PROTOTYPE SISTEM INFORMASI AKADEMIK DENGAN TEKNOLOGI SINGLE SIGN ON

(STUDI KASUS PADA STIKOM SURABAYA)

Yoppy Mirza Maulana[1)]

1) Program Studi Sistem Informasi, STIKOM Surabaya, email: yoppy@stikom.edu

Abstract: Single Sign On (SSO) adalah sebuah metode kontrol akses yang memungkinkan sebuah user untuk login sekali dan mendapatkan akses ke sistem-sistem perangkat lunak yang berbeda tanpa harus login kembali. SSO menggunakan server otentikasi terpusat dimana semua sistem dan aplikasi menggunakan server tersebut untuk tujuan otentikasi (proses validasi user). Pada Penelitian ini dibuat Prototipe Aplikasi SSO yang dapat dimanfaatkan Sistem Informasi Akademik (SIA) STIKOM Surabaya berbasis web. Prototipe SSO yang akan dibuat menggunakan mekanisme otentikasi yang dikembangkan dari kerberos dan disesuaikan dengan fungsi-fungsi kebutuhan pembuatan aplikasi. Untuk keamanan proses otentikasinya digunakan AES (Advanced Encryption Standard) dengan algoritma Rijndael. Pembuatan sistem informasi ini menggunakan bahasa Microsoft Visual Basic .Net 2005. Dan diharapkan penelitian ini dapat memberikan manfaat terhadap pengembang aplikasi SIA dan sivitas akademika STIKOM Surabaya.

Keywords: Sistem Informasi Akademik, Otentifikasi, *Single Sign On*, *Kerberos*

1. Pendahuluan

STIKOM Surabaya adalah lembaga pendidikan yang bergerak dalam bidang teknologi informasi, yang selalu meningkatkan pelayanannya terhadap mahasiswa khususnya dalam memperoleh informasi akademiknya. Sistem Informasi Akademik (SIA) dikembangkan dari bermacam-macam aplikasi pendukung seperti *Student Integrated Information System* (SIIS), Sistem Informasi Cyber Campus (SICyCa), e-Kelas, e-Learning, dan lain sebagainya.

Tiap aplikasi pendukung tersebut dikembangkan sebagai modul independen sebelum akhirnya disatukan dengan aplikasi lainnya untuk mendukung fungsi SIA di STIKOM. Permasalahan yang timbul dengan pengembangan secara independen ini adalah mekanisme otentikasi terdesentralisasi. Artinya masing-masing aplikasi meminta pemakai untuk melakukan otentikasi agar dapat menggunakan layanan-layanan yang tersedia. Konsekuensinya adalah pemakai harus menyediakan *credential* mereka, yaitu *user id* dan *password*, lebih dari satu kali pada saat akan mengakses aplikasi yang berbeda. Hal ini menyebabkan meningkatkan kompleksitas manajemen identitas dan celah keamanan.

Oleh karena itu, pada penelitian ini penulis membuat mekanisme otentikasi terpusat atau prototipe aplikasi *Single Sign On* (SSO) dimana untuk mengakses aplikasi SIA mahasiswa hanya sekali melakukan otentikasi dengan menggunakan satu identitas saja.

Diharapkan dengan implementasi teknologi SSO, pengembang SIA diberi kemudahan dalam mengelola user seluruh mahasiswanya, dan bagi sivitas akademika STIKOM dapat dengan mudah dalam mengakses berbagai aplikasi SIA dan terjamin keamanannya.

2. Single Sign On (SSO)[(4)]

Single sign-on (SSO) adalah sebuah metode kontrol akses yang memungkinkan sebuah user untuk log in sekali dan mendapatkan akses ke sistem-sistem software yang ada tanpa harus log in kembali. Karena berbagai aplikasi dan sistem tersebut mempunyai mekanisme otentikasi yang berbeda, SSO harus bisa melakukan pengubahan dan menyimpan berbagai *credentials* yang berbeda sesuai dengan penggunaannya. Keuntungan SSO meliputi:

- Mengurangi tingkat kejenuhan user dalam penggunaan password.
- Mengurangi waktu yang digunakan untuk memasukkan password kembali untuk identitas yang sama.
- Mengurangi biaya IT seiring dengan berkurangnya user yang meminta bantuan mengenai password.
- Keamanan di semua tingkat masuk/akses/keluar ke sistem.

