**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](#_Toc504045273)

[DAFTAR GAMBAR vi](#_Toc504045274)

[DAFTAR TABEL vii](#_Toc504045275)

[BAB I PENDAHULUAN 1](#_Toc504045276)

[I.1 Latar Belakang 1](#_Toc504045277)

[I.2 Rumusan Masalah 4](#_Toc504045278)

[I.3 Tujuan 4](#_Toc504045279)

[I.4 Batasan Masalah 5](#_Toc504045280)

[I.5 Metodologi 5](#_Toc504045281)

[I.6 Jadwal Pelaksanaan Tugas Akhir 6](#_Toc504045282)

[BAB II STUDI LITERATUR 7](#_Toc504045283)

[II.1 Autentikasi dan Autorisasi 7](#_Toc504045284)

[II.2 Sistem Autentikasi Identitas 8](#_Toc504045285)

[II.2.1 *Password* 9](#_Toc504045286)

[II.2.2 *One Time Password* 9](#_Toc504045287)

[II.2.3 *Token* 9](#_Toc504045288)

[II.2.4 *Biometrics* 9](#_Toc504045289)

[II.2.5 Tanda Tangan Digital dan *Certification Authorities* 9](#_Toc504045290)

[II.3 *Multi Authentication* *Factor* 9](#_Toc504045291)

[II.4 *Single Sign-On* 9](#_Toc504045292)

[II.5 *Web Single Sign*-*On* 9](#_Toc504045293)

[II.6 Pekerjaan Terkait 9](#_Toc504045294)

[II.7 Single Sign-On 9](#_Toc504045295)

[II.7.1 Definisi Umum 9](#_Toc504045296)

[II.7.2 Kelebihan dan Kekurangan 10](#_Toc504045297)

[II.8 Autentikasi dan Autorisasi 10](#_Toc504045298)

[II.8.1 Autentikasi 10](#_Toc504045299)

[II.8.2 Autorisasi 11](#_Toc504045300)

[II.9 Mekanisme *Single Sign-On* 11](#_Toc504045301)

[II.10 Penggunaan *Single Sign-On* 15](#_Toc504045302)

[II.10.1 Social Single Sign-On 16](#_Toc504045303)

[II.10.2 Internal *Single Sign-On* 17](#_Toc504045304)

[II.11 Keamanan *Single Sign-On* 18](#_Toc504045305)

[II.11.1 Mengurangi Jumlah *Password* 18](#_Toc504045306)

[II.11.2 *Security* *Policies* 19](#_Toc504045307)

[II.11.3 Penyimpanan *Credential* 19](#_Toc504045308)

[II.11.4 Mengirimkan *Credential* lewat Jaringan 19](#_Toc504045309)

[II.12 Contoh Implementasi *Single Sign-On* 20](#_Toc504045310)

[II.12.1 OneLogin SAML 20](#_Toc504045311)

[II.12.2 Auth0 23](#_Toc504045312)

[II.12.3 OpenID Connect 27](#_Toc504045313)

[BAB III ANALISIS PERMASALAHAN DAN RANCANGAN SOLUSI 29](#_Toc504045314)

[III.1 Analisis Permasalahan 29](#_Toc504045315)

[III.1.1 *Workflow* General *Single Sign-On* 29](#_Toc504045316)

[III.2 Analisis Kebutuhan 31](#_Toc504045317)

[III.2.1 Deskripsi Aplikasi 31](#_Toc504045318)

[III.2.2 Kebutuhan Fungsional Aplikasi 32](#_Toc504045319)

[III.3 Rancangan Solusi 33](#_Toc504045320)

[III.3.1 Aspek Pertimbangan *Single Sign-On* 33](#_Toc504045321)

[III.3.2 Rencana Desain *Single Sign-On* 35](#_Toc504045322)

[III.3.3 Pembangunan Single Sign-On 35](#_Toc504045323)

[DAFTAR PUSTAKA 37](#_Toc504045324)

DAFTAR GAMBAR

[Gambar II.1.Mekanisme *log-on* tanpa *single sign-on* 9](#_Toc471296608)

[Gambar II.2.Mekanisme *log-on* dengan *single sign-on* sederhana secara general 10](#_Toc471296609)

[Gambar II.3. *Single sign-on* pada sistem jaringan yang terhubung 11](#_Toc471296610)

[Gambar II.4. Tombol *log-in* yang disediakan oleh *service provider* untuk melakukan *single sign-on* menggunakan *social single sign-on* 13](#_Toc471296611)

