Analisa Implementasi *Single Sign On* Pada *Learning Management System* dan *Internet Protocol Television*

Ragil Widiharso, Achmad Affandi, Djoko Suprajitno Rahardjo.
Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember
Kampus ITS, Sukolilo, Surabaya 60111
Email : ragil.widiharso09@mhs.ee.its.ac.id

***Abstrak* ---- Berkembangnya teknologi *e-learning* saat ini mengarah pada pemanfaatan teknologi lain untuk menunjang *Learning Management System* (LMS). *Internet Protocol Television* (IPTV) merupakan salah satu teknologi yang dapat dimanfaatkan untuk menunjang LMS. Baik LMS maupun IPTV mempunyai sistem otentikasi tersendiri yang berbeda satu sama lain. Untuk mengintegrasikan teknologi IPTV berbasis *web* dalam lingkup LMS, diperlukan sistem *Single Sign On* (SSO), sebuah metode untuk memberikan ijin pada pengguna dalam mengakses beberapa aplikasi sekaligus tanpa harus *login* berulang kali. Pada tugas akhir ini untuk integrasi aplikasi LMS dan IPTV akan digunakan metode SSO berbasis *Central Authentication Service (*CAS) yang memanfaatkan *directory* pada struktur *Lightweigh Directory Access Protocol* (LDAP) untuk manajemen *user*, sistem tersebut akan dibangun di atas platform *Hypertext Transfer Protocol Secure* (HTTPS). Untuk mengetahui kinerja *server* CAS dan LDAP dalam melayani *user*, dilakukan *load test* pada *Local Area Network* (LAN) dengan variasi *bandwith* 64 kbps, 128 kbps, 256 kbps, 512 kbps, 1024 kbps, dan tanpa batasan *bandwith*.**

Kata kunci : CAS, LDAP, LMS, IPTV

I. PENDAHULUAN

Semakin berkembangnya kompleksitas LMS saat ini, mengarah pada pendekatan desain LMS yang memanfaatkan teknologi lain yang terpisah dengan LMS itu sendiri. Teknologi lain yang digunakan sebagai alat untuk mendukung LMS tersebut terpisah secara fisik dan bukan bagian dari LMS, alat tersebut bisa berupa aplikasi *web stand alone* yang digunakan oleh LMS untuk menyediakan *e-learning*. [1]

Salah satu teknologi yang dapat diterapkan pada LMS salah satunya adalah *Internet Protocol Television* (IPTV). IPTV merupakan suatu layanan yang memberikan konten audio visual dan juga interaktif berbasis *Internet Protocol* (IP). Layanan IPTV menawarkan sebuah kesempatan interaksi yang lebih baik dengan *user* dan layanan yang ditawarkan lebih atraktif [2].

Baik LMS dan IPTV mempunyai sistem otentikasi tersendiri untuk masuk kedalam sistem masing-masing. Yang mana dibutuhkan minimal dua *credential user* (*username* dan *password*) untuk bisa mengakses layanan-layanan tersebut. Banyaknya *credential user* ini menyebabkan ketidak-efektifan dalam sisi pengelolaan *database,* dan juga *user* diharuskan untuk menghafal banyak identitas untuk mengakses *resource* dari masing-masing aplikasi. *Multiple ID* yang diperlukan untuk mengakses aplikasi *web* ini dapat diminimalisir dengan metode *Single Sign-On* (SSO), sebuah metode dimana *user* hanya butuh sekali *log in* untuk bisa mengakses *Network resources* yang ada. [3]

Karena proses otentikasi pada SSO melibatkan *credential user*, perlu koneksi yang aman untuk menyokong SSO. Koneksi yang aman disini adalah koneksi terenkripsi, dimana *username* dan *password* yang dimasukkan oleh *user* telah terenkripsi. Ini untuk melindungi pemilik akun dari pembajakan. Koneksi *client-server* yang terenkripsi dapat menggunakan protokol *Hypertext Transfer Protocol Secure* (HTTPS). HTTPS merupakan kombinasi antara protokol *Hypertext Transfer Protocol* (HTTP) dan *Secure Socket Layer* (SSL) [4]. Untuk membangun suatu koneksi HTTPS diperlukan suatu *key* dan sertifikat digital. Membangkitkan *key* dan sertifikat merupakan langkah kritis untuk mambangun sistem SSO.

Pada tugas akhir ini akan dilakukan implementasi metode SSO berbasis *Central Authentication Service* (CAS) sebagai pusat otentikasi dan *Lighweight Directory Access Protocol* (LDAP) untuk management *user* untuk mengakses layanan LMS berbasis *Modular Object Oriented Dynamic Learning Environment* (moodle) dan *web based* IPTV. Sehingga hanya perlu sebuah *user identity* untuk dapat mengakses LMS berbasis moodle dan *web based* IPTV. Proses komunikasi data antara *client* dengan *server* SSO akan dibangun di atas platform protokol keamanan HTTPS.

