**PEMBANGUNAN *SINGLE SIGN-ON* UNTUK PENGINTEGRASIAN APLIKASI LAYANAN PUBLIKKOTA BANDUNG**

**Laporan Tugas Akhir**

**Disusun sebagai syarat kelulusan tingkat sarjana**

**Oleh**

**Luqman Arifin Siswanto**

**NIM: 13513024**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2016**

**PEMBANGUNAN *SINGLE SIGN-ON* UNTUK PENGINTEGRASIAN APLIKASI LAYANAN PUBLIKKOTA BANDUNG**

**Laporan Tugas Akhir**

**Oleh**

**Luqman Arifin Siswanto**

**NIM: 13513024**

**Program Studi Teknik Informatika**

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Bandung, 3 Januari 2017

Mengetahui,

Pembimbing,

Adi Mulyanto, ST., MT.

NIP. 19631126 198803 1 002

DAFTAR ISI

[DAFTAR ISI iii](#_Toc520315832)

[DAFTAR GAMBAR vi](#_Toc520315833)

[DAFTAR TABEL vii](#_Toc520315834)

[BAB I PENDAHULUAN 1](#_Toc520315835)

[I.1 Latar Belakang 1](#_Toc520315836)

[I.2 Rumusan Masalah 4](#_Toc520315837)

[I.3 Tujuan 4](#_Toc520315838)

[I.4 Batasan Masalah 5](#_Toc520315839)

[I.5 Metodologi 5](#_Toc520315840)

[I.6 Sistematika Pembahasan 6](#_Toc520315841)

[I.7 Jadwal Pelaksanaan Tugas Akhir 6](#_Toc520315842)

[BAB II STUDI LITERATUR 8](#_Toc520315843)

[II.1 Autentikasi dan Autorisasi 8](#_Toc520315844)

[II.2 Sistem Autentikasi Identitas 9](#_Toc520315845)

[II.2.1 *Password* 9](#_Toc520315846)

[II.2.2 *Token* 10](#_Toc520315847)

[II.2.3 *Biometrics* 12](#_Toc520315848)

[II.2.4 Tanda Tangan Digital 12](#_Toc520315849)

[II.2.5 Sertifikat Digital 12](#_Toc520315850)

[II.3 *Multi-Factor Authentication* 13](#_Toc520315851)

[II.4 *Single Sign-On* 13](#_Toc520315852)

[II.5 *Web Single Sign*-*On* 17](#_Toc520315853)

[II.5.1 *Auth0* 17](#_Toc520315854)

[II.5.2 WSO2 *Identity Server* 20](#_Toc520315855)

[II.5.3 *Central Authentication Service (CAS)* 24](#_Toc520315856)

[II.5.4 *Java Open Single Sign-On (JOSSO)* 26](#_Toc520315857)

[II.6 Pekerjaan Terkait 27](#_Toc520315858)

[II.6.1 *CAS++* 27](#_Toc520315859)

[II.6.2 Kristanto, 2011 28](#_Toc520315860)

[BAB III ANALISIS PERMASALAHAN DAN RANCANGAN SOLUSI 30](#_Toc520315861)

[III.1 Analisis Permasalahan 30](#_Toc520315862)

[III.1.1 Auth0 30](#_Toc520315863)

[III.1.2 WSO2 Identity Server 31](#_Toc520315864)

[III.1.3 Central Authentication Server 31](#_Toc520315865)

[III.1.4 JOSSO 31](#_Toc520315866)

[III.2 Analisis Solusi 31](#_Toc520315867)

[III.3 Rancangan Solusi 31](#_Toc520315868)

[III.3.1 Desain *Web* *Single Sign-On* 31](#_Toc520315869)

[III.3.2 Desain Dokumentasi *Single Sign-On* 31](#_Toc520315870)

[III.3.3 Desain Aplikasi *Service Provider* 31](#_Toc520315871)

[BAB IV IMPLEMENTASI DAN EVALUASI 32](#_Toc520315872)

[IV.1 Lingkungan Implementasi 32](#_Toc520315873)

[IV.2 Implementasi 32](#_Toc520315874)

[IV.2.1 *Web Single Sign-On* 32](#_Toc520315875)

[IV.2.2 Dokumentasi *Single Sign-On* 32](#_Toc520315876)

[IV.2.3 Aplikasi *Service Provider* 32](#_Toc520315877)

[IV.3 Evaluasi 32](#_Toc520315878)

[IV.3.1 Skenario Uji *Web Single Sign-On* 33](#_Toc520315879)

[IV.3.2 Kemudahan Integrasi 33](#_Toc520315880)

[IV.3.3 Keamanan 33](#_Toc520315881)

[IV.3.4 Ketergantungan terhadap Teknologi Tertentu 33](#_Toc520315882)

[IV.3.5 Kemampuan Autentikasi 33](#_Toc520315883)

[IV.3.6 Kemampuan Autorisasi 33](#_Toc520315884)

[IV.3.7 Kemampuan Manajemen Akun 33](#_Toc520315885)

[IV.3.8 Integrasi Sosial Media 33](#_Toc520315886)

[IV.3.9 Dukungan Dokumentasi 33](#_Toc520315887)

[DAFTAR PUSTAKA 34](#_Toc520315888)

DAFTAR GAMBAR

[Gambar ‎II.1 Sistem yang Mekanisme Autentikasinya tanpa SSO 15](#_Toc520246088)

[Gambar ‎II.2 Sistem dengan SSO 16](#_Toc520246089)

[Gambar ‎II.3 Cara Kerja Auth0 19](#_Toc520246090)

[Gambar ‎II.4 Mekanisme Kerja CAS 21](#_Toc520246091)

[Gambar ‎II.5. Sequence diagram untuk SAML 26](#_Toc520246092)

[Gambar ‎II.6. Diagram *Sequence* untuk SAML ketika user sudah terautentikasi 27](#_Toc520246093)

[Gambar ‎II.7. Diagram keterhubungan antara aplikasi, Auth0, dan user. 28](#_Toc520246094)

[Gambar ‎II.8. Tampilan halaman Auth0 Hosted Lock 29](#_Toc520246095)

[Gambar ‎II.9. Diagram *sequence* Auth0 apabila user mengunjungi app pertama kalinya. 30](#_Toc520246096)

[Gambar ‎II.10. Tampilan halaman Auth0 Hosted Lock bila user pernah *login* sebelumnya 31](#_Toc520246097)

[Gambar ‎II.11. Diagram *sequence* Auth0 untuk kunjungan berikutnya. 31](#_Toc520246098)

[Gambar ‎II.12. Diagram *sequence* OpenID Connect. 33](#_Toc520246099)

[Gambar ‎III.1. Diagram *sequence* *workflow* general *single sign-on* 35](#_Toc520246100)

DAFTAR TABEL

[Tabel ‎I.1. Rencana Pelaksanaan Tugas Akhir I 6](#_Toc520246101)

[Tabel ‎III.1. Aspek yang perlu dikaji dari *single sign-on* sesuai kebutuhan fungsional 39](#_Toc520246102)

[Tabel ‎III.2.Perbedaan antar implementasi *single sign-on* 39](#_Toc520246103)

# PENDAHULUAN

## Latar Belakang

Indonesia menggunakan salah satu negara yang memiliki jumlah pengakses internet terbesar di dunia. Pada Januari 2016, total populasi Indonesia mencapai 259,1 juta orang (Kamil, 2016). Dari ratusan penduduk Indonesia, 88 juta orang merupakan pengguna internet aktif, atau sekitar 34% dari penduduk total. Bahkan menurut data yang bersumber dari UN and US Census Bureau, *mobile connection* di Indonesia berjumlah 326,3 juta koneksi (126% dari jumlah penduduk), bahkan melebihi dari jumlah penduduk Indonesia sendiri. Artinya tiap orang bisa saja memiliki lebih dari satu koneksi *mobile*. Hal ini menandakan bahwa penggunaan teknologi sudah cukup menyebar di kalangan masyarakat ramai. Dengan intensitas penggunaan internet yang sangat tinggi di Indonesia, bisa dikatakan perilaku penduduk Indonesia sudah beralih ke teknologi. Ini menyebabkan tantangan baru kepada masyarakat Indonesia secara umum, apakah penduduk Indonesia bisa memanfaatkan fakta penggunaan internet ini sebagai *advantage*?

Tren saat ini yang terjadi di dunia adalah implementasi *smart city* pada kota-kota di dunia. Definisi *smart city* di sini adalah usaha pengintegrasian teknologi informasi dan internet untuk mengelola aset kota, termasuk layanan publik. Integrasi data dan aplikasi dalam pengelolaan *smart city* akan mengundang banyak kelebihan, misalnya lebih mudah bagi pemerintah kota nantinya untuk memonitor di level eksekutif dalam pengambilan keputusan. Dengan pengintegrasian sistem layanan publik, seluruh data dan aplikasi akan terletak di satu tempat yang sama sehingga lebih mudah dikontrol dan dimonitor.

