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

**Laporan Tugas Akhir I**

**Disusun sebagai syarat kelulusan mata kuliah**

**IF4091/Tugas Akhir I dan Seminar**

**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 I**

**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 iv](#_Toc471302871)

[DAFTAR GAMBAR vi](#_Toc471302872)

[DAFTAR TABEL vii](#_Toc471302873)

[BAB I PENDAHULUAN 1](#_Toc471302874)

[I.1 Latar Belakang 1](#_Toc471302875)

[I.2 Rumusan Masalah 4](#_Toc471302876)

[I.3 Tujuan 4](#_Toc471302877)

[I.4 Batasan Masalah 5](#_Toc471302878)

[I.5 Metodologi 5](#_Toc471302879)

[I.6 Jadwal Pelaksanaan Tugas Akhir 6](#_Toc471302880)

[BAB II STUDI LITERATUR 7](#_Toc471302881)

[II.1 *Single* *Sign-On* 7](#_Toc471302882)

[II.1.1 Definisi Umum 7](#_Toc471302883)

[II.1.2 Kelebihan dan Kekurangan 7](#_Toc471302884)

[II.2 Autentikasi dan Autorisasi 8](#_Toc471302885)

[II.2.1 Autentikasi 8](#_Toc471302886)

[II.2.2 Autorisasi 8](#_Toc471302887)

[II.3 Mekanisme *Single Sign-On* 8](#_Toc471302888)

[II.4 Penggunaan *Single Sign-On* 12](#_Toc471302889)

[II.4.1 Social Single Sign-On 13](#_Toc471302890)

[II.4.2 Internal *Single Sign-On* 14](#_Toc471302891)

[II.5 Keamanan *Single Sign-On* 15](#_Toc471302892)

[II.5.1 Mengurangi Jumlah *Password* 15](#_Toc471302893)

[II.5.2 *Security* *Policies* 16](#_Toc471302894)

[II.5.3 Penyimpanan *Credential* 16](#_Toc471302895)

[II.5.4 Mengirimkan *Credential* lewat Jaringan 16](#_Toc471302896)

[II.6 Contoh Implementasi *Single Sign-On* 17](#_Toc471302897)

[II.6.1 OneLogin SAML 17](#_Toc471302898)

[II.6.2 Auth0 20](#_Toc471302899)

[II.6.3 OpenID Connect 24](#_Toc471302900)

[BAB III Analisis permasalahan dan rancangan solusi 26](#_Toc471302901)

[III.1 Analisis Permasalahan 26](#_Toc471302902)

[III.1.1 *Workflow* General *Single Sign-On* 26](#_Toc471302903)

[III.2 Analisis Kebutuhan 28](#_Toc471302904)

[III.2.1 Deskripsi Aplikasi 28](#_Toc471302905)

[III.2.2 Kebutuhan Fungsional Aplikasi 29](#_Toc471302906)

[III.3 Rancangan Solusi 30](#_Toc471302907)

[III.3.1 Aspek Pertimbangan *Single Sign-On* 30](#_Toc471302908)

[III.3.2 Rencana Desain *Single Sign-On* 32](#_Toc471302909)

[III.3.3 Pembangunan Single Sign-On 32](#_Toc471302910)

[DAFTAR PUSTAKA 34](#_Toc471302911)

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.

## Jadwal Pelaksanaan Tugas Akhir

Berikut adalah rencana kegiatan dan jadwal pengerjaan tugas akhir.

Tabel .. 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

Dalam bab ini akan dibahas mengenai studi literatur yang mencakup definisi *single sign-on*, penggunaan single sign-on saat ini, metode-metode *single sign-on* yang sudah ada sebelumnya, dan studi lain terkait *single-sign on*.

## *Single* *Sign-On*

### Definisi Umum

*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.

### Kelebihan dan Kekurangan

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. Apabila aplikasi sudah ada dan *single sign-on* akan diintegrasi, *cost* untuk mengintegrasikan single sign-on dengan aplikasi yang sudah ada cukup besar.

## Autentikasi dan Autorisasi

### Autentikasi

Autentikasi adalah proses di mana *credential* yang diberikan oleh user akan dibandingkan dengan *credential* pada *database* yang berisi informasi-informasi user pada server autentikasi. Apabila credential yang diberikan cocok, maka proses telah selesai dan user diberikan ijin akses.