Pemilihan metode SSO berbasis CAS yang digunakan pada tugas akhir ini didasarkan bahwa CAS merupakan metode SSO yang mendukung library dari *client* untuk PHP, bahasa pemrograman yang digunakan untuk membuat *web* based IPTV. Dan juga telah terdapat integrasi dengan moodle, jenis LMS yang digunakan dalam tugas akhir ini.

II. DASAR TEORI

2.1 Central Authentication Service (CAS)

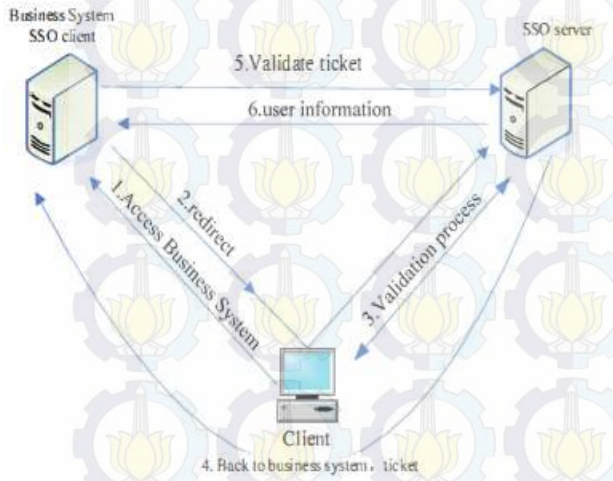
CAS merupakan protokol SSO yang tujuannya adalah untuk memberikan ijin pada pengguna dalam mengakses beberapa aplikasi, sekaligus menyediakan *credential* pengguna (seperti *user id* dan *password*) hanya sekali, dan mengizinkan aplikasi *web* untuk

meng-otentikasi pengguna tanpa mendapatkan akses ke *security credential* pengguna. CAS kemudian dikembangkan sebagai sebuah *software open source* dengan komponen *server* Java dan mendukung *library* dari *client* untuk Java, PHP, Perl, dan lainnya.[5]

Secara singkat prinsip kerja dari SSO berbasis CAS dapat dijelasakan sebagai berikut [6]:

1. *Client* mengakses SSO *client* (*business system*).
2. SSO *client* akan *redirect browser client* ke SSO *server*.
3. *User* akan memasukkan *username* dan *password* untuk otentikasi.
4. *User* akan melewati proses otentikasi dan kembali ke SSO *client* dengan sebuah tiket.
5. SSO *client* untuk mengkonfirmasi apakah pengguna sah.
6. Pengguna sah dapat mengakses informasi yang terdapat pada SSO *client*.

Skema ini berlaku untuk semua aplikasi yang terintegrasi dengan *server* CAS

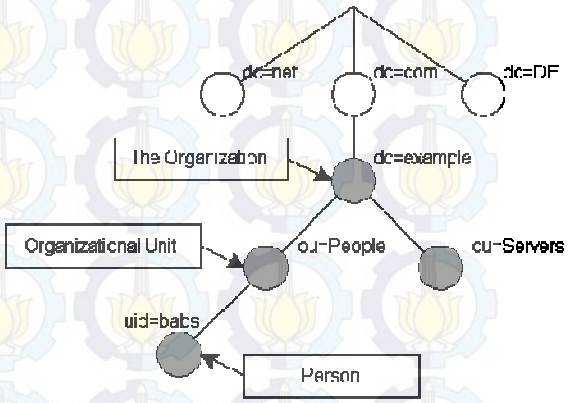


Gambar 2.1 Skema Single Sign On (SSO) [6]

2.2 *Lightweight Directory Access Protocol* (LDAP)

Lightweight Directory Access Protocol (LDAP) merupakan pengembangan dari protokol X-500. LDAP adalah sebuah protokol yang mengatur mekanisme pengaksesan layanan direktori (*Directory Service*) yang dapat digunakan untuk mendeskripsikan banyak informasi seperti informasi tentang *people, organizations, roles, services*, dan banyak entitas lainnya. LDAP menggunakan model *client-server*, dimana *client* mengirimkan *identifier* data kepada *server* menggunakan protokol TCP/IP dan *server* mencoba mencarinya pada DIT (*Directory Information Tree*) yang tersimpan di *server*. Dalam terminologi komputer, *directory service* bisa dikatakan sebagai suatu database tempat penyimpanan data, yang dapat di gunakan untuk memberikan informasi yang berkaitan dengan objeknya. Bagian direktori berisi kumpulan informasi tentang *user* seperti *sure name, first name, phone number, User ID, mail address* dll.