[Gambar II.5. Spotify menyediakan pilihan untuk *login* menggunakan akun Facebook dan Google. 14](#_Toc471296612)

[Gambar II.6. Sequence diagram untuk SAML 18](#_Toc471296613)

[Gambar II.7. Diagram *Sequence* untuk SAML ketika user sudah terautentikasi 19](#_Toc471296614)

[Gambar II.8. Diagram keterhubungan antara aplikasi, Auth0, dan user. 20](#_Toc471296615)

[Gambar II.9. Tampilan halaman Auth0 Hosted Lock 21](#_Toc471296616)

[Gambar II.10. Diagram *sequence* Auth0 apabila user mengunjungi app pertama kalinya. 22](#_Toc471296617)

[Gambar II.11. Tampilan halaman Auth0 Hosted Lock bila user pernah *login* sebelumnya 23](#_Toc471296618)

[Gambar II.12. Diagram *sequence* Auth0 untuk kunjungan berikutnya. 23](#_Toc471296619)

[Gambar ‎II.13. Diagram *sequence* OpenID Connect. 25](#_Toc471296620)

[Gambar ‎III.1. Diagram *sequence* *workflow* general *single sign-on* 27](#_Toc471296621)

DAFTAR TABEL

[Tabel II.1. Rencana Pelaksanaan Tugas Akhir I 6](#_Toc471296622)

[Tabel ‎III.1. Aspek yang perlu dikaji dari *single sign-on* sesuai kebutuhan fungsional 30](#_Toc471296623)

[Tabel ‎III.2.Perbedaan antar implementasi *single sign-on* 30](#_Toc471296624)

# 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.

### WSO2 *Identity Provider*

### *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 alam pembuatan mekanisme autentikasi. Cara kerja protokol CAS dapat dilihat pada Gambar II.x.



Gambar ‎II.3 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++*

### Kristanto, 2011

## Contoh Implementasi *Single Sign-On*

Ada banyak implementasi *single sign-on* saat ini yang kita temui. Tiap-tiap implementasi tentunya memiliki perbedaan masing-masing, berupa kelebihan, kekurangan, dan mekanisme yang berbeda dalam melakukan banyak hal seperti menyimpan *credential*, membagikan akses untuk *credential*, dan lainnya.

Bab ini akan membahas secara detil mengenai metode dari tool yang mengimplementasi *single sign-on*.

### OneLogin SAML

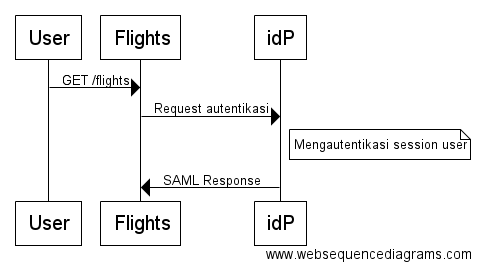
SAML (*Security Assertion Markup Language*) adalah protokol standar open-source untuk *web browser* *single sign-on* menggunakan token. SAML menghilangkan seluruh *password* dalam proses autentikasi. SAML menggunakan *digital signature* untuk mengirim token keamanan yang diberikan oleh *identity provider* ke aplikasi. Token keamanan ini berisi

SAML berbasis XML, merupakan produk dari OASIS *Security Services Technical Committee*. Banyak vendor besar perangkat lunak sudah men-support penggunaan SAML seperti Google, Microsoft, Salesforce.

#### Mekanisme OneLogin SAML

Misalkan seorang user akan mengakses dua web-app yang divendori oleh perusahaan yang sama. Satu untuk *booking* penerbangan, satunya untuk rental mobil. Dalam istilah SAML, user disebut *principal*. Dua aplikasi tersebut disebut *service provider*.

Misalkan user tersebut akan mengakses <http://flight.example.com/flights>, aplikasi flight akan me-*redirect* ke <https://sso.example.com?SAMLRequest>=... untuk melakukan *single sign-on*. Ketika user akan mengakses *resource* /flights yang diproteksi, *service provider* (aplikasi flight) tidak menampilkan *login page*, tetapi akan melakukan *redirect* ke *identity provider*, dalam hal ini adalah <https://sso.example.com> dengan *request* autentikasi yang disertakan pada parameter secara eksplisit. *Identity provider* menggunakan parameter sebagai informasi, *service provider* mana yang melakukan *request*. User akan *log-in* (mengetikkan *email* dan *password*-nya) melalui pada https://sso.example.com. *Identity provider* kemudian bertindak sebagai otoritas penyimpan *session*, menyimpan fakta bahwa user punya *session* yang valid. *Session* ini akan berguna nantinya.

