambar 4.5.



Gambar 4.5 *Reply Message Window* yang asli dan yang telah dimodifikasi

4.2.2 Analisis dari Sisi Fitur

Kemampuan bahwa Jabber *support* terhadap algoritma kriptografi yang simetris maupun asimetris, yang telah dilakukan beberapa modifikasi terhadap pesan aslinya dapat dilihat pada Tabel 4.1 berikut merupakan kapabilitas protokol Jabber yang telah dimodifikasi, dapat dijadikan sebagai pembandingan dari Tabel 3.1 sebelum dilakukan modifikasi.

Tabel 4.1 *Properties* pada teknologi sistem keamanan yang ada saat ini dibandingkan dengan yang *support* Jabber dan telah mengalami modifikasi

| <i>Security Property</i> | <i>Deskripsi</i> | <i>Existing Technology</i> | <i>Jabber Support</i> |
|---------------------------------|--|------------------------------------|--|
| Autentikasi | Memastikan sebuah entitas siapa yang telah mengklaim | JAASI, Kerebos, Microsoft Passport | <i>Jabber Authentication Protocol</i> dan SASL |
| Autorisasi | Menentukan apakah entitas telah mendapat izin untuk mengakses data atau mengontrol sumber daya | JAAS, <i>access control list</i> | Autorisasi biner. User yang tidak diautentikasi dan dijamin benar, dan autentikasi oleh user yang lainnya. |
| Integrity | Memastikan bahwa data tidak dirusak | <i>Message digest</i> | <i>Support</i> terhadap <i>message digest</i> |
| Non-repudiation | Memastikan bahwa <i>author</i> data akan selalu dapat diautentikasi | <i>Digital signature</i> | <i>Support</i> terhadap <i>digital signature</i> |
| Confidentiality | Memastikan bahwa data hanya dapat dibaca oleh entitas yang resmi | Enkripsi | <i>Support</i> terhadap enkripsi secara total |

BAB V

KESIMPULAN

Jika ditinjau dari pangsa pasar sistem keamanan saat ini, Jabber bukanlah kandidat yang bagus untuk sistem IM khususnya dalam bidang industri, hal ini karena masih banyak kekurangan dari segi dukungan sistem keamanan, format data dan kerjasamanya.

Dalam tugas akhir ini telah dibahas tentang sistem keamanan yang mungkin diterapkan pada sistem IM pada sisi *client* protokol Jabber serta telah diimplementasikan bagaimana mengamankan protokol Jabber secara lebih mudah dengan melakukan beberapa modifikasi untuk mendukung beberapa algoritma kriptografi yang dan protokol yang ada seperti *digital certificates*, *digital signature* dan *public key infrastructure*.

Berdasarkan hasil dalam tugas akhir ini, protokol Jabber akan benar-benar sukses jika *developer* Jabber dalam melakukan desain yang dapat mendukung *library-library* kriptografi atau *event-event* yang memiliki standar *library* yang bersifat *open source* sebagai tulang punggung untuk keamanan *task-task* yang berada pada lingkungan Jabber karena protokol Jabber menggunakan format XML.

DAFTAR PUSTAKA

1. <http://www.jabber.org> “*What Is jabber,*” www page, 2003,.
2. Robin Cover, “*IETF Charters Extensible Messaging and Presence Protocol(XMPP) Working Group.*,” WWW page, 2002, <http://xml.coverpages.org/>.
3. Iain Shigeoka, *Instant Messaging in Java The Jabber Protocols*, Manning Publications Co., 2002.
4. Stephen Lee and Terence Smelser, *Jabber Programming*, Hungry Minds, Inc., 2002.
5. J. Miller P. Saint-Andre, “*XMPP Core Draft-IETF-XMPP-Core-12*” www page, May 2003, Expire on November 2, 2003.
6. P. Saint-Andre, “*End-to-End Object Encryption in XMPP,*” www page, May 2003, Expire on November 18, 2003.
7. Paul Lloyd, “*A Framework For Securing Jabber Conversations.*,” www page, 2002, <http://www.jabber.org/>.
8. Naokhiko Uramoto Makoto Murata Andy Clark Yuichi Nakamura Ryo Neyama Kazuya Kosaka Hiroshi Maruyama, Kent Tamura and Satoshi Hada, *XML and Java Second Edition, Developing Web Applications*, Prentic, 2002.
9. Donald E. Eastlake III and Kitty Niles, *Secure XML The new Syntax for Signature and Encryption*, Addison-Wesley, Person Education, 2002.
10. Cay S. Horstmann and Gary Cornell, *Core Java2, Volume II-Advanced Features*, Addison-Wesley, Person Education, 2002.
11. <http://java.sun.com/>. “*Java Cryptography Extension,*” www page, 2003,
12. H.X. Mel and Doris Baker, *Cryptography Decrypted*, Addison-Wesley, Person Education, 2001.
13. Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory*, Prentice-Hall, Inc., 2002.
14. Symantec Security Response, “*Secure Instant Messaging,*” White Paper, May 2003, <http://www.symantec.com/avcenter/whitepapers.html>.