Pemerintah Kota Bandung merupakan salah satu kota yang sedang menggalakkan program *smart city*. Salah satu roadmap yang menjadi bagian program *smart city* Bandung adalah layanan publik. Layanan publik ini mencakup kependudukan, perijinan, pengaduan, perpajakan, industri, bisnis, dan publikasi informasi. Dalam program layanan publik, diharapkan teknologi informasi didukung internet mampu mengkatalis kebutuhan masyarakat lewat program *smart city* ini.

Layanan publik dalam *smart city* Bandung meliputi kependudukan, perijinan, pengaduan, perpajakan, bisnis, publikasi. Layanan ini diimplementasi dalam bentuk *service*. Masing-masing layanan akan direpresentasikan menjadi satu aplikasi yang independen. Artinya akan ada aplikasi kependudukan, aplikasi perijinan, dan seterusnya.

Sementara itu, tren pengembangan perangkat lunak berevolusi dari waktu ke waktu. Di era 1980-an, orang cenderung mengembangkan aplikasi di *desktop*, pengembangan aplikasi berbasis *web* belum populer pada saat itu. Sementara itu, di masa kini, hampir semua aplikasi yang digunakan sudah berbasis *web* dan terkoneksi ke jaringan internet, walaupun tidak dipungkiri aplikasi *desktop* saat ini masih tetap digunakan.

Dengan bergesernya *trend* pengembangan aplikasi menuju ke arah berbasis internet, kini banyak juga yang perusahaan yang mengembangkan SaaS (*Software as a Service*) (Goth, 2008). Produk yang ditawarkan oleh perusahaan dapat diakses secara *real-time* dan langsung melalui internet. Banyak bidang yang menawarkan produknya melalui internet, seperti misalnya *e-commerce*, *banking*, *travel agent*, *e-government*, penyedia jasa infrastruktur, dan lain lain.

Banyak institusi yang menyediakan banyak *service* aplikasi sekaligus. Setiap aplikasi tentu membutuhkan *credential* untuk autentikasi dan autorisasi. Tidak efisien apabila tiap aplikasi dalam satu vendor yang sama memiliki mekanisme autentikasi sendiri-sendiri (Volchkov, 2001). Ini mengakibatkan redundansi *credential* yang sebenarnya tidak diperlukan.

Institusi biasanya memiliki lebih dari 70 aplikasi yang membutuhkan sistem supaya user dapat log-in dan menyimpan credential (nama, email, password) (Novell, 2011). Fakta yang terjadi pada perusahaan ini membuktikan bahwa ada kebutuhan besar terhadap solusi single sign-on supaya mekanisme autentikasi dan autorisasi menjadi efektif.

Bayangkan apabila perusahaan seperti Google yang memiliki banyak aplikasi seperti Youtube, Gmail, Blogger, Maps, dll. Setiap aplikasi tentu membutuhkan mekanisme autentikasi supaya menjamin hanya user yang terautentikasi-lah yang mampu mengakses informasi pribadi dan servis dari aplikasi (Goth, 2008). Apabila setiap satu aplikasi menyimpan informasi *credential* berupa email dan password, redundansi akan terjadi. Dalam satu *enterprise*, rata-rata user akan menggunakan antara 5-10 aplikasi sekaligus (Novell, 2011). Misalnya, user kemungkinan besar akan menggunakan banyak servis Google secara bersamaan, misal menggunakan Youtube sekaligus Gmail. Tanpa *single sign-on*, user harus mengelola tiap password untuk tiap aplikasi.

Di sisi *enterprise*, Google harus menyimpan *credential* masing-masing user untuk tiap aplikasi. Ini tentunya tidak efisien. Bagaimana bisa aplikasi dalam satu vendor, memiliki mekanisme autentikasi yang berbeda. Bagaimana jika user cukup login sekali saja untuk satu vendor (Google), kemudian user bisa mengakses seluruh servis yang disediakan Google? (Goth, 2008).

*Single sign-on* adalah solusi dari permasalahan autentikasi multi-servis. Ide utama *single sign-on* adalah pemusatan autentikasi beberapa servis menjadi *single-entry point* untuk autentikasi (Volchkov, 2001). Dengan *single sign-on*, permasalahan autentikasi akan disentralisasi menjadi satu mekanisme autentikasi. Mekanisme *single sign-on* ini akan mengatur dan memberikan autorisasi kepada *credential* supaya bisa mengakses bagian tertentu pada servis.

Single sign-on adalah mekanisme di mana user cukup mengautentikasi dirinya satu kali saja kemudian user diberi akses ke banyak web servis, tanpa memasukkan banyak *password* (Volchkov, 2001). User diberi akses oleh *identity provider* (IdP) berupa *token* atau *cookies* yang dapat digunakan untuk mengakses beberapa *service provider* (SP).

Secara umum, implementasi *single sign-on* melibatkan IdP sebagai penyedia identitas yang mengatur autentikasi dan autorisasi dan SP sebagai penyedia servis yang bertugas mengkonsumsi autentikasi dari IdP. Apabila bicara ke level detil implementasi, ada banyak mekanisme berbeda yang dilakukan oleh SSO yang sudah ada seperti yang dilakukan oleh SAML, Auth0, OAuth, OpenID, dan LastPass.

Dengan latar belakang kebutuhan akan layanan publik variatif pada *smart city* Bandung yang membutuhkan mekanisme autentikasi dan autorisasi yang terintegrasi, kemudian didukung dengan adanya tawaran teknologi yang memungkinkan diimplementasinya *single sign-on*, oleh karena itu, pengintegrasian *single sign-on* pada sistem aplikasi citizen memungkinkan untuk dilakukan.

## Rumusan Masalah

Perbedaan implementasi *single sign-on* oleh banyak vendor *single sign-on* baik berbayar maupun *open-source* menyebabkan developer harus mempertimbangkan banyak aspek di antara berbagai mekanisme *single sign-on* untuk mendapatkan preferensi yang diinginkan. Aspek yang perlu dipertimbangkan antara lain adalah keamanan, kompleksitas, dan performansi.

Berdasarkan kebutuhan tersebut, diperlukan mekanisme *single sign-on* yang tepat sesuai dengan kebutuhan pengembang. Oleh karena itu, rumusan masalah dalam tugas akhir ini adalah menentukan sebuah mekanisme *single sign-on* yang terbaik sesuai studi dan kajian yang telah dilakukan, untuk kemudian diimplementasi pada sistem aplikasi *citizen* Bandung.

## Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut.

1. Menentukan teknik yang tepat untuk mengimplementasi *single sign-on* dari hasil kajian kelebihan dan kekurangan *single sign-on*.
2. Membangun *single sign-on* pada sistem aplikasi *citizen* Kota Bandung menggunakan metode yang sudah dikaji dengan baik pada tahapan sebelumnya.

## Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Tugas akhir ini berfokus pada kajian *single sign-on*. Sementara, fungsionalitas aplikasi yang menjadi *service provider* untuk *single sign-on* bukan merupakan lingkup pekerjaan tugas akhir ini.

## Metodologi

Metodologi yang digunakan dalam pengerjaan tugas akhir ini meliputi:

1. Analisis masalah. Pada tahap ini, dilakukan analisis terhadap permasalahan berkaitan dengan topik yang diangkat pada tugas akhir ini. Analisis masalah ini meliputi kajian terhadap bermacam model dan desain *single sign-on* yang pernah didesain orang sebelumnya.
2. Pembangunan aplikasi *single sign-on*. Setelah masalah dianalisis dan dilakukan *requirement engineering*, dibangunlah aplikasi *single sign-on* yang sesuai dengan kebutuhan yang diberikan. Aplikasi ini harus mampu memenuhi kebutuhan fungsional yang telah didefinisikan pada tahap analisis masalah sebelumnya. Aplikasi *single sign-on* ini bertindak sebagai sentra pusat autentikasi bagi *service* lain yang menjadi *service provider* bagi sistem aplikasi *citizen* Kota Bandung.
3. Pembangunan aplikasi *service provider*. Walaupun pembangunan aplikasi *service provider* bukan merupakan fokus utama dalam pengerjaan tugas akhir ini karena topik sebenarnya adalah *single sign-on*, aplikasi yang bertindak sebagai *service provider* harus tetap dibangun walau dengan fungsionalitas yang bersinggungan langsung dengan *single sign-on* saja. Ini tetap harus dilakukan untuk tujuan pengujian fungsionalitas *single sign-on* nantinya.
4. Evaluasi dan penarikan kesimpulan. Pada tahap ini dilakukan evaluasi terhadap *single sign-on* yang telah dibangun. Setelah itu, dilakukan juga penarikan kesimpulan yang didasari oleh hasil evaluasi pengujian aplikasi single sign-on.

## Sistematika Pembahasan

Penulisan tugas akhir ini dibagi menjadi 5 bab, yaitu bab I Pendahuluan, bab II Studi Literatur, bab III Analisis Permasalahan dan Rancangan Solusi, bab IV Implementasi dan Pengujian, dan bab V Kesimpulan dan Saran. Bab I Pendahuluan memberikan gambaran mengenai tugas akhir ini. Bab I terdiri dari latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.