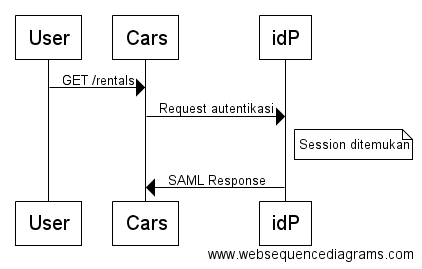


Gambar ‎II.4. Sequence diagram untuk SAML

Setelah *identity provider* berhasil mengautentikasi user, IdP mengirimkan response ke *service provider* mengatakan bahwa autentikasi telah berhasil. IdP kemudian memberikan informasi seperti email, nama, dan kontak lain ke *service provider. Service provider* melakukan update terhadap informasi ini dengan cara diberikan ke *identity provider*.

Setelah selesai urusan dengan aplikasi *flight*, user melakukan *request* ke <https://cars.example.com/rentals>. Awalnya aplikasi rental mobil ini melakukan *redirect* ke <https://sso.example.com?SAMLRequest>=..., tetapi kemudian di-*redirect* kembali ke <http://cars.example.com/rentals> tanpa perlu mengetikkan *password*-nya kembali.

Mengapa user tidak perlu diautentikasi kembali? Ingat bahwa saat dia di-*redirect* ke *identity provider* ketika *booking flight*, *identity provider* menyimpan *session* bahwa user sudah terautentikasi. Ketika *service provider* lain yakni <https://cars.example.com> mengirimkan permintaan autentikasi, *identity provider* tahu bahwa user sudah terautentikasi sebelumnya karena ada *session* yang aktif. Dalam hal ini, user tidak perlu diautentikasi kembali, karena itu *identity provider* me-*redirect* kembali ke aplikasi car rental.



Gambar ‎II.5. Diagram *Sequence* untuk SAML ketika user sudah terautentikasi

#### *Binding* OneLogin SAML

*Binding* adalah metode untuk mendefinisikan bagaimana SAML dan bagaimana protokol *request*-*response* dapat dipertukarkan antar sistem menggunakan protokol komunikasi dan *framework*.

Secara sederhana dalam kasus *single sign-on*, *binding* mendefinisikan bagaimana user di-*redirect* antara *identity provider* dan *service provider*. Pada SAML, *binding* menjelaskan bagaimana pesan seharusnya di-*encode* dan bagaimana layer *transport* yang dilewati. Untuk web *single sign-on*, biasanya digunakan HTTP *Redirect Binding* dan HTTP *Post Binding*. *Redirect* dari *service provider* ke *identity provider* biasanya menggunakan *redirect binding*, sementara *response* dari *identity provider* ke *service provider* harus menggunakan *post binding*. Normalnya *binding* menggunakan HTTP, tetapi SAML juga mendukung metode *binding* lain seperti SOAP, URI, dan Artifact.

### Auth0

Auth0 adalah servis yang menyediakan bagaimana user terautentikasi ke aplikasi.



Gambar ‎II.6. Diagram keterhubungan antara aplikasi, Auth0, dan user.

Sebagai developer, kita bisa menghubungkan aplikasi apapun (ditulis dalam bahasa apapun dengan teknologi apapun) ke Auth0 dan mendefinisikan metode bagaimana user harus terautentikasi ke aplikasi tersebut.

Auth0 bukan bertindak sebagai *identity provider*, tetapi hanya sebagai servis pusat yang bekerja sebagai pengintegrasi beberapa autentikasi yang dapat kita atur. Yang bertindak sebagai *identity provider* adalah vendor *identity provider* yang sudah di-setting sebelumnya, contohnya Facebook, Google, Twitter, dst.

Berikut adalah metode autentikasi yang didukung oleh Auth0:

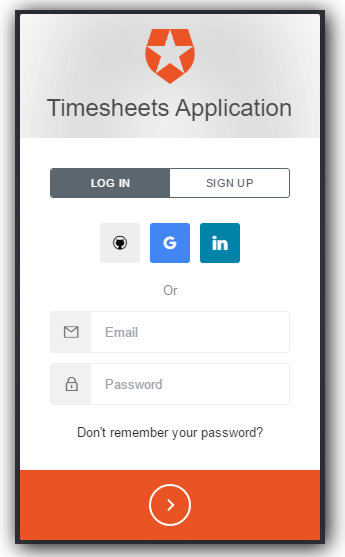
1. Credential standar: *username* dan *password*.
2. *Login* media sosial: Google, Facebook, Twitter, OAuth2, OAuth1, dan OpenID *identity provider*.
3. *Enterprise directory*: LDAP, Google APPs, Office 365, ADFS, AD, SAML-P, WS-Federation, etc.
4. Sistem tanpa *password*: TouchID, OTP, SMS, email.

