

**PERANCANGAN DAN IMPLEMENTASI PROSES
AUTENTIKASI PADA *RESTFUL WEB SERVICE*
MENGUNAKAN *SEED BASED AUTHENTICATION***

TESIS

**Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Magister dari
Institut Teknologi Bandung**

**Oleh
Auliak Amri
NIM: 23215077
(Program Studi Magister Teknik Elektro)**



**INSTITUT TEKNOLOGI BANDUNG
Juni 2017**

**PERANCANGAN DAN IMPLEMENTASI PROSES
AUTENTIKASI PADA *RESTFUL WEB SERVICE*
MENGUNAKAN *SEED BASED AUTHENTICATION***

Oleh
Auliak Amri
NIM: 23215077
(Program Studi Magister Teknik Elektro)

Institut Teknologi Bandung

Menyetujui
Pembimbing

Tanggal 21 Juni 2017

(Ir. Budi Rahardjo, M.Sc., Ph.D.)

ABSTRAK

PERANCANGAN DAN IMPLEMENTASI PROSES AUTENTIKASI PADA *RESTFUL WEB SERVICE* MENGUNAKAN *SEED BASED AUTHENTICATION*

Oleh

Auliak Amri

NIM: 23215077

(Program Studi Magister Teknik Elektro)

Web service memungkinkan dilakukannya komunikasi *machine-to-machine* melalui protokol standar web untuk penggunaan *resource* yang disediakan oleh *service provider*. Metode yang paling banyak digunakan pada *web service* adalah REST karena keunggulannya yang lebih cepat, lebih sederhana, dan kemampuannya dalam menangani berbagai macam model aplikasi IoT. Namun demikian, *RESTful web service* memiliki kelemahan pada konsep *stateless*, yaitu server tidak dapat menyimpan status client sehingga tidak dapat menggunakan *session* yang mengakibatkan perlu dilakukannya autentikasi setiap kali client meminta *resource* ke server. Hal tersebut dapat mengakibatkan *overload* pada CPU dan penggunaan sumber daya jaringan yang tinggi. Selain itu, *RESTful web service* memiliki kelemahan terhadap serangan keamanan yang disebabkan oleh penggunaan username dan password statis. Proses autentikasi pada *RESTful web service* dirancang dengan menggunakan *Seed Based Authentication*. Pada metode ini, *username* dan URI dibangkitkan secara dinamis setiap dilakukan autentikasi. *Seed* digunakan sebagai input *Pseudorandom Number Generator* dan selanjutnya diproses secara paralel di client dan server. Perancangan proses autentikasi dilakukan dengan menggabungkan *Token Based Authentication* karena kelebihanannya dalam menjaga konsep *stateless* dan *Seed Based authentication* yang dapat menangani masalah keamanan yang disebabkan oleh penggunaan username dan password statis. Implementasi dilakukan dengan menyusun prototipe yang terbagi menjadi dua bagian, yaitu *Authentication Server* dan *Resource Server*. Hasil pengujian dan evaluasi menunjukkan bahwa fungsi dalam prototipe berjalan dengan baik dan sesuai dengan spesifikasi dalam perancangan. Waktu eksekusi tahap inisialisasi, identifikasi, dan autentikasi relatif sama. Tahap sinkronisasi memiliki waktu eksekusi yang lebih lama dibandingkan dengan ketiga tahapan yang lain. Pada setiap tahap, fungsi pada skrip PHP yang memiliki distribusi waktu eksekusi terbesar adalah fungsi yang berhubungan dengan database, seperti update dan pengambilan data client. Evaluasi keamanan menunjukkan bahwa metode ini lebih baik dibandingkan metode autentikasi lainnya seperti *HTTP Basic Authentication*, *HTTP Digest Authentication*, dan *Token Based Authentication*. Namun demikian, metode ini memakan waktu yang lebih lama dalam proses autentikasi jika dibandingkan dengan *Token Based Authentication*.

Kata kunci: REST, *web service*, *internet of thing*, *seed based authentication*

ABSTRACT

DESIGN AND IMPLEMENTATION OF AUTHENTICATION PROCESS FOR RESTFUL WEB SERVICE USING SEED BASED AUTHENTICATION

By

Auliak Amri

NIM: 23215077

(Master of Electrical Engineering)

Web service allows machine-to-machine communication via standard web protocols for the client to get resources provided by the service provider. The most widely used method of web service is REST because it is faster, simpler, and has the ability to handle a wide variety of application models of IoT. However, RESTful web service has some limitations in the stateless concept of the REST. Server doesn't save the status of the client, so it can not use a session. Without using a session, server have to authenticate whenever the client requests resource to the server. Furthermore, it may cause CPU overload and waste abundant amount of network resources. In addition, some of the methods used in the authentication process still have a weakness against security attacks caused by static username and password used in the authentication process. The authentication process in RESTful web service is designed using Seed Based Authentication. In this method, the username and URI are dynamically generated each time authentication is performed. A Seed is used as input of Pseudorandom Number Generator and then processed in parallel on the client and server. The design of the authentication process for RESTful Web Services combines the Token Based Authentication method because of its advantages in keeping the concept of statelessness and Seed Based Authentication to deal with the weakness of the authentication process with static username and password. Implementation is conducted by creating a prototype that is divided into two parts, Authentication Server and Resource Server. Test and evaluation results show that the functions in the prototype run well and fit with the specifications in the design. The execution time of initialization, identification, and authentication phase is relatively the same. The synchronization phase has a longer execution time compared to the other three stages. At each stage, PHP scripts that have the largest time execution are database related functions, such as update and retrieval of client data. Security evaluation shows that this method is better than other authentication methods, such as HTTP Basic Authentication, HTTP Digest Authentication, and Token Based Authentication. However, this method takes a longer time in the authentication process when compared with Token Based Authentication.

Keywords: REST, web service, internet of thing, seed based authentication

PEDOMAN PENGGUNAAN TESIS

Tesis S2 yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin pengarang dan harus disertai dengan kaidah ilmiah untuk menyebutkan sumbernya.

Sitasi hasil penelitian Tesis ini dapat ditulis dalam bahasa Indonesia sebagai berikut:

Amri, A. (2017): *Perancangan dan implementasi proses autentikasi pada RESTful web service menggunakan Seed Based Authentication*, Tesis Program Magister, Institut Teknologi Bandung.

dan dalam bahasa Inggris sebagai berikut:

Amri, A. (2017): *Design and implementation of authentication process for RESTful web service using Seed Based Authentication*, Master's Program Thesis, Institut Teknologi Bandung.

Memperbanyak atau menerbitkan sebagian atau seluruh tesis haruslah seizin Dekan Sekolah Pascasarjana, Institut Teknologi Bandung.

*Dipersembahkan kepada kedua orang tua tercinta,
Mudjiono dan Sri Umi Rosdayanti*

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT karena berkat rahmat dan hidayah-Nya karya tulis ini dapat terselesaikan dengan baik. Tesis yang berjudul “Perancangan dan implementasi proses autentikasi pada *RESTful web service* menggunakan *Seed Based Authentication*” ini disusun sebagai syarat kelulusan Program Studi Magister Teknik Elektro jalur pilihan Rekayasa dan Manajemen Keamanan Informasi (RMKI).

Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih kepada pihak-pihak yang telah membantu dalam penyelesaian tesis ini, yaitu:

1. Ir. Budi Rahardjo, M.Sc., Ph.D. selaku dosen pembimbing penulis yang telah memberikan nasihat dan bimbingan sehingga tesis ini dapat terselesaikan dengan baik,
2. Dr. Ir. Aciek Ida Wuryandari, MT., Dra. Harlili, M.Sc., dan Dr. M. Ari Anggorowati selaku dosen penguji yang telah memberikan masukan dan saran untuk penyempurnaan tesis ini,
3. Ir. Yudi Satria Gondokaryono, M.Sc., Ph.D. selaku dosen wali RMKI angkatan 2015,
4. Dr.Ing.-Ir. Suhardi, MT. selaku koordinator Program Magister Teknik Elektro jalur pilihan RMKI dan LTI Kerma BPS,
5. H. Mudjiono dan Hj. Sri Umi Rosdayanti selaku kedua orang tua penulis yang selalu memberikan semangat dan doa untuk penulis,
6. Badan Pusat Statistik yang telah memberi kesempatan kepada penulis untuk melanjutkan studi di Institut Teknologi Bandung,
7. Teman seperjuangan tugas belajar BPS ITB angkatan IV dan mahasiswa/i RMKI angkatan 2015,
8. Semua pihak yang telah membantu dalam kelancaran studi dan penyelesaian tesis ini, baik dari BPS maupun STEI ITB.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam penyusunan tesis ini. Oleh karena itu, kritik dan saran penulis harapkan untuk perbaikan dan penyempurnaan penelitian ini. Semoga karya tulis ini dapat bermanfaat dalam pengembangan ilmu pengetahuan, khususnya di bidang keamanan informasi.

Bandung, Juni 2017

Auliak Amri

DAFTAR ISI

ABSTRAK	i
<i>ABSTRACT</i>	ii
PEDOMAN PENGGUNAAN TESIS.....	iii
KATA PENGANTAR	v
DAFTAR ISI.....	vii
DAFTAR LAMPIRAN.....	ix
DAFTAR GAMBAR DAN ILUSTRASI.....	x
DAFTAR TABEL.....	xii
DAFTAR SINGKATAN DAN LAMBANG.....	xiii
Bab I Pendahuluan	1
I.1 Latar Belakang	1
I.2 Rumusan Masalah	4
I.3 Tujuan Penelitian	4
I.4 Batasan Masalah.....	4
I.5 Sistematika Penulisan	5
Bab II Tinjauan Pustaka	7
II.1 Web Service	7
II.2 RESTful Web Service	8
II.3 Keamanan Web Service	9
II.4 Metode Autentikasi	12
II.5 Macam-Macam Metode Autentikasi.....	13
II.6 Autentikasi pada Web Service	14
II.7 Two-legged authorization dan three-legged authorization	15
II.8 Psudeorandom Number.....	17
II.9 Seed Based Authentication	18
II.10 ID-based Encryption (IBE)	22
II.11 ID-based Authentication pada RESTful Web Service	24
II.12 Unified Modeling Language (UML).....	26
II.13 Yii Framework	32
II.14 Xdebug	33
II.15 Literature Map.....	34
Bab III Metodologi	37
Bab IV Perancangan	39
IV.1 Perancangan Proses Autentikasi	40
IV.2 Perancangan Database.....	64
IV.3 Perancangan Antarmuka	65
Bab V Implementasi dan Pengujian	67
V.1 Implementasi	67
V.2 Pengujian Prototipe	76

Bab VI	Kesimpulan dan Saran	117
VI.1	Kesimpulan.....	117
VI.2	Saran	118
DAFTAR PUSTAKA.....		119
LAMPIRAN		121

DAFTAR LAMPIRAN

Lampiran A	Source Code AuthControllerProvider.php Aplikasi Authenticater pada Authenticater Server	122
Lampiran B	Source Code AuthServiceProvider.php Aplikasi Authenticater pada Authenticater Server.....	126
Lampiran C	Source Code CClient.php Aplikasi Authenticater pada Authenticater Server	133
Lampiran D	Source Code Client.php Aplikasi Authenticater pada Authenticater Server	137
Lampiran E	Source Code AuthControllerProvider.php pada Resource Server .	141
Lampiran F	Source Code AuthServiceProvider.php pada Resource Server.....	143
Lampiran G	Source Code CClient.php pada Resource Server.....	145
Lampiran H	Source Code Client.php pada Resource Server.....	147
Lampiran I	SQL database authdb pada Authentication Server.....	150
Lampiran J	Antarmuka Aplikasi Administrator.....	152

DAFTAR GAMBAR DAN ILUSTRASI

Gambar II.1. Proses pertukaran pesan dalam web service (W3C, 2004)	8
Gambar II.2. End-to-end security pada web service (W3C, 2004)	10
Gambar II.3. Proses autentikasi menggunakan HTTP Basic Authentication.....	14
Gambar II.4. Proses autentikasi menggunakan Token Based Authentication.....	15
Gambar II.5. Sequence diagram untuk two-legged authorization	16
Gambar II.6. Sequence diagram untuk three-legged authorization	17
Gambar II.7. Proses pada ID-based Encryption (Lee dkk., 2015).....	23
Gambar II.8. Alur proses autentikasi pada ID-based Authentication.....	25
Gambar II.9. Contoh notasi class pada class diagram	29
Gambar II.10. Struktur aplikasi pada Yii Framework	32
Gambar II.11. Literature Map penelitian	35
Gambar III.1. Tahapan pada Design Research methodology (DRM)	37
Gambar IV.1. Seed Based Authentication pada RESTful web service	39
Gambar IV.2. Diagram pemodelan UML.....	40
Gambar IV.3. Use Case Diagram	42
Gambar IV.4. Activity Diagram pada tahap inisialisasi	44
Gambar IV.5. Activity Diagram pada tahap identifikasi.....	46
Gambar IV.6. Activity Diagram pada tahap Autentikasi	48
Gambar IV.7. Activity Diagram pada tahap sinkronisasi.....	49
Gambar IV.8. Class diagram aplikasi administrator pada Authenticater Server...	51
Gambar IV.9. Class Diagram aplikasi authenticater pada Authenticater Server ..	54
Gambar IV.10. Class Diagram pada Resource Server	56
Gambar IV.11. Sequence Diagram pada tahap inisialisasi.....	58
Gambar IV.12. Sequence diagram pada tahap identifikasi dan autentikasi	61
Gambar IV.13. Sequence Diagram pada tahap sinkronisasi	62
Gambar IV.14. Entity Relationship Diagram (ERD) database client.....	64
Gambar IV.15. Form registrasi developer	65
Gambar IV.16. Form login	65
Gambar IV.17. Form untuk pendaftaran aplikasi	66
Gambar IV.18. Form untuk mengelola aplikasi	66
Gambar V.1. Struktur direktori aplikasi administrator.....	72
Gambar V.2. Antarmuka utama aplikasi administrator	72
Gambar V.3. Struktur database authdb.....	73
Gambar V.4. Struktur database resource, resourcedb	75
Gambar V.5. Gambar arsitektur pengujian.....	76
Gambar V.6. Gambar arsitektur jaringan pada virtual machine untuk pengujian.	77
Gambar V.7. Gambar ilustrasi black box testing.....	78
Gambar V.8. Skenario uji kinerja tahap inisialisasi	95
Gambar V.9. Skenario uji kinerja tahap identifikasi	95
Gambar V.10. Skenario uji kinerja tahap autentikasi	96
Gambar V.11. Skenario uji kinerja tahap sinkronisasi	96
Gambar V.12. Grafik total waktu menurut jumlah iterasi	98
Gambar V.13. Grafik rata-rata waktu menurut jumlah iterasi.....	99
Gambar V.14. Call Graph dari output xdebug pada proses inisialisasi.....	108
Gambar V.15. Call Graph dari output xdebug pada proses identifikasi.....	109

Gambar V.16. Call Graph dari output xdebug pada proses autentikasi	109
Gambar V.17. Call Graph dari output xdebug pada proses sinkronisasi untuk request pertama	110
Gambar V.18. Call Graph dari output xdebug pada proses sinkronisasi untuk request kedua.....	111
Gambar V.19. Grafik perbandingan total waktu eksekusi Token Based Authentication dan Seed Based Authentication	116

DAFTAR TABEL

Tabel II.1. Notasi pada Use Case Diagram	27
Tabel II.2. Notasi pada Activity Diagram	28
Tabel II.3. Notasi visibilitas pada Class Diagram	30
Tabel II.4. Notasi pada Sequence Diagram	31
Tabel V.1. Tabel spesifikasi lingkungan pengujian	77
Tabel V.2. Tabel token yang dihasilkan dari berbagai macam format root file	80
Tabel V.3. Hasil pengujian pada sistem admin	81
Tabel V.4. Hasil pengujian proses autentikasi Client dengan Authentication Server.....	88
Tabel V.5. Tabel total waktu hasil pengukuran pada uji kinerja.....	97
Tabel V.6. Tabel rata-rata waktu hasil pengukuran pada uji kinerja.....	98
Tabel V.7. Tabel output analisis regresi data uji kinerja pada tahap inisialisasi ...	99
Tabel V.8. Tabel output analisis regresi data uji kinerja pada tahap identifikasi	101
Tabel V.9. Tabel output analisis regresi data uji kinerja pada tahap autentikasi	103
Tabel V.10. Tabel output analisis regresi data uji kinerja pada tahap sinkronisasi	105
Tabel V.11. Tabel total dan rata-rata waktu eksekusi Token Based Authentication dan Seed Based Authenticon.....	115

DAFTAR SINGKATAN DAN LAMBANG

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
SOA	<i>Service Oriented Architecture</i>	1
REST	<i>Representational State Transfer</i>	1
SOAP	<i>Simple Object Access Protocol</i>	1
IoT	<i>Internet of Things</i>	1
URI	<i>Uniform Resource Identifiers</i>	1
HTTP	<i>Hypertext Transfer Protocol</i>	2
CPU	<i>Central Processing Unit</i>	2
ID	<i>Identifier</i>	2
OAuth	<i>Open Authorization</i>	2
URL	<i>Uniform Resource Locator</i>	4
MS-CHAP	<i>Microsoft Challenge Handshake Authentication Protocol</i>	4
RS	<i>Resource Server</i>	4
AS	<i>Authentication Server</i>	4
IBE	<i>ID-based Encryption</i>	22
IBA	<i>ID-based Authentication</i>	24
LAMBANG		
n	<i>Sequence number</i>	19
P	Fungsi yang mengimplementasikan <i>Pseudorandom Number Generator</i>	19
$S_u(n)$	Karakter sepanjang 16 byte yang diekstrak dari root file	19
S_{url}^C	URLSeed <i>client</i>	19
S_{url}^S	URLSeed <i>server</i>	19
$S_m(n)$	Karakter sepanjang 16 byte yang diekstrak dari <i>root file</i> , namun berbeda dengan $S_u(n)$	19
S_{unm}^C	UNMSeed <i>client</i>	19
S_{unm}^S	UNMSeed <i>server</i>	19
$T_{url}(n)$	Token yang dibangkitkan dari URLSeed	19
$T_{url}^C(n)$	URLToken <i>client</i>	19
$T_{url}^S(n)$	URLToken <i>server</i>	19
$T_{unm}(n)$	Token yang dibangkitkan dari UNMSeed	19
$T_{unm}^C(n)$	UNMToken <i>client</i>	19
$T_{unm}^S(n)$	UNMToken <i>server</i>	19
S_{url}^{AS}	URLSeed <i>Authentication Server</i>	44
S_{unm}^{AS}	UNMSeed <i>Authentication Server</i>	44
$T_{url}^{AS}(n)$	URLToken <i>Authenticatoin Server</i>	44
$T_{unm}^{AS}(n)$	UNMToken <i>Authentication Server</i>	44

Halaman ini sengaja dikosongkan

Bab I Pendahuluan

I.1 Latar Belakang

Setiap sistem memiliki *resource* seperti gambar, video, informasi, atau apapun yang dapat direpresentasikan dalam sistem komputer. *Service* dibuat untuk menyediakan antarmuka bagi client agar dapat menggunakan *resource* tersebut. *Web service* memungkinkan dilakukannya komunikasi *machine-to-machine* pada jaringan melalui protokol standar web (W3C, 2004) antara client (*service requester*) dan server (*service provider*). *Service Oriented Architecture* (SOA) menjadi konsep yang digunakan pada *web service*, yaitu untuk menggunakan kembali (*reuse*) dan mengintegrasikan sistem.

RESTful web service merupakan *web service* yang menggunakan metode REST pada arsitekturnya. Saat ini REST adalah metode yang banyak digunakan, menggantikan metode yang lebih lama, yaitu SOAP. Keterbatasan terdapat pada metode SOAP, generasi pertama *web service*, saat menangani berbagai macam perangkat (Aihkisalo dan Paaso, 2012). Dengan demikian, banyak perusahaan penyedia *web service* beralih ke metode REST, seperti Google misalnya, mulai menggunakan metode ini sejak 5 Desember 2006, dan juga Amazon sebagian besar, sekitar 85%, antarmukanya menggunakan REST di tahun 2003 (Lee dkk., 2015).

Web service dengan metode REST lebih cepat dibandingkan dengan SOAP dalam melakukan proses *request-response*. Perbedaannya bisa mencapai 5-10 kali lebih cepat. Faktor yang paling mempengaruhi adalah ukuran pesan pada REST yang lebih kecil dibandingkan dengan SOAP (Aihkisalo dan Paaso, 2012).

Web service juga menjadi konsep yang digunakan pada *Internet of Things* (IoT). Kelebihan dalam menggunakan *RESTful web service* pada IoT yaitu kemampuannya yang dapat menangani berbagai macam model aplikasi IoT (Alghamdi dkk., 2013). Dengan metode ini, IoT dapat diidentifikasi oleh *Uniform Resource Identifiers* (URI) yang unik. *Resource* dan koneksi dapat dikelola dengan baik oleh URI yang dapat membantu IoT dalam melakukan *self-configuration*

(Feng dkk., 2009). Selain itu, metode ini menggunakan metode HTTP dasar yaitu *get*, *put*, *post*, dan *delete*, serta dua tambahan lainnya yaitu *head* dan *option* (Fielding, 2000).

Namun demikian, *RESTful web service* masih memiliki beberapa kelemahan, yaitu pada konsep *stateless*, dimana server tidak dapat menyimpan status dari client sehingga tidak dapat menggunakan *session* pada koneksi yang dilakukan antara client dan server (Lee dkk., 2015). Oleh karena itu, masalah lain akan timbul, yaitu pada proses autentikasi. Setiap kali client melakukan permintaan *resource*, maka server akan meminta autentikasi dari client. Proses autentikasi yang dilakukan oleh berbagai macam client dan dilakukan berulang kali dapat mengakibatkan kelebihan beban pada CPU dan penggunaan sumber daya jaringan yang tinggi.