Suatu *directory service* akan memiliki item yang dijadikan sebagai root. Untuk sebuah titik root, secara umum di tunjukkan dengan suatu attribut dc (*Domain Component*) atau o (*Organization*) mungkin juga ou (*Organization Unit*). Kemudian pada titik daun (*leaf*) biasanya akan berisi item dengan atribut uid (*User ID*) ataupun cn (*Common Name*). *Directory service* biasanya menyimpan informasi dalam bentuk struktur tree yang dinamakan *Directory Information Tree* (DIT). Untuk alamat relatif sering disebut sebagai RDN (*Relative Distinguish Name*) sedangkan alamat yang absolut disebut DN (*Distinguish Name*) [7].



Gambar 2.3 Contoh *Directory Tree* pada LDAP [7]

2.3 *Secure Socket Layer* (SSL)

Protokol SSL memfasilitasi enkripsi untuk data yang rahasia dan membantu menjamin integritas informasi yang ditukarkan antara *web*site dan *web browser*. SSL beroperasi antara protokol komunikasi TCP/IP (*Transmission Control Protocol / Internet Protocol*) dan aplikasi. SSL bertindak sebagai layer baru antara lapisan transport (TCP) dengan aplikasi. SSL dipanggil ketika sebuah referensi dimulai dengan **https://**, *browser* menginisiasi sebuah sesi kepada *server* pada port TCP/443. SSL akan menegosiasikan *link* yang aman dan transfer data di atasnya. Jika negosiasi gagal, maka tidak ada data yang ditransfer. Dalam perkembangannya, protokol SSL berubah menjadi *Transport Layer Security* (TLS).

SSL disusun oleh dua sub-protokol [4]:

- *SSL Handshaking*, yaitu sub-protokol untuk membangun koneksi (kanal) yang aman untuk berkomunikasi.
- *SSL record*, yaitu sub-protokol yang menggunakan koneksi (kanal) yang sudah aman. SSL *record* membungkus seluruh data yang dikirim selama koneksi.

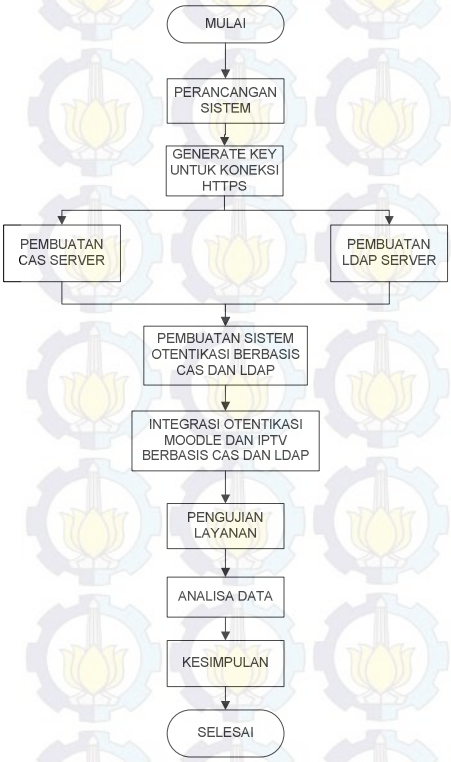
III. PERANCANGAN SISTEM

Sistem SSO pada tugas akhir ini menggunakan sebuah *server* otentikasi CAS yang berfungsi untuk melakukan otentikasi *user* serta memberikan Service Ticket (ST) kepada *client*. Dengan tiket ini *user* dapat melakukan akses ke beberapa aplikasi yang memerlukan otentikasi *user* yang terhubung dengan sistem. LMS pada tugas akhir ini menggunakan *Modular Object Oriented Dynamic Learning Environment* (moodle), kemudian IPTV yang digunakan adalah *web* based IPTV.

Manajemen data *user* yang digunakan dalam sistem SSO ini meggunakan struktur direktori dari *Lightweight Directory Access Protocol* (LDAP), untuk itu perlu dibangun sebuah *server* LDAP yang

menangani manajemen *user* untuk sistem SSO ini. *Server* CAS akan memvalidasi *user* menggunakan directory yang ada pada LDAP. Pada *server* LDAP nanti perlu konfigurasi untuk mendeskripsikan *user*.