### Autorisasi

Autorisasi adalah proses di mana seorang administrator memberi hak kepada user tertentu untuk mengakses *resource* tertentu. Autorisasi juga meliputi proses ketika sistem mengecek apakah user yang bersangkutan memiliki hak akses pada *resource* yang dilindungi tersebut.

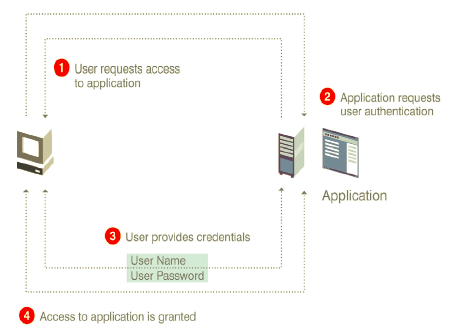
## Mekanisme *Single Sign-On*

Untuk mengetahui bagaimana *single sign-on* bekerja, kita perlu tahu bagaimana perbedaan mekanisme *single sign-on* di metode yang berbeda. Tujuan kita mengetahui mekanisme *single sign-on* adalah untuk mengetahui bagaimana teknologi berikut berjalan pada proses *single sign-on*.

Normalnya, suatu user dalam sehari rata-rata melakukan 5-10 login yang berbeda (Novell, 2011). Karena alasan ini, produk yang mengimplementasi *single sign-on* dituntut agar dapat menyimpan login detail supaya dapat memberikan *credential* yang benar ketika suatu saat dibutuhkan kembali. Walaupun konsep dalam mekanisme *single sign-on* sama untuk sebagian besar produk, tetap ada perbedaan dalam bagaimana *credential* user disimpan dan diakses (Novell, 2011).

*Single sign-on* didesain untuk mengurangi jumlah login yang harus diingat user, juga supaya user semakin merasa efisien. *Single sign-on* menyediakan *credential* secara otomatis apabila user sudah login ke aplikasi sebelumnya.

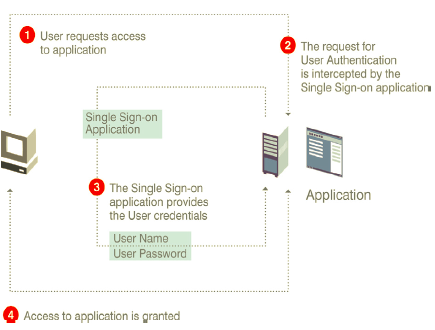
Berikut adalah mekanisme *sign-on* pada aplikasi biasa tanpa *single sign-on*.



Gambar ..Mekanisme *log-on* tanpa *single sign-on*

Mekanisme *logon* tanpa *single sign-on* sebenarnya cukup trivial. *Credential* yang dimasukkan oleh user diproses/validasi secara langsung oleh aplikasi. Apabila sebuah perusahaan memiliki 50 aplikasi, maka dengan mekanisme ini akan ada 50 metode autentikasi yang berbeda juga.

Berikut adalah mekanisme *sign-on* pada aplikasi menggunakan *single sign-on* sederhana.



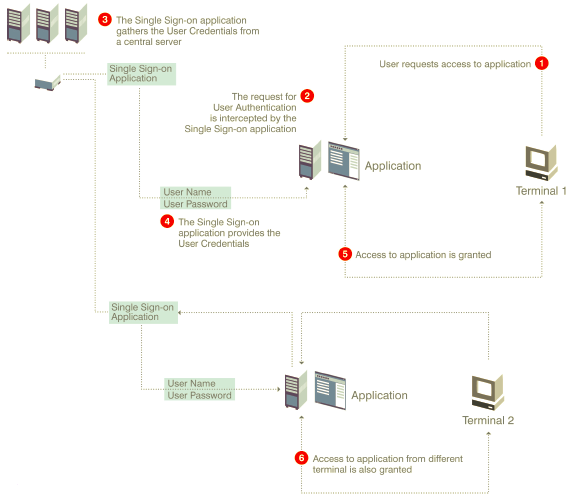
Gambar ..Mekanisme *log-on* dengan *single sign-on* sederhana secara general

Berikut adalah mekanisme *log-on* dalam sistem yang mengimplementasi *single sign-on*:

1. User melakukan *request* ke aplikasi.
2. Request ke aplikasi dipotong (*intercept*) oleh aplikasi *single sign-on*.
3. Aplikasi *single sign-on* memberikan *credential* user kepada aplikasi.
4. Akses ke aplikasi diberikan.