Di samping itu, kelemahan juga terdapat pada proses autentikasi. Metode autentikasi yang saat ini digunakan pada *RESTful web service* diantaranya *HTTP Basic Authentication*, *HTTP Digest Authentication*, dan *Token Based Authentication (OAuth)*. *HTTP Basic Authentication* (Franks dkk., 1999) adalah cara yang paling dasar yaitu menggunakan ID dan *password* dalam proses autentikasi. ID dan *password* tersebut tidak dienkripsi sehingga rentan terhadap serangan *replay attack*, *injection attack*, dan *middleware hijacking* (Jo dkk., 2014). Metode yang lebih lanjut adalah *HTTP Digest Authentication* (Franks dkk., 1999). Meskipun menggunakan ID dan *password* yang terenkripsi, metode ini masih memiliki kelemahan terhadap serangan *Man-in-the-Middle*. Metode lainnya adalah *Token Based Authentication (OAuth)* (Peng dkk., 2009). Metode ini menggunakan token yang dibangkitkan secara dinamis dalam proses autentikasi.

Permasalahan konsep *stateless* pada *RESTful web service* terjadi ketika autentikasi dilakukan dengan menggunakan *HTTP Basic Authentication* dan *HTTP Digest Authentication*. Dalam proses permintaan *resource* oleh client, metode ini memerlukan *session* sehingga *RESTful web service* tidak lagi memiliki prinsip *stateless*. Apabila *session* tidak digunakan, maka proses autentikasi akan dilakukan setiap kali client meminta *resource*. Oleh karena itu, *Token Based Authentication*

(*OAuth*) lebih banyak digunakan karena kemampuannya dalam menjaga prinsip *stateless* pada REST (Lee dkk., 2015).

Metode autentikasi telah banyak dikembangkan untuk proses identifikasi dan verifikasi pengguna dalam sistem komputer. Metode-metode tersebut diantaranya autentikasi dengan *one-time password*, *challenge response* (Vapen dkk., 2010), *biometric*, yang menggunakan karakteristik tubuh pengguna seperti sidik jari (Kumar dkk., 2008), telapak tangan, telapak kaki (Yun dkk., 2007), retina, dan sebagainya, dan *location based authentication* (Zhang dkk., 2012).

Metode autentikasi dengan menggunakan *username* dan *password* adalah cara yang paling mudah. Namun demikian, metode ini memerlukan kemampuan pengguna untuk dapat menyimpan *password* secara aman dari pihak lain yang dapat mengeksploitasinya (Inglesant dan Sasse, 2010). Selain itu, pengguna perlu mengganti *password* tersebut secara rutin. Berbagai macam cara telah diusulkan untuk perbaikan metode autentikasi dengan membuat *password* yang lebih dinamis. Sebagai contoh, autentikasi dilakukan dengan menggunakan *password* yang dikombinasikan dengan tanggal hari ini sehingga *password* akan berubah setiap harinya. Hal ini akan memberikan tingkat keamanan yang lebih tinggi dibandingkan dengan yang statis.

Token Based Authentication (OAuth) memanfaatkan token yang dibangkitkan secara dinamis. Namun demikian, token digunakan sebagai faktor kedua dalam proses autentikasi. ID dan *password* masih diperlukan, sebagai faktor pertama, untuk mendapatkan token tersebut. Padahal, autentikasi yang menggunakan *password* masih memiliki kelemahan keamanan terhadap beberapa serangan keamanan seperti *dictionary attack*, *interception*, dan *brute force attack* (Dell'Amico dkk., 2010) (Canvel dkk., 2003) (Joshi dkk., 2009).

Untuk mengatasi masalah ini, *Seed Based Authentication* (Nassar dan Chen, 2015) dapat digunakan dalam proses autentikasi. Metode ini mengganti *username* dan *password*, yang digunakan sebagai faktor pertama, dengan token yang dibangkitkan

secara dinamis dengan menggunakan *Parallel Pseudorandom Numbers Generator*. *Pseudorandom Numbers Generator* dijalankan secara paralel pada client dan server dengan menggunakan seed yang sama (Salmon dkk., 2011). Untuk meningkatkan keamanan, tiga elemen yang digunakan pada proses autentikasi, yaitu URL, *username*, dan *password*, dibuat menjadi rahasia dan dinamis dengan menggunakan *Pseudorandom Numbers* tersebut. Konsep terdekat dengan metode ini adalah autentikasi dengan *random challenge*, *Microsoft Challenge Handshake Authentication* (MS-CHAP), namun metode ini lemah dan telah diretas (Cassola dkk., 2013).

I.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas maka dibentuk rumusan masalah yaitu bagaimana proses autentikasi pada *RESTful web service* dengan menggunakan *Seed Based Authentication*.

I.3 Tujuan Penelitian

Tujuan umum penelitian ini adalah melakukan perancangan dan implementasi proses autentikasi pada *RESTful web service* dengan menggunakan *Seed Based Authentication*. Tujuan khusus penelitian ini adalah sebagai berikut.

1. Melakukan perancangan proses autentikasi pada *RESTful web service* dengan menggunakan *Seed Based Authentication*.
2. Melakukan implementasi hasil perancangan dalam sebuah prototipe.
3. Melakukan pengujian dan evaluasi terhadap prototipe yang telah disusun.

I.4 Batasan Masalah

Penelitian ini dilakukan pada proses autentikasi *web service* yang menggunakan metode REST pada arsitekturnya. Proses autentikasi dilakukan antara client yang berupa aplikasi web dan *service provider* yang terdiri dari *Resource Server* (RS) dan *Authentication Server* (AS). Tipe *authorization* yang digunakan adalah *two-legged authorization*, karena proses autentikasi hanya melibatkan dua pihak, yaitu client dan *service provider*, tanpa melibatkan *end user*.

I.5 Sistematika Penulisan

Sistem penulisan tesis ini terbagi menjadi beberapa bab yaitu:

1. Bab I Pendahuluan

Bab ini menjelaskan bagaimana latar belakang, rumusan masalah, tujuan penelitian, dan batasan masalah dalam melakukan penelitian.

2. Bab II Tinjauan Pustaka

Landasan teori yang digunakan sebagai dasar dalam melakukan penelitian ini akan paparkan dalam bab ini.

3. Bab III Metodologi

Pada bab ini dijelaskan metodologi apa yang digunakan serta tahapan-tahapan dalam melakukan penelitian.

4. Bab IV Perancangan

Bab IV berisi tentang bagaimana perancangan proses autentikasi pada *RESTful web service* menggunakan *Seed Based Authentication*.

5. Bab V Implementasi dan Pengujian

Bab ini memuat implementasi hasil perancangan dalam prototipe dan pengujian terhadap prototipe yang telah disusun.

6. Bab VI Kesimpulan dan Saran

Kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya diberikan pada bab ini.

Halaman ini sengaja dikosongkan

Bab II Tinjauan Pustaka

II.1 Web Service

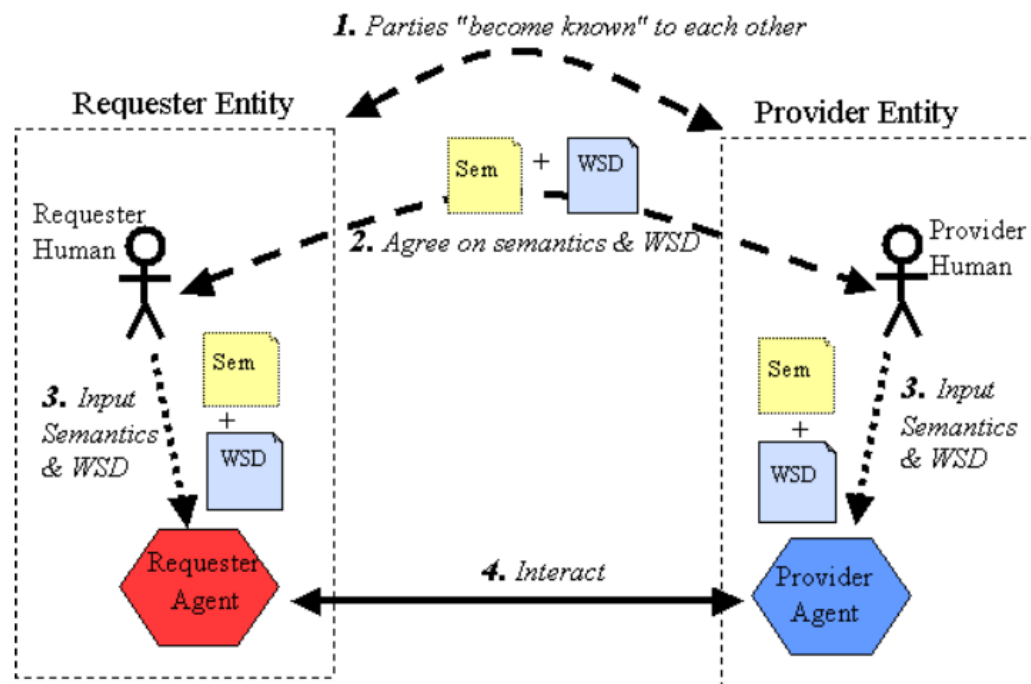
Web service adalah sistem *software* yang dirancang untuk mendukung interaksi mesin-ke-mesin melalui jaringan. *Web service* memiliki antarmuka yang dideskripsikan dengan format yang dapat diproses oleh mesin (misalnya WSDL). Sebuah sistem berinteraksi dengan *web service* menggunakan pesan SOAP dengan cara yang telah ditentukan dalam deskripsi antarmukanya, biasanya interaksi menggunakan HTTP dengan XML, sesuai dengan standar yang berhubungan dengan web (W3C, 2004).

Tujuan dari *web service* adalah untuk menyediakan fungsionalitas dari suatu individu atau organisasi, disebut dengan *provider entity*, yang menyediakan sebuah *agent* untuk mengimplementasikan sebuah *service*. *Web service* yang disediakan oleh *provider entity* nantinya akan digunakan oleh suatu individu atau organisasi, disebut dengan *requester entity*, dengan menggunakan *requester agent* untuk melakukan pertukaran pesan dengan *agent* dari *provider entity*. Banyak dokumentasi yang menggunakan istilah *service provider* yang merujuk pada *provider entity* dan *provider agent*. Begitu pula dengan istilah *service requester* yang merujuk pada *requester entity* dan *requester agent*.

Pertukaran pesan dapat dilakukan antara *requester entity* dan *provider entity* dengan terlebih dahulu masing-masing pihak menyetujui mekanisme dan semantik dari pertukaran pesan. Mekanisme pertukaran pesan didokumentasikan dalam *Web Service Description* (WSD). WSD merupakan spesifikasi antarmuka *web service* yang ditulis dalam WSDL. Spesifikasi *web service* meliputi format pesan, tipe data, dan *transport protocol*.

Gambar II.1 menjelaskan bagaimana *requester entity* dapat menggunakan sebuah *web service*. Hal ini dimulai dengan saling mengenali antara *requester entity* dan *provider entity*. Selanjutnya, masing-masing *entity* menyetujui deskripsi dan semantik dari *service* yang akan menjadi panduan dalam pertukaran pesan antara

requester dan *provider agent*. Setelah itu, deskripsi dan semantik dikenali oleh *requester* dan *provider agent*. Terakhir, *agent* tersebut melakukan pertukaran pesan, melakukan tugas-tugas atas nama *requester* dan *provider entity*.



Gambar II.1. Proses pertukaran pesan dalam *web service* (W3C, 2004)

Dalam sistem *World Wide Web*, *agent* mengidentifikasi objek atau *resource* dengan *Uniform Resource Identifiers* (URI). *Agent* menggambarkan *resource* dengan berbagai macam format data yang dipahami secara luas, seperti XML, HTML, CSS, JPEG, PNG, dan sebagainya.

II.2 *RESTful Web Service*

RESTful web service adalah *web service* yang menggunakan metode REST pada arsitekturnya. REST merupakan *architectural style* pada aplikasi web yang diusulkan oleh Fielding (2000) dan telah banyak digunakan untuk membangun *web service*. REST adalah bagian dari WWW yang memiliki lebih banyak batasan-batasan. Pada REST, *agent* menyediakan semantik antarmuka yang seragam, yaitu *create*, *retrieve*, *update*, dan *delete*. Selain itu, interaksi REST bersifat *stateless* yang berarti pesan tidak bergantung pada status dari client.

Saat ini *web service* yang berbasis REST telah banyak digunakan, meninggalkan metode yang lama yaitu SOAP (Lee dkk., 2015). Web service berbasis REST memiliki karakteristik sebagai berikut (Fielding, 2000).

1. *Addressability*

Semua objek dalam IoT dapat diidentifikasi melalui URI yang unik.

2. *Connectedness*

Karakteristik ini terbentuk karena adanya *addressability* (Feng, 2009). Setiap objek dapat dihubungkan satu dengan lainnya dan dikelola oleh URI.

3. *Homogenous Interface*

Dalam menggunakan *resource*, metode yang digunakan adalah *get*, *put*, *post*, dan *delete*. Ada juga metode tambahan yaitu *head* dan *option*.

4. *Statelessness*

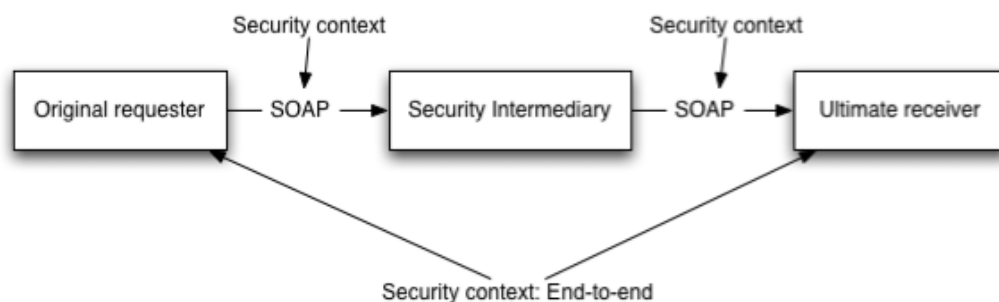
Server tidak menyimpan status dari client, sehingga koneksi antara keduanya tidak dapat menggunakan *session*.

II.3 Keamanan Web Service

Ancaman terhadap *web service* melibatkan sistem *host*, aplikasi, dan infrastruktur jaringan. Implementasi *web service* memerlukan mekanisme *point-to-point* dan *end-to-end security* bergantung pada tingkat ancaman atau risiko. Keamanan merupakan keseimbangan antara risiko dan biaya penanggulangan. Pelaksanaan tindak lanjut ancaman keamanan dilakukan sesuai dengan toleransi dari risiko.

Dari sisi arsitektur, terdapat tiga konsep penting dalam keamanan yaitu *resource* yang harus aman, mekanisme bagaimana *resource* menjadi aman, dan kebijakan. Kebijakan dapat dibagi menjadi dua, yaitu *permission policies* dan *obligatory policies*. *Permission policies* menyangkut pada aksi dan akses yang diizinkan pada entitas, sedangkan *obligatory policies* merupakan aksi yang harus dilakukan oleh entitas. Kedua kebijakan tersebut memiliki mekanisme yang berbeda yaitu *permission guard* yang merupakan mekanisme untuk memverifikasi aksi atau akses yang diminta apakah diizinkan dan *audit guard* yang merupakan mekanisme untuk memverifikasi apakah persyaratan sudah terpenuhi.

Mekanisme keamanan jaringan tradisional seperti *Transport Layer Security* (SSL/TLS), *Virtual Private Networks* (VPN), *IPSec* (*Internet Protocol Security*), dan *Secure Multipurpose Internet Mail Exchange* (S/MIME) tidak cukup untuk memberikan keamanan *end-to-end*. Pesan pada *web service* dapat melalui berbagai macam perantara sebelum mencapai tujuannya. Oleh karena itu, *message-level security* menjadi penting, tidak hanya *point-to-point* (*transport-level*) *security*.



Gambar II.2. *End-to-end security* pada *web service* (W3C, 2004)

Ilustrasi *end-to-end security* pada *web service* dapat dilihat pada Gambar II.2. *Requester agent* berkomunikasi dengan *ultimate receiver* melalui satu atau lebih perantara. Konteks keamanan pada pesan SOAP adalah *end-to-end*. Namun demikian, perantara dapat mengakses informasi pesan sehingga konteks keamanan menjadi antara perantara dan *requester agent* dan antara perantara dan *ultimate receiver*.

Berikut ini adalah ancaman keamanan terhadap pesan yang dikirim.

1. *Message Alteration*

Attacker dapat mengubah sebagian atau keseluruhan pesan sehingga mempengaruhi integritas pesan. Hal ini dilakukan *attacker* misalnya dengan menghapus sebagian pesan, mengubah pesan, atau menyisipkan informasi tambahan pada pesan. Modifikasi tersebut dapat dilakukan pada bagian *header*, *body*, atau bahkan *attachment* pesan.

2. *Confidentiality*

Ancaman keamanan ini terjadi karena informasi pesan atau bagian tertentu pesan diakses oleh pihak yang tidak berhak mengakses pesan tersebut.

3. *Man-in-the-Middle*

Attacker dapat melakukan penyerangan terhadap perantara pesan dan menahan pesan antara *requester* dan *receiver* akhir. Pesan tersebut kemudian dapat diakses atau bahkan diubah oleh *attacker*.

4. *Spoofing*

Pada serangan ini, *attacker* mengambil identitas pihak tertentu untuk menyerang target. Entitas target tetap mengira bahwa komunikasi dilakukan dengan pihak yang terpercaya. Serangan ini biasanya digunakan sebagai teknik untuk meluncurkan bentuk serangan lain.

5. *Denial of Service (DoS)*

Tujuan serangan ini adalah untuk mencegah user yang sah dalam mengakses *service*. Serangan DoS mengacaukan operasional pihak yang diserang dan memutuskan hubungan dengan dunia luar.

6. *Replay Attack*

Replay attack dilakukan *attacker* dengan menahan pesan dan kemudian menjawab kembali pesan tersebut kepada target. Teknik autentikasi yang sesuai adalah menggunakan *time stamp* dan *sequence number* pada pesan sehingga dapat mencegah *replay attack*.

Persyaratan keamanan pada *web service* adalah sebagai berikut.

1. Mekanisme autentikasi

Autentikasi dilakukan untuk memverifikasi identitas *requester* dan *provider agent*. Beberapa metode yang dapat dilakukan untuk autentikasi misalnya *password*, *one time password*, dan sertifikat. Penggunaan *password* saja tidak cukup sehingga diperlukan kombinasi dengan metode autentikasi lain.

2. Otorisasi

Otorisasi dilakukan untuk mengontrol akses ke *resource*. Setelah proses autentikasi dilakukan, mekanisme otorisasi mengontrol *requester* dalam mengakses *resource* yang sesuai. Hak akses dari *requester* ditentukan dalam kebijakan *web service*.

3. Integritas dan kerahasiaan data

Teknik enkripsi data dan *digital signature* dapat digunakan untuk tujuan integritas dan kerahasiaan data. Integritas data menjamin bahwa informasi tidak diubah selama proses transmisi tanpa terdeteksi sedangkan kerahasiaan data menjamin bahwa data hanya dapat diakses oleh pihak yang memiliki hak akses.

4. Integritas transaksi dan komunikasi

Bisnis proses harus dilakukan dengan benar dan operasional dieksekusi dengan cara yang tepat.

5. *Non-repudiation*

Non-repudiation melindungi suatu pihak dari penyangkalan suatu kejadian transaksi yang dilakukan pihak lain. Hal ini dilakukan dengan menyediakan bukti mengenai kejadian transaksi yang dapat digunakan untuk menyelesaikan pertentangan.

6. Integritas *end-to-end* dan kerahasiaan pesan

Integritas dan kerahasiaan pesan harus dapat dijamin meskipun terdapat perantara.

7. *Audit trails*

Audit trails digunakan untuk mengikuti jejak akses dan *behavior* dari pengguna. Hal ini dapat dilakukan oleh *agent* yang berfungsi sebagai *audit guard* yang dapat mengawasi *resource* dan *agent* lain.

8. Kebijakan keamanan

Kebijakan keamanan harus didefinisikan dan dapat dilaksanakan di berbagai macam *platform* dengan *privilege* yang beragam.

II.4 Metode Autentikasi

Mekanisme autentikasi digunakan pada sistem komputer untuk mengizinkan akses hanya kepada pengguna yang sah. Secara umum, autentikasi terdiri dari dua bagian, yaitu identifikasi dan verifikasi. Identifikasi merupakan tahapan yang digunakan untuk mengenali siapa pengguna yang akan mengakses sistem, sedangkan tahapan verifikasi melakukan pemeriksaan apakah pengguna tersebut adalah benar seperti identitas yang diberikan.

II.5 Macam-Macam Metode Autentikasi

Berbagai macam metode autentikasi telah dikembangkan untuk mengganti atau memperkuat autentikasi yang dilakukan dengan password.

1. *Graphical password*

2. *One time password*

Metode autentikasi *one time password* menggunakan token yang dibangkitkan oleh perangkat khusus.

3. *Challenge response*

Proses yang dilakukan pada autentikasi dengan menggunakan metode *challenge response* yaitu server membangkitkan *challenge* yang dikirim ke pengguna. Setelah itu, pengguna membuat sebuah respons dan mengirimkannya kembali.

4. *Out of band messaging*

Proses autentikasi dengan metode ini dilakukan dengan mengirimkan pesan melalui email atau pesan teks yang harus direspons pengguna. Metode ini memberikan keamanan lebih, namun dapat mengganggu bagi pengguna yang sering mengakses sistem.

5. *Biometrics*

Autentikasi biometrik menggunakan karakteristik tubuh pengguna seperti sidik jari, telapak tangan, telapak kaki, dan retina. Kelemahan metode ini adalah jika informasi digital biometrik dicuri maka akan sulit untuk diganti karena informasi tersebut merupakan bagian tubuh pengguna. Beberapa pengguna mungkin enggan memberikan informasi biometriknya karena alasan privasi. Selain itu, metode ini tidak akurat 100%. Penggunaan metode ini memerlukan peralatan *scanning* khusus.