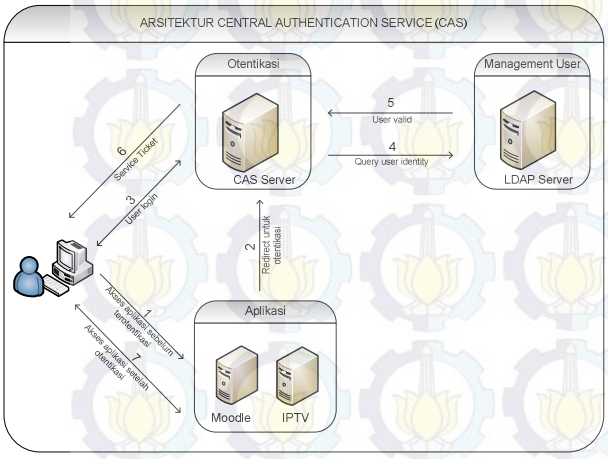
Berikut diagram alir pengerjaan tugas akhir ini :



Gambar 3.1 Flowchart perancangan dan implementasi

3.1 Arsitektur Sistem

Dalam tugas akhir ini, terdapat dua bagian sistem. Bagian pertama adalah sistem otentikasi dan bagian kedua adalah aplikasi yang berbasis *web*. Pada sistem otentikasi terdapat dua bagian, yaitu *server* CAS dan *server* LDAP. Sedangkan pada sistem aplikasi terdapat LMS berbasis moodle dan juga *web* based IPTV.



Gambar 3.2 Arsitektur sistem

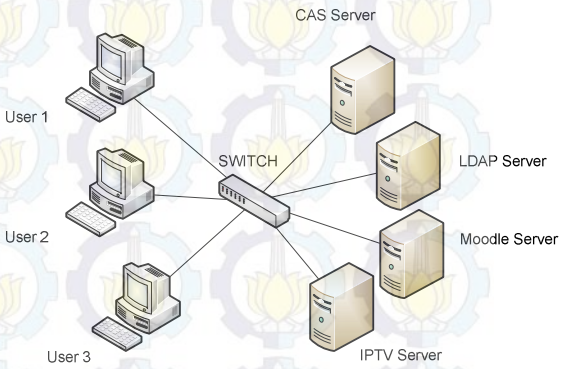
Ketika *user* pertama kali mengakses sebuah aplikasi, dalam hal ini moodle ataupun *web* based IPTV, *user* akan di-*redirect* ke halaman *login* CAS untuk melakukan proses otentikasi. Ketika *user* memasukkan identitas pada halaman CAS, *server* CAS akan mengecek validitas dari identitas tersebut pada *server* LDAP. Jika *username* dan *password* sesuai, maka *server* LDAP akan memberitahukan pada *server* CAS bahwa identitas tersebut eksis. *Server* CAS kemudian akan meng-generate Service Ticket (ST) untuk kemudian dikirimkan ke *user*.

Setelah otentikasi berhasil, *user* akan di-*redirect* kembali pada aplikasi pertama yang diaksesnya dengan membawa ST yang telah didapat dari CAS *server*. Jika *user* ingin mengakses aplikasi lain, *user* tidak perlu melakukan *login* lagi karena sudah mempunyai ST yang tersimpan dalam cookies *browser*. Dalam hal ini diasumsikan bahwa *user* menggunakan *browser* yang sama ketika melakukan otentikasi dan belum di tutup.

Sistem ini dibangun dengan pola komunikasi gabungan *client-server* dan *server-server* yang diamankan dengan protokol security TLS yang melalui jalur komunikasi TCP/IP, komunikasi *client-server* terjadi antara *web browser user* ke *server* CAS, sedangkan komunikasi *server-server* terjadi saat terjadi proses otentikasi dari *server* CAS dengan *server* LDAP untuk validasi *user*. Semua koneksi antar pada sistem ini melalui koneksi https pada port 443.

3.2 Konfigurasi Jaringan

Pada tugas akhir ini menggunakan *Local Area Network* (LAN), dimana *user* dengan *server* SSO dan *server* aplikasi berada pada satu jaringan dengan menggunakan switch untuk menghubungkan masing-masing end device. Konfigurasi jaringan yang digunakan pada tugas akhir ini:



Gambar 3.3 Konfigurasi jaringan

3.3 Konfigurasi *Transport Layer Security* (TLS)

Untuk dapat membuat koneksi https, diperlukan certificate dan *key* untuk mengenkripsi data yang ditransmisikan. Untuk perusahaan atau korporasi besar, biasanya membeli *certificate* dan *key* dari perusahaan security. Tetapi *certificate* dan *key* tersebut dapat di generate sendiri. Pembangkitan sertifikat dan *key* ini berkaitan dengan protokol SSL. Untuk sistem operasi windows, certificate dan *key* SSL dapat di-*generate* dari *Java Virtual Machine*. Berikut perintahnya