SSO menggunakan server otentikasi terpusat dimana semua sistem dan aplikasi menggunakan server tersebut untuk tujuan otentikasi, dan mengkombinasikannya dengan teknik-teknik dimana user tidak perlu mengentrikan *credential* mereka lebih dari 1 (satu) kali.

3. Identifikasi dan Otentikasi[(1)]

3.1 Definisi Identifikasi[(1)]

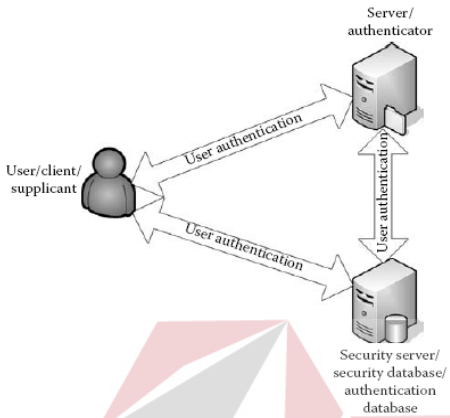
Identifikasi menyediakan identitas user ke sistem keamanan. Identitas ini umumnya disediakan dalam bentuk user ID. Sistem keamanan akan mencari diseluruh objek yang diketahuinya dan menemukan seperangkat *privilege* sesuai dengan user ID tersebut. Setelah selesai, user telah diidentifikasi.

3.2 Definisi Otentikasi[(1)]

Otentikasi adalah proses validasi identitas user. Fakta bahwa user telah direpresentasikan oleh sebuah object (dari user ID-nya). Untuk lebih meyakinkan hal

ini user harus menyediakan bukti bahwa identitas yang diberikan ke sistem adalah benar. Otentikasi adalah proses untuk meyakinkan identitas user dengan memverifikasi bukti yang telah disediakan oleh user.
Ada 3 komponen yang terlibat dalam proses otentikasi seperti tampak pada gambar 1:

1. User
   Pihak yang dalam proses otentikasi menyediakan identitasnya dan bukti, dan sebagai hasil dia yang akan diotentikasi.



Gambar 1: Komponen user dari sistem authentication

2. Otentikator
   Pihak yang dalam proses otentikasi menyediakan resource ke *client* (*user*) dan perlu untuk mengecek identitas user untuk meng-otorisasi dan mengaudit akses user ke resource. Otentikator ini bisa juga disebut sebagai server.
3. Security authority / database:
   Tempat penyimpanan atau mekanisme untuk mengecek *credential* user. Bisa berupa file, atau server otentikasi terpusat, atau seperangkat server otentikasi terdistribusi yang menyediakan sistem untuk otentikasi user dalam perusahaan atau internet.

4. AES (Advanced Encryption Standard)[5]

National Institute of Standards and Technology (NIST) Pada tahun 1972 dan 1974 menerbitkan permintaan publik untuk pembuatan standar enkripsi. Pada saat itu adalah DES (*Data Encryption Standard*). DES adalah sebuah algoritma kriptografi simetrik yang mempunyai panjang kunci 56 bit dan blok data 64 bit. Dengan semakin majunya teknologi, para kriptografer merasa bahwa panjang kunci untuk DES terlalu pendek, sehingga keamanan algoritma ini dianggap kurang memenuhi syarat. Untuk mengatasi hal ini, akhirnya muncul triple DES.

Namun teknologi yang tidak pernah berhenti berkembang *Triple* DES juga kurang memenuhi syarat dalam standar enkripsi. Akhirnya NIST mengadakan kompetisi untuk standar kriptografi yang terbaru, yang dinamakan AES (*Advanced Encryption Standard*). Dari hasil kompetisi tersebut, NIST memilih 5 finalis AES, yaitu : Mars, RC6, Rijndael, Serpent, dan Twofish. Kompetisi ini akhirnya dimenangkan oleh Rijndael dan secara resmi diumumkan oleh NIST pada tahun 2001.