Secara general, mekanisme *log-on* pada *single sign-on* adalah seperti di atas. Namun, bagaimana *credential* disimpan dan digunakan untuk mengakses aplikasi (autorisasi), akan berbeda-beda untuk setiap implementasi *single sign-on*. Studi tentang hal ini akan dibahas pada subbab selanjutnya.

*Single sign-on* dalam sistem *standalone*, *credential* user dapat disimpan pada satu *workstation* lokal. Ini dapat berlaku karena tidak seharusnya user dapat mengakses *credential* user lain. Maka dari itu, tidak apa-apa apabila *single sign-on* ditanam pada workstation masing-masing user. Masalahnya, dalam sistem yang terhubung (*network environment*), sistem ini sangat terbatas karena *single sign-on* hanya terdapat pada lokal. *Single sign-on* yang hanya terdapat pada lokal juga tidak dapat berjalan apabila user memiliki lebih dari satu *workstation*.



Gambar .. *Single sign-on* pada sistem jaringan yang terhubung

Tahapan *single sign-on* dalam jaringan yang terhubung sebagai berikut.

1. User melakukan *request* ke aplikasi.
2. Request ke aplikasi dipotong (*intercept*) oleh aplikasi *single sign-on*.
3. Aplikasi *single sign-on* mengambil *credential* user dari server pusat.
4. Aplikasi *single sign-on* memberikan *credential* user kepada aplikasi.
5. Akses ke aplikasi diberikan.

Apabila pada suatu saat user yang sudah terautentikasi dengan *single sign-on* akan *login* kembali, maka berikut tahapannya:

1. User melakukan *request* ke aplikasi.
2. Request ke aplikasi dipotong (*intercept*) oleh aplikasi *single sign-on*.
3. Aplikasi *single sign-on* menghubungi server pusat, apakah user yang bersangkutan masih memiliki *session* yang aktif atau tidak.
4. Aplikasi *single sign-on* memberi tahu ke aplikasi bahwa user yang bersangkutan sudah memiliki *credential*.
5. Akses ke aplikasi diberikan.

Dalam kasus aplikasi yang terhubung pada jaringan, *credential* dari banyak user harus dapat diakses dari banyak client. Satu user bisa saja akan *login* dari beberapa komputer. Oleh karena itu, diperlukan server pusat atau sistem *single sign-on* berbasis direktori yang dapat diakses lewat jaringan supaya *credential* dapat diakses secara bersamaan.

Apabila server *single sign-on* hanya satu, server berikut dituntut memiliki *availibility* yang tinggi. Oleh karena itu, banyak perusahaan membuat server *single sign-on* secara terdistribusi di lebih dari satu server.

Dalam organisasi yang besar dengan jumlah user yang banyak, koneksi ke *remote* server tidak dijamin. Ide yang bagus apabila perusahaan menggabungkan metode penyimpanan *credential* di lokal dan pada server pusat. Ini dapat meningkatkan *availability* apabila suatu saat server *down*. Apabila penyimpanan gabungan ini diterapkan, maka perlu dipikirkan mekanisme untuk sinkronisasi data antara lokal dengan server *remote*.

## Penggunaan *Single Sign-On*

Penggunaan *single sign-on* dapat dibagi menjadi dua, *social single sign-on* dan internal *single sign-on*. Secara singkat, *social single sign-on* dapat digunakan secara bebas oleh publik. Sementara, internal *single sign-on* hanya meliputi sistem pada domain yang tertutup. Aplikasi yang dapat diakses internal *single sign-on* pun hanya terbatas pada aplikasi yang dikembangkan oleh organisasi yang bersangkutan.