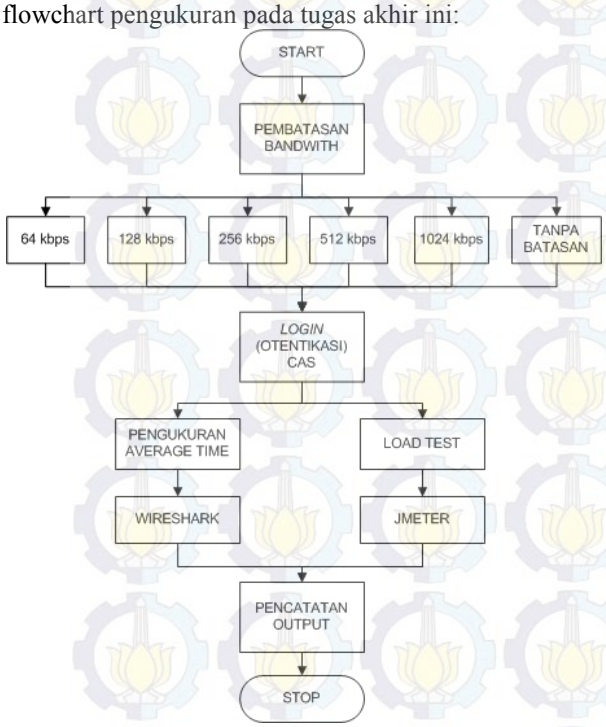
```
C:\Program Files\Java\jdk1.7.0_01\bin> keytool -
genkey -alias tomcat -keypass changeit -keyalg RSA
```

Selanjutnya akan ada beberapa pertanyaan yang harus dijawab. Pertanyaan tersebut merupakan suatu mekanisme untuk mendefinisikan sertifikat tersebut

digunakan oleh siapa. Dalam menjawab pertanyaan perlu diperhatikan bahwa untuk mengisi first and last name harus disamakan nama domain yang akan dipakai, misal menggunakan domain contoh.net. Maka yang diisikan untuk menjawab first and lastname adalah contoh.net. Ini berdasarkan pengalaman penulis dalam pengerjaan tugas akhir ini.

3.4 Metode Pengukuran

Untuk mengetahui performa dari *server* SSO akan dilakukan *load test* untuk menguji kemampuan sistem. Juga akan dicari waktu rata-rata yang dibutuhkan oleh *server* SSO untuk melayani *user*. Beberapa parameter teknis dari *load test* yang akan dianalisa diantaranya adalah *response error* dan *average time* dari *server* CAS dan LDAP ketika menerima banyak *request* otentikasi dalam waktu yang bersamaan. Semua pengujian dilakukan pada *Local Area Network* (LAN), dengan memvariasikan lebar *bandwith*. Berikut flowchart pengukuran pada tugas akhir ini:



Gambar 4.1 Flowchart pengukuran

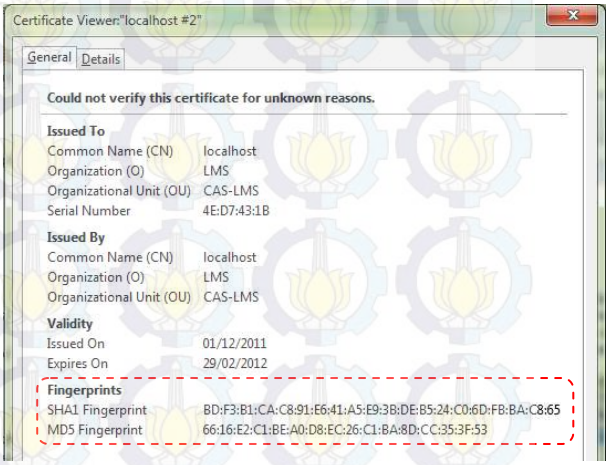
Pengukuran *response time* adalah pengukuran waktu yang dibutuhkan oleh *server* SSO dalam melayani satu *user* agar dapat terotentikasi. Sedangkan *load test* digunakan untuk mengetahui kemampuan *server* SSO dalam melayani *user* ketika diberi beban berlebih. Beban berlebih dalam hal ini adalah ketika ada banyak *request* otentikasi yang datang dalam waktu yang bersamaan. Baik pengukuran *average time* dan *load test* akan dilakukan dengan memvariasikan lebar *bandwith* untuk mengetahui kinerja dari *server* SSO pada lingkungan yang berbeda, lingkungan dalam hal ini adalah lebar *bandwith*.

Dalam pengukuran *average time*, pada tugas akhir ini menggunakan *software packet sniffing*, wireshark, dengan menjumlahkan durasi yang yang dibutuhkan antara *client* dengan *server* dari mulai paket pertama hingga paket terakhir yang saling ditukarkan. Sedangkan untuk pengukuran *load test*, digunakan software Apache JMeter. JMeter merupakan aplikasi java untuk memuat perilaku uji fungsional, mengukur kinerja, menguji *server* atau *script* atau perilaku objek, dibawah beban berat. Bisa juga digunakan untuk mensimulasikan beban berat pada jaringan. Dalam hal ini, JMeter dapat membangkitkan *request* yang banyak dalam waktu yang bersamaan. Untuk pembatasan *bandwith*, pada tugas akhir ini digunakan *software Traffic Shaper XP*

IV. ANALISA DAN PENGUKURAN

4. 1 Uji Fungsional TLS

Seperti dijelaskan sebelumnya bahwa koneksi baik antara *server* CAS dengan *server* LDAP ataupun *user* dengan *server* CAS, menggunakan *Transport Layer Security* (TLS). Dengan meng-*generate key* dan sertifikat sendiri, perlu dilihat apakah *key* dan sertifikat tersebut dapat berfungsi. Untuk mengetahuinya adalah dengan mengakses halaman CAS melalui koneksi https, kemudian mengecek apakah *browser* berkomunikasi dengan *server* menggunakan *key* dan sertifikat yang telah dibangkitkan sebelumnya.