6. *Behavioral pattern*

Proses identifikasi dilakukan dengan menggunakan pola aktivitas pengguna seperti pola ketikan, *gesture*, penekanan *mouse*, dan sebagainya. Aktivitas-aktivitas tersebut unik untuk setiap pengguna sehingga dapat digunakan dalam proses identifikasi. Namun demikian, akurasi tidak 100% pada setiap kondisi.

7. *Location based authentication*

Login pada sistem hanya terbatas pada alamat IP atau lokasi tertentu.

II.6 Autentikasi pada Web Service

Metode autentikasi yang saat ini digunakan pada *web service* diantaranya *HTTP Basic Authentication*, *HTTP Digest Authentication*, *Token Based Authentication* (*OAuth*).



Gambar II.3. Proses autentikasi menggunakan *HTTP Basic Authentication*

1. *HTTP Basic Authentication*

Metode *HTTP Basic Authentication* (Franks dkk., 1999) menggunakan ID dan *password* untuk melakukan proses autentikasi. Proses autentikasi diawali dengan client mengirimkan *request* dan kemudian server mengirim pesan “*HTTP 401 Not Authorized*” melalui *WWW-Authenticated HTTP Header*. Pada proses ini ID dan password dari client tidak dilakukan enkripsi atau *hash* melainkan hanya dilakukan proses *encode* dengan *Base64* dan kemudian disimpan pada *HTTP Authentication Header*. Metode autentikasi ini memiliki kelemahan yaitu rentan terhadap serangan seperti *replay attack*, *injection attack*, dan *middleware hijacking* (Jo dkk., 2014). Proses autentikasi menggunakan metode *HTTP Basic Authentication* dapat dilihat pada Gambar II.3.

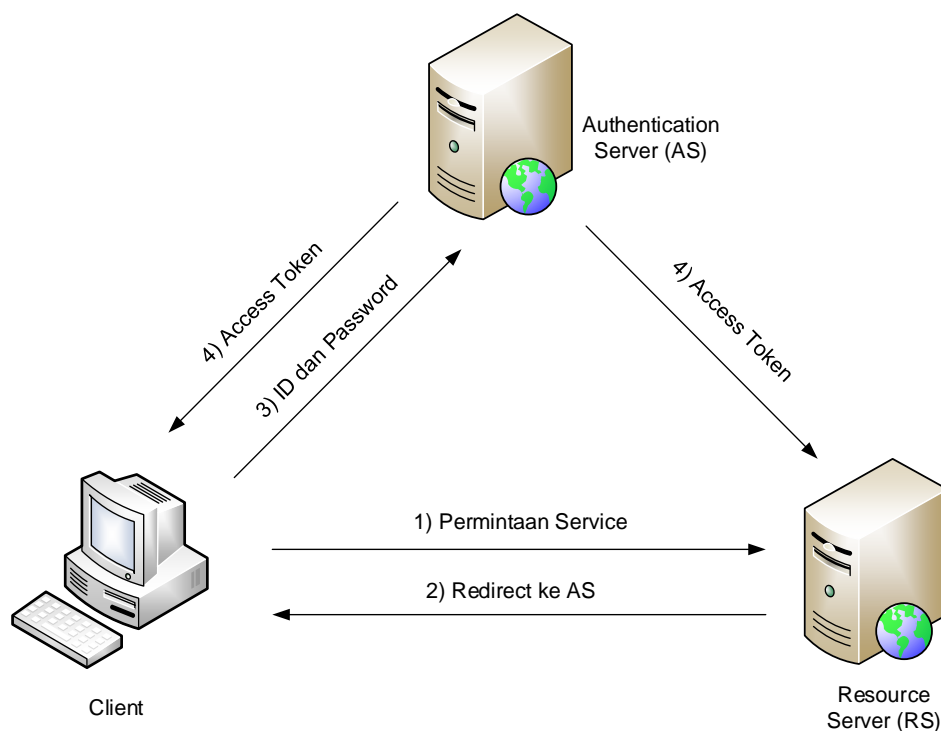
2. *HTTP Digest Authentication*

Proses autentikasi pada *HTTP Digest Authentication* (Franks dkk., 1999) sama seperti *HTTP Basic Authentication* namun ID dan *password* dienkrpsi menggunakan fungsi *hash* seperti MD5. Meskipun ID dan *password* terenkrpsi, namun metode ini masih memiliki kelemahan terhadap serangan *Man-in-the-Middle* yang dapat memperoleh informasi antara client dan server dikarenakan tidak adanya metode konfirmasi dari server ke client sehingga

fungsi dari metode ini tidak berjalan sebagaimana mestinya. Serangan ini mengubah *HTTP Digest Authentication* menjadi *HTTP Basic Authentication*.

3. *Token Based Authentication (OAuth)*

Proses autentikasi pada *Token Based Authentication* (Peng dkk., 2009) dilakukan dengan menggunakan token yang dibangkitkan secara dinamis. Alur proses autentikasi dengan metode ini digambarkan pada Gambar II.4. Pada saat client melakukan *request* ke *Resource Server*, maka akan dialihkan ke *Authentication Server*. Setelah itu, client melakukan login ke *Authentication Server* menggunakan ID dan *password*. *Authentication Server* merespons dengan memberikan token yang akan digunakan untuk komunikasi antara client dan *Resource Server*. Dalam melakukan permintaan *service*, user menyertakan token yang kemudian akan diverifikasi oleh *Resource Server*.

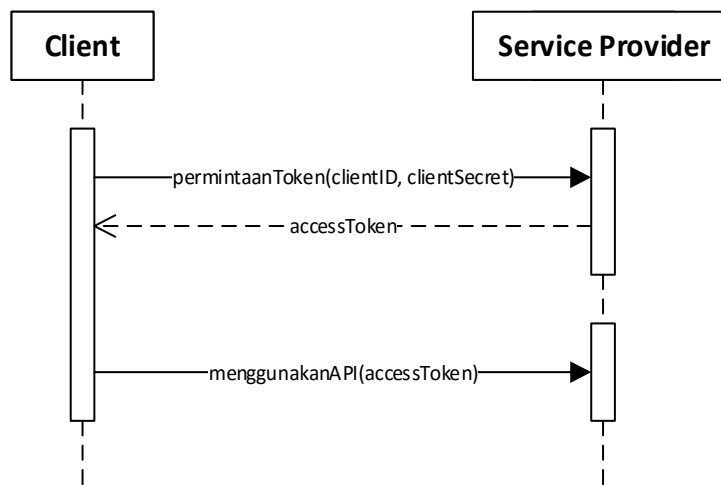


Gambar II.4. Proses autentikasi menggunakan *Token Based Authentication*

II.7 *Two-legged authorization dan three-legged authorization*

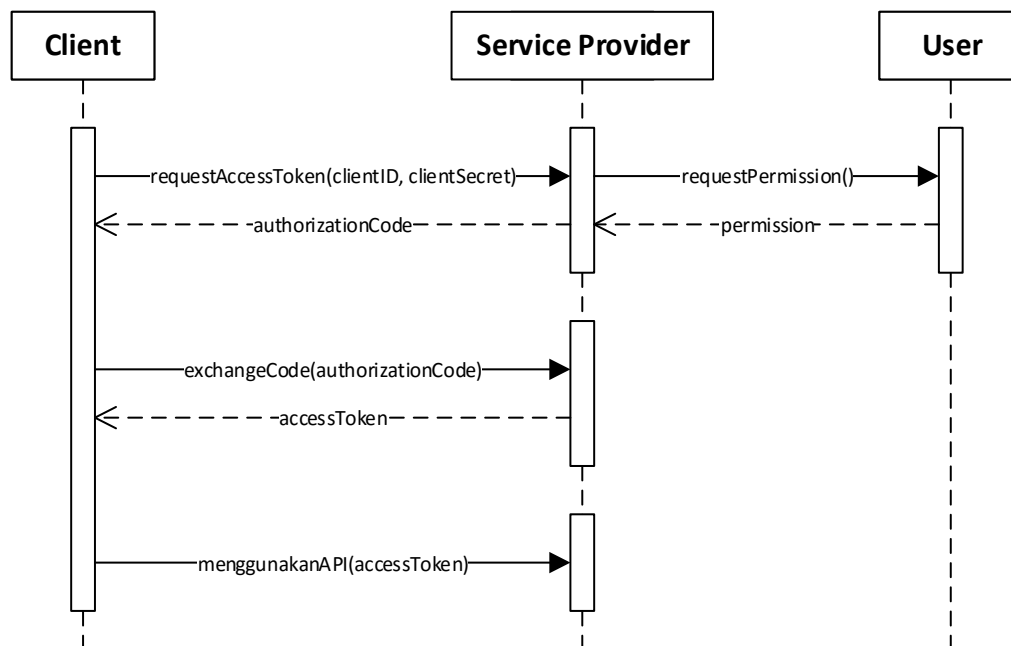
Sebuah aplikasi dapat menggunakan *public* maupun *private data*. Aplikasi yang menggunakan *public data* tidak perlu mendapatkan *authorization* dari *end user*.

Tipe *authorization* untuk aplikasi ini disebut dengan *two-legged authorization* karena *authorization* terjadi antara dua pihak, yaitu aplikasi (Client) dan sumber *public data* (Service Provider). Proses yang terjadi dalam *two-legged authorization* diilustrasikan dengan *sequence diagram* pada Gambar II.5. Client melakukan permintaan *access token* dengan menyediakan *Client ID* dan *Client secret* kepada Service Provider. Setelah itu, Service Provider mengirimkan *access token* kepada Client. Access token digunakan oleh Client untuk mengakses *public data* pada Service Provider.



Gambar II.5. *Sequence diagram* untuk *two-legged authorization*

Untuk mendapatkan akses *private data*, aplikasi membutuhkan *authorization* dari *end user*. Authorization terjadi antara tiga pihak, yaitu *end user* (User), aplikasi (Client), dan sumber *private data* (Service Provider), sehingga disebut dengan *three-legged authorization*. Proses yang terjadi yaitu diawali dengan permintaan *access token* oleh Client kepada Service Provider. Setelah itu, Service provider melakukan permintaan *permission* kepada User. Sebuah *permission* diberikan oleh User kepada Service Provider. Service Provider memberikan *authorization code* kepada Client. Authorization Code digunakan oleh Client untuk mendapatkan *access token*. Terakhir, Client mengakses *private data* pada Service Provider dengan menyediakan *access token* tersebut. *Sequence diagram* untuk *three-legged authorization* diberikan pada Gambar II.6.



Gambar II.6. *Sequence diagrami untuk three-legged authorization*

II.8 Psudeorandom Number

Random number dapat digunakan sebagai *key* dalam proses autentikasi antara dua pihak yang saling berkomunikasi. *Key* berfungsi untuk mencegah *replay attack*. *Random number* dapat mempersulit *attacker* untuk menebak *key* tersebut. Selain itu, *random number* juga dapat digunakan untuk membangkitkan *session key*, yaitu *key* yang hanya digunakan dalam periode waktu tertentu.

Pembangkitan *random number* dilakukan dengan menggunakan teknik algoritma kriptografi yang sudah tertentu (*deterministic*) sehingga hasilnya berupa *random number* yang secara statistik tidaklah acak. Namun, bila algoritma yang digunakan cukup baik, maka *random number* tersebut dapat melewati uji keacakan. Bilangan ini disebut dengan *pseudorandom number*. *Pseudorandom number* menggunakan input dengan nilai yang tetap, disebut sebagai *seed*, untuk membangkitkan keluaran berupa rangkaian *bit* dengan menggunakan algoritma tersebut (Stallings, 2011).

Pseudorandom Number Generator (PRNG) menggunakan algoritma yang pada dasarnya adalah sekuensial. Teknik PRNG yang banyak digunakan adalah algoritma yang diusulkan oleh Lehmer, yang dikenal sebagai *linear congruential method*. Rangkaian *random number* $\{X_n\}$ dihasilkan melalui persamaan (II.1). Jika m , a , c , dan X_0 adalah bilangan *integer*, maka teknik tersebut akan menghasilkan rangkaian bilangan *integer* dengan tiap bilangan berada pada batas $0 \leq X_n < m$.

$$X_{n+1} = (aX_n + c) \bmod m \quad (\text{II.1})$$

dengan:

m = Modulus, $m > 0$

a = Bilangan pengali, $0 < a < m$

c = Bilangan penambah, $0 \leq c < m$

X_0 = Nilai awal, atau *seed*, $0 \leq X_0 < m$

II.9 Seed Based Authentication

Seed Based Authentication (Nassar dan Chen, 2015) menggunakan *Pseudorandom Numbers* untuk membangkitkan password secara dinamis pada setiap proses autentikasi. *Pseudorandom Numbers Generator* dijalankan secara paralel pada client dan server dengan menggunakan *seed* yang sama (Salmon dkk., 2011). Untuk meningkatkan keamanan, tiga elemen yang digunakan pada proses autentikasi, yaitu URL, *username*, dan *password*, dibuat menjadi rahasia dan dinamis dengan menggunakan *Pseudorandom Numbers* tersebut.

Seed Based Authentication menggunakan notasi-notasi sebagai berikut.

1. Root File

File yang ditentukan oleh pengguna untuk diekstrak menjadi *seed* yang selanjutnya akan digunakan untuk membangkitkan token oleh PRNG. File ini dapat berupa file teks, kode QR, ataupun file biometrik seperti gambar sidik jari, iris mata, wajah, dsb.

2. SeqNum, n

Angka ini menandakan sudah berapa kali token dibangkitkan. SeqNum digunakan oleh PRNG untuk melakukan sinkronisasi token antara client dan server.

3. PRNG, P

Fungsi yang digunakan untuk membangkitkan *pseudorandom number*.

4. URLSeed, $S_u(n)$

Karakter sepanjang 16 *byte* yang diekstrak dari *root file*. Seed ini digunakan untuk membangkitkan token baik di client maupun di server selama proses autentikasi. URLSeed pada client dinotasikan dengan S_{url}^C sedangkan URLSeed pada server dinotasikan dengan S_{url}^S .

5. UNMSeed, $S_m(n)$

Karakter sepanjang 16 *byte* yang diekstrak dari *root file*. Seed ini berbeda dengan URLSeed karena diekstrak dari bagian yang berbeda dengan bagian *root file* untuk mengekstrak URLSeed. S_{unm}^C adalah notasi untuk UNMSeed pada client sedangkan S_{unm}^S adalah notasi untuk UNMSeed pada server.

6. URLToken, $T_{url}(n)$

URLToken dibangkitkan menggunakan fungsi P dengan input URLSeed. Token pada client, $T_{url}^C(n)$, dan token pada server, $T_{url}^S(n)$, dihitung dengan persamaan berikut.

$$T_{url}^C(n) = P(S_{url}^C(n)) \quad (II.2)$$

$$T_{url}^S(n) = P(S_{url}^S(n)) \quad (II.3)$$

7. UNMToken, $T_{unm}(n)$

Token ini dibangkitkan dengan fungsi P yang diberi input UNMSeed. UNMToken pada client, $T_{unm}^C(n)$, dan UNMToken pada server, $T_{unm}^S(n)$, dihitung dengan persamaan berikut.

$$T_{unm}^C(n) = P(S_{unm}^C(n)) \quad (II.4)$$

$$T_{unm}^S(n) = P(S_{unm}^S(n)) \quad (II.5)$$

Proses pada *Seed Based Authentication* dibagi menjadi empat tahap yaitu inisialisasi, identifikasi, autentikasi, dan sinkronisasi.

1. Inisialisasi

Tahap inisialisasi adalah tahap awal dimana pengguna melakukan registrasi untuk menentukan dan mengirimkan *root file* ke server melalui *secure channel*, seperti HTTPS. *Root file* ini kemudian digunakan untuk mengekstrak dua seed, URLSeed dan UNMSeed, yang dilakukan di client dan server dengan menggunakan fungsi yang telah didefinisikan sebelumnya. Sebagai contoh, URLSeed dapat berupa 16 *byte* yang diambil dari bagian tengah file dan UNMSeed diambil dari 16 *byte* terakhir file. Server dan client kemudian membangkitkan token pertamanya pada *sequence number* 0. Client menghitung $T^C_{url}(n)$ dan $T^C_{unm}(n)$ menggunakan persamaan (II.2) dan (II.4), sedangkan server menghitung $T^S_{url}(n)$ dan $T^S_{unm}(n)$ menggunakan persamaan (II.3) dan (II.5).

Proses inisialisasi kemudian dilanjutkan dengan client mengirimkan kedua token ke server. Token ini kemudian dilakukan perbandingan dengan token pada server. Jika $T^C_{url}(0)$ sama dengan $T^S_{url}(0)$ dan $T^C_{unm}(0)$ sama dengan $T^S_{unm}(0)$ maka server akan mengirimkan pesan sukses yang menandakan bahwa proses ini telah berhasil. Setelah itu, server akan menghapus *root file* dan hanya akan menyimpan kedua *seed* yang telah diekstrak dari *root file* tersebut. Selain itu, server juga akan menyimpan token yang dibangkitkan menggunakan *sequence number* berikutnya, $n + 1$.

2. Identifikasi

Tahap identifikasi dilakukan saat pengguna akan melakukan login ke aplikasi pada server. Proses ini diawali dengan pengguna mengirimkan URLToken dan UNMToken yang dibangkitkan dengan menggunakan fungsi PRNG dengan input masing-masing *seed* dan *sequence number* yang berlaku yaitu $T^C_{url}(n)$ dan $T^C_{unm}(n)$. Setelah itu, kedua token dikirim ke server dengan HTTPS POST request, *www.domain.com/uri/T^C_{url}(n)/T^C_{unm}(n)/n*. Proses ini berhasil jika

server dapat menemukan pasangan token sesuai dengan token dan *sequence number* yang dikirimkan yaitu sebagai berikut.

$$\{T^C_{url}(n), T^C_{unm}(n)\} == \{T^S_{url}(n), T^S_{unm}(n)\} \quad (II.6)$$

Jika server tidak dapat menemukan token yang dimaksud maka server mengirimkan pesan *error*, misalnya *404 error code*. Jika proses ini berhasil maka client dan server akan menambahkan *sequence number* menjadi $n + 1$. Kemudian server membangkitkan token dengan menggunakan *sequence number* tersebut untuk digunakan dalam proses identifikasi berikutnya.

3. Autentikasi

Proses autentikasi dilakukan setelah tahap identifikasi berhasil dilakukan antara client dan server. Proses ini diawali dengan server mengirimkan empat *sequence number* yang acak dan berbeda kepada client yaitu $\{x, y, u, v\}$. Client diminta untuk menyediakan *hash* dari empat token yang dibangkitkan dari *sequence number* tersebut dalam waktu yang ditentukan oleh server. Empat token tersebut adalah URLToken untuk *sequence number* x dan y dan UNMToken untuk *sequence number* u dan v . Kemudian, keempat token tersebut digabung untuk dilakukan penghitungan nilai *hash* dengan menggunakan fungsi h seperti di bawah ini.

$$H^C = h(T^C_u(x), T^C_u(y), T^C_m(u), T^C_m(v)) \quad (II.7)$$

Setelah itu client mengirimkan H^C ke server yang kemudian akan dibandingkan dengan nilai *hash* yang dihitung oleh server menggunakan *seed* dan *sequence number* yang sama.

$$\begin{aligned} h(T^S_u(x), T^S_u(y), T^S_m(u), T^S_m(v)) == \\ h(T^C_u(x), T^C_u(y), T^C_m(u), T^C_m(v)) \end{aligned} \quad (II.8)$$

Proses autentikasi berhasil jika nilai *hash* yang dibandingkan sesuai, seperti pada persamaan (II.8). Apabila client tidak melakukan respons pada waktu yang telah ditentukan, maka server akan membatalkan proses autentikasi dan client harus melakukan proses *login* dari awal.

4. Sinkronisasi

Proses sinkronisasi dilakukan apabila *sequence number* pada client dan server tidak sama (*out of sync*). Hal ini dapat terjadi misalnya disebabkan oleh kesalahan jaringan yang mengakibatkan client dapat memperbaharui *sequence number* sedangkan server tidak. Selain itu, serangan *denial-of-service* dapat juga menjadi penyebab *out of sync* pada *sequence number*. Selama terjadi serangan ini, server dapat membuat *expire* secara sepihak pada *sequence number* yang berlaku untuk alasan keamanan.

Untuk melakukan sinkronisasi, client melakukan permintaan sinkronisasi dengan menyediakan *sequence number* yang berlaku dan tokennya, $\{T_{url}^C(n), T_{unm}^C(n), n\}$. Selanjutnya, server memastikan client terautentikasi dengan mengirimkan empat *random number* seperti pada proses autentikasi. Client kemudian menyediakan *hash* dari empat token yang dibangkitkan dari empat *sequence number* tersebut dan server membandingkan nilai *hash* client dengan nilai *hash* yang dihitung oleh server. Apabila proses ini berhasil maka server akan mengirimkan *sequence number* yang valid untuk digunakan pada proses identifikasi.