Input dan output dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data yang telah terbentuk dalam satu kelompok 128 bit tersebut disebut juga sebagai blok data atau *plaintext* yang kemudian akan dienkripsi menjadi *ciphertext*. Cipher key dari AES terdiri dari key yang memiliki panjang 128 bit, 192 bit, atau 256 bit.

5. Kerberos[3]

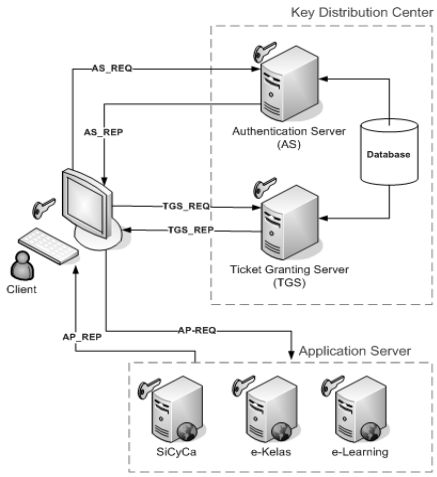
Model Kerberos sebagain dibuat berdasarkan protokol otentikasi pihak ketiga oleh *Needham* and *Schroeder's* dengan modifikasi seperti yang disarankan oleh *Denning* dan *Sacco*. Rancangan asli dan implementasi kerberos versi 1 sampai 4 karya dua mantan anggota *staff project athena*, *Steve Miller* dari *DEC* dan *Clifford Neuman* dari *University of Southern California*, bersama dengan *Jerome Saltzer Technical Director of Project Athena* dan *Jeffrey Schiller*, *MIT Campus Network Manager*. Banyak anggota dari project athena juga memberikan kontribusi terhadap kerberos yang versi empatnya sekarang dapat ditemukan di berbagai di internet.

Kerberos menyediakan suatu cara memastikan identitas user (misalnya woskstation atau server jaringan) pada jaringan terbuka (tidak terlindungi). Hal ini dicapai dengan tidak mengandalkan pada otentikasi oleh sistem operasi host, tanpa mempercayai alamat host tanpa perlu pengamanan fisik di jaringan, dan dengan asumsi bahwa setiap paket melwati jaringan dapat dibaca, diubah dan ditambah sewaktu-waktu (catatan meskipun banyak aplikasi fungsi kerberos hanya pada tahap inisialisasi. Pada koneksi jaringan dan mengasumsikan tidak ada kehadiran ”pembajak” yang mungkin sewaktu-waktu dapat mengambil alih koneksi tersebut. Pemakain semacam ini mempercayai alamat host). Kerberos melakukan otentikasi dengan kondisi tersebut diatas sebagai layanan otentikasi pihak ketiga menggunakan kriptograpi konvesional, contohnya shared secret key. sesuatu rahasia yang hanya pengguna dan server yang mengetahuinya. Menunjukkan identitas dilakukan tanpa pengguna harus membuka rahasia tersebut. Ada suatu cara untuk membuktikan bahwa kita mengetahui rahasia tersebut tanpa mengirimnya ke jaringan.

6. Deskripsi Umum Aplikasi

Perangkat lunak ini dikembangkan berdasarkan konsep sistem Kerberos dan disesuaikan dengan fungsi-fungsi kebutuhan pembuatan aplikasi. Dalam mendukung prototipe aplikasi SSO yang dikembangkan berdasarkan konsep sistem kerberos ini, pada proses otentikasinya digunakan algoritma enkripsi simetris yaitu AES dengan key 256 bit.

Aplikasi yang dibangun dalam penelitian ini adalah sebuah aplikasi dimana mahasiswa dapat mengakses beberapa *server* aplikasi (SiCyca, e-Kelas, e-Learning) dengan sekali otentikasi, seperti tampak pada gambar 2:



Gambar 2 : Gambaran Umum Sistem

Pada aplikasi ini user diminta untuk memasukkan flashdisk yang didalamnya terdapat *secret key client* dan kemudian dibaca oleh aplikasi untuk proses otentikasi sampai dapat mengakses layanan *application server* seperti SiCyca, e-Learning dan e-Kelas. Berikut adalah rancangan tiap-tiap proses pada prototipe aplikasi SSO.