Bab II Studi Literatur memaparkan hasil studi literatur terkait autentikasi dan autorisasi, *single sign-on*, beserta kakas dalam pengembangan *single sign-on*.

Bab III Analisis Permasalahan dan Desain Solusi berisi hasil analisis dari permasalahan yang ada pada tugas akhir ini serta usulan solusi yang diajukan untuk memecahkan permasalahan tersebut.

Bab IV Implementasi dan Pengujian membahas implementasi serta hasil pengujian. Pengujian yang dilakukan terkait perbandingan antara kakas yang digunakan satu sama lain atas berbagai parameter.

Bab V Kesimpulan dan Saran menjelaskan mengenai kesimpulan dari tugas akhir ini dan saran untuk mengembangkan lebih lanjut mengenai tugas akhir ini agar menghasilkan hasil yang lebih baik.

## Jadwal Pelaksanaan Tugas Akhir

Berikut adalah rencana kegiatan dan jadwal pengerjaan tugas akhir.

Tabel ‎I.1. Rencana Pelaksanaan Tugas Akhir I

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan | November 2016 | | | | Desember 2016 | | | | Januari 2017 | | | | Februari 2017 | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Analisis permasalahan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Pembangunan aplikasi single sign-on |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Pembangunan aplikasi service provider |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Evaluasi dan penarikan kesimpulan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| No. | Kegiatan | Maret 2017 | | | | April 2017 | | | | Mei 2017 | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Analisis permasalahan |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Pembangunan aplikasi single sign-on |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Pembangunan aplikasi service provider |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Evaluasi dan penarikan kesimpulan |  |  |  |  |  |  |  |  |  |  |  |  |

# STUDI LITERATUR

## Autentikasi dan Autorisasi

Autentikasi adalah proses bagaimana sistem mengetahui dengan siapa ia berkomunikasi (Tanenbaum, 2007). Menurut Neuman (1994), autentikasi adalah verifikasi dari identitas seseorang yang memberikan data beserta data yang diberikan.

Autorisasi adalah teknik yang menentukan proses apa saja yang boleh dilakukan (Tanenbaum, 2007). Definisi lain dari Neuman (1994), autorisasi adalah proses yang menentukan apakah sebuah pihak diperbolehkan untuk melakukan operasi tertentu.

Menurut Halonen (2000), autentikasi dan autorisasi seharusnya memiliki karakteristik sebagai berikut:

1. Kebenaran (*Correctness*)

Hasil dari setiap autentikasi dan autorisasi haruslah akurat.

1. Kemungkinan untuk anonim dan privasi (*Possibility to anonymity and privacy*)

Autentikasi terhadap identitas hanya diperlukan apabila benar-benar diperlukan. Apabila tidak perlu membuka identitas, maka tidak perlu identitas dibuka. Jika memungkinkan, autorisasi sebaiknya tidak dilakukan tanpa membuka identitas.

1. Kecepatan (*Speed*)

Proses autentikasi seharusnya dilakukan dengan cepat.

1. Ketahanan terhadap serangan (*Attack resistance*)

Metode autentikasi seharusnya dapat tahan terhadap jenis serangan, misalnya serangan mid-in-the-middle.

1. Murah (*Inexpensiveness*)

Proses autentikasi dan autentikasi seharusnya memerlukan ongkos tidak mahal, baik ongkos pengembangan dan pemeliharaan.

1. Kemudahan penggunaan (*User friendliness*)

Metode autentikasi sedapat mungkin mudah digunakan dan dimengerti. Pada kondisi ideal, pengguna tidak perlu mengeluarkan usaha ketika melakukan autentikasi.

1. Universal (*Universality*)

Metode autentikasi sebaiknya sedapat mungkin sama pada semua layanan.

## Sistem Autentikasi Identitas

Autentikasi adalah hal paling mendasar dari keamanan *service* karena autentikasi yang terpercaya dibutuhkan untuk memberi kontrol akses, menentukan siapa yang berhak menerima atau mengubah informasi, dan melawan penyanggahan. (Ford, 1997).

Menurut Ford (1998), ada 5 metode untuk mengautentikasi:

1. Sesuatu yang diketahui pengaku (something the claimant knows).
2. Sesuatu yang dimiliki pengaku (something the claimant owns).
3. Sesuatu tentang siapa pengaku (something the claimant is).
4. Pengaku pada tempat atau waktu tertentu (claimant is at a particular place at particular time).
5. Autentikasi yang diberikan oleh pihak ketiga yang terpercaya (authentication is established by a trusted third party).

### *Password*

Password merupakan metode yang paling mudah dan umum digunakan untuk autentikasi (Ford, 1998). Password umumnya diasosiasikan dengan username, sehingga metoda ini dapat mendukung banyak pengguna. Tetapi, psasword juga merupakan metode yang paling tidak aman untuk autentikasi dan hal ini merupakan salah satu kerentanan utama dari metode password. Namun, kelemahan ini bisa ditangkal dengan menggunakan tool password generator sehingga kita tidak perlu mengingat password yang panjang, dan cukup menggunakan tool password generator yang diamankan dengan 1 master password saja.

Kriteria keamanan pada metode password terletak hanya pada panjangnya string password. Beberapa teknik penyerangan yang dapat dilakukan pada metode password adalah external disclosure, guessing, communications eavesdropping, replay attacks, dan host compromise (Ford, 1998). Oleh karena itu, dewasa ini metode autentikasi password sering kali digunakan bersama dengan metode autentikasi lainnya sehingga memiliki tingkat keamanan yang lebih tinggi, atau yang sering disebut Multi-Factor Authentication (MFA).

### *Token*

Token adalah metode yang digunakan untuk meningkatkan keamanan dari sisten autentikasi (Ford, 1998). Salah satu metode autentikasi token adalah physical tokens yang sering digunakan oleh bank. Tujuannya adalah memberikan pengamanan tambahan pada transaksi yang dilakukan. Jika seandainya PIN nasabah bank diketahui oleh pencuri atau pihak yang tidak berhak, maka pencuri tidak dapat melakukan transaksi jika tidak memiliki token. Metode pembangkitan token juga memerlukan suatu alat fisik dan memperhitungkan variabel waktu sehingga token menjadi sulit ditebak oleh penyerang.

Dewasa ini virtual token juga sering digunakan sebagai metode autentikasi. Cara pembuatan virtual token berbeda dengan cara pembuatan physical token yang memerlukan alat atau benda fisik untuk menghasilkan token. Virtual token merupakan token yang dapat dihasilkan secara virtual dengan bantuan perangkat lunak atau software. Sedangkan untuk representasi datanya, physical token dan virtual token mungkin saja memiliki representasi data yang sama, misalnya string atau angka. Contoh implementasi virtual token adalah One Time Password, atau yang sering disingkat OTP. OTP adalah virtual token yang memungkinkan user menerima password yang dibangkitkan secara acak oleh provider OTP dan berlaku hanya dalam jangka waktu tertentu (biasanya 30 detik). Contoh OTP yang secara luas digunakan oleh masyarakat adalah Google Authenticator dan Authy. Kedua aplikasi tersebut merupakan aplikasi yang berjalan pada device Android dapat diunduh di Playstore secara gratis.

Contoh penggunaan virtual tokens pada aplikasi web adalah session dan cookie. Session disimpan pada server dan digunakan untuk menyimpan state dari pengguna (Whitehead, 2006). Session dapat diumpamakan sebagai kumpulan dari key yang memiliki value tertentu. Masing-masing session memiliki session ID yang dihasilkan secaraotomatis dan berbeda untuk setiap sesion. Session ID inilah yang dapat dianggap sebagai token.

Cookie merupakan kumpulan dari pasangan parameter dan nilainya. Cookie dibuat atau dihasilkan oleh server, dan diberikan kepada web browser. Cookie umumkan disimpan pada media penyimpanan lokal milik device client dan dapat diakses web browser. Cookie dapat digunakan untuk memberikan tanda bahwa pengguna sudah melakkan atau mendapatkan sesuatu dari server. Dengan kata lain, cookie merupakan suatu mekanisme untuk menambahkan state pada HTTP request (Whitehead, 2006).

Pada umumnya, session dan cookie dapat dikombinasikan sebagai teknik manajemen state pada aplikasi web (Whitehead, 2006). State diperlukan pada aplikasi web untuk mengetahui apakah pengguna pernah mengunjungi web pada waktu sebelumnya atau apakah pengguna sudah melakukan tahapan yang benar untuk sebuah proses pada aplikasi web. Server menyimpan informasi state dari pengguna dengan menggunakan session. Cookie berisi informasi, misalnya session ID, yang digunakan untuk mencari session milik pengguna pada server. Cookie dihasilkan oleh server dan diberikan kepada web browser pada HTTP response, kemudian cookie dikirimkan kembali oleh web browser pada server pada HTTP request selanjutnya.