II.10 ID-based Encryption (IBE)

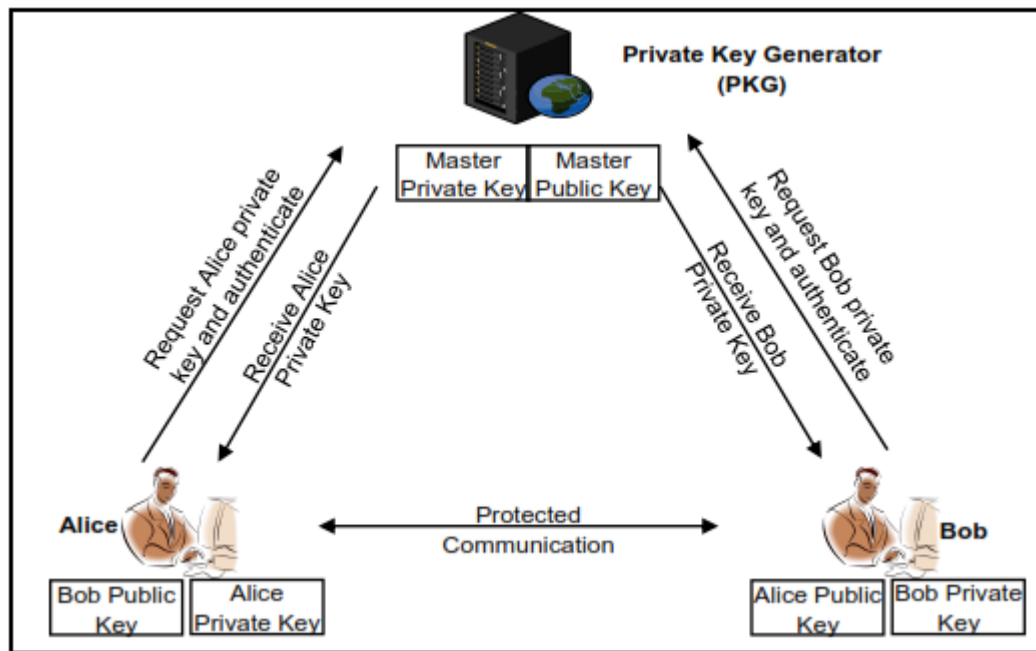
ID-Based Encryption (IBE) merupakan teknik enkripsi yang menggunakan algoritma kunci asimetrik. IBE berbeda dengan RSA dalam hal pembangkitan kunci *private* dan *public* yang mana pada RSA kedua kunci tersebut dibangkitkan pada waktu yang bersamaan oleh pengguna, sedangkan pada IBE kunci *private* dibuat oleh Private Key Generator (PKG).

PKG merupakan pihak ketiga yang terpercaya untuk membangkitkan *private key* dari ID unik client seperti alamat email. Sementara itu, *public key* akan dibangkitkan oleh pengirim jika dibutuhkan oleh penerima dan PKG. Gambar II.7 adalah diagram yang menggambarkan proses pada IBE.

Pada IBE terdapat empat langkah proses yaitu *setup*, *extract*, *encrypt*, dan *decrypt*.

1. Setup

Pada langkah ini PKG membuat *master private key* dan *master public key*. *Master private key* disimpan secara aman dan digunakan untuk membangkitkan *private key* client. Hal ini berbeda dengan *master public key* yang terbuka untuk publik.



Gambar II.7. Proses pada *ID-based Encryption* (Lee dkk., 2015)

2. Extract

PKG membangkitkan *private key* client menggunakan *master private key* dan ID client.

$$K_{Alice-pri} = G(MK_{pri}, Alice_{email}) \quad (II.9)$$

$$K_{Bob-pri} = G(MK_{pri}, Bob_{email}) \quad (II.10)$$

3. *Encrypt*

Pengirim melakukan enkripsi pada pesan menggunakan *public key* penerima yang dibangkitkan oleh PKG.

$$K_{Bob-pub} = G(MK_{pub}, Bob_{email}) \quad (II.11)$$

$$C = E(K_{Bob-pub}, M) \quad (II.12)$$

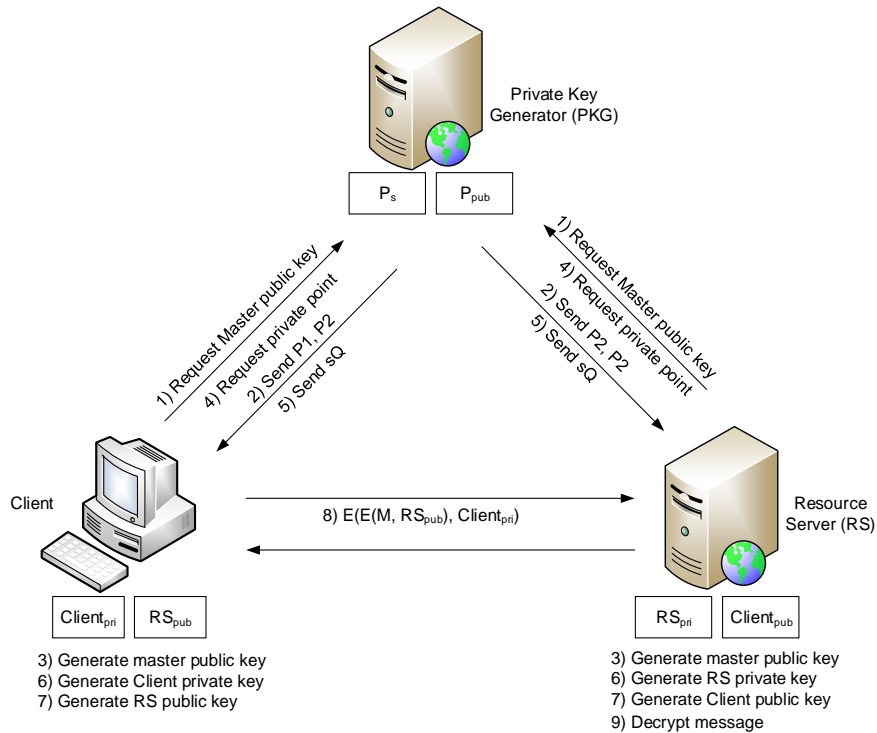
4. *Decrypt*

Pesan dilakukan dekripsi menggunakan *private key* penerima yang dibangkitkan oleh PKG.

$$M = D(K_{Bob-pub}, C) \quad (II.13)$$

II.11 ID-based Authentication pada *RESTful Web Service*

Setiap objek dari IoT direpresentasikan menggunakan URI yang unik. *ID-based Authentication* (IBA) menggunakan IBE dan REST URI dalam proses autentikasi. Gambar II.8 merupakan alur proses autentikasi menggunakan ID-based Authentication pada *RESTful web service*.



Gambar II.8. Alur proses autentikasi pada *ID-based Authentication*

Pada gambar di atas proses autentikasi diasumsikan bahwa PKG telah membangkitkan *master public key* dan *master private key*. Dengan demikian, proses selanjutnya yang dilakukan adalah *extract*, *encrypt*, dan *decrypt*.

1. *Extract*

Pada tahap ini PKG mengirimkan $p1$, $p2$, dan sQ kepada RS dan client untuk membuat *private key* dan *public key*.

- a. Client membuat *private key* melalui *private point* (sQ) dan *master key* (P_{pub}). Selain itu, client membangkitkan *public key* dari RS.

$$Client_{pri} = sQ \cdot P_{pub} \quad (II.14)$$

$$RS_{pub} = URL_{ss} \cdot P_{pub} \quad (II.15)$$

- b. RS membuat *private key* melalui *private point* (sQ') dan *master key* (P_{pub}). RS juga membangkitkan *public key* dari client.

$$RS_{pri} = sQ' \cdot P_{pub} \quad (II.16)$$

$$Client_{pub} = URL_{client} \cdot P_{pub} \quad (II.17)$$

2. *Encrypt*

Client melakukan enkripsi pada pesan (M) menggunakan *public key* dari RS (RS_{pub}). Setelah itu, pesan dienkrip lagi menggunakan *private key* dari client ($Client_{pri}$) sebagai *digital signature*.

$$C = E(M, RS_{pub}) \quad (II.18)$$

$$C_s = E(C, Client_{pri}) \quad (II.19)$$

3. *Decrypt*

RS melakukan verifikasi pesan yang diterima (C_s) dengan *public key* dari client ($Client_{pub}$). Setelah itu, pesan tersebut dilakukan dekripsi menggunakan *private key* dari RS (RS_{pri}).

$$C = D(C_s, Client_{pub}) \quad (II.20)$$

$$M = D(C, RS_{pri}) \quad (II.21)$$

II.12 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan standar untuk pengembangan sistem dan software (Miles dan Hamilton, 2006). UML memudahkan dalam menggambarkan persyaratan sistem, komponen yang dibutuhkan, dan fungsinya. Selain itu, UML memungkinkan rancangan untuk diberikan kepada pihak lain tanpa adanya salah penafsiran pada rancangan sistem. Sebuah model merupakan abstraksi dari sesuatu yang nyata. Model tersebut adalah bentuk sederhana dari sebuah sistem sehingga dapat dipahami dan dievaluasi lebih mudah dari pada melalui sistem yang sebenarnya. UML memiliki beberapa kelebihan sebagai berikut.

1. *It's a formal language*

Setiap bagian dari UML memiliki makna yang didefinisikan dengan baik sehingga model sistem tidak akan disalahartikan.

2. *It's concise*

Bahasa pemodelan UML menggunakan notasi yang mudah dan sederhana.

3. *It's comprehensive*

UML menggambarkan semua aspek penting sebuah sistem.

4. *It's scalable*

UML dapat menangani perancangan sistem baik dengan skala kecil maupun besar.

5. *It's built on lessons learned*

UML menjadi *best practice* dari komunitas *object-oriented* selama 15 tahun terakhir.

6. *It's the standard*

UML dikontrol oleh kelompok standar terbuka dengan kontribusi yang aktif dari para vendor dan akademisi yang tidak terikat pada produk tertentu.


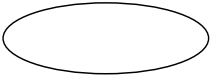

Sebuah bahasa pemodelan dapat berupa *pseudo-code*, *actual code*, gambar, diagram, atau deskripsi dalam bentuk paragraf. Elemen yang membentuk bahasa

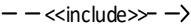

pemodelan disebut dengan notasi. Pemodelan pada UML dilakukan dengan menggunakan diagram, diantaranya *use case diagram*, *activity diagram*, *class diagram*, dan *sequence diagram*. Penjelasan diagram-diagram tersebut beserta notasinya diberikan pada bagian berikut.

II.12.1 Use Case Diagram

Use case merupakan titik awal dari setiap pengembangan, perancangan, pengujian, dan pendokumentasian sistem berorientasi objek. *Use case* menggambarkan persyaratan fungsionalitas yang disediakan oleh sistem. *Use case* dapat membantu dalam menyusun pengujian sistem. Dengan menggunakan *use case*, kasus pengujian dapat dibentuk dengan baik karena *use case* mendefinisikan persyaratan dan kriteria sukses sistem. Notasi yang digunakan pada *use case diagram* dapat dilihat pada Tabel II.1.

Tabel II.1. Notasi pada *Use Case Diagram*



Notasi	Keterangan
	<u>Actor</u> <i>Actor</i> adalah pihak eksternal di luar sistem yang berinteraksi dengan sistem. <i>Actor</i> dapat berupa orang maupun sistem lain. <i>Actor</i> digambarkan dengan “stick man” dan diberi label dengan nama yang sesuai.
	<u>Use case</u> <i>Use case</i> mendefinisikan penggunaan sistem oleh seorang <i>actor</i> dalam menyelesaikan fungsi tertentu. <i>Use case</i> diidentifikasi dari persyaratan sistem. <i>Use case</i> digambarkan dengan sebuah oval dengan nama yang mendeskripsikan interaksi <i>actor</i> dengan sistem.
	<u>Communication line</u> <i>Communication line</i> menghubungkan <i>actor</i> dengan <i>use case</i> untuk menunjukkan partisipasi <i>actor</i> dalam <i>use case</i> .




	<p><u>Include relationship</u></p> <p>Sebuah <i>use case</i> dapat digunakan kembali oleh <i>use case</i> lainnya dengan menggunakan <i>include relationship</i> sehingga dapat mencegah penggambaran <i>use case</i> yang berulang. <i>Include relationship</i> menyatakan bahwa <i>use case</i> yang berada di pangkal tanda panah menggunakan kembali semua tahapan dari <i>use case</i> yang disertakan.</p>
	<p><u>Extend relationship</u></p> <p>Sama seperti <i>include relationship</i>, <i>extend relationship</i> menunjukkan penggunaan kembali <i>use case</i> lainnya namun dilakukan secara opsional. Tahapan tambahan dari sebuah <i>use case</i> dapat digambarkan dengan <i>extend relationship</i> untuk menunjukkan bahwa tahapan tersebut bersifat opsional.</p>

II.12.2 Activity Diagram

Berbeda dengan *use case* yang menunjukkan apa yang seharusnya dapat dilakukan oleh sistem, *activity diagram* menunjukkan bagaimana sistem mencapai tujuannya. *Activity diagram* menggambarkan proses yang terjadi di dalam sistem. *Activity diagram* baik untuk memodelkan proses bisnis, yaitu satu set tahapan untuk mencapai tujuan bisnis. Notasi yang digunakan pada *activity diagram* diberikan pada Tabel II.2.

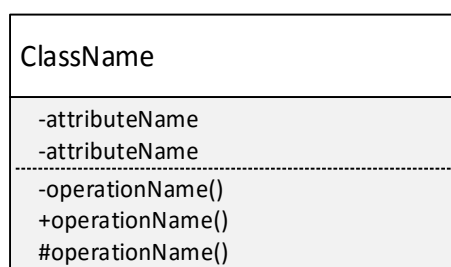
Tabel II.2. Notasi pada *Activity Diagram*

Notasi	Keterangan
	<p><u>Initial node</u></p> <p><i>Initial node</i> menyatakan awalan dari sebuah aktivitas yang digambarkan dengan sebuah lingkaran berwarna hitam.</p>
	<p><u>Action</u></p>

	<i>Action</i> merupakan tahapan penting dalam sebuah aktivitas. <i>Action</i> dapat berupa tindakan, penghitungan, atau tahapan penting dalam sebuah proses.
	<u>Edge</u> Alur aktivitas digambarkan dengan sebuah garis panah yang disebut dengan <i>edge</i> . Mata panah menunjukkan arah aliran dari satu <i>action</i> ke <i>action</i> berikutnya.
	<u>Decision</u> <i>Decision</i> dapat dianalogikan dengan pernyataan <i>if-else</i> dalam sebuah kode. Terdapat dua <i>edge</i> yang keluar dari sebuah <i>decision</i> yang masing-masing diberi label dengan kondisi boolean.
	<u>Final node</u> <i>Final node</i> menyatakan akhir dari sebuah aktivitas.

II.12.3 Class Diagram

Class merupakan bagian penting dari sistem berorientasi objek. Sebuah struktur sistem dibentuk dari kumpulan objek. *Class* mendeskripsikan berbagai jenis objek yang dapat dimiliki oleh sebuah sistem untuk memenuhi persyaratan sistem. *Class diagram* menggambarkan *class* tersebut serta hubungan antar *class*.



Gambar II.9. Contoh notasi *class* pada *class diagram*

Class mendefinisikan karakteristik suatu objek. *Class* terdiri dari *attribute* dan *operation*. *Attribute* menunjukkan informasi yang dapat dimiliki oleh suatu objek, sedangkan *operation* menunjukkan fungsi yang dapat dilakukan suatu objek. *Class*

dalam UML digambarkan oleh sebuah persegi panjang yang dibagi menjadi 3 bagian, yaitu nama *class*, *attribute*, dan *operation*, seperti pada Gambar II.9. Class memiliki visibilitas dari *attribute* dan *operation* terhadap class lainnya. Notasi visibilitas digambarkan sebelum nama *attribute* dan *operation*. Terdapat empat visibilitas yang dapat diaplikasikan pada *class* diagram, seperti pada Tabel II.3.

Tabel II.3. Notasi visibilitas pada Class Diagram

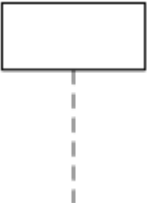




Notasi	Keterangan
+	<u>Public visibility</u> <i>Public visibility</i> menunjukkan bahwa <i>attribute</i> atau <i>operation</i> dapat diakses oleh <i>class</i> lainnya.
#	<u>Protected visibility</u> <i>Protected visibility</i> memiliki tingkat visibilitas yang lebih rendah dibandingkan dengan <i>public visibility</i> . <i>Attribute</i> dan <i>operation</i> dengan <i>protected visibility</i> dapat diakses oleh <i>method</i> yang terdapat pada <i>class</i> tersebut dan pada <i>class</i> yang meng- <i>extend class</i> tersebut.
~	<u>Package visibility</u> <i>Package visibility</i> menunjukkan bahwa <i>attribute</i> atau <i>operation</i> dapat diakses oleh <i>class</i> yang berada dalam satu <i>package</i> .
-	<u>Private visibility</u> <i>Attribute</i> atau <i>operation</i> yang dideklarasikan dengan <i>private visibility</i> hanya dapat diakses oleh <i>method</i> dalam <i>class</i> tersebut.

II.12.4 Sequence Diagram

Use case menggambarkan fungsi apa yang seharusnya dapat dilakukan sistem. *Class* menggambarkan berbagai macam jenis objek yang membentuk struktur sistem. Berbeda dengan kedua diagram tersebut, *sequence diagram* menggambarkan bagaimana sebenarnya sistem menjalankan fungsinya. *Sequence diagram* menunjukkan bagaimana interaksi antar bagian yang membentuk sistem.

Saat sebuah *use case* dieksekusi, interaksi yang akan dijalankan dan urutannya dapat digambarkan pada *sequence diagram*. Notasi yang digunakan pada *sequence diagram* diberikan pada Tabel II.4.

Tabel II.4. Notasi pada *Sequence Diagram*

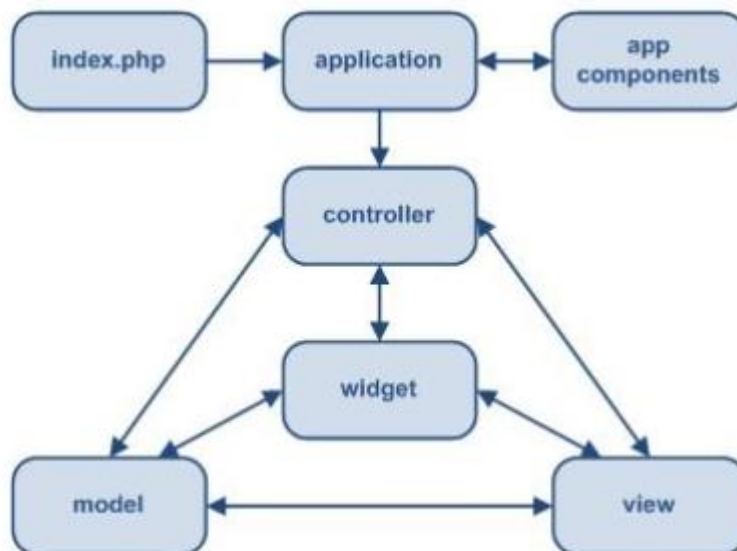
Notasi	Keterangan
	<u><i>Participant</i></u> <i>Sequence diagram</i> terdiri dari kumpulan <i>participant</i> , bagian dari sistem yang saling berinteraksi selama proses berlangsung. Setiap <i>participant</i> memiliki <i>lifeline</i> yang berjalan ke arah bawah. <i>Lifeline</i> menyatakan keberadaan <i>participant</i> tersebut.
	<u><i>Activation bar</i></u> <i>Activation bar</i> menunjukkan bahwa <i>participant</i> dalam keadaan aktif dalam waktu tertentu.
	<u><i>Synchronous message</i></u> Interaksi dalam <i>sequence diagram</i> terjadi saat <i>participant</i> (<i>message caller</i>) mengirimkan pesan kepada <i>participant</i> lain (<i>message receiver</i>). Pesan digambarkan dengan tanda panah. <i>Message caller</i> mengirimkan <i>Synchronous message</i> kepada <i>message receiver</i> jika ingin menunggu <i>return value</i> sebelum menjalankan interaksi berikutnya.
	<u><i>Asynchronous message</i></u> <i>Message caller</i> mengirimkan <i>asynchronous message</i> tanpa menunggu <i>return value</i> dari <i>message receiver</i> . Beberapa interaksi dapat terjadi secara bersamaan.
	<u><i>Return message</i></u> <i>Return message</i> merupakan notasi yang opsional yang digunakan pada bagian akhir <i>activation bar</i> . Notasi ini menunjukkan bahwa aliran kontrol kembali kepada

	<i>participant</i> yang mengirimkan pesan. Dalam sebuah kode, <i>return message</i> terdapat pada akhir sebuah <i>method</i> atau pada pernyataan <i>return</i> .
--	---

II.13 Yii Framework

Yii merupakan *framework* PHP untuk pengembangan aplikasi web, baik yang sederhana maupun kompleks. Yii adalah *framework* dengan kode sumber yang terbuka. Yii memiliki kinerja yang tinggi dibandingkan dengan framework lain jika dilihat dari nilai *request per second* (RPS).

Yii mengimplementasikan pola pengembangan *model-view-controller* (MVC). MVC bertujuan untuk memisahkan logika bisnis dengan antarmuka sehingga pengembang dapat dengan mudah membuat perubahan pada setiap bagian tanpa mempengaruhi bagian lain. Dalam MVC, model merepresentasikan data dan *business rule*, view berisi bagian-bagian antarmuka seperti teks dan form input, dan *controller* adalah yang mengelola komunikasi antara *model* dan *view*. Struktur aplikasi yii ditunjukkan pada Gambar II.10.



Gambar II.10. Struktur aplikasi pada Yii Framework (Yii, n.d.)

II.14 Xdebug

Xdebug merupakan ekstensi tambahan PHP yang menyediakan fungsi *debugging* dan *profiling* (Xdebug, n.d.). Informasi yang dapat disediakan oleh Xdebug diantaranya sebagai berikut.

1. Stack Traces

Xdebug menampilkan *stack trace* ketika PHP memiliki *notice*, *warning*, dan *error*.

2. Function Traces

Xdebug mencatat *log* pemanggilan fungsi, termasuk parameter dan nilai yang dikembalikan.

3. Code Coverage Analysis

Fitur ini menunjukkan baris mana pada skrip yang dieksekusi selama proses *request*.

4. Profiling PHP Scripts

Xdebug Profiler dapat menemukan *bottleneck* pada skrip PHP dan hasilnya dapat divisualisasikan dengan perangkat seperti KcacheGrind atau WinCacheGrind.

5. Remote Debugging

Xdebug menyediakan antarmuka untuk client *debugger* berinteraksi dengan skrip PHP yang sedang dieksekusi.

Xdebug Profiler adalah salah satu fitur yang disediakan oleh ekstensi Xdebug. Xdebug Profiler memiliki kemampuan dalam menganalisa kode PHP dan mengidentifikasi *bottleneck*. Fitur ini dapat menentukan kode program bagian mana yang lambat sehingga bagian tersebut memerlukan perbaikan.