7. Perancangan Aplikasi

1. Perancangan pembuatan database

Pada pembuatan prototipe aplikasi SSO ini terlebih dahulu dibuat suatu database KDC menggunakan SQL Server 2005, yang digunakan untuk menampung dua tabel yaitu :

1. Tabel UserClient digunakan untuk menyimpan data user yang berisi IdClient dan Secret Key, seperti tampak pada tabel 1.

Tabel 1 : Tabel User Client

Field	TipeData	Keterangan
IdClient	Varchar(11)	Identitas Client
SecretKey	Varchar(256)	Secret Key

2. Tabel Server digunakan untuk menyimpan data server yang berisi IdServer, Secret Key, dan nama Url, seperti tampak pada tabel 2.

Tabel 2 : Tabel Server

Field	TipeData	Keterangan
IdServer	Varchar(50)	Identitas Server
SecretKey	Varchar(256)	Secret Key
UrlName	Varchar(500)	Nama url Server

3. Pada *Application Server* juga dibuat database history menggunakan SQL Server 2005, untuk menampung dua buah tabel yaitu serverkey dan ticket. Serverkey yang digunakan untuk menyimpan data server yang berisi IdServer dan Secret Key seperti tampak pada tabel 3 dan database history yang digunakan untuk menampung Tabel Tiket yang digunakan untuk menyimpan history dari user mahasiswa dalam mengakakses *application server*. Pada tabel tiket ini disimpan parameter url, Id Server, status, Id Client, waktu akses *client*, waktu berlaku tiket, waktu logout dan IPAddress Client, seperti tampak pada tabel 4.

Tabel 3 : Tabel ServerKey

Field	TipeData	Keterangan
IdServer	Varchar(11)	Identitas Client
SecretKey	Varchar(256)	Secret Key

Tabel 4 : Tabel Ticket

Field	TipeData	Keterangan
ParamUrl	Varchar(900)	Parameter URL
Idv	Varchar(50)	Id Server
Status	Char(1)	Status Tiket
IdClient	Varchar(11)	Identitas Client
Waktuakses	datetime	Waktu Akses Client
MasaTiket	datetime	Massa Berlakunya Tiket
WaktuLogout	datetime	Waktu Logout
IPAddressClient	Varchar(50)	Alamat Client

2. Perancangan proses otentikasi

Pada tahap ini, user meminta AS untuk mendapatkan *Ticket Granting Ticket*. Didalam Permintaan ini adalah Id Client, gabungan antara Id Client, MacAddress, IpAddress, dan Id Tgs (AS) dan *timespan* yang dienkripsi menggunakan *secret key client* dan dikirim dalam bentuk base64, seperti tampak pada gambar 3:

$$\text{AS\_REQ} = ID_c || E(K_c, [ID_c || ID_{tgs} || TS_1])$$

Gambar 3 : Proses Otentikasi

Ketika pesan masuk, sebelumnya AS memeriksa apakah IdClient ada di database KDC. Jika tidak ada, maka pesan error dikirimkan kepada pengguna. Jika ada, maka AS akan memproses sebagai berikut:

1. Secara random, AS akan membuat *session key* yang akan menjadi rahasia antara pengguna dan TGS, sebutlah ($K_{c,tgs}$).
2. AS membuat *Ticket Granting Ticket*. Di dalamnya terdapat Id Client, alamat client (MacAddress dan IpAddress), Id TGS, tanggal dan waktu, lifetime,

dienkrip berdasarkan *session key* ($K_{c,tgs}$), dan semuanya dienkrip oleh *secret key* tgs, sebutlah $Ticket_{tgs}$.

3. Merespon client dengan mengirimkan Id TGS, timespan, liftime dan beberapa id layanan (idv) beserta $Ticket_{tgs}$ dan dienkripsi berdasarkan *session key* ($K_{c,tgs}$) dan semua pengiriman dienkrip menggunakan *secret key client* (K_c) kemudian di kirim dalam bentuk base64.