#### Mekanisme Auth0

Dalam kasus *single sign-on* menggunakan Auth0, Auth0 Authorization Server bertindak sebagai server pusat. Server pusat ini dapat diakses secara *remote* dari aplikasi yang kita inginkan. Seluruh yang bisa terkoneksi ke protokol HTTP, sejatinya bisa menggunakan mekanisme autentikasi *single sign-on* Auth0.

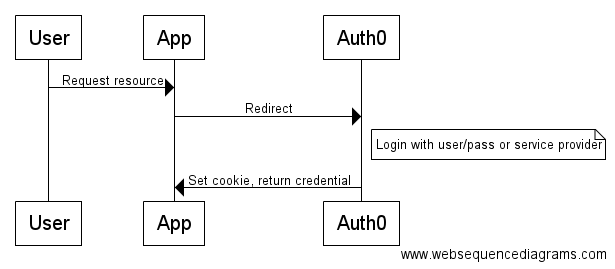
Berikut adalah langkah-langkah ketika user mengunjungi aplikasi yang terlindungi Auth0 untuk pertama kalinya:

1. Aplikasi akan melakukan *redirect* user ke halaman Auth0 Hosted Lock di mana user bisa *login*.
2. Auth0 akan melakukan pengecekan apakah *cookie single sign-on* pernah ada sebelumnya.
3. Karena ini adalah pertama kalinya user mengunjungi Hosted Lock Page, maka dari itu tidak ada *single sign-on cookie* yang ada. User akan diberikan form berisi *username, password,* dan mungkin beberapa *identity provider* lain yang sudah di-*support* oleh aplikasi ini sebelumya seperti LinkedIn, GitHub, Google. Tampilan dari Hosted Lock Page akan berbeda-beda tergantung pada *identity provider* yang sudah dikonfigurasikan untuk aplikasi ini.



Gambar ‎II.7. Tampilan halaman Auth0 Hosted Lock

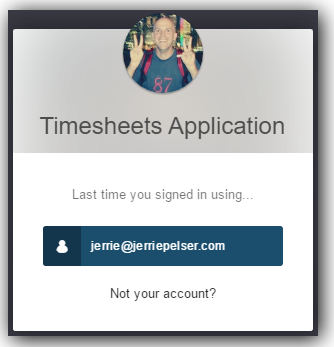
1. User akan login, Auth0 akan menyimpan *single sign-on cookie*.
2. Auth0 akan mengembalikan user ke aplikasi *service provider* dan mengembalikan id\_token yang berisi identitas user.



Gambar ‎II.8. Diagram *sequence* Auth0 apabila user mengunjungi app pertama kalinya.

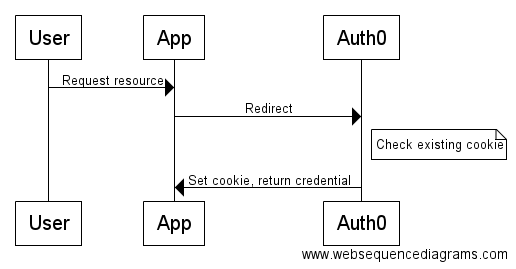
Sekarang bagaimana jika user kembali ke web untuk kunjungan berikutnya? Hal ini yang akan terjadi.

1. Aplikasi me-*redirect* user ke Auth0 Hosted Lock di mana user bisa *login*.
2. Auth0 akan mengecek apakah ada *single sign-on cookie*.
3. Ketika Auth0 menemukan ada *single sign-on cookie*, Auth0 tidak menampilkan form dengan *username* dan *password*, tetapi akan menampilkan *lock screen* yang menandakan bahwa user yang bersangkutan sudah pernah *login* sebelumnya. User dapat hanya mengkonfirmasi bahwa mereka ingin *login* dengan metode yang sama.



Gambar ‎II.9. Tampilan halaman Auth0 Hosted Lock bila user pernah *login* sebelumnya

1. Auth0 akan memperbarui *single sign-on cookie* apabila diperlukan.
2. Auth0 akan mengembalikan *user* ke aplikasi dan mengembalikan id\_token berisi identitas user.



Gambar ‎II.10. Diagram *sequence* Auth0 untuk kunjungan berikutnya.