Gambar 4.2 Tampilan *browser* yang menunjukkan *Key* dan *Certificate* yang digunakan.

Dari gambar 4.2 terlihat bahwa *key* dan sertifikat yang digunakan adalah *key* dan sertifikat yang ditujukan untuk "*Local*host" dengan nama organisasi "LMS" dan unit organisasi "CAS-LMS". *Key* dan sertifikat dibuat pada tanggal 01/12/2011 dan akan kadaluarsa pada tanggal 29/02/2012. Berarti bahwa *key* dan sertifikat yang dibangkitakan telah berfungsi.

4. 2 *Response Time*

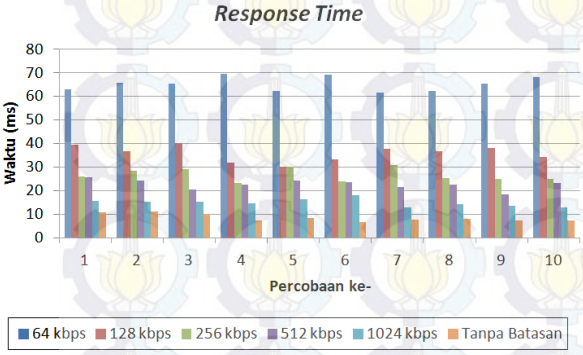
Pengujian *response time* dilakukan untuk mengetahui seberapa lama *server* SSO dalam melayani satu *user* untuk terotentikasi. Pengujian dilakukan dengan mengakses halaman *login* CAS. *User* melakukan *login*, kemudian wireshark akan merekan durasi waktu yang dibutuhkan oleh CAS dari ketika *user* mengklik tombol *login* hingga masuk ke halaman utama, dimana *user* berarti telah terotentikasi oleh *server* SSO.

Pengukuran *response time* ini dilakukan dalam lebar *bandwith* yang berbeda-beda, yaitu; 64 kbps, 128 kbps, 256 kbps, 512 kbps, 1024 kbps, dan tanpa batasan *bandwith*. Penentuan besarnya lebar *bandwith*

yang digunakan dalam tugas akhir ini adalah karena rata-rata *bandwith* yang ditawarkan oleh provider layanan internet untuk masyarakat umum berkisar pada angka tersebut. Pengukuran *response time* dilakukan 10 kali, ini dilakukan hanya untuk memperbanyak sample, sehingga nanti didapatkan rata-rata yang cukup akurat.

Tabel 4.1 Tabel hasil pengukuran *response time*

Sample ke-	*Response Time* (ms)					
	64 kbps	128 kbps	256 kbps	512 kbps	1024 kbps	100 Mbps
1	63,0875	39,629	26,1494	25,586	15,5243	10,7692
2	65,6939	36,5374	28,49	24,416	15,1455	10,998
3	65,4699	40,2914	29,0513	20,423	15,1149	9,9203
4	69,5627	31,9552	23,221	22,41	14,4722	7,4186
5	62,1446	30,1865	30,2996	24,31	16,2174	8,3569
6	69,0381	33,1434	23,9436	23,468	18,1664	6,561
7	61,4525	37,7011	30,8114	21,454	12,6766	7,7146
8	62,1164	36,614	25,3379	22,622	14,0736	7,957
9	65,4672	38,1741	24,9863	18,234	13,6537	7,4089
10	68,149	34,3107	24,9863	23,221	12,9058	7,3942
Rata-rata	65,22	35,854	26,73	22,62	14,795	8,45



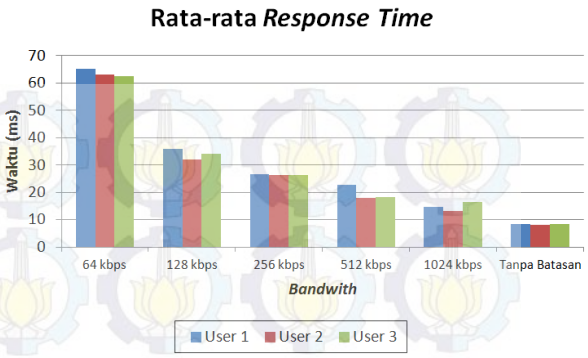
Gambar 4.1 Grafik *response time*

Dari grafik 4.1 dan gambar 4.1 terlihat bahwa *response time* cenderung lebih cepat ketika *bandwith* juga lebih lebar. Ketika lebar *bandwith* sebesar 64 kbps, rata-rata *response time* 65,22 ms. Sedangkan ketika lebar *bandwith* sebesar 1024 kbps, rata-rata *response time* 14,795 ms, 4 X lebih cepat.