AS_REP
$=(K_c,[K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Idv||Ticket_{tgs}])$

$Ticket_{tgs}=E(K_{tgs},[K_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2])$

3. Perancangan proses mendapatkan tiket

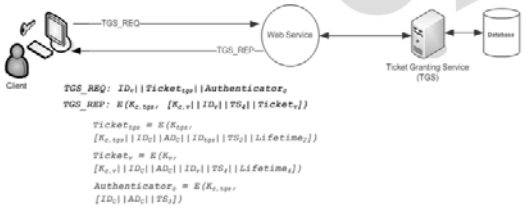
Pada tahap ini seperti pada gambar 4, pengguna yang sudah terotentikasi ingin mengakses layanan tetapi belum mempunyai tiket yang sesuai, maka *client* mengirimkan (TGS_REQ) kepada *Ticket Granting Service* dengan konstruksi sebagai berikut :

1. Membuat *authenticator* berdasarkan *Id Client*, timestamp dari mesin pengguna kemudian mengenkripsi semuanya dengan *session key* yang dibagi bersama dengan TGS ($K_{c,tgs}$).
2. Membuat paket permintaan yang berisi: Id layanan tidak dienkripsi, *Ticket Granting Ticket* yang sudah dienkripsi menggunakan key dari TGS, dan authenticator yang baru saja dibuat.

$TGS\_REQ = ID_v||Ticket_{tgs}||Authenticator_c$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2])$

$Authenticator_c = E(K_{c,tgs}, [ID_C||AD_C||TS_3])$



Gambar 4 : Proses mendapatkan tiket

Selanjutnya ketika pesan sebelumnya tiba, pertama-tama TGS memverifikasi bahwa Id dari layanan yang diminta ada di dalam database KDC. Jika ada, TGS membuka TGT dan mengekstrak session key TGS yang digunakan untuk mendekripsikan *authenticator*. Agar layanan tiket dapat dibuat, TGS memeriksa apakah Id Client yang terdapat pada authenticator cocok dengan yang ada pada TGT.

Kondisi-kondisi tersebut di atas membuktikan bahwa TGT benar-benar kepunyaan pengguna yang melakukan permintaan dan kemudian TGS memulai proses sebagai berikut:

1. Secara random, TGS membuat *session key* yang akan menjadi rahasia antara pengguna dengan layanan, sebutlah $K_{c,v}$.
2. TGS membuat layanan tiket yang di dalamnya berisi (Id Client, MacAddres dan IPAddress), id layanan, tanggal dan waktu, lifetime, dan $K_{c,v}$. Tiket ini disebut $Ticket_v$.

TGS mengirim pesan balasan yang berisi : $Ticket_v$. yang baru saja dibuat dan dienkripsi menggunakan *secret key* milik layanan (K_v), Id layanan, *timestamp*, dan session key yang baru ($K_{c,v}$), semuanya dienkripsi menggunakan session key ($K_{c,tgs}$) yang diekstrak dari TGT.

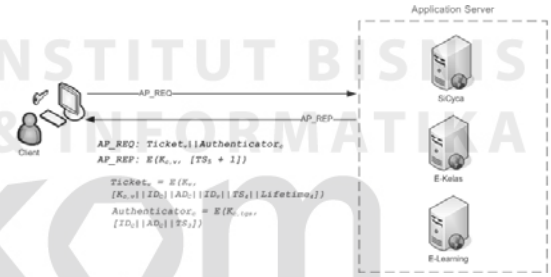
$TGS\_REP : E(K_{c,tgs}, [K_{c,v}||ID_v||TS_4||Ticket_v])$

$Ticket_v = E(K_v, [K_{c,v}||ID_C||AD_C||ID_v||TS_4||Lifetime_4])$

4. Perancangan proses mendapatkan tiket

Pada tahap ini seperti pada gambar 5, user yang mempunyai *credential* untuk mengakses layanan, dapat meminta server layanan agar ia dapat mengakses sumber daya melalui pesan AP_REQ. Tidak seperti pesan-pesan sebelumnya dimana KDC terlibat, AP_REQ adalah dimana user dapat menggunakan *credentialnya* untuk membuktikan identitasnya kepada *server*. Adapun langkah-langkahnya sebagai berikut:

1. User membuat *authenticator* yang berisi (Id Client, MacAddress, IpAddress) dan *timestamp*, dan semuanya dienkripsi menggunakan *session key* layanan ($K_{c,v}$).