### *Biometrics*

Metode biometrics menekankan bahwa autentikasi akan mengidentifikasi dan memverifikasi seseorang berdasarkan dari ciri-ciri fisik atau perilaku yang dimilikinya. Metode autentikasi dengan biometrics memiliki kekurangan utama yaitu harga atau biaya untuk mengimplementasikannya sangat mahal dibandingkan dengan metode autentikasi lainnya (Ford, 1998). Contoh ciri-ciri fisik yang dapat digunakan pada metode biometrics adalah sidik jari (fingerprint), retina mata, tulisan tangan, bentuk wajah, bentuk gigi. Sedangkan, contoh ciri perilaku yang dapat digunakan adalah kebiasaan gerak seseorang, suara dalam melafalkan kata-kata kalimat.

Salah satu metode biometrics yakni sidik jari memiliki akurasi yang sangat tinggi. Sidik jari manusia merupakan komponen yang identitas manusia yang unik. Dua fingerprint yang sama hanya ditemukan sekali dalam setiap 1048 tahun (Jain, 2005).

### Tanda Tangan Digital

Tanda tangan digital atau digital signatures adalah tanda tangan yang diberikan pada data digital (Munir, 2005). Tanda tangan berupa nilai yang berdasarkan dari isi dan pengirim pesan. Proses umum pemberian tanda tangan melibatkan kriptografi asimetri dan fungsi hash. Tanda tangan digital dapat digunakan oleh penerima pesan untuk memverifikasi bahwa pengirim pesan memiliki kunci privat yang sah dan isi pesan tidak diubah selama proses pengiriman (Ford, 1998). Penerima pesan dapat dianalogikan sebagai aplikasi dan pengirim pesan adalah pengguna, atau sebaliknya. Sebagai metode autentikasi, pengguna juga dapat mengirimkan tanda tangan digitalnya untuk diverifikasi.

### Sertifikat Digital

Sertifikat Digital adalah Certification Authorities mengeluarkan sertifikat digital yang menyatakan hubungan antara kunci publik dan pemiliknya (Ford, 1998). Isi informasi dari sertifikat digital berupa nama an organisasi pemilik kunci publik, kunci publik, dan waktu kadaluarsa sertifikat. Sertifikat digital dapat digunakan untuk mengautentikasi seorang pengguna atau sebuah website ataupun sebuah institusi. Certification Authorities memberikan cara untuk mengirimkan kunci publik beserta informasi pemiliknya pada dunia digital melalui sertifikat digital (Ford, 1998). Namun, pengenalan CA yang sah dan dipercaya untuk mengeluarkan sertifikat igital harus dapat dilakukan oleh pihak penerima pesan. Pengenalan CA yang sah menjadi masalah baru yang ditimbulkan dari penggunaan metode certificat authorities untuk metode autentikasi.

## *Multi-Factor Authentication*

Multi-Factor Authentication (MFA) atau autentikasi banyak faktor adalah autentikasi yang menggunakan lebih dari satu faktor atau metode (Federal Deposit Insurance Corporation, 2004). Faktor untuk melakukan autentikasi adalah lima metode autentikasi yang sudah dijelaskan pada subbab 2.2. MFA dengan dua metode autentikasi seringkali disebut two factor authentication (2FA).

Semakin banyak metode autentikasi yang digunakan, semakin baik keamanan sebuah sistem (Federal Deposit Insurance Corporation, 2004). Tetapi, setiap kelebihan kadangkala terdapat kelemahan yang harus diterima sebagai konsekuensinya. Kelemahannya adalah lamanya waktu yang dibutuhkan untuk autentikasi sebanding dengan banyaknya metode autentikasi yang digunakan.

## *Single Sign-On*

Single sign-on adalah proses autentikasi user yang memperbolehkan seorang pengguna dengan sebuah identitas yang terdaftar untuk mengakses berbagai aplikasi (Pashadilis, 2003). Single sign-on disingkat SSO. Menurut He (2007), single sign-on dalam definisi yang lebih sederhana adalah proses penggunaan identitas pengguna yang sama secara bersama-sama. *Single sign-on* adalah bagian dari sistem yang memungkinkan user mengakses beberapa servis/aplikasi/*resource* melalui satu titik entri. *Single sign-on* merupakan properti akses kontrol dari beberapa sistem yang berhubungan tetapi independen (Volchkov, 2001). Dengan properti ini, user dapat login menggunakan satu ID dan *password* untuk mendapatkan akses ke sistem yang terhubung atau sistem dengan banyak *username* dan *password* yang berbeda. *Single-sign on* biasanya menyimpan satu set credential (contoh: nama, email, password) untuk mengakses beberapa aplikasi. *Single sign-on* mengautentikasi user, apakah user yang bersangkutan diberikan hak untuk mengakses *resource* yang ada. Pada *back-end*, *single sign-on* sangat membantu untuk mengelola aktivitas user sekaligus memonitor akun. Contoh sederhana versi *single sign-on* dapat dicapai lewat penyimpanan cookie di browser untuk domain tertentu, namun hanya dapat dicapai apabila situs memiliki domain DNS yang sama. Secara umum, *single sign-on* terdiri dari Identity Provider (IdP) yang berfungsi untuk mengelola credential/session milik user. Sementara itu single-sign on juga memiliki Service Provider (SP) yang merupakan konsumer dari autentikasi yang disediakan IdP.

Misalkan dalam sebuah sistem terdiri dari beberapa aplikasi dan sistem tersebut memiliki mekanisme autentikasi dalam single sign-on. Penguna harus melakukan login pada setiap aplikasi dan aplikasi akan mengautentikasi pengguna tersebut. Pengguna juga harus mendaftarkan dirinya dan mengingat sejumlah identitas untuk diautentikasi pada sejumlah aplikasi. Hal ini juga dapat menyebabkan human error jika pengguna lupa terhadap identitasnya atau mekanisme autentikasi error pada sebuah aplikasi.

Pengelolaan pengguna menjadi pekerjaan yang merepotkan jika pengguna dan/atau aplikasi semakin bertambah banyak. Hal ini dapat diperburuk dengna pengelolaan pengguna yang harus dilakukan pada masing-masing aplikasi jika aplikasi-aplkikasi tidak menggunakan single sign-on.

Pada sistem yang mekanisme autentikasinya tanpa single sign-on, aplikasi juga mengalami kerugian yaitu terdapat overhead proses autentikasi. Overhead pada autentikasi maksudnya adalah proses autentikasi dilakukan berkali-kali pada setiap aplikasi meskipun proses autentikasi tersebut cukup dilakukan sekali oleh sebuah aplikasi saja. Dengan menggunakan single sign-on, proses autentikasi cukup dilakukan sekali pada aplikasi yang menangani single sign-on



Gambar ‎II.1 Sistem yang Mekanisme Autentikasinya tanpa SSO

.

Beberapa poin kerugian, yaitu kemungkinan terjadinya human error, kerumitan pada pengelolaan pengguna dan overhead pada proses autentikasi, dapat terjadi pada sistem yang tidak menggunakan single sign-on. Gambar II.x merupakan contoh sistem yang mekanisme autentikasinya tanpa single sign-on. Sistem pada gambar tersebut terdiri dari aplikasi A, B, dan C. Pengguna akan melakukan login terlebih dahulu untuk menggunakan aplikasi yang diinginkannya. Setiap aplikasi memiliki mekanisme autentikasinya sendiri sehingga pengguna harus melakukan login pada setiap aplikasi untuk menggunakannya.

Mekanisme autentikasi pada Gambar II.x kurang efektif dan efisien jika dibandingkan dengan mekanisme autentikasi pada Gambar II.x+1. Mekanisme autentikasi pada Gambar II.x+1 menggunakan single sign-on untuk mekanisme autentikasi seluruh aplikasi. Sehingga, pengguna cukup terautentikasi sekali saja untuk menggunakan aplikasi A atau B atau C. Autentikasi dapat dilakukan oleh salah satu aplikasi ataupun mengunakan aplikasi pihak ketiga yang dipercaya oleh masing-masing aplikasi A, B, dan C.



Gambar ‎II.2 Sistem dengan SSO

Berikut adalah kelebihan implementasi single sign-on:

1. Satu *password* untuk masuk ke dalam sistem.
2. Login otomatis, apabila user sudah memiliki session sebelumnya.
3. Lebih mudah untuk mengelola user.
4. Satu klik untuk mengambil (*revoke*) akses user untuk sebagian atau seluruh bagian sistem.

Berikut adalah kekurangan implementasi *single sign-on*:

1. *Single point of failure*. Apabila server *single sign-on* *down*, maka seluruh aplikasi tidak memiliki mekanisme untuk autentikasi user. Dapat di-handle dengan mereplika server *single sign-on* menjadi beberapa replika supaya *availability* semakin besar.
2. *Single point of attack*. Penyerang dapat bertindak sebagai aplikasi yang sah untuk mencuri informasi yang dikirimkan SSO server kepada pengguna (Pashadilis, 2003). Sehingga, penyerang dapat mengetahui informasi berharga yang digunakan untuk komunikasi antara SSO server dan pengguna.
3. Apabila aplikasi sudah ada dan *single sign-on* akan diintegrasi, *cost* untuk mengintegrasikan single sign-on dengan aplikasi yang sudah ada cukup besar. Hal ini biasanya rumit, terutama untuk pengimplementasian SSO pada sebuah sistem lama. Pada kondisi ini, implementasi SSO lebih diarahkan kepada penambahan fitur untuk meningkatkan user experience dan efektivitas dan efisiensi dari aplikasi. Akibatnya, implementasi SSO harus memperhatikan fungsionalitas yang sudah ada dan biasa digunakan pada sistem lama.