Profiling diaktifkan dengan memberikan nilai pada parameter `xdebug.profiler_enable` menjadi 1 pada `php.ini`. Output xdebug profiler terdapat pada direktori yang dinyatakan dalam parameter `xdebug.profiler_output_dir`. Nama file output tersebut diawali dengan “cachegrind.out.” dan diakhiri ID proses (PID) dari PHP atau Apache. Profiler dapat diaktifkan secara selektif untuk bagian tertentu program dengan cara memberikan nilai 1 pada parameter

xdebug.profiler_enable_trigger. Selanjutnya profiler akan dijalankan jika ada variabel GET/POST dengan nama XDEBUG_PROFILE. Hal ini dapat dilakukan dengan memberikan nilai 0 pada xdebug.profiler_enable.

Keluaran dari xdebug profiler memiliki format file yang kompatibel dengan *cachegrind* sehingga analisis dapat dilakukan dengan aplikasi KCacheGrind di Linux atau QcacheGrind di Windows dan Mac OSX. Pada aplikasi tersebut, bagian *Flat Profile* memberikan informasi mengenai semua fungsi pada skrip yang dijalankan yang diurutkan berdasarkan waktu yang dihabiskan pada fungsi tersebut dan anaknya. Kolom “*Self*” menunjukkan waktu yang dihabiskan pada fungsi tersebut (tidak termasuk anaknya), kolom “*Called*” menunjukkan berapa kali fungsi tersebut dipanggil, dan kolom “*Function*” menunjukkan nama fungsi tersebut.

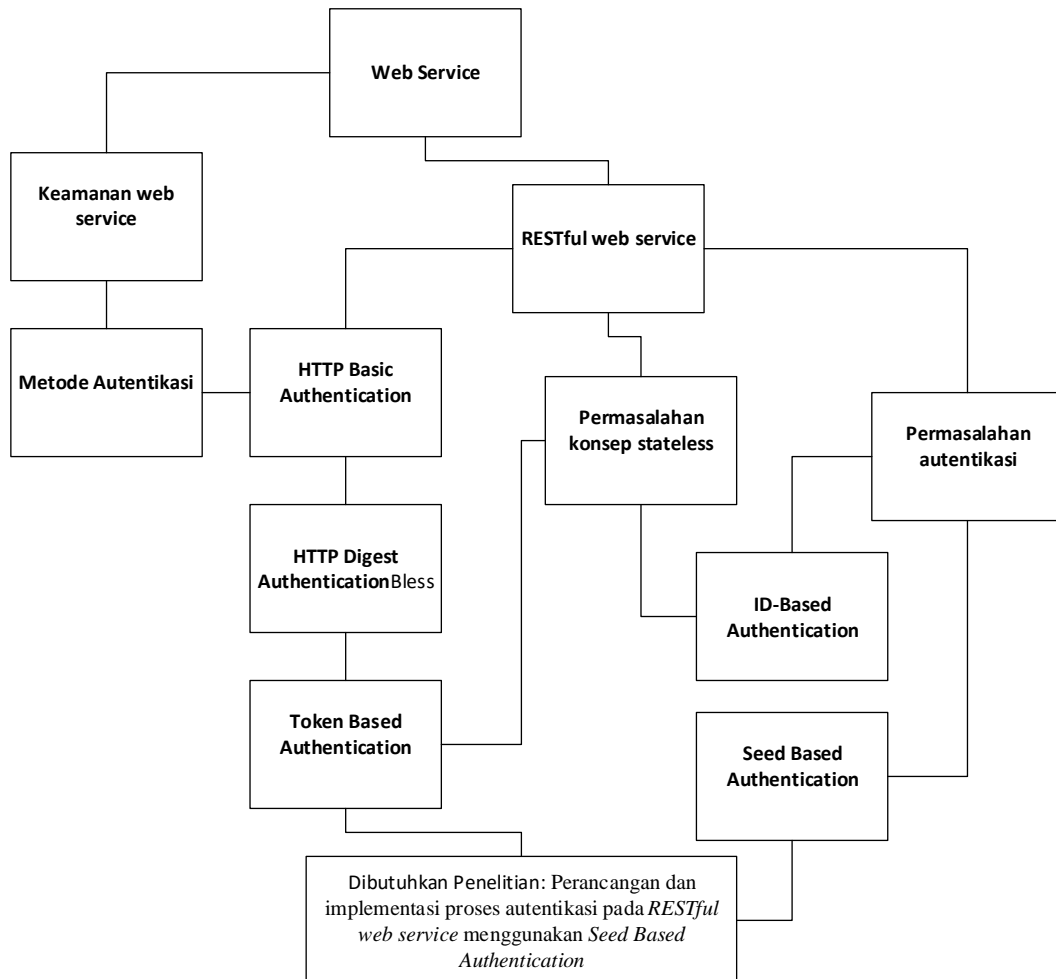
Pada bagian kanan terdapat dua panel, yaitu panel atas dan panel bawah. Kolom “*Cost*” pada panel atas menunjukkan *time spend* dari fungsi terpilih saat dipanggil dari fungsi yang ada di *list*. Angka pada kolom *Cost* akan selalu berjumlah 100%. Kolom *Cost* pada panel bawah menunjukkan *time spend* saat memanggil fungsi dari *list*. Apabila nilai *cost* dijumlahkan maka tidak akan mencapai 100% karena fungsi tersebut juga akan memakan waktu dalam proses eksekusi.

Tab “*All Callers*” dan “*All Calls*” menunjukkan *direct call* dimana fungsi tersebut dipanggil secara berurutan. Panel atas menunjukkan semua fungsi memanggil fungsi yang sedang terpilih secara langsung maupun tidak langsung dengan fungsi lainnya. Kolom “*Distance*” menunjukkan berapa banyak panggilan fungsi diantara fungsi tersebut dengan fungsi yang sedang terpilih.

II.15 Literature Map

Literature map dari penelitian ini dapat dilihat pada Gambar II.11. Tinjauan pustaka penelitian ini dimulai dari definisi *web service* dan bagaimana arsitekturnya. Kemudian tinjauan pustaka dilanjutkan pada keamanan *web service*, metode autentikasi dan jenis-jenisnya. RESTful *web service* kemudian dikaji bagaimana

keterbatasan pada metode tersebut dan metode yang dapat digunakan untuk menyelesaikan permasalahan tersebut.

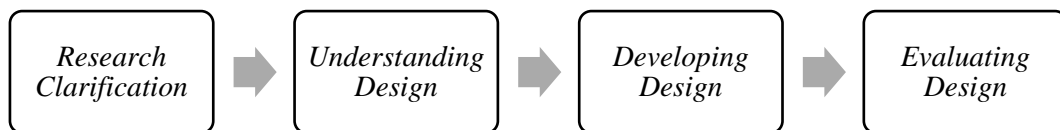


Gambar II.11. *Literature Map* penelitian

Halaman ini sengaja dikosongkan

Bab III Metodologi

Metode yang digunakan pada penelitian ini adalah *Design Research Methodology* (DRM) (Blessing dkk., 2009). Dalam metode ini, tahapan yang dilakukan adalah *Research Clarification*, *Understanding Design*, *Developing Design*, dan *Evaluating Design*. Gambar III.1 adalah bagan yang menggambarkan tahapan tersebut.



Gambar III.1. Tahapan pada *Design Research methodology* (DRM)

1. *Research Clarification*

Tahapan ini dilakukan untuk mengidentifikasi permasalahan dan menentukan tujuan penelitian. Studi literatur dilakukan pada tahapan ini, diantaranya mengenai *web service*, REST, keamanan *web service*, dan berbagai macam metode autentikasi.

2. *Understanding Design*

Pada tahapan ini dilakukan pemahaman perancangan yang akan dibuat sebagai solusi permasalahan yang telah diidentifikasi pada tahapan sebelumnya. Solusi yang akan diajukan dalam penelitian ini adalah melakukan perancangan proses autentikasi dengan metode *Seed Based Authentication* yang dapat mengatasi permasalahan autentikasi pada *RESTful web service*.

Perancangan proses autentikasi pada *RESTful web service* menggabungkan metode Token Based Authentication dan Seed Based Authentication. Hal ini dilakukan karena kelebihan *Token Based Authentication* dalam menjaga konsep *stateless* pada *RESTful web service* dan kelebihan *Seed Based Authentication* dalam mengatasi permasalahan autentikasi yang dilakukan dengan *password* dan *username* yang statis.

3. *Developing Design*

Perancangan proses autentikasi pada *RESTful web service* dilakukan pada tahapan ini. Perancangan proses autentikasi mengikuti tahapan pada *Seed Based Authentication*, yaitu inisialisasi, identifikasi, autentikasi, dan sinkronisasi. Perancangan yang dibuat meliputi perancangan arsitektur, proses, database, dan antarmuka.

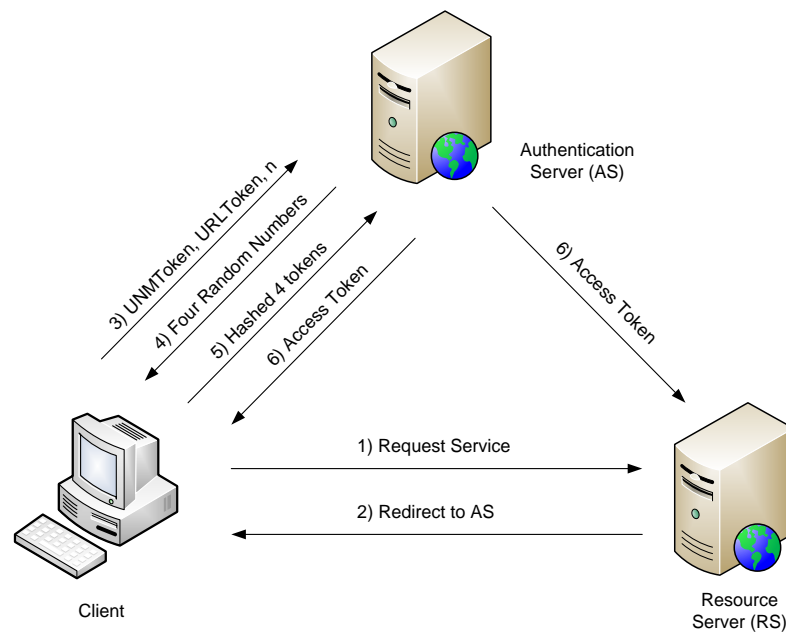
4. *Evaluating Design*

Pada tahapan ini dilakukan evaluasi dan pengukuran sejauh mana perancangan proses autentikasi pada *RESTful web service* yang diajukan dapat mengatasi permasalahan penelitian. Evaluasi dilakukan dengan mengimplementasikan hasil perancangan dalam sebuah prototipe. Setelah itu, pengujian dilakukan terhadap prototipe tersebut. Pengujian dilakukan dengan melakukan simulasi proses autentikasi oleh client yang akan mengakses *resource* pada *Service Provider*. Pengujian prototipe meliputi pengujian fungsional, pengujian kinerja, evaluasi keamanan, dan evaluasi perbandingan dengan metode lain.

Bab IV Perancangan

Bab ini membahas perancangan proses autentikasi pada *RESTful web service*. Hasil perancangan akan digunakan sebagai pedoman dalam pembuatan prototipe *web service* pada tahap implementasi. Perancangan yang dilakukan diantaranya perancangan proses autentikasi, *database*, dan antarmuka.

Seperti telah dijelaskan pada Bab I, *Token Based Authentication* memiliki kelebihan dalam menjaga konsep *stateless* pada *RESTful web service*. Metode autentikasi lainnya adalah *Seed Based Authentication* yang memiliki kelebihan dalam mengatasi permasalahan autentikasi yang dilakukan dengan *password* dan *username* yang statis. Oleh karena itu, perancangan proses autentikasi pada *RESTful web service* akan menggabungkan kedua metode ini.



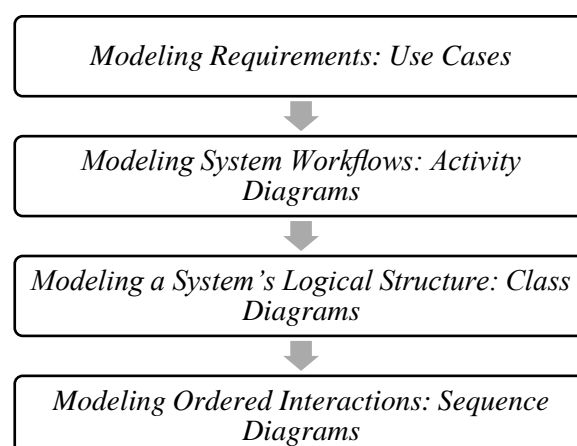
Gambar IV.1 *Seed Based Authentication* pada *RESTful web service*

Proses autentikasi dengan *Token Based Authentication* diilustrasikan pada Gambar II.4. Untuk memperoleh access token, client tetap menyediakan ID dan *password*. Perancangan dilakukan dengan mengganti ID dan *password* dengan proses autentikasi pada *Seed Based Authentication*. Gambar IV.1 mengilustrasikan *Seed*

Based Authentication pada *RESTful web service*. Proses autentikasi dilakukan antara suatu aplikasi, sebagai *client*, dan *service provider* yang terdiri dari *Authentication Server* (AS) dan *Resource Server* (RS). Setelah *client* melakukan permintaan *resource*, yaitu proses 1), *Resource Server* meminta proses autentikasi sehingga *client* dialihkan ke *Authentication Server* seperti pada proses 2). Selanjutnya autentikasi dilakukan, yaitu pada proses 3) s.d. 6), hingga masing-masing *client* dan *Resource Server* memperoleh *access token* yang akan digunakan untuk mengakses *resource* oleh *client*. Perancangan proses autentikasi yang lebih rinci dijelaskan pada bagian selanjutnya yang mengikuti tahapan pada *Seed Based Authentication*, yaitu tahap inisialisasi, identifikasi, autentikasi, dan sinkronisasi.

IV.1 Perancangan Proses Autentikasi

Perancangan proses autentikasi digambarkan menggunakan perangkat *Unified Modeling Language* (UML). UML merupakan standar bahasa untuk pemodelan pengembangan sistem dan *software* (Miles dan Hamilton, 2006). Model merupakan abstraksi dari sistem nyata yang menggambarkan komponen-komponen sistem beserta fungsinya. Dengan dibuatnya model, sebuah sistem dapat dipahami dan dievaluasi dengan lebih mudah karena sistem tersebut digambarkan dalam bentuk yang lebih sederhana. Bahasa pemodelan dapat berupa *pseudo-code*, *actual code*, gambar, diagram, atau paragraf deskripsi. Terdapat 4 (empat) diagram pada UML yang digunakan untuk menggambarkan perancangan yaitu *Use Case Diagram*, *Activity Diagram*, *Class Diagram*, dan *Sequence Diagram*.



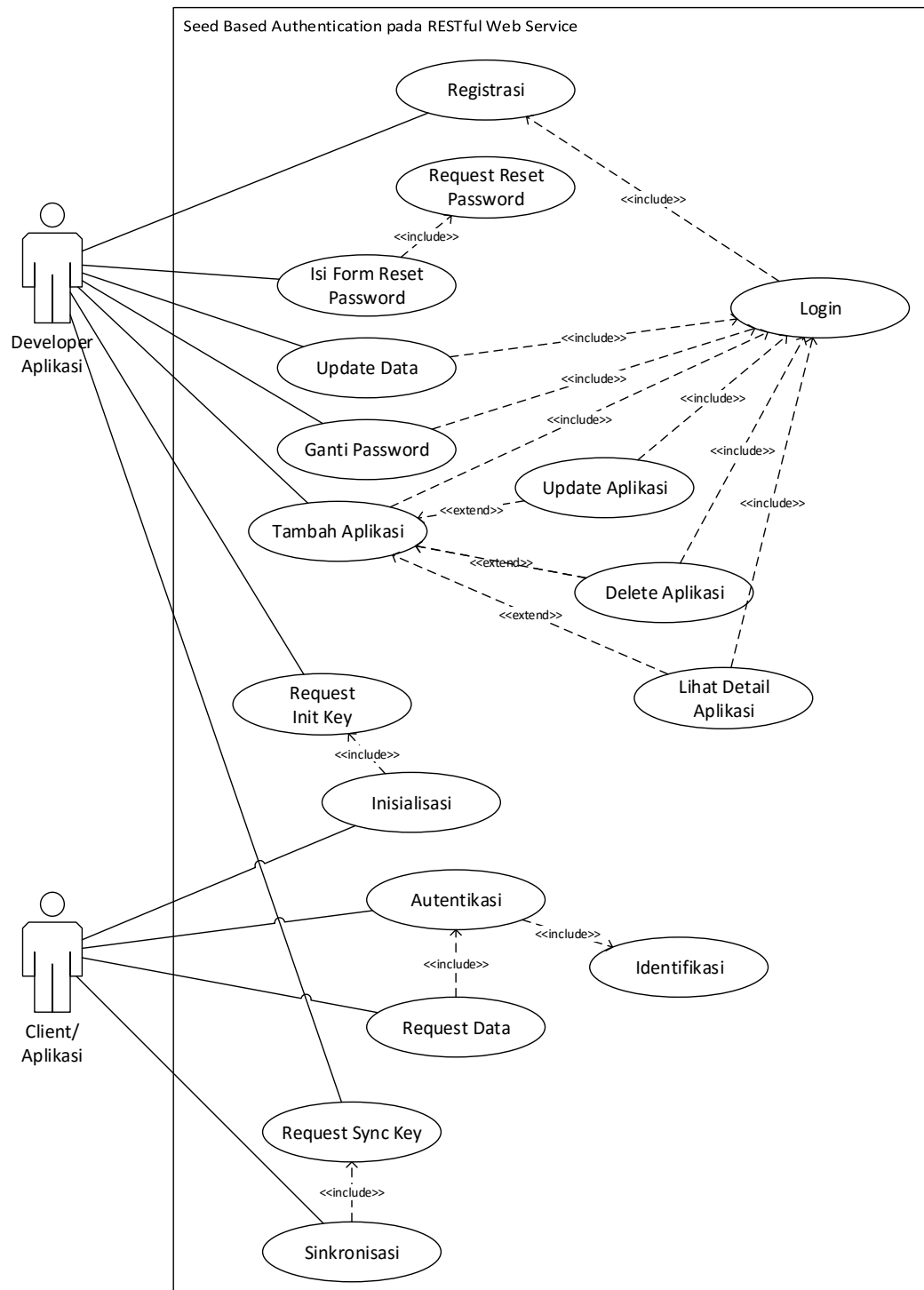
Gambar IV.2. Diagram pemodelan UML

IV.1.1 Use Case Diagram

Use case menggambarkan persyaratan fungsional sistem dan merupakan keluaran pertama dari sebuah perancangan model. *Use case* menunjukkan apa yang seharusnya sistem lakukan dari sudut pandang luar sistem. *Use case* dapat membantu dalam menyusun *test case* dan prosedur pengujian sistem karena *use case* memuat persyaratan dan kriteria sukses sistem.

Dari sudut pandang luar sistem, terdapat dua *actor* yang berhubungan dengan sistem yaitu *developer* aplikasi dan client/aplikasi. *Developer* aplikasi berinteraksi dengan sistem untuk mendaftarkan dan mengelola aplikasinya, yaitu melakukan *update* aplikasi, menghapus aplikasi, dan melihat detail aplikasi. Untuk melakukan fungsi ini, *developer* aplikasi diharuskan untuk *login* ke dalam sistem dengan terlebih dahulu dilakukan registrasi *developer* aplikasi tersebut ke dalam sistem. Apabila *developer* aplikasi lupa terhadap informasi passwordnya maka dapat mengisi *form reset password* dengan terlebih dahulu melakukan *request reset password*. Selain itu, *developer* aplikasi juga dapat melakukan pengelolaan data dirinya dan penggantian *password*.

Actor kedua adalah client/aplikasi yang berinteraksi dengan sistem dengan melakukan fungsi inisialisasi yang dapat dilakukan setelah *actor developer* aplikasi melakukan *request initialization key*. Selain itu, *actor client* berinteraksi dengan sistem dengan melakukan fungsi permintaan *resource*. Fungsi ini dapat dilakukan setelah *actor client* melakukan fungsi autentikasi dengan terlebih dahulu melakukan fungsi identifikasi. Terakhir, *actor client* dapat melakukan fungsi sinkronisasi yang dapat dilakukan setelah *actor developer* aplikasi melakukan fungsi permintaan *synchronization key*. *Use case* diagram dari persyaratan ini dapat dilihat pada Gambar IV.3.



Gambar IV.3. Use Case Diagram

IV.1.2 Activity Diagram

Activity diagram menentukan bagaimana sistem menyelesaikan fungsi-fungsinya. *Activity diagram* dibuat dengan menggambarkan setiap aksi yang saling terkait untuk mencapai tujuan sistem. Serangkaian aksi tersebut disebut juga dengan proses

bisnis. *Activity diagram* adalah salah satu diagram UML yang mudah dipahami karena digambarkan dengan notasi yang mirip dengan notasi *flowchart* sehingga berguna untuk mendeskripsikan proses kepada khalayak yang lebih luas. Berikut ini adalah *activity diagram* untuk setiap tahap pada *Seed Based Authentication* untuk *RESTful web service*, yaitu tahap inisialisasi, identifikasi, autentikasi, dan sinkronisasi.

1. *Activity diagram* untuk tahap inisialisasi

Tahapan inisialisasi dilakukan untuk mendaftarkan aplikasi ke dalam sistem. Proses diawali dengan registrasi *developer* aplikasi ke dalam sistem. Setelah itu, *developer login* ke dalam sistem untuk mendaftarkan aplikasinya. Satu *developer* dapat mendaftarkan satu atau lebih aplikasinya. Di dalam sistem, *developer* aplikasi dapat melakukan pengelolaan datanya, seperti email, *username*, dan *password*, dan data aplikasinya.