Gambar 5 : Proses mengakses SIA

2. Membuat paket permintaan yang berisi tiket layanan $Ticket_v$ yang dienkripsi menggunakan secret key layanan (K_v) dan authenticator yang baru saja dibuat.

AP_REQ = $Ticket_v||Authenticator_c$

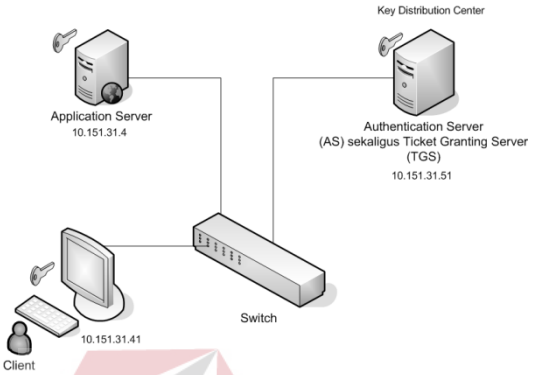
Ketika permintaan sebelumnya tiba, server layanan membuka tiket menggunakan secret key layanan (K_v) yang digunakan untuk mengekstrak session key ($K_{c,v}$) yang digunakan untuk layanan mendekripsikan authenticator. Untuk mengotentikasi user, maka server memeriksa kondisi-kondisi sebagai berikut:

1. Client yang ada dalam authenticator cocok dengan yang ada di tiket
2. MacAddress dan IpAddress (diekstrak dari tiket) diperiksa apakah MacAddress dan IpAddress sama dengan yang ada pada tiket dan Authenticator.

8. Uji Coba dan Evaluasi

1. Lingkungan Pelaksanaan Uji Coba

Pada sub bab ini akan dijelaskan mengenai lingkungan uji coba prototipe aplikasi SSO yang terdiri dari Pc *Client*, PC *Server Key Distribution Center* (AS dan TGS), dan PC *Application Server* seperti yang terlihat pada gambar 6. Selain perangkat keras juga digunakan dalam uji coba ini adalah perangkat lunak yang dapat di lihat pada tabel 5.



Gambar 6 : Skenario Uji coba Single Sign On

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengujian ini dapat dilihat pada tabel 5.

Tabel 5 : Lingkungan Pelaksanaan Uji Coba

Spesifikasi PC Server *Authentication* dan *Ticket Granting Ticket*	Prosesor : Memory : Hardisk : Sistem Operasi : Bahasa Pemrograman: Web Server :	Intel Pentium 4, 2.26GHz 2 GB 40 GB Windows XP Professional SP2 .Net (web-service) Internet Information Services
Spesifikasi PC Server *Application Server*	Prosesor : Memory : Hardisk : Sistem Operasi : Bahasa Pemrograman Web Server :	Intel Pentium 4, 2.26GHz 2 GB 40 GB Windows Server 2003 ASP. Net Internet Information Services
Spesifikasi PC Client	Prosesor : Memory : Hardisk : Sistem Operasi : Bahasa Pemrograman:	Intel Celeron M Processor 370 512 MB 40 GB Windows XP Professional SP2 VB.Net

2. Uji Coba Proses Otentikasi

Uji coba proses otentikasi adalah proses validasi user terhadap AS berdasarkan secret key-nya. Pada uji Coba ini secret key yang digunakan adalah milik id client: 99410104001 dengan SecretKey: "f85OqGMgsYKPkoD0kXrmX1jEgWG/+g==". Pada setiap pengiriman paket dari client, paket dilindungi dengan enkripsi.

Ada dua hal uji coba proses otentikasi yang dilakukan pada saat pengiriman paket ke AS

1. Mengirimkan paket yang benar (paket yang dienkripsi secara benar oleh *authenticator) atau belum mendapatkan modifikasi dari pihak lain* seperti pada gambar 7.