### OpenID Connect

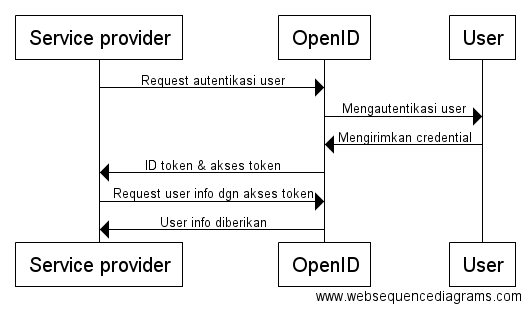
OpenID Connect adalah *framework* untuk identitas yang dibangun di atas protokol OAuth 2.0 yang mengembangkan *workflow* autorisasi OAuth 2.0 dengan menambahkan mekanisme autentikasi. Seperti yang kita ketahui, bahwa OAuth merupakan protokol yang berfungsi untuk mengatur autorisasi antar *service* *provider*, seperti *resource* mana yang bisa diakses oleh *service* tertentu, seberapa tingkat level aksesnya, dan sebagainya. OpenID dapat memberikan user keleluasaan untuk menggunakan *identity provider* yang mereka mau asalkan *identity provider* tersebut mendukung OpenID Connect.

Seperti mekanisme *single sign-on* yang lain, OpenID juga terbagi menjadi *identity provider* dan *service provider*. Server yang mengimplementasi OpenID Connect disebut *identity provider*, sementara *service* yang menggunakan OpenID Connect disebut dengan *service provider*. Untuk mengautentikasi menggunakan OpenID Connect, *service provider* harus mengetahui informasi mengenai *identity provider*, seperti *endpoint* autorisasi, *token* *endpoint*, dan seterusnya.

#### Mekanisme OpenID Connect

Berikut adalah mekanisme login menggunakan OpenID Connect.

1. *Service provider* mengirimkan *request* ke OpenID provider untuk mengautentikasi user.
2. OpenID provider mengautentikasi user
3. OpenID provider mengirimkan ID token dan akses token ke *service provider*.
4. *Service provider* mengirimkan *request* *endpoint* *user* *info* kepada OpenID provider menggunakan akses token yang sudah diberikan.
5. *Endpoint user info* diberikan oleh OpenID provider ke *service provider*.



Gambar ‎II.11. Diagram *sequence* OpenID Connect.

# ANALISIS PERMASALAHAN DAN RANCANGAN SOLUSI

## Analisis Permasalahan

### Analisis *Web Single Sign-On*

### Analisis Protokol *Single Sign-On*

### Analisis Metode Autentikasi

### Analisis Metode Autorisasi

### Kebutuhan Fungsional *Single Sign-On*

### Kebutuhan Fungsional Aplikasi *Service Provider*

## Rancangan Solusi

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

### Desain *Blueprint* Dokumentasi *Single Sign-On*

### Desain Aplikasi *Service Provider*

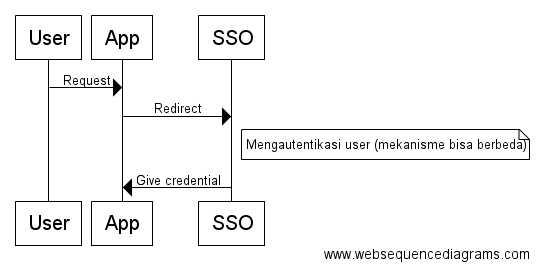
Pada bab ini akan dipaparkan analisis permasalahan yang ada dalam studi dan implementasi *single sign-on*. Selain itu, akan dipaparkan juga rumusan solusi yang akan diajukan dalam bentuk langkah-langkah pekerjaan yang akan dilakukan.

## Analisis Permasalahan

*Single sign-on* bukan merupakan solusi baru yang mengatasi permasalahan multi-autentikasi pada perusahaan. Sejarah mengatakan bahwa OpenID sudah menyelesaikan masalah *single sign-on* sejak 2005 (Sun, 2011). Hingga saat ini, telah banyak diajukan solusi permasalahan dalam bentuk implementasi *single sign-on oleh* beberapa perusahaan maupun organisasi nonprofit. Tetapi, tidak seluruh implementasi menawarkan solusi yang sempurna.

### *Workflow* General *Single Sign-On*

Ada banyak implementasi *single sign-on* yang mengatasi permasalahan multiautentikasi. Masing-masing implementasi memiliki perbedaan detil masing-masing dalam metode penyimpanan *credential*, bagaimana hak akses diberikan, dan bagaimana mengelola *credential* diberikan untuk aplikasi yang berbeda. Namun, sebagian besar implementasi *single sign-on* mengerucut pada sebuah persamaan yang saling bersinggungan. *Workflow* untuk *single sign-on* sebagian besar sama, yakni sebagai berikut.