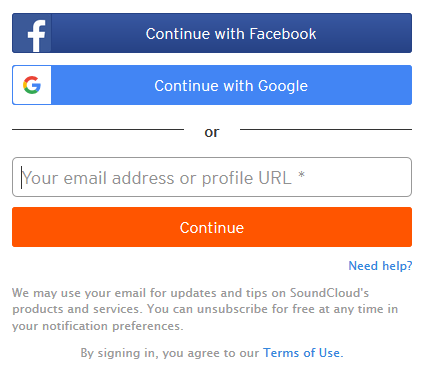
### Social Single Sign-On

D:\My Documents\Informatika ITB Sem. 7\Tugas Akhir I & Seminar\picts\social sso.png

Gambar .. Tombol *log-in* yang disediakan oleh *service provider* untuk melakukan *single sign-on* menggunakan *social single sign-on*

*Public single sign-on* adalah *single sign-on* yang tersedia untuk aplikasi publik dan dapat digunakan secara bebas. Beberapa *social single sign-on* adalah berikut.

1. Google. *Single sign-on* pada Google akan memberikan akses ke YouTube, Gmail, Google Analytics, Maps, Photos, dan sebagainya. Banyak aplikasi lain (tidak dikembangkan Google) yang menyediakan login via Google.
2. Facebook Connect. Banyak aplikasi yang memberikan izin untuk masuk ke servis mereka menggunakan *login with* Facebook, seperti Spotify, Soundcloud, Traveloka, BukaLapak.
3. Google+.
4. Auth0. Sebuah aplikasi bisa menggunakan Auth0 sebagai *identity provider*. Ketika aplikasi yang bertindak sebagai *service provider* membutuhkan *credential*, maka aplikasi akan meminta *request* ke Auth0, kemudian aplikasi akan menerima *credential* dari Auth0.



Gambar .. Spotify menyediakan pilihan untuk *login* menggunakan akun Facebook dan Google.

### Internal *Single Sign-On*

Internal *single sign-on* adalah aplikasi *single sign-on* yang diimplementasi pada internal sistem, akibatnya hanya *service provider* internal perusahaan yang dapat menggunakan kelebihan *single sign-on* ini. *Single sign-on* jenis internal tidak dapat diintegrasikan secara bebas seperti *social single sign-on* yang sudah dibahas sebelumnya.

Beberapa contoh internal *single sign-on*:

1. *Single sign-on* OL Akademik ITB. Ketika user berhasil *login* satu kali saja di alamat ol.akademik.itb.ac.id, selanjutnya beberapa aplikasi internal OL Akademik akan dapat diakses, seperti Pendaftaran & Perwalian Mahasiswa, Status dan Hasil Studi Mahasiswa, Penjurusan, dan Kuesioner.
2. *Single sign-on* pada judgels *online judge*. Setelah user *login* pada *single sign-on,* user akan dapat mengakses beberapa aplikasi lain seperti *Training Gate* dan *Competition Gate*.

## Keamanan *Single Sign-On*

Keamanan jaringan antara server dengan servis jelaslah penting. Salah satu fokus utama dalam pemilihan implementasi *single sign-on* adalah keamanan. Semakin kompleks jaringan komputer dalam suatu sistem, maka ketergantungan terhadap keamanan yang lebih baik akan semakin tinggi.

Kontrol akses merupakan komponen utama dalam sistem keamanan IT apapun (Novell, 2011). Kontrol akses harus mampu memberikan teknologi yang mengamankan informasi sensitif sistem dari serangan luar.

Sistem autentikasi sebagian besar masih tidak berubah. Mekanisme login standar menggunakan *username* dan *password* untuk memastikan user adalah orang yang benar masih digunakan sampai sekarang.

### Mengurangi Jumlah *Password*

Apabila ada 6 aplikasi yang saling terhubung, manakah yang lebih bagus, membangun 6 metode autentikasi berbeda atau membangun satu layer bersama untuk autentikasi yang digunakan untuk 6 aplikasi berbeda? Untuk menjawab pertanyaan ini, kita harus benar-benar mempertimbangkan apakah dengan 6 metode autentikasi yang berbeda, apakah ini benar-benar meningkatkan keamanan sistem atau hanya mempersulit *workflow*. Apakah manusia benar-benar mampu mengingat 6 *password* berbeda secara bersamaan atau user suatu saat akan menulis password pada suatu tempat supaya tidak lupa?

Penelitian oleh Novell pada 2011 membuktikan bahwa user tidak mampu mengingat lebih dari tiga *password* secara bersamaan. Ini membuktikan bahwa semakin sedikit *password* yang harus diingat user, maka semakin bagus. Semakin sedikit *password* yang diingat user, maka user akan cenderung membuat password yang kuat. Semakin banyak *password* yang perlu diingat user, maka user akan cenderung membuat *password* yang lemah supaya mudah diingat seperti 1111, abcd.