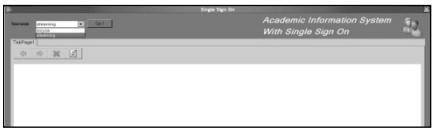
Gambar 7 : Paket yang benar dikirim ke AS

Apabila paket tersebut valid maka AS akan merespon client dengan memberikan tiket untuk mengakses TGS. Respon AS dalam bentuk enkripsi dan

dikirim dalam bentuk base64, seperti tampak pada gambar 8.

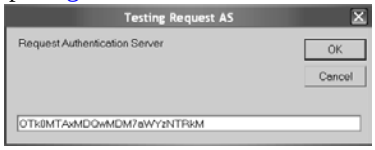
Gambar 8 : Response AS terhadap Client untuk paket yang benar

Kemudian response AS diterjemahkan dalam program bahwa user mendapatkan id layanan yang digunakan untuk meminta tiket kepada TGS, seperti tampak pada gambar 9.



Gambar 9 : mendapatkan Id layanan

2. Mengirimkan paket yang telah dimodifikasi (mendapatkan serangan *Man In The Middle Attack*). Misalkan kita mengubah paket yang benar dengan menghapus beberapa karakter dari paket yang ada seperti pada gambar 10.



Gambar 10 : Paket yang telah diubah

Apabila paket tersebut tidak valid maka AS akan merespon client yang diterjemahkan oleh program seperti tampak pada gambar 11.

Gambar 11 : Respon AS terhadap paket yang salah

3. Uji Coba Proses Meminta Tiket kepada TGS

Pada proses ini setelah mendapatkan akses terhadap *Ticket Granting Service* maka client dapat meminta layanan SIA dan TGS mengenerate-kan tiket yang akan digunakan digunakan client untuk mengakses layanan SIA.

Ada dua hal uji coba proses meminta tiket yang dilakukan pada saat pengiriman paket ke TGS.

1. Mengirimkan paket yang benar (paket yang dienkripsi secara benar oleh *authenticator) atau belum mendapatkan modifikasi dari pihak lain* seperti pada gambar 12.

Gambar 12 : Paket yang benar dikirim ke TGS

Apabila paket tersebut valid maka TGS akan merespon client dengan memberikan tiket untuk mengakses SIA. Respon TGS dalam bentuk enkripsi dan dikirim dalam bentuk base64, seperti tampak pada gambar 13.

Gambar 13 : Response TGS terhadap *client* untuk paket yang benar

Kemudian respon TGS diterjemahkan dalam program untuk mendapatkan tiket yang digunakan dalam mengakses layanan server aplikasi. Adapun format tiket seperti tampak pada gambar 14.

```
http://10.151.31.51/elearning/default.aspx?parameter=d0Raek
5LdVJ5TVlxTEZzcmRNVzk1Y2FYb3FsT1VxOW40N1QxY
kVOcGk1clJrN0N3VUhnRWpaK0luaDZWSzd1bFArWkpFT
G40T3BrRWIydE5oSkxXb0s2L1BRbzhZWHJxdmZ0SUd4Z
ElYWERhVDFWczB1MHFLRUcvL01JLzVOMkw7MGdWc
3Y4UFdxajJVMUZNYjZ6cmhQcGxrT1c4RkprcGFxby9UQ
UJ2NndsLzd1SXI4STdvbmI1VGdDdTlaYzR4cmZraUR1Skp
LbjlUTUxEV0xobUE1YlE9PQ==
```

Gambar 14 : Format tiket yang digunakan dalam mengakses layanan server aplikasi

2. Mengirimkan paket yang telah dimodifikasi (mendapatkan serangan *Man In The Middle Attack*). Misalkan kita mengubah paket yang benar dengan menghapus beberapa karakter dari paket yang ada seperti pada gambar 15.

Gambar 15 : Paket yang telah diubah

Apabila paket tersebut tidak valid maka TGS akan merespon client yang diterjemahkan oleh program seperti tampak pada gambar 16.