Gambar .. Diagram *sequence* *workflow* general *single sign-on*

Apabila ditimbang dan diteliti dengan seksama, ada beberapa persamaan global di antara beberapa implementasi *single sign-on* yang dapat ditambil titik temunya. Persamaan tersebut adalah sebagai berikut.

#### Mekanisme *identity provider* dan *service provider*

*Single sign-on* memiliki *term* untuk *identity provider* dan *service provider*. *Identity provider* adalah *service* yang bertujuan untuk mengelola dan menyediakan *credential*. *Service provider* adalah aplikasi yang mengkonsumsi autentikasi dari *identity provider*.

#### Servis sentral untuk autentikasi

Untuk permasalahan autentikasi, seluruh mekanisme *single sign-on* memiliki satu servis sentral yang khusus didedikasikan untuk autentikasi. Apa yang dilakukan *service provider* di belakang ketika menyimpan dan mengirim *credential* bisa saja berbeda. Tetapi tujuan adanya servis sentral tetap sama yakni untuk autentikasi.

#### Aplikasi tidak mengautentikasi user

Aplikasi sebagai *service provider* tidak melakukan autentikasi. Sebagai gantinya, setiap *service provider* membutuhkan user untuk diautentikasi, aplikasi akan menyerahkan permasalahan autentikasi kepada *identity provider*. *Identity provider* akan menyelesaikan autentikasi dengan user, apabila sudah selesai, *identity provider* akan memberikan semacam token/*cookies* user kepada aplikasi untuk digunakan pada akses berikutnya.

## Analisis Kebutuhan

Pada subbab ini, akan dipaparkan mengenai kebutuhan aplikasi *single sign-on* termasuk deskripsi aplikasi dan apa yang menjadi kebutuhan fungsional.

### Deskripsi Aplikasi

Aplikasi *single sign-on* merupakan sentral pusat autentikasi pada sistem aplikasi-aplikasi citizen di Kota Bandung. Sistem aplikasi *citizen* ini terdiri dari beberapa service, yakni sebagai berikut:

1. Administrasi kependudukan. Untuk tujuan administrasi Kartu Tanda Penduduk, Kartu Keluarga, Akta Kelahiran, dan lain-lain.
2. Perijinan. Untuk pengajuan izin terkait pendirian industri dan bisnis.
3. Perpajakan. Untuk pembayaran pajak.
4. Keluhan publik. Sebagai jembatan masyarakat menyampaikan keluhan ke pemerintah kota.

Aplikasi *single sign-on* harus mampu menyediakan layanan autentikasi user untuk masuk ke aplikasi-aplikasi yang disebutkan di atas. Dengan demikian, ketika user akan mengakses salah satu dari aplikasi, user harus melewati sistem autentikasi *single sign-on*. Setelah user berhasil diautentikasi, user akan diberi ijin akses pada aplikasi bersangkutan.

Dengan adanya *single sign-on*, *user* (penduduk Kota Bandung) harapannya cukup memiliki satu akun untuk masuk ke seluruh layanan publik Kota Bandung. Apabila ini tercapai, maka sistem aplikasi akan menjadi sangat efisien.

### Kebutuhan Fungsional Aplikasi

#### Autentikasi

Autentikasi adalah proses pemastian bahwa user yang akan *login* merupakan orang yang benar seperti yang ia klaim. Setiap warga Bandung, tentunya punya ID unik berupa nomor KTP. Nomor KTP yang berbeda akan menandakan bahwa mereka adalah orang yang berbeda. Dari nomor KTP ini, dapat di-*generate* *username* dan *password* untuk tiap orang.

Aplikasi *single sign-on* harus mampu melakukan autentikasi menggunakan *username* dan *password* yang terpusat. Untuk menjamin, *username* supaya tetap unik, setiap *username* dapat di-*binding* dengan nomor KTP yang sudah jelas merupakan ID unik bagi penduduk Bandung.

Aplikasi *single sign-on* harus bisa menerima autentikasi sebagai satu-satunya gerbang utama untuk mengakses sistem aplikasi *citizen*. Artinya, *user* harus memasukkan *credential* melalui aplikasi ini sebelum bisa mengakses *resource* pada aplikasi *citizen*. Sebagai server pusat autentikasi, aplikasi *single sign-on* juga harus mampu mengautentikasi user apakah memiliki *credential* yang tepat dengan cara yang aman.