Kemudian pengujian *response time* dilakukan dengan menggunakan *user* yang berbeda. Jadi pengujian sama seperti tabel 4.1, akan tetapi menggunakan nama *user* yang berbeda. Setelah itu dihitung nilai rata-ratanya dan kemudian dibandingkan dengan pengukuran yang menggunakan nama *user* lain.

Tabel 4.2 Tabel perbandingan rata-rata *response time*

	Average Response Time (ms)					
	64 kbps	128 kbps	256 kbps	512 kbps	1024 kbps	100 Mbps
User 1	65,2182	35,8543	26,7277	22,6141	14,795	8,4499
User 2	63,0813	31,9941	26,3407	18,032	13,080	8,057
User 3	62,443	34,0112	26,2723	18,3746	16,465	8,4923



Gambar 4.2 Rata-rata *response time* dari tiga *user* yang berbeda

Dari pengukuran pada tugas akhir ini, didapat bahwa rata-rata antara *user* 1, *user* 2, dan *user* 3 tidak begitu jauh berbeda. Dan juga memiliki kecenderungan yang sama dengan bahwa semakin lebar *bandwith*, maka *response time* akan semakin cepat.

4.3 *Load Test*

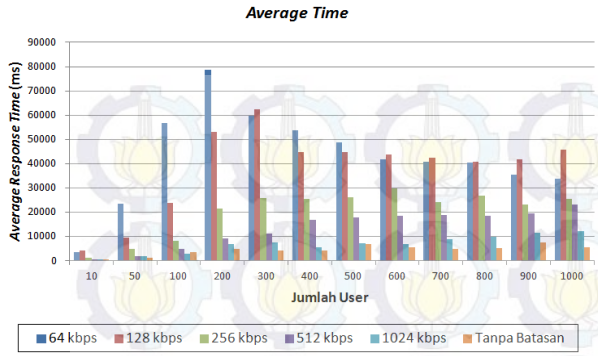
Load test dilakukan untuk mengetahui kemampuan *server* SSO jika diberi beban berlebih, sehingga nantinya akan diketahui pada keadaan yang seperti apa *server* SSO dapat bekerja dengan normal. Jadi pada *load test* ini, *server* SSO akan diberi banyak *request* untuk otentikasi dalam waktu yang bersamaan.

Dalam *load test* ini digunakan untuk JMeter. Pada JMeter dapat dibangkitkan *request* dalam jumlah banyak dengan selang waktu yang dapat di atur. *Average response time* akan langsung dimunculkan oleh JMeter.

Pengukuran *average response time* ini untuk mengetahui seberapa lama *server* SSO melayani *request* yang datang. Jadi ketika ada banyak *request* otentikasi, berapa lama *server* SSO akan melayani seluruh otentikasi tersebut. Berikut hasilnya:

Tabel 4.3 Tabel *average response time* saat *load test*.

Jumlah *Request*	Waktu (ms)					
	64 kbps	128 kbps	256 kbps	512 kbps	1024 kbps	100 Mbps
10	3591	4266	978	389	328	272
50	23574	9296	4640	1711	1818	1292
100	56771	23705	8164	4725	2891	3495
200	78817	53125	21583	8985	6856	4880
300	59784	62373	25657	11065	7343	4106
400	53658	44883	25373	16906	5469	4009
500	48852	44731	26235	17731	6991	6817
600	41883	43718	29666	18611	6779	5374
700	40780	42397	23980	18858	8847	4846
800	40338	40797	26632	18549	9631	5275
900	35497	41750	23165	19325	11542	7561
1000	33695	45778	25599	23011	12082	5325



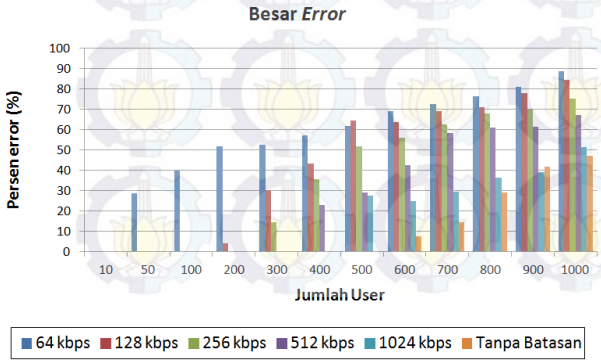
Gambar 4.3 *Average response time* pada *load test*

Pada saat lebar *bandwith* 64 kbps, *response time* tertinggi berada pada saat 200 *request* bersamaan. Untuk lebar *bandwith* 128 kbps, *response time* tertinggi saat 300 *request* bersaman. Untuk lebar *bandwith* 512 kbps dan 1024 kbps, *response time* tertinggi saat 1000 reequest bersamaan. Dan untuk tanpa batasan *bandwith* terjadi saat 900 *request* bersamaan.