### *Security* *Policies*

Tingkat kebutuhan keamanan akan berbeda-beda untuk setiap aplikasi yang terhubung dengan *single sign-on* tergantung dengan kebutuhan bisnis masing-masing aplikasi. Maka dari itu, perlu dipertimbangkan fleksibilitas *single sign-on* dalam mengatasi perbedaan kebutuhan level keamanan pada tiap aplikasi.

### Penyimpanan *Credential*

Karena banyaknya kelemahan yang ditemukan dalam banyak sistem operasi, satu-satunya solusi untuk penyimpanan *credential* adalah dengan enkripsi. Enkripsi mampu melindungi informasi sensitif supaya tidak dapat diakses oleh sembarang orang.

Enkripsi adalah proses untuk mengubah data secara matematis menggunakan algoritma yang sudah ada dan nilai kunci. Ada banyak algoritma sudah ada yang dapat digunakan. Pilihan terbaik untuk melindungi data tergantung pada tipe enkripsi yang perlu dilakukan. Enkripsi bisa saja simetrik (kunci yang sama untuk enkripsi/dekripsi) atau asimetrik (menggunakan mekanisme kunci publik dan privat). Untuk melindungi data sensitif, algoritma enkripsi yang digunakan harus memenuhi standar industri yang telah disepakati, seperti misalnya Triple DES (*Data Encryption Standard*) dan AES (*Advanced Encryption Standard*).

### Mengirimkan *Credential* lewat Jaringan

Mengamankan penyimpanan *credential* adalah hal penting, tetapi proses pengiriman juga tidak kalah pentingnya. Komunikasi antara *service provider* (aplikasi) dengan server pusat penyedia credential dilakukan melalui jaringan. Penyampaian pesan berupa data ini tidak sepenuhnya aman dari serangan berupa *wiretapping* atau *traffic monitoring*. Untuk meyakinkan keamanan *credential* user selama pengiriman data, mekanisme *single sign-on* harus memastikan data yang dikirimkan terenkripsi. Ini bisa dicapai dengan menggunakan protokol HTTPS untuk transmisi data.

## 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

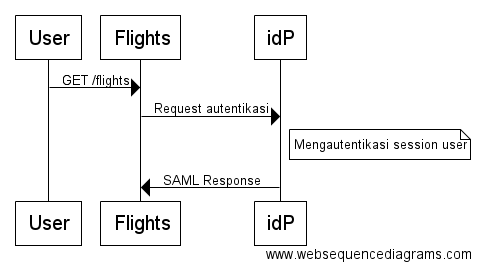
SAML (*Security Assertion Markup Language*) adalah protokol standar open-source untuk *web browser* *single sign-on* menggunakan token. SAML menghilangkan seluruh *password* dalam proses autentikasi. SAML menggunakan *digital signature* untuk mengirim token keamanan yang diberikan oleh *identity provider*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

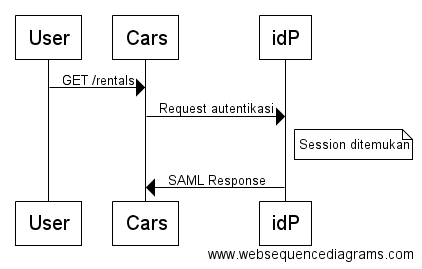


Gambar .. Sequence diagram untuk SAML

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 .. 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 .. 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:

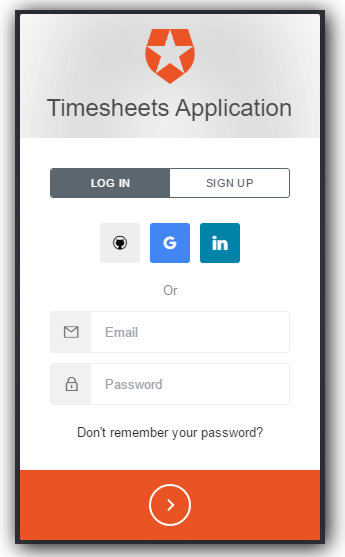
1. Credential standar: *username* dan *password*.
2. *Login* media sosial: Google, Facebook, Twitter, OAuth2, OAuth1, dan OpenID *identity provider*.
3. *Enterprise directory*: LDAP, Google APPs, Office 365, ADFS, AD, SAML-P, WS-Federation, etc.
4. Sistem tanpa *password*: TouchID, OTP, SMS, email.