Aplikasi *single sign-on* juga harus menjamin seluruh mekanisme autentikasi dilakukan dengan cara yang aman, termasuk penyimpanan *credential* dan transmisi *credential* ketika komunikasi antara server *single sign-on* dengan aplikasi *service provider*.

#### Autorisasi

Aplikasi *single sign-on* harus memiliki kapabilitas untuk akses level user. Bisa saja untuk user tertentu memiliki akses ke 5 aplikasi, tapi bisa saja user lain hanya memiliki akses hanya ke satu aplikasi saja. Aplikasi *single sign-on* harus mampu memberikan fleksibilitas yang mengatur *user* mana yang bisa mengakses *resource* mana saja.

#### Manajemen Akun

Aplikasi *single sign-on* harus memiliki kemampuan untuk mengatur autorisasi user. Artinya ada level akses administrator yang memiliki kemampuan mengakses daftar user, menambah user, menghapus user, mengubah hak akses level user.

## Rancangan Solusi

Secara singkat, kebutuhan fungsional *single sign-on* mencakup tiga hal, yakni: autentikasi terpusat, autorisasi *service* per *user*, manajemen *user*. Untuk menjawab rumusan masalah yang sudah dikemukakan di bab 1 sebelumnya, harus dilakukan pertimbangan untuk memilih solusi mana yang paling *feasible* sesuai dengan kebutuhan fungsional *single sign-on*.

### Aspek Pertimbangan *Single Sign-On*

Sesuai dengan kebutuhan fungsional, kita harus menimbang aspek-aspek sesuai dengan kebutuhan yang telah dijabarkan. Dari kebutuhan fungsional, dapat diturunkan aspek-aspek yang perlu dikaji dari tiap implementasi *single sign-on*. Aspek-aspek tersebut adalah sebagai berikut.

Tabel ‎III.1. Aspek yang perlu dikaji dari *single sign-on* sesuai kebutuhan fungsional

| Kebutuhan Fungsional | Aspek |
| --- | --- |
| General | Basis, *propietary*, *message exchange*, keamanan |
| Autentikasi | *Scope*, *integration support*, bertindak sebagai, bagaimana session disimpan, kemampuan autentikasi |
| Autorisasi | *Scope*, *integration support*, kemampuan autorisasi |
| Manajemen akun | *Scope*, *integration support*, kemampuan manajemen akun |

Tabel ‎III.2.Perbedaan antar implementasi *single sign-on*

|  | SAML | Auth0 | OpenID Connect |
| --- | --- | --- | --- |
| Basis | XML-*based* | Token-*based* | HTTP Post |
| *Proprietary* | *Open-source* | *Open-source* | *Open-source* |
| *Scope* | Hanya mendukung autentikasi antara *service provider* dan *identity provider* saja.  Dapat di-*custom* menjadi bermacam-macam *workflow* sesuai kebutuhan. Misal: SSO-*first redirect/redirect*, *destination-first redirect*/POST | Mendukung protokol SAML, OAuth 2.0, OpenID Connect. Memiliki keterhubungan yang luas dengan *identity provider* lain, misal: *connect with* Facebook, Google, Twitter. | Dibangun di atas OAuth 2.0 (*framework* autorisasi) yang ditambah dengan mekanisme autentikasi. Melakukan standarisasi beberapa hal yang dibebaskan di OAuth 2.0, contohnya *scope instance*, *endpoint discovery*, pendaftaran *client* secara dinamis |
| *Message exchange* | HTTP Redirect, HTTP Post. Mendukung SOAP, Artifact | HTTP Redirect, HTTP Post. Mendukung SOAP, URI, Artifact | HTTP Redirect, HTTP Post |
| *Integration Support* | Baik. Mendukung banyak *toolkit* | Baik. Mendukung banyak identity provider lain. Punya *dashboard* untuk manajemen *identity provider* yang sudah di-*setup* | Kurang baik. Dokumentasi pengembangan tidak terlalu lengkap. |
| Bertindak sebagai | *Identity Provider* | Tidak benar-benar menjadi *identity provider*. Hanya sebagai pengintegrasi banyak identity provider yang sudah ada. | *Identity provider* dengan dukungan *credential* terbatas. Hanya mendukung informasi dasar berupa *username* dan *password* |
| Bagaimana *session* disimpan | Dalam kasus web SSO, SAML Token, disimpan sebagai *cookie* pada *browser*. | Disimpan pada *cookie* di *browser*. | Disimpan pada *cookie* di *browser*. |
| Keamanan level *transport* | TLS 1.0 | TLS 1.2 dengan minimal 128-bit enkripsi AES. | TLS, sama dengan OAuth 2.0. |
| Keamanan level *message* | XML Signature dan XML Encryption. | TLS 1.2, AES\_128\_GCM,  ECDHE\_RSA | TLS. |
| Autentikasi | *Capable* dalam melakukan autentikasi *single sign-on* | *Capable* dalam melakukan autentikasi *single sign-on* | *Capable* dalam melakukan autentikasi *single sign-on* |
| Kemampuan autorisasi | Tidak mendukung autorisasi. Autorisasi harus dilakukan secara *hard-code* di *app-level* berdasarkan atribut pada token keamanan pada SAML *header*. | Karena dibangun di atas protokol OAuth 2.0, Auth0 mendukung fitur autorisasi. | Karena dibangun di atas protokol OAuth 2.0, Open ID Connect mendukung fitur autorisasi. |
| Manajemen user | Dilakukan di level *app*. Kemampuan manajemen akan mengikuti kemampuan autorisasi. | Dilakukan di level *app*. Kemampuan manajemen akan mengikuti kemampuan autorisasi. | Dilakukan di level *app*. Kemampuan manajemen akan mengikuti kemampuan autorisasi. |