Proses autentikasi akan dilakukan untuk setiap aplikasi yang mengakses *resource* pada *Resource Server*. Pada saat mendaftarkan aplikasinya, *developer* melakukan upload *root file* ke *Authentication Server* untuk selanjutnya digunakan dalam proses autentikasi. Setelah melakukan registrasi aplikasi, *developer* aplikasi akan mendapatkan *initialization key* yang akan digunakan saat aplikasinya melakukan *request* inisialisasi ke *Authentication Server*.

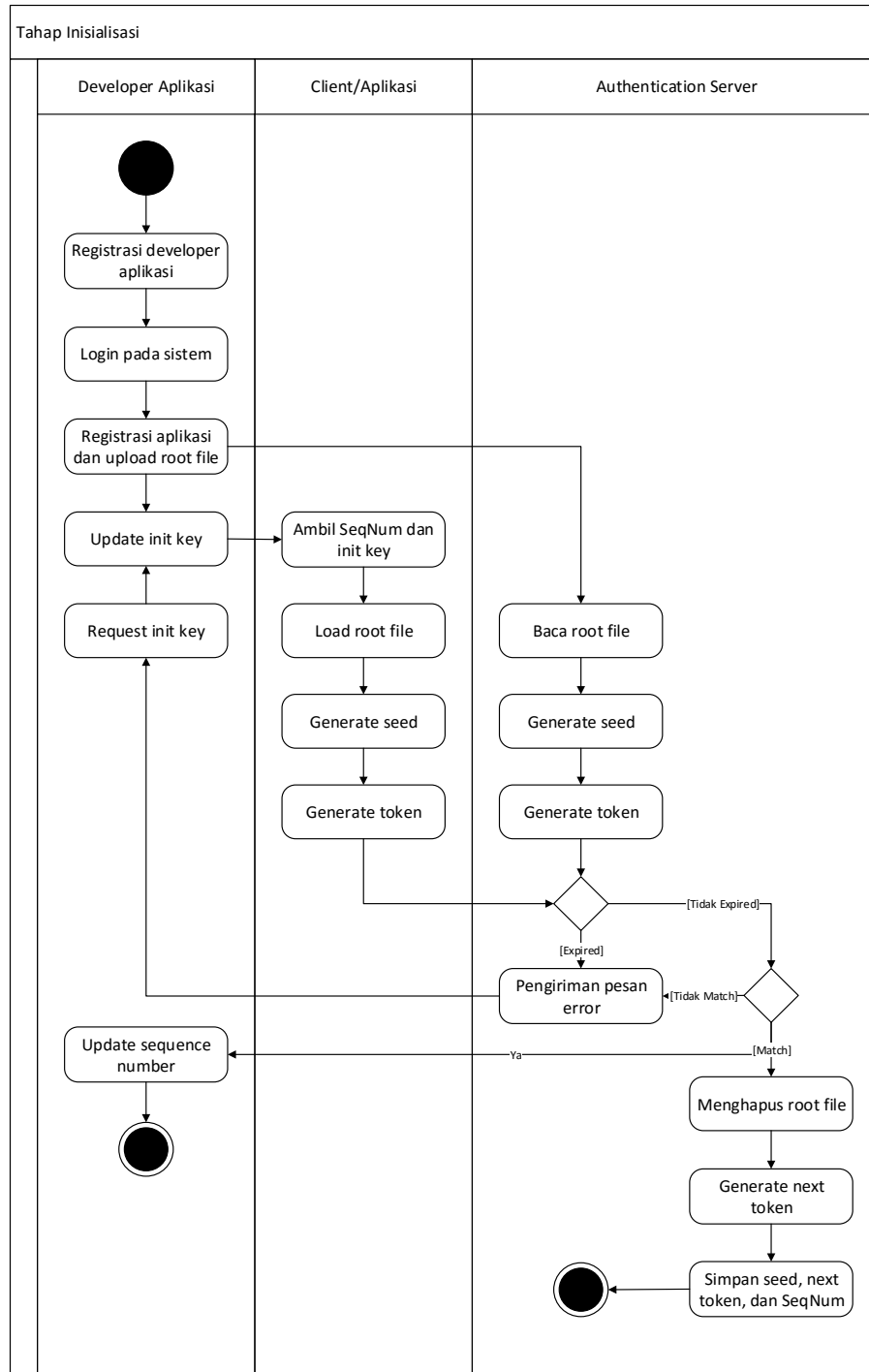
Setelah itu, *developer* aplikasi memasang modul autentikasi pada aplikasinya. *Root file* disimpan pada tempat yang dapat diakses oleh aplikasi tersebut. Sebelum dapat melakukan autentikasi, pengujian perlu dilakukan apakah modul autentikasi telah dipasang dengan benar dan dapat berfungsi dengan baik.

Proses pengujian dilakukan dengan masing-masing aplikasi dan *Authentication Server* membangkitkan dua *seed*, URLSeed dan UNMSeed, dari *root file* dengan menggunakan fungsi yang sama. Setelah itu, kedua *seed* tersebut digunakan untuk membangkitkan token, yaitu pada *sequence number* 0. Token pada aplikasi, sebagai client, dihitung dengan $T_{url}^C(n)$ dan $T_{unm}^C(n)$ seperti pada persamaan (II.2) dan (II.4), sedangkan pada *Authentication Server*

dihitung dengan $T_{url}^{AS}(n)$ dan $T_{unm}^{AS}(n)$ seperti pada persamaan (IV.1) dan (IV.2).

$$T_{url}^{AS}(n) = P(S_{url}^{AS}(n)) \quad (IV.1)$$

$$T_{unm}^{AS}(n) = P(S_{unm}^{AS}(n)) \quad (IV.2)$$



Gambar IV.4. Activity Diagram pada tahap inisialisasi

Proses ini dilanjutkan dengan aplikasi mengirimkan $T^C_{url}(0)$ dan $T^C_{unm}(0)$ ke *Authentication Server*. Setelah itu, token tersebut dibandingkan dengan token pada *Authentication Server*, $T^{AS}_{url}(0)$ dan $T^{AS}_{unm}(0)$. Jika hasil perbandingan menunjukkan bahwa token sesuai, maka proses inisialisasi berhasil dilakukan.

Selanjutnya, *Authentication Server* mengirimkan pesan sukses ke client dan client menyimpan *sequence number*. Sementara itu, *Authentication Server* menghapus *root file* dan hanya akan menyimpan kedua *seed*, UNMSeed dan URLSeed, yang dibangkitkan dari *root file* tersebut. Selain *seed*, *Authentication Server* juga menyimpan token pada *sequence number* berikutnya ($n + 1$) untuk digunakan dalam proses autentikasi berikutnya. *Activity diagram* untuk tahap inisialisasi diberikan pada Gambar IV.4.

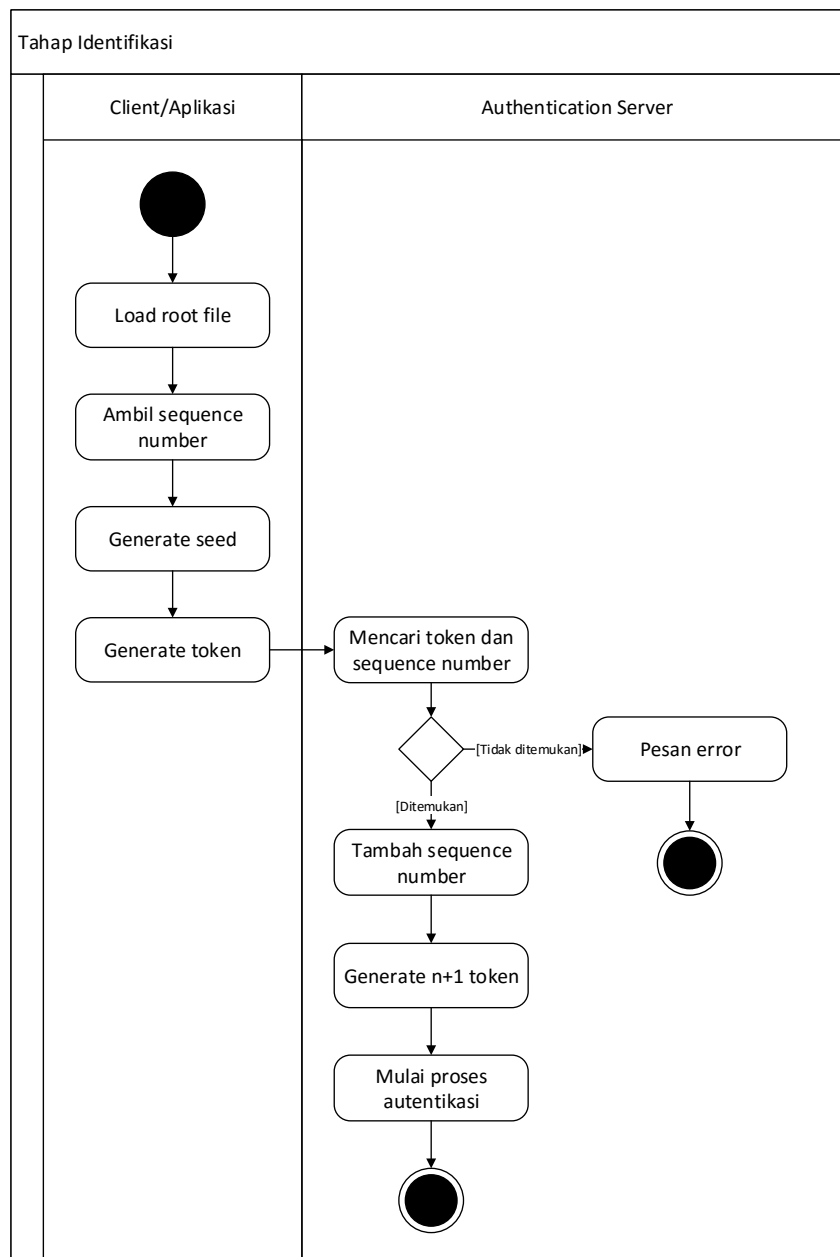
2. *Activity diagram* untuk tahap identifikasi

Tahap identifikasi dilakukan saat client melakukan permintaan *resource* ke *Resource Server*. Proses identifikasi dan autentikasi diperlukan jika *access token* yang disertakan tidak sesuai atau telah habis masa berlakunya. Client akan dialihkan ke *Authentication Server* untuk melakukan proses identifikasi dan autentikasi. Proses ini dapat dilihat pada Gambar IV.1, proses 1) s.d. 3).

Proses identifikasi diawali dengan client mengirimkan URLToken dan UNMToken yang dibangkitkan dengan menggunakan fungsi PRNG dengan input masing-masing *seed* dan *sequence number* yang berlaku, yaitu $T^C_{url}(n)$ dan $T^C_{unm}(n)$. Setelah itu, kedua token dikirim ke *Authentication Server* melalui POST request, $www.domain.com/uri/T^C_{url}(n)/T^C_{unm}(n)/n$. Proses ini berhasil jika *Authentication Server* dapat menemukan pasangan token sesuai dengan token dan *sequence number* yang dikirimkan seperti pada persamaan (IV.3).

$$\{T^C_{url}(n), T^C_{unm}(n)\} == \{T^{AS}_{url}(n), T^{AS}_{unm}(n)\} \quad (IV.3)$$

Jika tidak dapat menemukan token yang dimaksud maka *Authentication Server* mengirimkan pesan *error*. Jika proses ini berhasil maka client dan *Authentication Server* akan menambahkan *sequence number* menjadi $n + 1$. Kemudian *Authentication Server* membangkitkan token dengan menggunakan *sequence number* tersebut untuk digunakan dalam proses identifikasi berikutnya. *Activity diagram* pada Gambar IV.5 menggambarkan secara rinci proses identifikasi yang dilakukan antara client dan AS.



Gambar IV.5. *Activity Diagram* pada tahap identifikasi

3. *Activity diagram* untuk tahap autentikasi

Proses autentikasi dilakukan setelah tahap identifikasi berhasil dilakukan antara client dan *Authentication Server* yang diilustrasikan pada Gambar IV.1, proses 4) s.d. 6). Proses ini diawali dengan *Authentication Server* mengirimkan empat *sequence number* yang acak dan berbeda kepada client yaitu $\{x, y, u, v\}$. Client diminta untuk menyediakan nilai *hash* dari empat token yang dibangkitkan dari *sequence number* tersebut dalam waktu yang ditentukan oleh *Authentication Server*. Empat token tersebut adalah URLToken untuk *sequence number* x dan y dan UNMToken untuk *sequence number* u dan v . Kemudian keempat token tersebut digabung untuk dilakukan penghitungan nilai *hash* dengan menggunakan fungsi h seperti pada persamaan (II.7). Setelah itu client mengirimkan H^C ke *Authentication Server* yang kemudian akan dibandingkan dengan nilai *hash* yang dihitung oleh *Authentication Server* menggunakan seed dan *sequence number* yang sama.

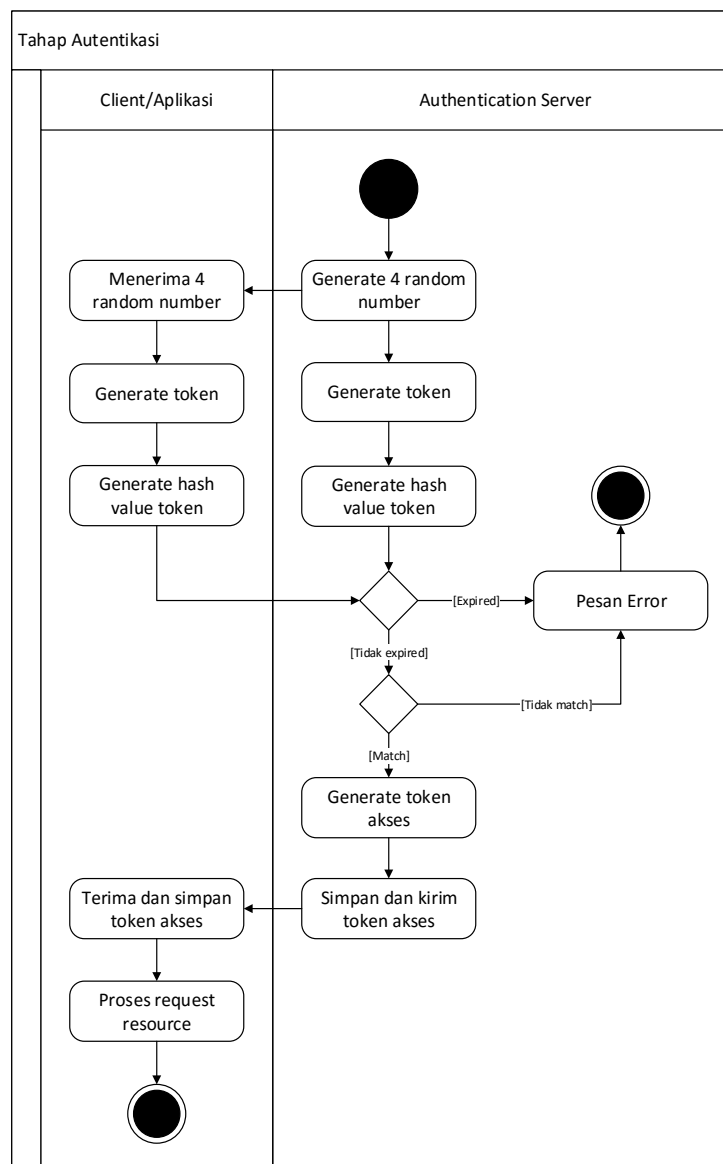
$$\begin{aligned} h(T_{u,u}^{AS}(x), T_{u,u}^{AS}(y), T_m^{AS}(u), T_m^{AS}(v)) = \\ h(T_u^C(x), T_u^C(y), T_m^C(u), T_m^C(v)) \end{aligned} \quad (IV.4)$$

Proses autentikasi berhasil jika nilai *hash* yang dibandingkan sesuai, seperti pada persamaan (IV.4). Kemudian *Authentication Server* mengirimkan *access token* kepada client dan *Resource Server* yang akan digunakan dalam komunikasi antara keduanya. Apabila client tidak melakukan respons pada waktu yang telah ditentukan, maka *Authentication Server* akan membatalkan proses autentikasi dan client harus melakukan proses autentikasi dari awal. *Activity diagram* pada tahapan autentikasi diberikan pada Gambar IV.6.

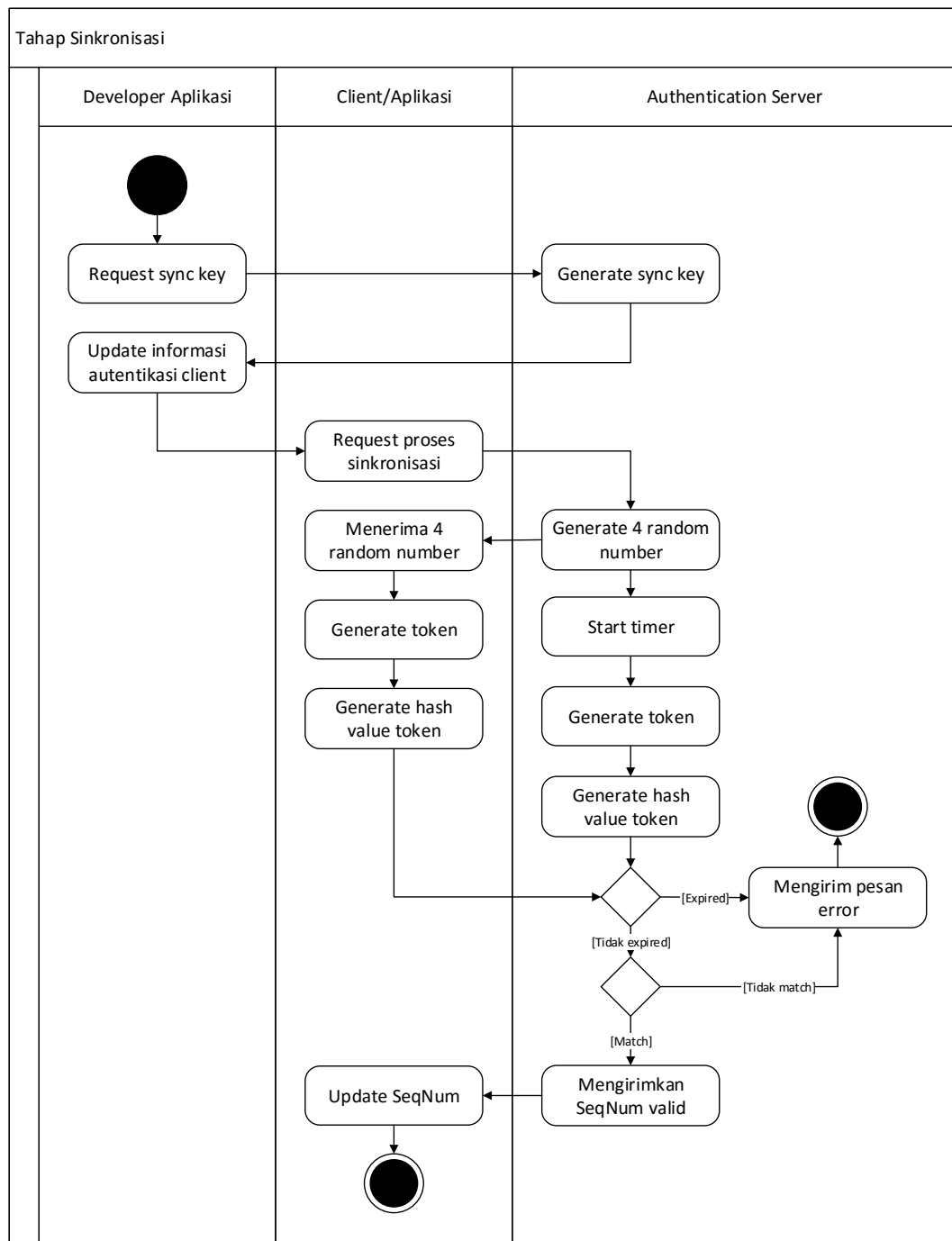
4. *Activity diagram* untuk tahap sinkronisasi

Untuk melakukan sinkronisasi, *developer* aplikasi melakukan *request synchronization key*. Kemudian, *Authentication Server* akan membangkitkan *synchronization key* dan menampilkannya pada aplikasi administrator. Setelah itu *developer* aplikasi melakukan *update synchronization key* pada aplikasinya. Selanjutnya, client melakukan *sync request* dengan menyediakan

synchronization key, *sequence number*, dan tokennya, $\{T_{url}^C(n), T_{unm}^C(n), n\}$. Selanjutnya, *Authentication Server* memastikan client terautentikasi dengan mengirimkan empat *random number* seperti pada proses autentikasi. Client kemudian menyediakan *hash* dari empat token yang dibangkitkan dari empat *sequence number* tersebut dan *Authentication Server* membandingkan nilai *hash* client dengan nilai *hash* yang dihitung oleh *Authentication Server*. Apabila proses ini berhasil maka *Authentication Server* akan mengirimkan *sequence number* yang valid untuk digunakan pada proses identifikasi. Gambar IV.7 adalah *activity diagram* pada tahap sinkronisasi.



Gambar IV.6. *Activity Diagram* pada tahap Autentikasi



Gambar IV.7. Activity Diagram pada tahap sinkronisasi

IV.1.3 Class Diagram

Class merupakan bagian penting dari sistem berorientasi objek. Sebuah sistem terdiri dari kumpulan berbagai jenis objek yang dideskripsikan oleh sebuah *class*. *Class diagram* menggambarkan *class* dan hubungan antar *class* tersebut. *Use case* mendeskripsikan sistem sebagai kumpulan persyaratan sistem, sedangkan *class* mendeskripsikan berbagai jenis objek yang diperlukan sistem untuk memenuhi persyaratan tersebut.