Gambar 16 : Respon TGS terhadap paket yang salah

4. Uji Coba Proses Mengakses Layanan SIA

Setelah membuat aplikasi SIA maka akan dilakukan uji coba, aplikasi ini akan berjalan jika mendapatkan request dari prototipe aplikasi SSO. Pada uji coba ini client request pada layanan SIA yang sebelumnya mendapat hasil genarate dari TGS. Pada saat mengakses SIA ada tiga hal yang dilakukan uji coba.

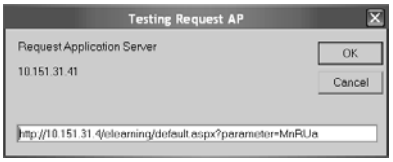
1. Mengirimkan tiket yang benar (tiket yang dienkripsi secara benar oleh *authenticator) atau belum mendapatkan modifikasi dari pihak lain* seperti pada gambar 17.

Gambar 17 : Tiket yang benar dikirim ke Server Aplikasi

Apabila tiket tersebut valid maka server aplikasi akan merespon client dengan memberikan layananya.

Gambar 18 : Layanan SIA yang di request oleh client

2. Mengirimkan tiket yang telah dimodifikasi (mendapatkan serangan *Man In The Middle Attack*). Misalkan kita mengubah tiket yang benar dengan menghapus beberapa karakter dari tiket yang ada seperti pada gambar 19.



Gambar 19 : Tiket yang telah dimodifikasi

Apabila tiket tersebut tidak valid maka TGS akan merespon client.

Gambar 20 : Tiket tidak dapat digunakan

3. Tiket yang benar dapat dimanfaatkan pihak lain (Serangan *Replay Attack*). Pada gambar 18 menunjukkan tiket yang digenerate oleh id client 99410104001 pada komputer dengan IP address 10.151.31.41. Tiket ini telah digunakan oleh Id tersebut sampai mendapatkan layanan SIA, apabila digunakan oleh pihak lain maka tiket tersebut tidak dapat digunakan, seperti tampak pada gambar 4.20. Hal ini dikarenakan pada aplikasi server adanya proses mencocokan antara authenticator dengan tiket yang digunakan dan kemudian setelah benar dibuatlah *session id* yang unik dari web server, jadi ketika digunakan pihak lain tiket tidak dapat digunakan.

7. Simpulan

Dari pelaksanaan uji coba pada bab sebelumnya, didapatkan kesimpulan sebagai berikut :

1. Dalam mengakses SIA melalui prototipe aplikasi SSO ini, dapat berjalan dengan baik jika penguna dalam mengakses aplikasi tersebut menggunakan *secret key*nya yang telah terdaftar pada database KDC.
2. Serangan *Man-In-The-Middle Attacks* terhadap prototipe aplikasi SSO ini, dapat dicegah karena setiap proses paket yang dikirim dalam bentuk enkripsi, ketika ada paket yang dimodifikasi maka paket tersebut tidak dapat didekripsi.
3. Serangan *Replay Attack* terhadap prototipe aplikasi SSO ini, dapat dicegah karena setiap proses paket yang dikirim dalam bentuk enkripsi, kemudian dibandingkan antara authenticator dengan tiket yang digunakan, setelah benar dibuatkan *session id* untuk setiap *client* yang mengakses layanan tersebut.

8. Daftar Rujukan

[01] Dobromir Todorov, *Mechanics of User Identification and Authentication*, 2007.

[02] *How To: Encrypt and Decrypt Data Using a Symmetric (Rijndael) Key* (C#/VB.NET) http://www.obviex.com/samples/Encryption.aspx

[03] Kohl, J. Neuman, C. *The Kerberos Network Authentication Service (V5)*. September 1993. http://www.ietf.org/rfc/rfc1510.txt

[04] *Single Sign on*, http://en.wikipedia.org/wiki/Single_sign-on

[05] Wihartantyo Ari Wibowo, *Advanced Encryption Standard*, algoritma *rijndael*, http://www.cert.or.id/~budi/courses/ec5010/projects/wihartantyo-report.doc

[06] William Stallings, *Cryptography and Network Security Principles and Practices, Fourth Edition*, November 16, 2005.