### Rencana Desain *Single Sign-On*

Berdasarkan kajian hingga saat ini, *single sign-on* rencananya akan dibangun menggunakan Auth0 karena fitur yang ditawarkan paling sesuai dengan kebutuhan fungsional. Auth0 memiliki kemampuan untuk autentikasi *single sign-on* dan kemampuan autorisasi. Sementara itu, Auth0 juga mendukung protokol SAML, OAuth 2.0, dan OpenID Connect sehingga misalkan Auth0 harus disambungkan dengan *service* yang mendukung protokol tersebut, Auth0 bisa dengan mudah diintegrasikan.

Apabila dalam iterasi studi literatur berikutnya ternyata ditemukan kakas yang lebih memungkinkan untuk mengimplementasi *single sign-on*, maka rencana desain *single sign-on* bisa saja berubah apabila kakas tersebut memang benar-benar mampu menjawab kebutuhan fungsional yang telah dipaparkan di atas.

### Pembangunan Single Sign-On

Berikut adalah rencana pembangunan aplikasi *single sign-on* yang akan dikerjakan.

#### Membangun Aplikasi Single Sign-On

Aplikasi utama *single sign-on* yang paling utama harus dikerjakan terlebih dahulu. *Single sign-on* ini bertugas sebagai pintu gerbang utama autentikasi bagi aplikasi-aplikasi lain. *Single sign-on* yang diimplementasi harus memiliki kebutuhan fungsional minimum sebagai berikut.

1. Autentikasi dilakukan menggunakan *username* dan *password* yang di-*binding* dengan KTP. Asumsikan user adalah penduduk yang legal memiliki KTP dan berdomilisi di Kota Bandung baik sementara atau tetap.
2. Autentikasi pada server pusat bersifat *single sign-on* sebagai akses ke aplikasi lain dalam cakupan sistem aplikasi *citizen*.
3. Autentikasi dilakukan secara aman baik baik ketika transmisi data maupun penyimpanan data.
4. Aplikasi mampu mendukung pengaturan pemberian level akses aplikasi kepada user, baik pengurangan akses maupun penambahan akses.
5. Aplikasi memiliki akun administrator yang dapat digunakan untuk mengatur autorisasi, pengelolaan, penambahan, dan pengurangan user.

#### Membangun Aplikasi *Citizen*

Aplikasi *citizen* ini terdiri dari beberapa komponen: kependudukan, perijinan, perpajakan, keluhan publik. Aplikasi citizen ini berkedudukan sebagai *service provider* yang mengkonsumsi *single sign-on* sebagai *identity provider*. Aplikasi yang diimplementasi harus memiliki kebutuhan minimum sebagai berikut.

1. Aplikasi dapat diintegrasikan dengan *single sign-on* sebagai *identity provider*.
2. Aplikasi dapat berkomunikasi dengan *identity provider* melalui media yang aman pada jaringan.
3. User dapat *login* pada aplikasi ini setelah terautentikasi pada *single sign-on*, baik setelah autentikasi pada *identity provider* maupun melalui *session* yang masih tersimpan apabila user pernah *login* beberapa saat sebelumnya.
4. Aplikasi dapat membatasi akses apabila user tidak diberi hak untuk melakukan aksi yang tidak seharusnya.

# IMPLEMENTASI DAN EVALUASI

## Lingkungan Implementasi

## Implementasi

### *Web Single Sign-On*

### *Blueprint* 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.

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.

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.