## *Web Single Sign*-*On*

Web single sign-on adalah single sign-on yang digunakan pada aplikasi web (Jasig, 2009). Web single sign-on biasanya disingkat web SSO. Bnayaknya jumlah layanan yang tersedia di internet menyebabkan ketidaknyamanan terhadap pengguna layanan. Pengguna biasanya harus login pada beberapa sistem layanan dan masing-masing sistem layanan memerlukan identitas untuk diautentikasi. Jika setiap sistem layanan yang memerlukan identitas yang spesifik dan berbeda, maka pengguna akan mendapatkan kesulitan yang lebih banyak. Salah satu kesultan dalah pengguna harus mengingat identitas yang digunakan pada setiap layanan.

Web single sign-on sangat mudah ditemukan dari web pada internet. Dewasa ini, beberapa grup atau komunitas sudah mengembangkan web single sign-on. Beberapa produk web single sign-on bersifat open source, tetapi ada juga produk yang bersifat komersil. Beberapa grup atau kounitas yang mengembangkan web single sign-on adalah:

1. Auth0 Inc dengan produknya Auth0.
2. WSO2 dengan produknya WSO2 Identity Server.
3. Apereo dengan produknya Central Authentication Service.
4. Atricore Inc dengan produknya Java Open Single Sign-On.

### *Auth0*

Auth0 adalah layanan Platform as a Service (PaaS) yang menyediakan single sign-on yang memudahkan developer untuk memungkinkan developer menambahkan SSO pada aplikasinya secara mudah tanpa memperdulikan bahasa, teknologi, dan protokol yang digunakan dalam autentikasi. Sebagai PaaS, Auth0 menyediakan platform single sign-on secara keseluruhan termasuk dashboard, media penyimpanan data user, dan halaman login untuk single sign-on. Auth0 dikembangkan oleh Auth0 Inc sebuah perusahan yang berpusat di Seattle, Amerika Serikat. Alamat website Auth0 adalah <https://auth0.com>. Sementara alamat website untuk manajemen dashboard Auth0 adalah <https://manage.auth0.com>

Auth0 dapat digunakan tanpa memerdulikan bahasa apa yang digunakan aplikasi service provider. Sejauh ini, banyak stack teknologi yang didukung Auth0 baik regular web application (server-side authentication) seperti Go, Java, Ruby on Rails, dan seterusnya, single-page application (client-side authentication) seperti NodeJS, AngularJS, dan JQuery, maupun mobile application (iOS dan Android) (Pace, 2015). Masing-masing jenis client server-side, client-side, dan mobile memiliki alur autentikasi yang berbeda-beda. Auth0 mendukung lebih dari 65 SDK (Software Development Kit) yang dapat digunakan developer untuk mengintegrasikan aplikasinya dengan single sign-on Auth0. Auth0 juga mendukung protokol SAML, OpenID, dan integrasi dengan banyak penyedia single sign-on (provider) seperti Slack, Dropbox, Yahoo. Selain itu, dokumentasi Auth0 sangat mudah dimengerti dan diaplikasikan karena banyak tersedia sample application yang benar-benar dapat dijalankan. Dengan demikian, Auth0 berkembang dengan sangat pesat, saat ini memiliki 2000+ enterprise customer, 42 juta lebih login dalam satu hari. Cara kerja Auth0 dapat dilihat pada Gambar II.x di bawah.



Gambar ‎II.3 Cara Kerja Auth0

Cara kerja Auth0 adalah sebagai berikut.

* 1. Admin, atau developer mendaftarkan aplikasi service provider ke dashboard Auth0.
  2. Dashboard Auth0 akan menyimpan informasi kepada server mengenai developer dan aplikasi miliknya.
  3. Developer mengarahkan (redirect) aplikasi service provider ke server Auth0.
  4. User ingin mengakses resource di aplikasi service provider yang dilindungi oleh autentikasi.
  5. Aplikasi service provider mengetahui user belum terautentikasi, maka service provider mengarahkan user ke halaman login di server Auth0. Server Auth0 ini dapat berupa cloud Auth0 yang dapat ter-deploy sehingga dapat diakses siapapun, ataupun server yang berada pada jaringan privat. Untuk layanan gratis, Auth0 hanya menyediakan server cloud saja sehingga kita tidak bisa menggunakan server Auth0 pada jaringan privat kita sendiri. Ini artinya seluruh data user dan aplikasi service provider disimpan pada media penyimpanan milik Auth0.
  6. User melakukan autentikasi pada halaman login server Auth0 dengan memasukkan credential.
  7. Setelah autentikasi pada server Auth0 berhasil dilakukan, server mengembalikan user ke service provider beserta token yang dapat digunakan untuk identifikasi user ketika mengakses resource berikutnya.
  8. Ketika user membutuhkan resource yang membutuhkan autentikasi, aplikasi tidak perlu lagi meredirect laman ke server Auth0 karena token sudah tersimpan pada aplikasi service provider untuk digunakan selanjutnya.

### WSO2 *Identity Server*

WSO2 Identity Server adalah web single sign-on open-source yang dibuat oleh WSO2. WSO Identity Server dibangun di atas WSO2 Carbon, sebuah framework manajemen identitas yang dibuat oleh WSO2. WSO2 mendukung banyak protokol SSO seperti OpenID Connect, OAuth, SAML SSO, maupun social authenticator yang popular seperti Google, Facebook, Yahoo, Microsoft, juga mendukung berbagai metode autentikasi lain seperti SMS OTP, Email OTP, Kerberos, dan lain-lain.

Berikut adalah arsitektur WSO2.



Gambar ‎II.4 Arsitektur WSO2 Identity Server (WSO2, 2015)

Aplikasi service provider yang menggunakan WSO2 Identity Server melakukan autentikasi pada sisi aplikasi service provider. Credential yang dimasukkan oleh resource owner (user) akan dikirimkan ke WSO2 Identity Server melalui Inbound Authenticator. Inbound Authenticator berfungsi untuk mengurusi permintaan user yang akan login. Permintaan ini diterima oleh modul Request Processor kemudian akan diteruskan ke Authentication Framework yang berada pada gambar di tengah atas. Permintaan user atas service provider dipetakan ke struktur data lokal milik WSO2 (claim mapping) yang terstandardisasi menurut WSO2. Kemudian permintaan login akan dilanjutkan menurut tipenya. Apabila permintaan login tersebut adalah login dengan username/password, maka permintaan akan diteruskan ke Local Authenticators, sementara apabila permintaan login menggunakan model autentikasi lain seperti SAML, OpenID Connect, dst maka permintaan akan diteruskan Federated Authenticators. Federated Authenticators bertindak sebagai komunikator antara WSO2 Identity Server dengan aplikasi eksternal seperti Google, Facebook, Yahoo, atau penyedia layanan identity server lain.

Setelah Local Authenticators atau Federated Authenticators selesai memvalidasi user, respon akan dikembalikan ke Authentication Framework. Authentication Framework akan melakukan pemetaan ulang dari Identity Server ke struktur data internal WSO2 (claim mapping), dan dari claim mapping ke service provider. Respons akan dikembalikan ke Inbound Authenticator. Modul Response Generator pada Inbound Authenticator akan mengembalikan respon login kepada user. Respon ini bisa berupa penolakan login atau penerimaan login.



Gambar ‎II.5 Mekanisme WSO2 Identity Server

Cara kerja WSO2 Identity Server adalah sebagai berikut:

* 1. Admin, atau developer mendaftarkan aplikasi service provider ke dashboard WSO2 Identity Server.
  2. Dashboard WSO2 akan menyimpan informasi kepada server mengenai developer dan aplikasi miliknya.
  3. Developer mengarahkan aplikasi service provider ke single sign-on ketika membutuhkan autentikasi user.
  4. User ingin mengakses resource di aplikasi service provider yang dilindungi oleh autentikasi.
  5. Aplikasi service provider mengirimkan credential user dan credential aplikasi untuk menukarkannya dengan token user yang bisa digunakan untuk login pada session berikutnya. Credential user diperlukan untuk membedakan suatu user dan uesr lain. Credential aplikasi diperlukan server WSO2 untuk memastikan aplikasi yang meminta login user tersebut adalah benar-benar aplikasi yang valid.
  6. Setelah autentikasi di server Auth0 berhasil dilakukan, server memberikan token kepada service provider yang dapat digunakan untuk identifikasi user ketika mengakses resource berikutnya.
  7. Ketika user membutuhkan resource yang membutuhkan autentikasi, aplikasi tidak perlu lagi login ke aplikasi, cukup ke laman ke server Auth0 karena token sudah tersimpan pada aplikasi service provider untuk digunakan selanjutnya.