IV.1.3.1 Authentication Server

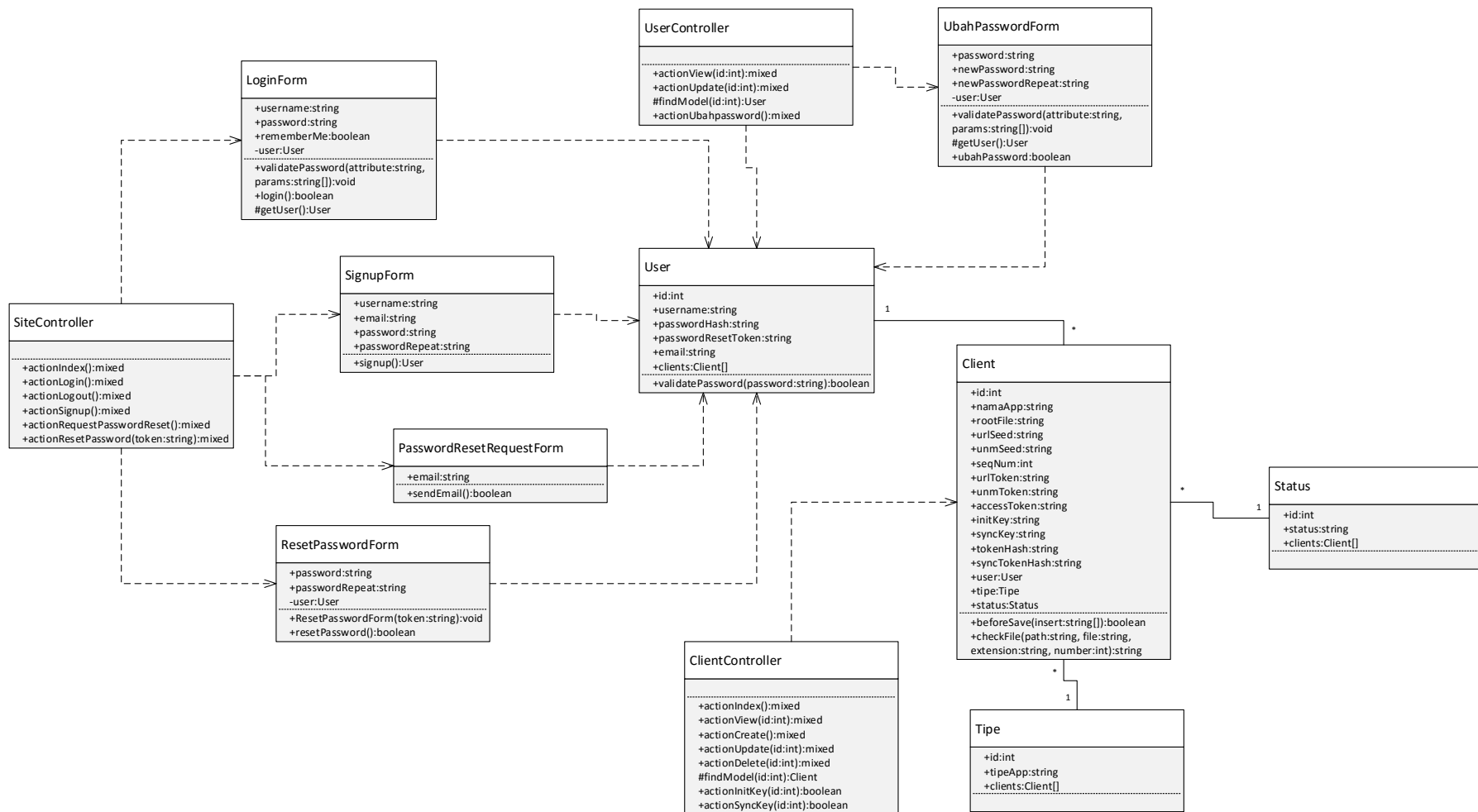
Authentication Server memiliki dua aplikasi, yaitu aplikasi administrator dan *authenticator*. *Class diagram* untuk aplikasi administrator diberikan pada Gambar IV.8. Penjelasan *class diagram* tersebut adalah sebagai berikut.

1. Class SiteController

Class SiteController memiliki 6 (enam) *method* yang menangani *request* dari *developer* aplikasi, yaitu *method* *actionIndex()*, *actionLogin()*, *actionLogout()*, *actionSignup()*, *actionRequestPasswordReset()*, dan *actionResetPassword()*. *Method* *actionIndex()* berfungsi dalam menampilkan halaman beranda aplikasi administrator. *Method* *actionLogin()* menampilkan *form login*. *Method* ini menggunakan *class* *LoginForm*. *Method* *actionLogout()* melakukan proses *logout developer* aplikasi dari sistem. *Method* *actionSignup()* menampilkan *form* registrasi *developer*. *Method* *actionRequestPasswordReset()* mengirimkan token ke email *developer* untuk proses *reset password*. *Method* *actionResetPassword()* memiliki parameter *string* token yang berfungsi untuk menampilkan *form reset password*.

2. Class SignupForm

Atribut dalam *class* *SignupForm* yaitu *username*, *email*, *password*, dan *passwordRepeat*. *Class* ini memiliki *method* *signUp()* yang berfungsi dalam memvalidasi dan menyimpan data *developer* saat melakukan registrasi.



Gambar IV.8. Class diagram aplikasi administrator pada Authenticator Server

3. Class LoginForm

Class LoginForm memiliki 3 (tiga) atribut utama yaitu *username*, *password*, dan *user*. *Method* pada *class* ini yaitu *validatePassword()* untuk memvalidasi *password* yang dimasukkan oleh *developer* aplikasi saat *login*, *login()* untuk proses *login*, dan *getUser()* untuk mendapatkan data *developer* sesuai dengan *username* yang dimasukkan saat proses *login*.

4. Class PasswordResetRequestForm

Class ini memiliki satu atribut, yaitu *email*. *Method* pada *class* ini adalah *sendEmail()*. *Method* ini berfungsi dalam menghasilkan token dan mengirimkannya ke email *developer* untuk proses *reset password*.

5. Class ResetPasswordForm

Class ResetPasswordForm memiliki atribut *password*, *passwordRepeat*, dan *user*. *Method* pada *class* ini adalah *resetPassword()* yang berfungsi dalam mengganti *password* lama dengan *password* yang dimasukkan dalam *form reset password*.

6. Class UserController

Class UserController memiliki empat *method* yaitu *actionView()*, *actionUpdate()*, *findModel()*, dan *actionUbahpassword()*. *Method* *actionView()* berfungsi dalam menampilkan data *developer*. *Method* *actionUpdate()* menampilkan *form update* data *developer*. *Method* *findModel()* mengambil data *developer* dari *database*. *Method* *actionUbahpassword()* menampilkan *form* untuk mengubah *password developer*.

7. Class UbahPasswordForm

Class ini memiliki empat atribut yaitu *password*, *newPassword*, *newPasswordRepeat*, dan *user*. *Method* dalam *class* ini diantaranya *validatePassword()* yang berfungsi dalam memvalidasi *password developer* yang dimasukkan dalam *form ubah password*, *getUser()* yang mengambil data *developer* dari *database*, dan *ubahPassword()* yang mengubah *password* lama dengan *password* yang dimasukkan dalam *form ubah password*.

8. *Class User*

Class User adalah *class* yang mewakili data *developer*. *Class* ini memiliki atribut diantaranya *id*, *username*, *passwordHash*, *passwordResetToken*, dan *email*, dan *clients*.

9. *Class ClientController*

Class ClientController memiliki *method* diantaranya *actionIndex()*, *actionView()*, *actionCreate()*, *actionUpdate()*, *actionDelete()*, *findModel()*, *actionInitKey()*, *actionSyncKey()*. *Method actionIndex()* berfungsi dalam menampilkan aplikasi yang telah didaftar oleh *developer*. *Method actionView()* menampilkan detail dari aplikasi. *Method actionCreate()* menampilkan *form*, memvalidasi, dan menyimpan data aplikasi saat registrasi aplikasi oleh *developer*. *Method actionUpdate()* menampilkan *form update* aplikasi, memvalidasi dan menyimpan data perubahan aplikasi. *Method actionDelete()* menghapus data aplikasi, termasuk *root file*. *Method findModel()* mengambil data aplikasi dari *database*. *Method actionInitKey()* membangkitkan *initialization key* untuk proses inisialisasi aplikasi. *Method actionSyncKey()* membangkitkan *synchronization key* untuk proses sinkronisasi aplikasi.

10. *Class Client*

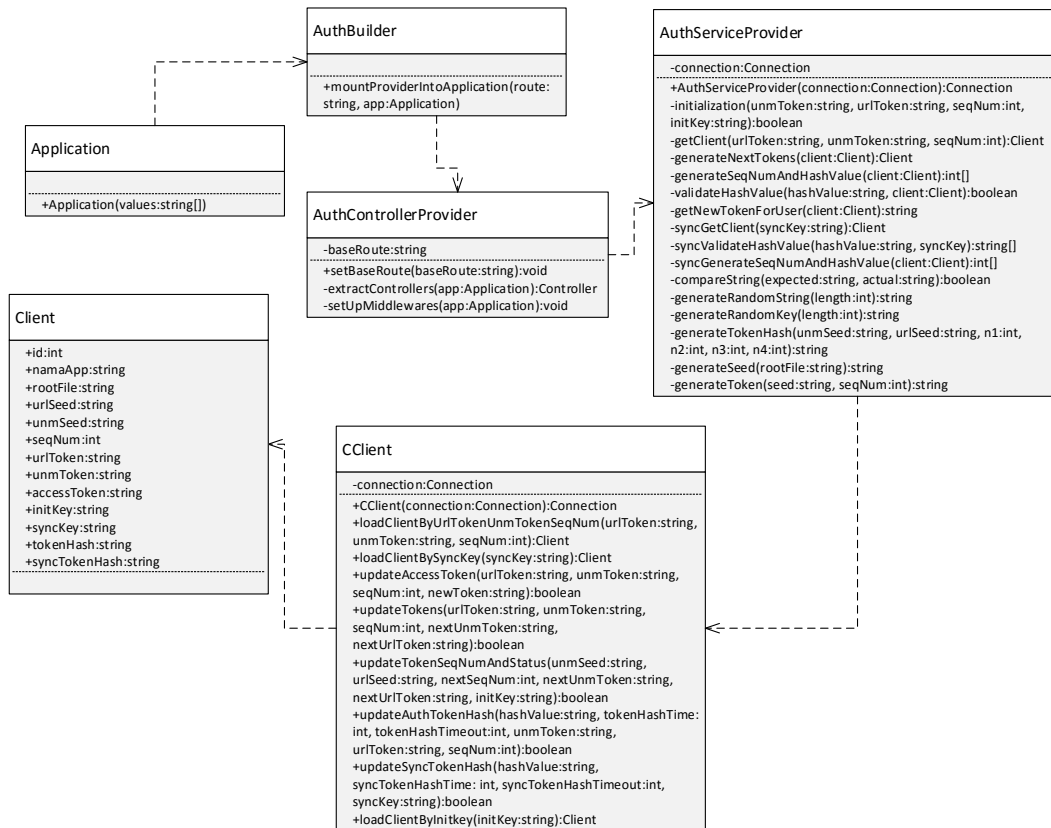
Class Client adalah *class* yang mewakili data aplikasi. *Class* ini memiliki atribut *id*, *namaApp*, *rootFile*, *urlSeed*, *unmSeed*, *seqNum*, *urlToken*, *unmToken*, *accessToken*, *initKey*, *syncKey*, *tokenHash*, *syncTokenHash*, *user*, *tipe*, dan *status*.

11. *Class Status*

Class Status mewakili data status aplikasi. *Class* ini memiliki atribut diantaranya *id*, *status*, dan *clients*.

12. *Class Tipe*

Class Tipe mewakili data tipe aplikasi. *Class* ini memiliki atribut diantaranya *id*, *tipeApp*, dan *clients*.



Gambar IV.9. Class Diagram aplikasi authenticater pada Authenticater Server

Class diagram untuk aplikasi authenticater diberikan pada Gambar IV.9. Penjelasan class diagram tersebut adalah sebagai berikut.

1. Class Application

Class Application merupakan class utama dari aplikasi authenticater. Class ini menggunakan Class AuthBuilder.

2. Class AuthBuilder

Class AuthBuilder memiliki method mountProviderIntoApplication(). Method ini berfungsi dalam meregistrasi route untuk proses autentikasi.

3. Class AuthControllerProvider

Class AuthControllerProvider memiliki satu atribut yaitu baseRoute. Method di dalam class ini diantaranya setBaseRoute(), extractControllers(), dan setUpMiddlewares().

4. *Class AuthServiceProvider*

Atribut pada *class AuthServiceProvider* adalah *conn* yang menyimpan informasi koneksi *database*. *Class* ini memiliki beberapa *method* yaitu *initialization()*, *getClient()*, *generateNextTokens()*, *generateSeqNumAndHashValue()*, *validateHashValue()*, *getNewTokenForUser()*, *syncGetClient()*, *syncValidateHashValue()*, *syncGenerateSeqNumAndHashValue()*, *compareString()*, *generateRandomString()*, *generateRandomKey()*, *generateTokenHash()*, *generateSeed()*, dan *generateToken()*.

5. *Class Client*

Class Client adalah *class* yang mewakili aplikasi dari *developer*. *Class* ini memiliki beberapa atribut, diantaranya *id*, *namaApp*, *rootFile*, *urlSeed*, *unmSeed*, *seqNum*, *urlToken*, *unmToken*, *accessToken*, *initKey*, *syncKey*, *tokenHash*, *syncTokenHash*, *user*, *tipe*, dan *status*.

6. *Class CClient*

Class CClient memiliki atribut *conn* untuk menyimpan informasi koneksi ke *database*. *Class* ini berisi operasi-operasi untuk memanipulasi data aplikasi. *Method* yang terdapat pada *class* ini diantaranya *loadClientByUrlTokenUnmTokenSeqNum()*, *loadClientBySyncKey()*, *updateAccessToken()*, *updateTokens()*, *updateTokenSeqNumAndStatus()*, *updateAuthTokenHash()*, *updateSyncTokenHash()*, dan *loadClientByInitkey()*.

IV.1.3.2 *Resource Server*

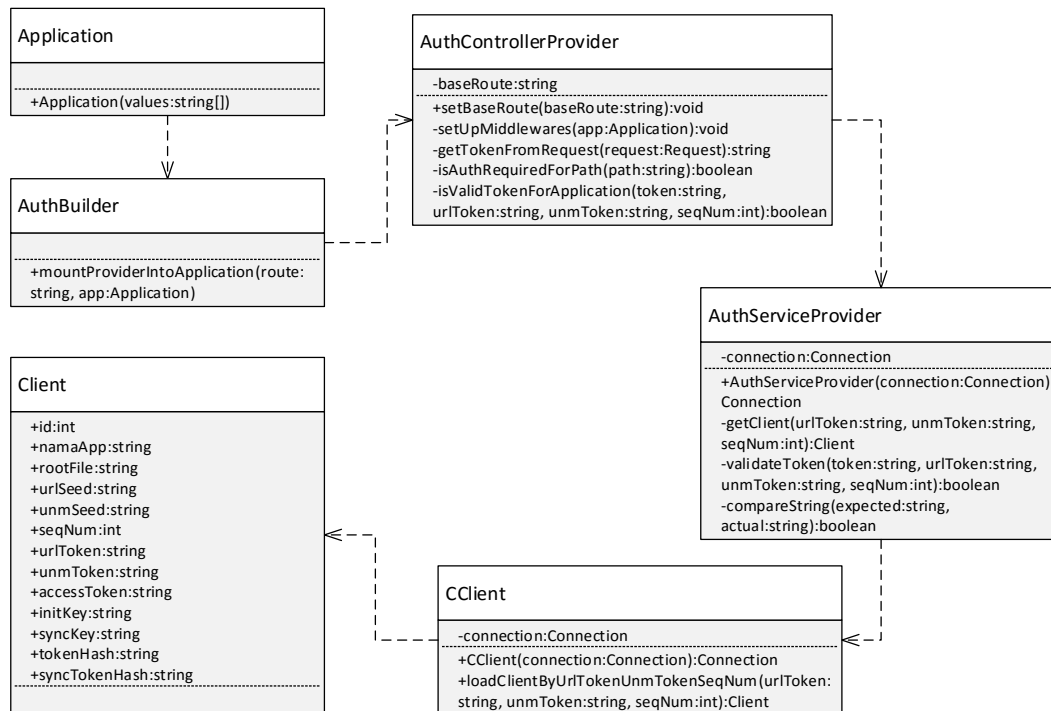
Class diagram untuk *Resource Server* diberikan pada Gambar IV.10. Penjelasan *class diagram* tersebut adalah sebagai berikut.

1. *Class Application*

Class utama dari aplikasi *Resource Server* adalah *class Application*. *Class* ini menggunakan *class AuthBuilder*.

2. *Class AuthBuilder*

Class AuthBuilder tidak memiliki atribut. *Class* ini memiliki satu *method* yaitu *mountProviderIntoApplication()* yang berfungsi dalam mendaftarkan *route* untuk proses autentikasi.



Gambar IV.10. *Class Diagram pada Resource Server*

3. Class AuthControllerProvider

Class AuthControllerProvider memiliki satu atribut, yaitu baseRoute. Method yang terdapat pada class ini diantaranya setBaseRoute(), setUpMiddlewares(), getTokenFromRequest(), isAuthRequiredForPath(), dan isValidTokenForApplication().

4. Class AuthServiceProvider

Class AuthServiceProvider memiliki atribut conn yang menyimpan informasi koneksi ke database. Class ini memiliki beberapa method, diantaranya getClient(), validateToken(), dan compareString(). Method getClient() berfungsi untuk mengambil data aplikasi dari database. Method validateToken() berfungsi untuk memvalidasi access token yang dikirim client. Method compareString() berfungsi untuk membandingkan token yang dikirim client dengan token di database Resource Server.

5. Class Client

Class Client adalah class yang mewakili aplikasi yang mengakses resource pada Resource Server. Class ini memiliki atribut diantaranya id, namaApp,

rootFile, urlSeed, unmSeed, seqNum, urlToken, unmToken, accessToken, initKey, syncKey, tokenHash, syncTokenHash, *user*, tipe, dan status..

6. *Class CClient*

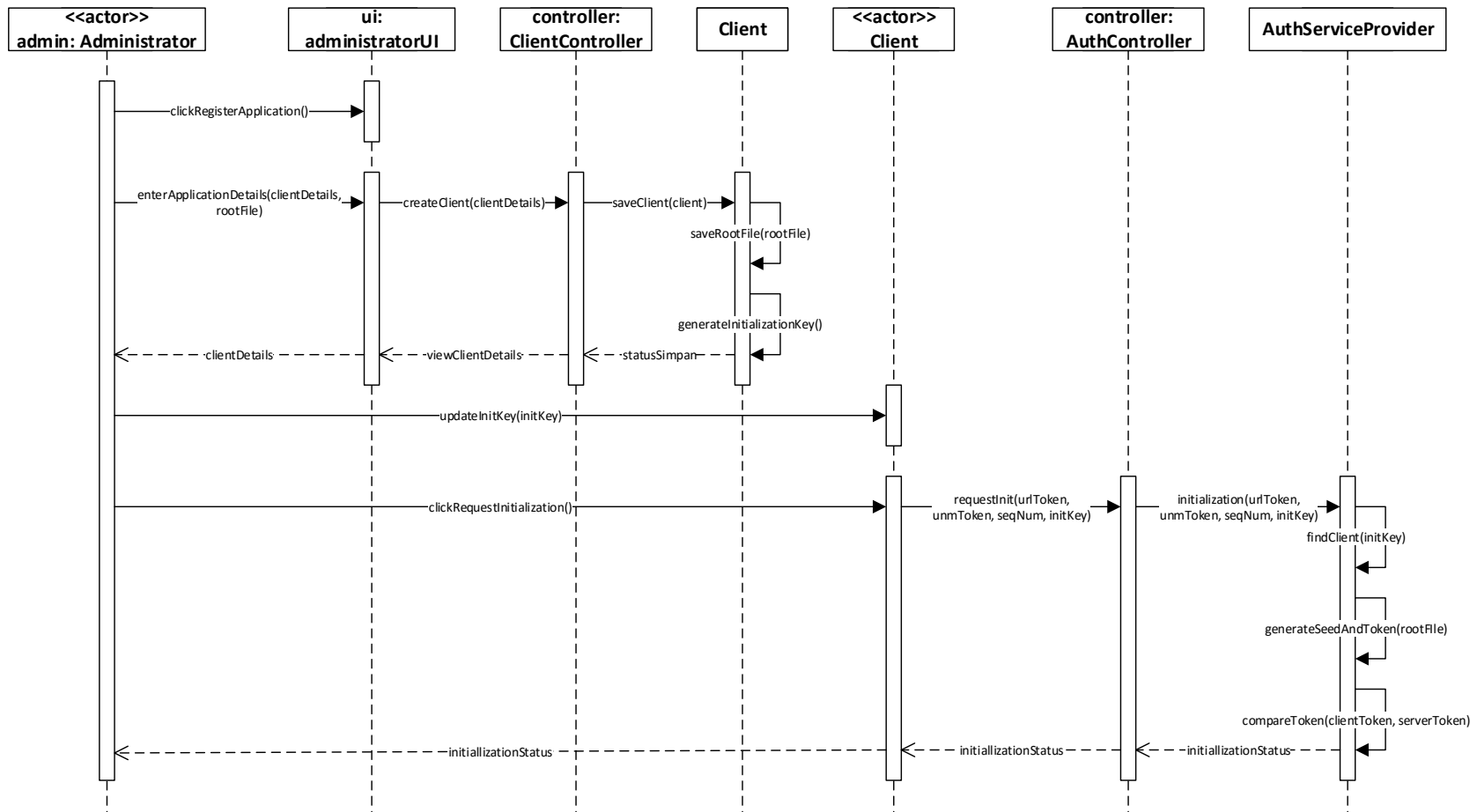
Class CClient berfungsi dalam melakukan manipulasi data aplikasi. *Class* ini memiliki satu atribut, yaitu *conn*, yang menyimpan informasi koneksi ke *database*. *Method* yang terdapat pada *class* adalah *loadClientByUrlTokenUnmTokenSeqNum()* yang berfungsi mengambil data aplikasi berdasarkan *UrlToken*, *UnmToken* dan *sequence number* yang dikirim oleh *client*.