#### 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.

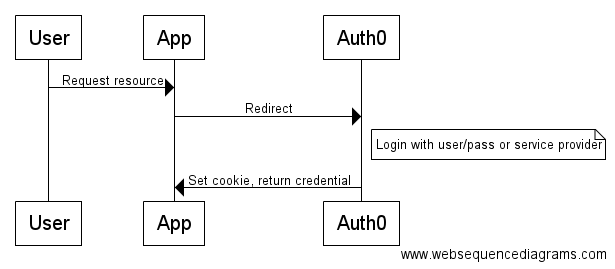
Berikut adalah langkah-langkah ketika user mengunjungi aplikasi yang terlindungi Auth0 untuk pertama kalinya:

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 .. Tampilan halaman Auth0 Hosted Lock

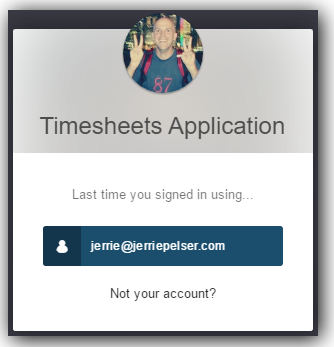
1. User akan login, Auth0 akan menyimpan *single sign-on cookie*.
2. Auth0 akan mengembalikan user ke aplikasi *service provider* dan mengembalikan id\_token yang berisi identitas user.



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

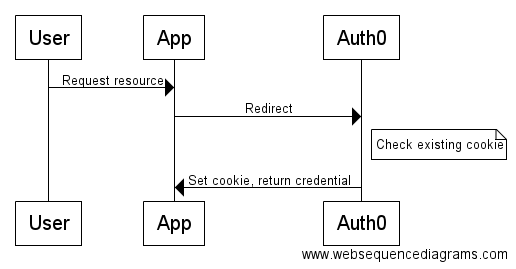
Sekarang bagaimana jika user kembali ke web untuk kunjungan berikutnya? Hal ini yang akan terjadi.

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 .. 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 .. Diagram *sequence* Auth0 untuk kunjungan berikutnya.

### OpenID Connect

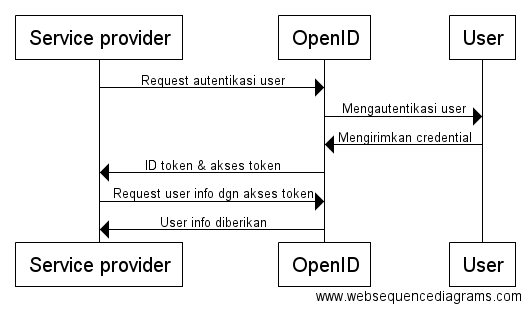
OpenID Connect adalah *framework* untuk identitas yang dibangun di atas protokol OAuth 2.0 yang mengembangkan *workflow* autorisasi OAuth 2.0 dengan menambahkan mekanisme autentikasi. Seperti yang kita ketahui, bahwa OAuth merupakan protokol yang berfungsi untuk mengatur autorisasi antar *service* *provider*, 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 .. Diagram *sequence* OpenID Connect.

# Analisis permasalahan dan rancangan solusi

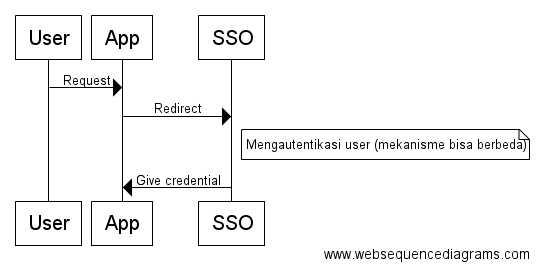
Pada bab ini akan dipaparkan analisis permasalahan yang ada dalam studi dan implementasi *single sign-on*. Selain itu, akan dipaparkan juga rumusan solusi yang akan diajukan dalam bentuk langkah-langkah pekerjaan yang akan dilakukan.

## 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 .. 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 ..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.

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.

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.

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

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.

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.

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

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

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

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.

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

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.