Secara garis besar, cara kerja WSO2 Identity Server kurang lebih mirip dengan Auth0 yang telah dijelaskan pada subbab sebelumnya. WSO2 Identity Server menyediakan dashboard developer/admin sama seperti yang Auth0 miliki. Yang menjadi perbedaan adalah server WSO2 IS disediakan secara gratis karena open-source dan kita bisa melakukan self-host dengan mudah.

Perbedaan lain adalah aplikasi service provider tidak di-redirect ke server milik SSO untuk login. Login dilakukan pada sisi aplikasi tetapi validasi dilakukan pada sisi server WSO2.

### *Central Authentication Service (CAS)*

Central Authentication Service merupakan web single sign-on yang dibuat oleh Apereo. Pada mulanya, Central Authentication Service hanyalah sebuah protokol single sign-on yang dibuat oleh Yale University. Kemudian, Central Authentication Service diimplementasikan dengan platform Java dan mendukung library dari client untuk Java, PHP, Perl, dan lainnya sehingga CAS menjadi sebuah penyedia web single sign-on.

Cara kerja protokol autentikasi CAS hampir mirip dengan cara kerja protokol Kerberos. Kemiripan protokol autentikasi dikarenakan CAS menggunakan Kerberos sebagai model dalam pembuatan mekanisme autentikasi. Cara kerja protokol CAS dapat dilihat pada Gambar II.x.



Gambar ‎II.6 Mekanisme Kerja CAS

Cara kerja protokol CAS adalah sebagai berikut (Jasig, 2009)

1. Pengguna mengakses sebuah aplikasi web atau service provider.
2. Pengguna tersebut akan di-redirect ke CAS login URL melalui koneksi HTTPS. Nama aplikasi web atau service provider diberikan sebagai parameter dari request.
3. Pengguna memasukkan ID, yang dapat berupa username, dan password. Kemudian, CAS server melakukan autentikasi terhadap pengguna. Jika autentikasi gagal, aplikasi web tidak akan bisa diakses dan pengguna akan diminta untuk mengisikan ID dan password yang benar.
4. Jika autentikasi berhasil, CAS server mengarahkan pengguna ke aplikasi web yang diinginkan dan menambahkan ticket pada parameter URL. CAS server membuat sebuah cookie yang disebut ticket granting cookie (TGC). Ticket granting cookie akan digunakan untuk menandakan bahwa pengguna berhasil diautentikasi. Untuk selanjutnya, ticket granting cookie dapat digunakan untuk melakukan autentikasi secara otomatis.
5. Aplikasi web atau service provider akan melakukan validasi keabsahan dari ticket dan kepemilikan ticket oleh pengguna yang sah. CAS ServerValidate yang disediakan oleh CAS server dapat digunakan untuk memvalidasi ticket.
6. Aplikasi web dan CAS server akan berkomunikasi melalui HTTPS dan ticket dan nama aplikasi akan dijadikan sebagai parameter. CAS server melakukan pengecekan bahwa ticket tersebut sah. Jika validasi berhasil, CAS server akan memberikan ID pengguna pada aplikasi.

Central Authentication Service terdiri dari dua komponen yaitu CAS server dan CAS client. CAS server menangani web single sign-on untuk aplikasi web. CAS server merupakan sebuah aplikasi web dengan platform Java.

CAS client akan membantu aplikasi web untuk menangani modul web single sign-on dari CAS server. CAS client akan dijadikan sebagai modul dari aplikasi web yang ingin mengunakan web single sign-on. CAS client harus disesuaikan dengan platform atau framework dari aplikasi web. CAS client juga dapat dikonfigurasi atau diimplementasi ulang jika memang diperlukan untuk memberikan dukungan web single sign-on pada aplikasi web.

Keunggulan dari CAS adalah CAS menyediakan banyak metode autentikasi yang dapat digunakan dengan aplikasi atau komponen lain. Contoh komponen yang digunakan CAS adalah active directory, LDAP, DBMS, X.509, SPNEGO, Legacy, dan lain-lain (Jasig, 2009). Metode autentikasi tersebut dapat dikonfigurasi pada CAS server.

### *Java Open Single Sign-On (JOSSO)*

JOSSO adalah sebuah infrastruktur open source single sign-on berbasiskan Java EE and Spring yang menyediakan solusi untuk autentikasi dan autorisasi pengguna secara terpusat tanpa melihat platform (Atricore, 2010). Alamat URL website JOSSO adalah sebagai berikut: <http://josso.org>. JOSSO merupakan sebuah perangkat lunak yang dibuat oleh Atricore Inc.

JOSSO dapat digunakan pada web server dengan platform yang berbeda-beda, misalnya Jboss, Tomcat, Weblogic, Apache, PHP, dan Microsoft IIS. Meskipun begitu, SSO gateway hanya dapat di-install pada web server yang men-support Java, sedangkan SSO agent dapat di-install pada seluruh web server.

Komponen utama JOSSO berjumlah tiga, yaitu SSO gateway, SSO agent, dan aplikasi rekanan. Sedangkan, entitas luar yang berinteraksi dengan JOSSO adalah penguna dengan web browsernya, identirt storage resource, dan application stage storage resource.

Penjelasan tiga komponen utama JOSSO adalah sebagai berikut:

1. SSO gateway / SSO server (identity provider/IdP)

SSO gateway merupakan SSO server atau dikenal sebagai identity provider. SSO gateway memiliki fungsi untuk mengelola SSO dari aplikasi web yang dapat menggunakan SSO. Pengguna yang dapat menggunakan SSO dapat dikelola melalui SSO gateway.

1. SSO agent

SSO agent bertugas melakukan penanganan web single sign-on terhadap aplikasi rekanan sesuai dengan lingkungan eksekusinya. Lingkungan eksekusi dalam prakteknya adalah web server dari aplikasi rekanan. SSO agent menggunakan identity services yang disediakan oleh SSO gateway.

1. Aplikasi rekanan atau partner application (service provider/SP)

Aplikasi rekanan atau partner application merupakan aplikasi yang menggunakan web SSO. Aplikasi rekanan juga dikenal sebagai service provider. Aplikasi rekanan memiliki ketergantungan atau dependensi terhadap SSO agent dan SSO gateway untuk menyediakan web single sign-on kepada pengguna.

JOSSO mengandalkan komponen identity provider, identity manager, dan session manager untuk menangani single sign-on yang dilakukan oleh setiap aplikasi rekanan. Session manager merupakan komponen yang mengelola session aplikasi, identity manager merupakan komponen yang mengelola session dari user yang sudah login, dan identiy provider merupakan komponen yang menyediakan daftar pengguna yang dapat menggunakan SSO.

Pada April 2011, versi JOSSO yang terbaru adalah 2.0.2, JOSSO2, mulai versi 2.0.0, merupakan generasi baru dari JOSSO. JOSSO generasi pertama (JOSSO1) dikembangkan hingga versi 1.8.4. Arsitektur JOSSO1 tidak berbeda dengan arsitektur JOSSO2. Namun, JOSSO2 men-support SAML2 untuk memberikan interoperability dengan sistem SSO pihak ketiga dan aplikasi software-as-a-service (SaaS). JOSSO2 juga memberikan pengelolaan SSO dengan menggunakan RIA (Rich Internet Application).

## Pekerjaan Terkait

### *CAS++*

CAS++ merupakan sistem open source single sign-on yang dikembangkan oleh Ardagna dan krunya. CAS++ merupakan pengembangan dari Central Authentication Services (Ardagna, 2006). Pengembangan yang dilakukan pada CAS++ adalah:

1. Menggunakan identity certificates dan intertigrasi dengan security layer pada Jboss. Identity certificates disebut juga sebagai digital certificates atau public key certificates.
2. Mekanisme autentikasi menggunakan Public Key Infrastrukture (PKI).
3. Keamanan pada CAS++ ditingkatkan dengan menggunakan:
   * 1. SSL untuk memberikan jalur komunikasi yang aman terhadap aplikasi web yang berpartisipasi.
     2. X.509 digital certificates digunakan untuk pertukaran credentials.
4. Menyediakan user management secara terpusat sehingga masing-masing aplikasi web tidak perlu melakukan atau menyediakan user management. Identitas pengguna akan disimpan secara terpusat pada SSO server.
5. Memiliki layanan yang dapat digunakan untuk aplikasi web berbeda platform.
6. Proses autentikasi menggunakan username dan password dan diperkuat dengan metode biometrics dan smart card. Metode biometrics yang digunakan adalah sidik jari atau fingerprint yang dibaca dengan fingerprint readers.

Dengan pengembangan yang dilakukan pada CAS++, CAS++ memiliki kelebihan terhadap aspek keamanan (Ardagna, 2006). Keamanan pada CAS++ ditingkatkan dengan menggunakan SSL, X.509 dan metode autentikasi yang menggunakan metode biometrics dan smart card. CAS++ menjadi lebih mudah untuk diguankan oleh appliaski web berbeda platform karena menggunakan protokol standar seperti HTTP, SSL, X.509 untuk melakukan pertukaran informasi antara CAS++ server dengan aplikasi web.