Average response time pada *load test* tersebut juga dipengaruhi oleh persentase *error*. Jadi ketika *average response time* cepat, belum tentu bahwa semua otentikasi tersebut berhasil, harus dilihat juga besarnya *error* dari jumlah *request* tersebut. *Average response time* yang cepat mungkin juga bahwa otentikasi tersebut gagal atau tidak dilayani. Untuk mengetahui besarnya persentase *error*, dapat dilihat tabel 4.4.

Tabel 4.4 Persentase *error* saat *load test*

Jumlah *Request*	Persentase *Error*					
	64 kbps	128 kbps	256 kbps	512 kbps	1024 kbps	100 Mbps
10	0	0	0	0	0	0
50	28,5	0	0	0	0	0
100	40	0	0	0	0	0
200	51,67	4	0	0	0	0
300	52,34	30,33	14,33	0	0	0
400	57	43,25	35,5	22,75	0	0
500	61,83	64,4	51,6	29,2	27,4	0
600	69,1	63,67	56,17	42,67	24,67	7,4
700	72,47	69,14	62,71	58,14	29,43	14,3
800	76,4	70,88	67,75	60,88	36,25	29,02
900	80,9	77,89	69,78	61,33	39,22	41,6
1000	88,5	84,5	75,1	67	51,3	47,23



Gambar 4.4 Persentase *error* pada *load test*

V. KESIMPULAN

Lebar *bandwith* mempengaruhi besarnya rata-rata *response time* satu *user* untuk terotentikasi pada *server* SSO, semakin besar lebar *bandwith*, maka *response time* akan semakin cepat.

Ketika semakin banyak *request* otentikasi yang masuk pada *server* SSO secara bersamaan, maka akan semakin banyak *request* yang gagal terotentikasi. Pada lebar *bandwith* 64 kbps, *server* SSO mengotentikasi 10 *request* bersamaan tanpa *error*. Ketika lebar *bandwith* 128 kbps, 100 otentikasi bersamaan tanpa *error*. Pada 256 kbps, 200 otentikasi bersamaan tanpa *error*. Pada 512 kbps, 300 otentikasi bersamaan tanpa *error*. Pada 1024 kbps, 400 otentikasi bersamaan tanpa *error*. Dan ketika tanpa batasan *bandwith* mampu melayani 500 otentikasi secara bersamaan.

REFERENCES

[1] Gonzalez, J., Rodriguez, M., Nistal, M., Rifon, L., "*Reverse OAuth: A Solution To Achieve Delegated Authorizations In Single sign-on e-Learning Systems*". Journal of Computer & *Security*, Volume 28, Page 843-856. 2009.

[2] Dai-Boong Lee, Hwangjun Song., *"QoE-aware mobile IPTV channel control algorithm over WiMAX Network"*, Journal of Visual Communication and Image Representation, Volume 21, Issue 3, Pages 245-25, April 2010.

[3] Nurdeni, Deden A., *"Implementasi Teknologi SSO di Lingkungan Teknik Informatika ITS"*, Tugas Akhir. Jurusan Teknik Informatika ITS. Surabaya. 2010.

[4] http://www.cisco.com/*web*/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html.

[5] http://www.ibm.com/developerworks/*web*/library/wa -singlesign/

[6] Hu, Jian., Sun, Qizhi., Chen Hongping., *"Application of Single sign-on (SSO) in Digital Campus"*. International Conference Broadband *Network* and Mul*time*dia technology (IC-BNMT). Pages 725 – 727. Beijing 2010

[7] Sari, Riri Fitri., Hidayat, Syarif., *"Integrating Web Server Applications With LDAP Authentication: Case Study on Human Resource Information System of UI"*. International Symposium on Communications and Information Technologies (ISCIT). Pages 307 – 312. Bangkok. 2006

RIWAYAT PENULIS

Ragil Widiharso, lahir di Pati pada 8 Maret 1989. Memulai pendidikannya di SD Negeri Gabus 1 tahun 1994, kemudian melanjutkan ke SLTPN 2 Pati dan lulus 2003. Melanjutkan pendidikan di SMAN 1 Pati lulus pada tahun 2006. Penulis menempuh studi pada Program Diploma Teknik Elektro, Fakultas Teknik, Universitas Gadjah Mada, Yogyakarta. Menyelesaikan studi Diploma pada 2009. Penulis melanjutkan pendidikan untuk meraih gelar Sarjana di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember melalui program Lintas Jalur pada tahun yang sama.