### Kristanto, 2011

Kristanto mencoba mengimplementasi web single sign-on pada web pada lingkungan informatika ITB. Masalah timbul ketika aplikasi-aplikasi yang digunakan pada lingkungan informatika ITB seperti milestone, moodle, oddyseus memiliki mekanisme loginnya masing-masing.

Kristanto menganalisis karakteristik masing-masing aplikasi, terutama struktur data yang digunakan dalam autentikasi user beserta protokol yang digunakan dalam web SSO. Kristanto kemudian mengintegrasikan JOSSO pada aplikasi lingkungan informatika ITB. Media penyimpanan untuk data user menggunakan LDAP.

Berikut adalah komponen yang digunakan Kristanto dalam pengembangan SSO pada lingkungan web informatika ITB:

1. SSO Server yang merupakan bawaan JOSSO. SSO server ini tidak diimplementasi ulang, hanya perlu dikonfigurasi.
2. SSO Agent adalah komponen yang menghubungkan aplikasi service provider dengan SSO server. SSO Agent mendefinisikan bagaimana aplikasi login, logout, dan endpoint milik SSO server. Untuk membuktikan bahwa single sign-on tidak peduli pada bahasa client, SSO Agent diperlukan sebanyak dua buah, masing-masing menggunakan bahasa Java dan PHP.
3. Aplikasi service provider. Terdapat dua buah yakni Java dan PHP dan diletakkan pada web server yang berbeda. Aplikasi service provider ini melakukan autentikasi melalui SSO Agent yang sudah diimplementasi sebelumnya.

Kesimpulan yang didapat adalah web single sign-on pada lingkungan informatika ITB dapat dilakukan dengan mengubah sedikit basis data dan beberapa atribut kelas pada aplikasi.

# ANALISIS PERMASALAHAN DAN RANCANGAN SOLUSI

## Analisis Permasalahan

Pada bab ini, akan dipaparkan analisis berdasarkan studi literatur pada bab sebelumnya. Permasalahan yang dipaparan merupakan acuan dari topik Tugas Akhir ini. Setelah analisis permasalahan tersebut, akan dijelaskan rancangan solusi yang akan dibangun.

### Auth0

Dalam tahap studi literatur, di antara tool web single sign-on yang dipelajari, Auth0 menyediakan fungsional dan Software Development Kit (SDK) yang paling lengkap. Auth0 tidak hanya mendukung Mengintegrasi aplikasi dengan single sign-on milik Auth0 sangat mudah, hampir tidak ada halangan. Selain itu, Auth0 memiliki dokumentasi Application Programming Interface (API) yang jelas dan benar-benar berjalan. Dokumentasi API sangat penting dalam pengembangan perangkat lunak, karena dalam prakteknya dokumentasi API merupakan kontrak yang sudah disediakan oleh penyedia API, yang mana developer harus mengikuti kontrak yang ada. Selain itu, Auth0 memiliki dashboard admin yang bisa digunakan untuk monitoring user, monitoring aplikasi service provider, monitoring identity provider eksternal yang terintegrasi dengan SSO Auth0. Dashboard admin ini memudahkan developer dalam monitoring banyak hal. Penggunaan Auth0 sebagai single sign-on gratis tetapi dibatasi, untuk penggunaan lebih, Auth0 memungut biaya.

Kelemahan Auth0 adalah ia merupakan perangkat lunak proprietary milik Auth0 Inc. Produknya Auth0 tidak open-source, artinya tidak bisa dikembangkan lebih lanjut oleh orang umum, hanya pegawai Auth0 Inc yang bisa mengembangkan Auth0. Selain itu, dalam penggunaan gratis, developer harus me-redirect aplikasi service provider ke laman login yang di-deploy pada cloud server milik Auth0, yang artinya seluruh data user, data aplikasi service provider tersimpan di basis data milik Auth0. Praktek ini tidak aman terutama apabila data user yang disimpan adalah data sensitif. Ini menyebabkan aplikasi service provider milik developer akan bergantung secara penuh pada ketersediaan server cloud Auth0. Apabila server cloud Auth0 down, maka aplikasi service provider tidak akan dapat melakukan segala hal yang berkaitan dengan manajemen user seperti autentikasi atau meminta data terkait user. Untuk mengatasi permasalahan ini, Auth0 Inc menyediakan Auth0 supaya bisa di-deploy secara self-host di server sendiri. Namun kelebihan ini hanya dapat dimanfaatkan bagi developer yang menggunakan Auth0 secara berbayar.

### WSO2 Identity Server

WSO2 Identity Server adalah produk SSO milik WSO2. WSO2 Identity Server dapat digunakan secara gratis karena open-source. Masyarakat luas secara umum bisa ikut mengembangkan WSO2 Identity Server melalui pull request di GitHub. WSO2 juga memungkinkan SSO di-deploy pada server privat (self-hosted). Dengan demikian, basis data, media penyimpanan data tentang aplikasi service provider juga dapat dikontrol secara langsung oleh developer sehingga dapat dijaman sendiri keamanan dan maintenance/uptime servernya. Seperti Auth0, WSO2 Identity Server juga memiliki dashboard untuk konfigurasi aplikasi service provider. Selain itu, WSO Identity Server juga mendukung banyak protokol SSO seperti SAML, OAuth, HTTP. WSO Identity Server juga menyediakan banyak integrasi dengan identity provider lain seperti Facebook, Google, Yahoo. WSO2

Kelemahan utama WSO2 Identity Server adalah dokumentasi yang kurang rapi dan learning curve yang cukup tajam. Sample application yang sangat dasar untuk SSO dengan protokol SAML dan OAuth milik WSO2 Identity Server bahkan tidak bisa dijalankan. Dengan demikian, developer yang baru menggunakan teknologi single sign-on tidak akan dapat mengintegrasikan aplikasinya dengan mudah. Dengan tingkat kompleksitas aplikasi yang tinggi seperti ini, dukungan teknis customer service hanya dapat diperoleh apabila developer men-subscribe WSO2 Identity Server secara berbayar. WSO2 Identity Server (SSO server) membutuhkan resource memori yang tinggi ketika dijalankan, yakni sekitar 800 MB dan 500 MB heap size. Kebutuhan memori yang besar terjadi karena WSO2 Identity Server diimplementasi menggunakan Java. WSO2 Identity Server menyediakan cloud server tetapi hanya bisa diakses secara berbayar. Selain itu, untuk monitoring user, WSO2 Identity Server tidak menyediakan dashboard secara terpusat sehingga harus dilakukan dari sisi aplikasi service provider. Dashboard yang disediakan WSO2 Identity Server hanya berfungsi sebagai monitoring aplikasi service provider, tetapi tidak untuk monitoring user.

### Central Authentication Server

Central Authentication Server pada mulanya adalah sebuah protokol untuk single sign-on, namun sekarang telah dikembangkan lebih lanjut menjadi tool single sign-on dan aplikasi client-nya banyak diimplementasikan menggunakan beberapa bahasa pemrograman. CAS merupakan project open-source milik Apereo. Dengan demikian, CAS server memungkinkan untuk di-deploy secara self-hosted. CAS mendukung beberapa protokol seperti CAS, SAML, OAuth, OpenID. CAS memiliki dokumentasi yang lengkap pada laman <https://apereo.github.io/cas/4.2.x/index.html>. CAS memiliki dashboard standalone (berbeda dari server) yang dapat digunakan untuk memonitor aplikasi service provider.

Walaupun memiliki dokumentasi yang lengkap, tetapi Apereo, organisasi pemilik project CAS, tidak menyediakan sample application yang bisa digunakan sebagai petunjuk dalam mengintegrasikan aplikasi dengan Central Authentication Server. Kelemahan CAS berikutnya adalah membutuhkan memori besar untuk dijalankan. CAS server membutuhkan memori minimal 8GB dan prosesor minimal 3,00 GHz (Apereo, 2015). Selain itu walaupun CAS memiliki dashbard untuk monitoring aplikasi service provider, CAS tidak memiliki dashboard untuk manajemen user sehingga developer harus membuat aplikasi client sendiri. Kelemahan lainnya dari CAS adalah pertukaran informasi antar komponen CAS dilakukan tidak menggunakan SSL sehingga rawan terhadap serangan man-in-the-middle. Selain itu, bahasa pemrograman client yang didukung oleh CAS hanya bahasa pemrograman kuno seperti Java, .NET, PHP, Perl. Bahasa modern seperti Python, Go, Ruby tidak didukung karena CAS merupakan teknologi lama. CAS juga tidak mendukung single sign-on untuk single page application (SPA) yang menggunakan berbagai framework JavaScript seperti React, Redux, VueJS yang akhir-akhir ini sedang trend.

## Analisis Solusi

Berdasarkan analisis permasalahan pada subbab sebelumnya, kali ini akan dibahas mengenai analisis solusi yang ditawarkan dalam tugas akhir ini. Secara garis besar, akan dibuat sebuah single sign-on baru yang memiliki fungsionalitas gabungan dari web single sign-on di atas, dan mengurangi/menghapus kelemahan dari web single sign-on yang sudah ada.

### Perbandingan Web Single Sign-On

### Kebutuhan Sistem

### Kebutuhan Fungsional Perangkat Lunak

### Kebutuhan Non-Fungsional Perangkat Lunak

### Diagram Use Case

### Alur Proses Single Sign-On

## Rancangan Solusi

### Desain *Web* *Single Sign-On*

### Desain Dokumentasi *Single Sign-On*

### Desain Aplikasi *Service Provider*

# IMPLEMENTASI DAN EVALUASI

## Lingkungan Implementasi

## Implementasi

### *Web Single Sign-On*

### Dokumentasi *Single Sign-On*

### Aplikasi *Service Provider*

## Evaluasi

Menurut Halonen (2000), autentikasi dan autorisasi seharusnya memiliki karakteristik sebagai berikut:

1. Kebenaran (*Correctness*)

Hasil dari setiap autentikasi dan autorisasi haruslah akurat.

1. Kemungkinan untuk anonim dan privasi (*Possibility to anonymity and privacy*)

Autentikasi terhadap identitas hanya diperlukan apabila benar-benar diperlukan. Apabila tidak perlu membuka identitas, maka tidak perlu identitas dibuka. Jika memungkinkan, autorisasi sebaiknya tidak dilakukan tanpa membuka identitas.

1. Kecepatan (*Speed*)

Proses autentikasi seharusnya dilakukan dengan cepat.

1. Ketahanan terhadap serangan (*Attack resistance*)

Metode autentikasi seharusnya dapat tahan terhadap jenis serangan, misalnya serangan mid-in-the-middle.

1. Murah (*Inexpensiveness*)

Proses autentikasi dan autentikasi seharusnya memerlukan ongkos tidak mahal, baik ongkos pengembangan dan pemeliharaan.

1. Kemudahan penggunaan (*User friendliness*)

Metode autentikasi sedapat mungkin mudah digunakan dan dimengerti. Pada kondisi ideal, pengguna tidak perlu mengeluarkan usaha ketika melakukan autentikasi.

1. Universal (*Universality*)

Metode autentikasi sebaiknya sedapat mungkin sama pada semua layanan.

### Skenario Uji *Web Single Sign-On*

### Kemudahan Integrasi

### Keamanan

### Ketergantungan terhadap Teknologi Tertentu

### Kemampuan Autentikasi

### Kemampuan Autorisasi

### Kemampuan Manajemen Akun

### Integrasi Sosial Media

### Dukungan Dokumentasi

DAFTAR PUSTAKA

Adabala, S., Matsunaga, A., Tsugawa, M., Figueiredo, R., & Fortes, J. A. B. *Single Sign-On in In-VIGO: Role-based Access via Delegation Mechanisms Using Short-lived User Identities*. ACIS Laboratory, University of Florida, Gainesville.

Ardagna, Claudio A., et al. (2006). *CAS++: an Open Source Single Sign-On Solution for Secure e-Services*. Crema: University Degli Study di Milano.

Apereo. (2015). Installation Requirements for CAS | Enterprise Single Sign-On for All. <https://apereo.github.io/cas/development/planning/Installation-Requirements.html#servlet-containers>. Diakses pada 25 Juli 2017.

Atricore. (2010). Simplified Identity an Access | About JOSSO. <http://www.josso.org/>. Diakses pada 18 Juli 2018.

Arianezhad, M., Kelley, T., Camp, L. J., & Stebila, D. *Comparative Eye Tracking of Experts and Novices in Web Single Sign-on*. School of Engineering Science, Simon Fraser University, Burnaby, B.C., Canada. Indiana University Bloomington, IN. Science and Engineering Faculty, Queensland University of Technology, Brisbane, Queensland, Australia.

Bauer, L., Bravo-Lillo, C., Fragkaki, E., & Melicher, W. *A Comparison of Users’ Perceptions of and Willingness to Use Google, Facebook, and Google+ Single-Sign-On Functionality*. Carniage Mellon University, Pittsburgh, PA.

Fleury, T., Basney, J., & Welch, V. *Single Sign-On for Java Web Start Applications Using MyProxy*. National Center for Supercomputing Applications.

Federal Deposit Insurance Corporation. Division of Supervision and Consumer Protection. (2004). *Putting and End to Account-Hijacking Identity Theft*. New York: FDIC.

Ford, M. D. (1998). *Identity Authentication and ‘E-Commerce‘*. The Journal of Information, Law, and Technology.

Ford, W., & Baum, M. S. (1997). Secure Electronic Commerce: Building the infrastructur for digital signature and encryption. New Jersey: Prentice Hall, Inc.

Goth, G. (2008). *Single Sign-on and Social Networks*. IEEE Computer Society, IEEE Distributed System Online, 1541-4922. vol. 8, no. 12.

Halonen, T. (2000). *Authentication and Authorization in Mobile Environment*. Finlandia: Helsinki University of Technology.

Harding, P., Johansson, L., & Klingenstein N. (2007). *Dynamic Security Assertion Markup Language Simplifying Single Sign-On*. IEEE Computer Society, IEEE Security & Privacy, 1540-7993/07.

He, Wenli. (2007). *Single Sign On*. University of Iowa.

Huhnlein, D. et al. *Options for Integrating eID and SAML*. ecsec GmbH, Ruhr-Universitat Bochum. Adolf Wurth GmbH & Co. KG.

Horst, T. W. V. D., & Seamons, K. E. (2007). *Simple Authentication for the Web*. Internet Security Research Lab, Brigham Young University, Provo, UT, USA.

Ishii, T., Inoie, A., & Okamoto, M. *Single Sign-on Using Portable IdP on USB Flash Drive*. Kanagawa Institute of Technology.

Jain, Anil. K. (2005). *On The Uniqueness of Fingerprints*. Dept. of Computer Science and Engineering, Michigan State University.

Jasig. (2009). *Central Authentication Service*. <https://wiki.jasig.org/display/CAS/Home>. Diakses pada 18 Juli 2018.

Kamil, R. (2016). *Smartcity Bandung 2016*. Presentation of Mayor of Bandung.

Kristanto, R. F. (2011). *Otentikasi dan Otorisasi Terpusat pada Aplikasi Web di Lingkungan Informatika ITB*. Program Studi Teknik Informatika, Institut Teknologi Bandung.

Munir M.T., Ir. Rinaldi. (2005). *Diktat Kuliah IF5054 Kriptografi*. Bandung: Institut Teknologi Bandung.

Neuman, B. Clifford, & Theodore Ts’o. (1994). Kerberos: An Authenticaton Service for Computer Networks. IEEE Communications Magazine, Vol 32, No 9, p. 33-38.

Novell. (2011). *Investigating Single Sign-on*. Novell Solution Provider. Waltham, MA.

OASIS. (2008). *Security Assertion Markup Language (SAML) V2.0 Technical Overview*.

Pace, Eugenio. (2015). *About Auth0 | The new way to solve Identity*. <https://auth0.com/about>. Diakses pada 24 Juli 2018.

Pashadilis, Andreas dan Chris J. Mitchell. (2003). *Single Sign-On using Trusted Platforms*. London: University of London.

Park, K., Lim, S. S., & Park, K. H. (2008). *Computationally Efficient PKI-Based Single Sign-On Protocol PKASSO for Mobile Devices*. IEEE Transactions on Computers, vol. 57, no. 6.

Sun, S.-T., & Beznosov, K. *The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems*. Laboratory for Education and Research in Secure Systems Engineering, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada.

Sun, S.-T., Boshmaf, Y., Hawkey, K., & Beznosov, K. *A Billion Keys, but Few Locks: The Crisis of Web Single Sign-On*. Laboratory for Education and Research in Secure System Engineering (LERSSE), Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada.

Sun, S.-T. et al. (2010). *What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID*. University of British Columbia, Vancouver, BC, Canada. Dalhousie University, Halifax, NS, Canada.

Sun, S.-T., Pospisil, E., Muslukhov, I., Dindar, N., Hawkey, K., and Beznosov, K. 2013. *Investigating users’perspectives of Web single sign-on: Conceptual gaps and acceptance model*. ACM Trans. Internet Technol. 13, 1, Article 2 (November 2013), 35 pages.

Tanenbaum, & Steen V., (2007). Distributed System: Principles and Paradigms 2nd Edition. Pearson Education. Vrije Universiteit, Amsterdam, The Netherlands.

Volchkov, A. (2001). *Revisiting Single Sign-On. A Pragmatic Approach in a New Context*. IEEE Computer Society Magazine IT Pro, 1520-9202/01.

Whitehead, Prof. Jim. (2006). *PHP Session Management*. CMPS.

WSO2 Inc, (2015). *WSO2 Identity Server Documentation | Architecture*. <https://docs.wso2.com/display/IS530/Architecture>. Diakses pada 25 Juli 2018.

Yang, R. Li, G., Lau, W. C., Zhang, K., & Hu, P. *Model-based Security Testing: an Empirical Study on OAuth 2.0 Implementations*. The Chinese University of Hong Kong.