IV.1.4 *Sequence Diagram*

Berbeda dengan *use case* diagram yang mendeskripsikan fungsi apa yang seharusnya dilakukan oleh sistem dan *class diagram* yang mendeskripsikan berbagai jenis bagian yang membentuk struktur sistem, *sequence diagram* mendeskripsikan bagaimana sebenarnya sistem melakukan fungsi-fungsi tersebut. Urutan interaksi antar bagian di dalam sistem digambarkan dengan *sequence diagram*. Saat suatu *use case* dieksekusi, interaksi mana yang menjadi pemicu dapat ditentukan dan bagaimana urutan interaksi tersebut terjadi.

Gambar IV.11 adalah *sequence diagram* pada tahap inisialisasi. Penjelasan *sequence diagram* tersebut adalah sebagai berikut.

- a. *Actor* eksternal Administrator mengklik tombol register aplikasi pada antarmuka aplikasi administrator.
- b. *Actor* eskternal Administrator memasukkan informasi detail aplikasi, termasuk memilih *root file* untuk proses autentikasi aplikasi.
- c. Informasi detail aplikasi diteruskan ke ClientController
- d. ClientController memanggil *method* *save()* pada Client untuk menyimpan informasi detail aplikasi. Client memanggil *method* *saveRootFile(rootFile)* untuk menyimpan *root file* dan *generateInitializationKey()* untuk membangkitkan *initialization key* yang akan digunakan oleh aplikasi pada proses inisialisasi.



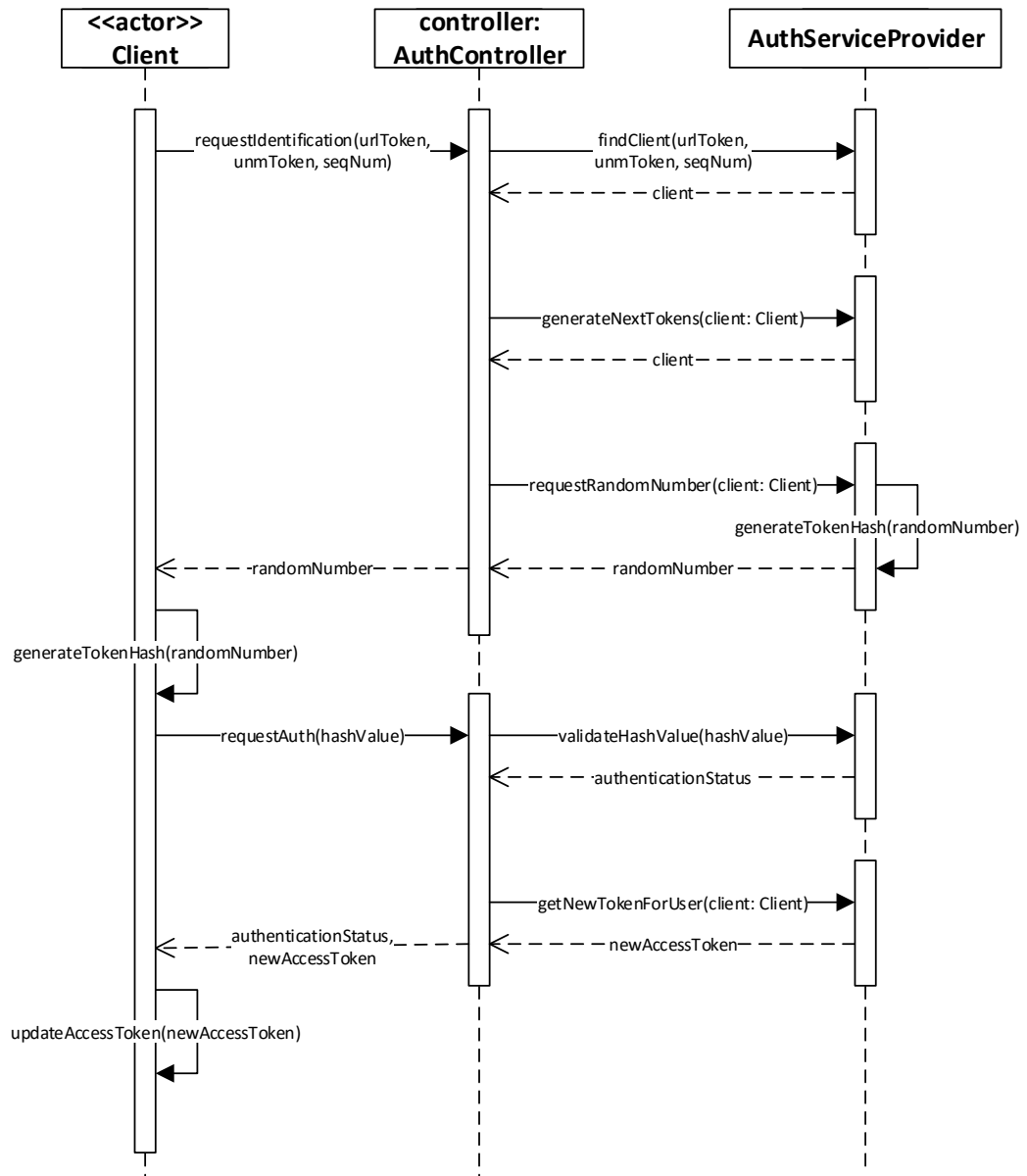
Gambar IV.11. *Sequence Diagram* pada tahap inisialisasi

- e. Client mengirimkan *statusSimpan* kepada *ClientController*. Kemudian *ClientController* mengarahkan *administratorUI* untuk menampilkan informasi detail aplikasi yang baru tersimpan. Setelah itu, *administratorUI* menampilkan detail aplikasi kepada *actor* eksternal *Administrator*.
- f. *Administrator* melakukan *update initialization key* pada aplikasinya, yaitu *actor* eksternal *Client*, untuk digunakan dalam proses inisialisasi.
- g. *Administrator* mengklik tombol *request* inisialisasi pada *Client*.
- h. Client mengirimkan *request* inisialisasi kepada *AuthController* dengan parameter *urlToken*, *unmToken*, *seqNum*, dan *initKey*.
- i. *AuthController* memanggil *method* *initialization()* pada *AuthServiceProvider* untuk melakukan proses inisialisasi *client* dengan mengirimkan *urlToken*, *unmToken*, *seqNum*, dan *initKey*.
- j. *AuthServiceProvider* memanggil *method* *findClient(initKey)* untuk mencari *Client* berdasarkan *initialization key*, *generateSeedAndToken(rootFile)* untuk mengekstrak *seed* dari *root file* dan membangkitkan *urlToken* dan *unmToken*, dan *compareToken(clientToken, serverToken)* untuk membandingkan token yang dikirim *client* dengan token yang dibangkitkan oleh *server*.
- k. *AuthServiceProvider* mengirimkan status inisialisasi ke *AuthController* dan kemudian diteruskan ke *Client*. Setelah itu, *Client* menampilkan status inisialisasi kepada *Administrator*.

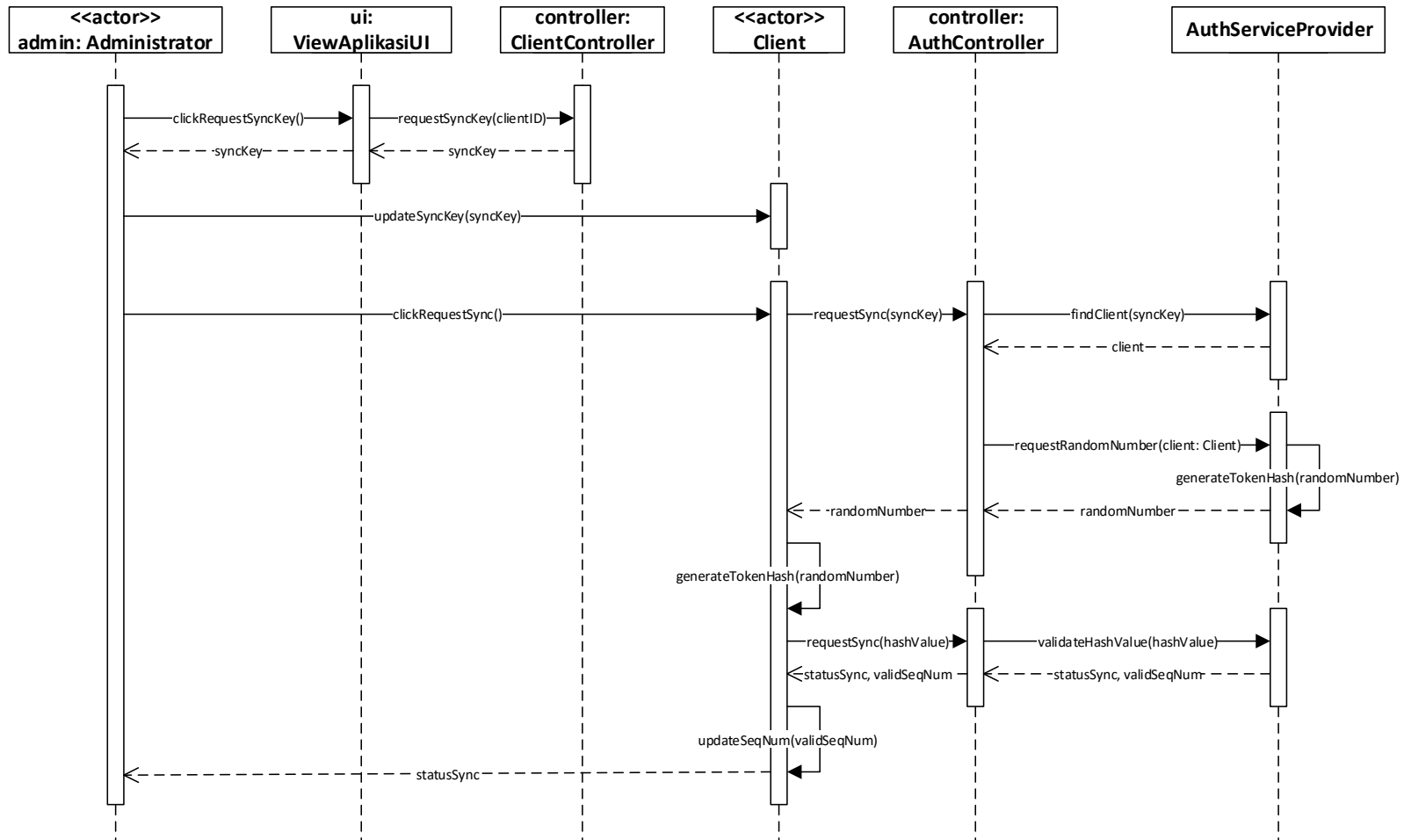
Gambar IV.12 adalah *sequence diagram* untuk tahap identifikasi dan autentikasi. Penjelasan *sequence diagram* tersebut adalah sebagai berikut.

- a. *Actor* eksternal *Client* melakukan *request* identifikasi kepada *AuthController* dengan mengirimkan parameter *urlToken*, *unmToken*, dan *seqNum*.
- b. *AuthController* memanggil *method* *findClient()* pada *AuthServiceProvider* untuk mencari *Client* berdasarkan *urlToken*, *unmToken*, dan *seqNum*.
- c. *AuthServiceProvider* mengirimkan objek *client* kepada *AuthController* sebagai *return value* dari *method* *findClient()*.
- d. *AuthController* memanggil *method* *generateNextToken()* dengan parameter *client* untuk membangkitkan *urlToken* dan *unmToken* yang akan digunakan pada proses identifikasi berikutnya.

- e. AuthServiceProvider mengirimkan objek *client* kepada AuthController sebagai *return value* dari *method* generateNextToken().
- f. AuthController memanggil *method* requestRandomNumber() dengan parameter objek *client* untuk membangkitkan *random number* yang akan digunakan dalam proses autentikasi. AuthController memanggil *method*-nya sendiri, yaitu generateTokenHash() dengan parameter *random number* untuk menghasilkan nilai *hash* dari token yang dibangkitkan dari *random number*.
- g. *Random number* dikirim kepada AuthController sebagai *return value* dari *method* requestRandomNumber() dan kemudian diteruskan kepada Client.
- h. Client menghitung nilai *hash* dari token yang dibangkitkan dari *random number* yang diterima. Client melakukan request autentikasi dengan mengirimkan nilai *hash* tersebut.
- i. Permintaan autentikasi diterima oleh AuthController. Kemudian AuthController memanggil *method* validateHashValue() dengan parameter nilai *hash* dari Client. *Method* ini berfungsi untuk membandingkan nilai *hash* pada *client* dengan nilai *hash* pada *server*.
- j. AuthServiceProvider mengirim status autentikasi *client* kepada AuthController sebagai *return value* dari *method* validateHashValue().
- k. AuthController memanggil *method* getNewTokenForUser() pada AuthServiceProvider untuk membangkitkan *access token* setelah proses autentikasi berhasil dilakukan.
- l. AuthServiceProvider mengirimkan *access token* kepada AuthController sebagai *return value* dari *method* getNewTokenForUser() dan meneruskannya kepada Client.
- m. Client memperbaharui data *access token* untuk digunakan dalam proses permintaan *resource* ke *Resource Server*.



Gambar IV.12. *Sequence diagram* pada tahap identifikasi dan autentikasi



Gambar IV.13. *Sequence Diagram* pada tahap sinkronisasi

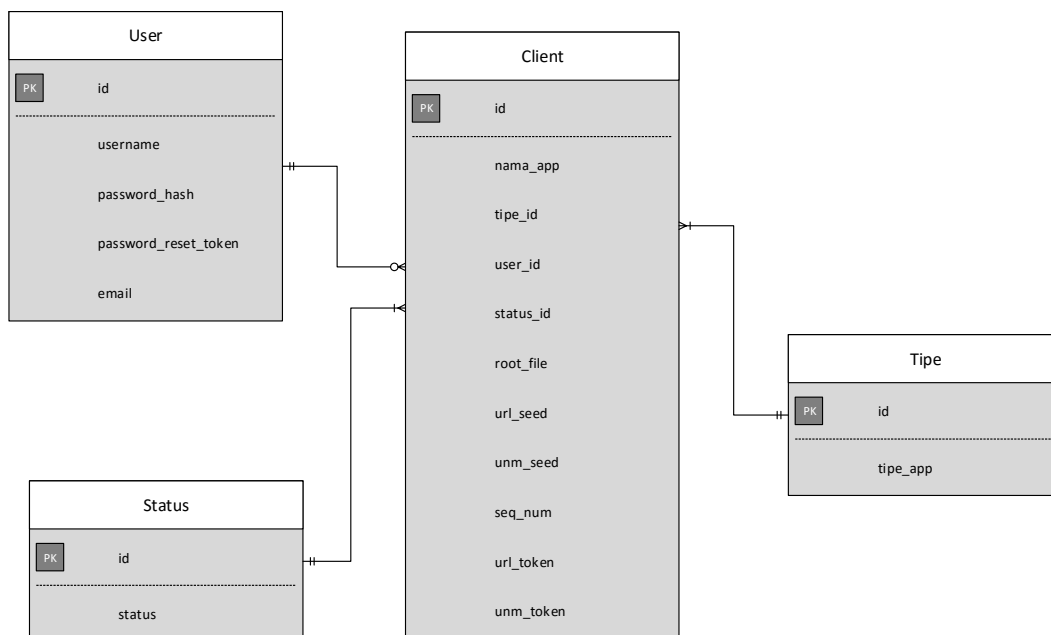
Gambar IV.13 adalah *sequence diagram* untuk tahap sinkronisasi. Penjelasan *sequence diagram* tersebut adalah sebagai berikut.

- a. *Actor* eksternal Administrator mengklik tombol *request synchronization key* pada antarmuka aplikasi administrator.
- b. Antarmuka ViewAplikasiUI melakukan *request synchronization key* dengan mengirimkan ID client.
- c. ClientController menerima *request* dan membangkitkan *synchronization key*. Kemudian *synchronization key* ditampilkan pada antarmuka dan diterima oleh Administrator.
- d. *Actor* eksternal Administrator memperbaharui *synchronization key* pada aplikasi Client.
- e. *Actor* eksternal Administrator mengklik tombol *request* proses sinkronisasi pada aplikasi Client. Kemudian Client melakukan *request* proses sinkronisasi ke AuthController.
- f. AuthController menerima *request* proses sinkronisasi dan kemudian memanggil *method* findClient() pada AuthServiceProvider untuk mengambil data *client* berdasarkan *synchronization key*.
- g. AuthServiceProvider mengirimkan objek *client* ke AuthController sebagai *return value* dari *method* findClient().
- h. AuthController melakukan *request random number* ke AuthServiceProvider dengan mengirimkan data *client*.
- i. AuthServiceProvider memanggil *method*-nya sendiri, generateTokenHash(), untuk menghitung nilai *hash* dari token yang dibangkitkan dari *random number*.
- j. AuthServiceProvider mengirimkan *random number* kepada AuthController. Kemudian AuthController meneruskannya ke Client.
- k. Client melakukan penghitungan nilai *hash* dari token yang dibangkitkan dari *random number* yang diterima.
- l. Client melakukan *request* sinkronisasi dengan mengirimkan nilai *hash*.
- m. AuthController memanggil *method* validateHashValue() dengan parameter nilai *hash* yang diterima dari Client. *Method* ini berfungsi untuk membandingkan nilai *hash* Client dengan nilai *hash* yang dihasilkan *Authentication Server*.

- n. AuthServiceProvider mengirimkan status sinkronisasi dan *sequence number* yang valid ke AuthController. Kemudian AuthController meneruskannya ke Client.
- o. Client memperbaharui *sequence number* dengan *sequence number* yang valid dan menampilkan status sinkronisasi kepada Administrator.

IV.2 Perancangan Database

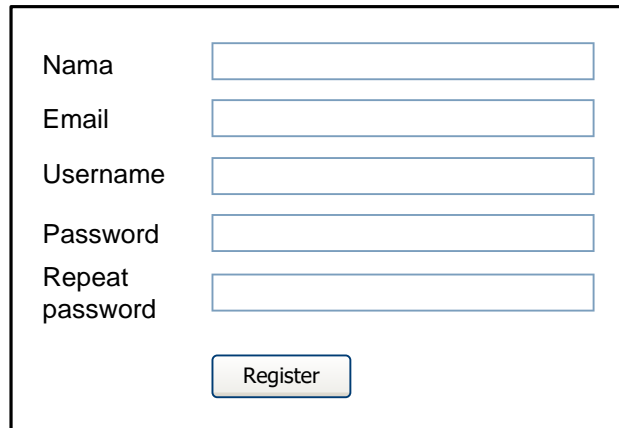
Database dirancang memiliki empat tabel yaitu tabel User, Client, Status, dan Tipe. Tabel User digunakan untuk menyimpan data *developer* aplikasi, tabel Client untuk menyimpan data aplikasi, tabel Status untuk menyimpan status aplikasi, dan tabel Tipe untuk menyimpan tipe aplikasi. Struktur database digambarkan dengan *Entity Relationship Diagram* (ERD) pada Gambar IV.14. Hubungan tabel User dan Client adalah *zero-to-many* yang berarti satu *developer* dapat memiliki nol atau banyak aplikasi. Tabel Status dan tabel Tipe memiliki hubungan *one-to-many* terhadap tabel Client.



Gambar IV.14. *Entity Relationship Diagram* (ERD) database client

IV.3 Perancangan Antarmuka

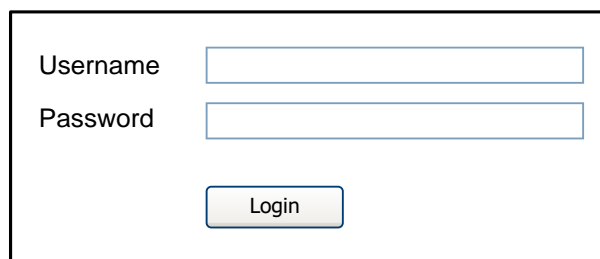
Pada tahap inialisai, terdapat proses registrasi aplikasi oleh *developer* aplikasi. Proses tersebut dilakukan pada sistem admin yang disediakan oleh *Service Provider*. Perancangan antarmuka sistem ini diantaranya form registrasi *developer*, form *login*, form registrasi aplikasi, dan halaman untuk mengelola data *developer* dan aplikasinya.



A registration form for a developer. It contains five text input fields stacked vertically, each preceded by a label: 'Nama', 'Email', 'Username', 'Password', and 'Repeat password'. Below the input fields is a single button labeled 'Register'.

Gambar IV.15. Form registrasi *developer*

Form registrasi *developer* digunakan untuk melakukan pendaftaran *developer* aplikasi ke Service Provider. Melalui form ini, *developer* memasukkan data nama, email, *username*, dan *password* yang digunakan untuk proses *login* ke dalam sistem. Form registrasi dirancang seperti pada Gambar IV.15.



A login form. It contains two text input fields stacked vertically, each preceded by a label: 'Username' and 'Password'. Below the input fields is a single button labeled 'Login'.

Gambar IV.16. Form *login*

Setelah melakukan registrasi, *developer* aplikasi masuk ke dalam sistem untuk mendaftarkan aplikasinya. Form *login* dirancang seperti pada Gambar IV.16.

Gambar IV.17. Form untuk pendaftaran aplikasi

Developer dapat mendaftarkan aplikasinya pada form pendaftaran aplikasi seperti terlihat pada Gambar IV.17. Pada form ini *developer* memasukkan nama dan tipe aplikasinya. Selain itu, *root file* diupload ke server untuk digunakan dalam proses autentikasi.

Gambar IV.18 menampilkan rancangan halaman kelola aplikasi. Pada halaman ini, *developer* dapat melihat dan mengelola aplikasi yang pernah didaftarkan, seperti menghapus, mengubah, dan menampilkan detail aplikasinya.

Nama aplikasi	Tipe aplikasi	Kelola aplikasi

Gambar IV.18. Form untuk mengelola aplikasi

Bab V Implementasi dan Pengujian

V.1 Implementasi

Implementasi dilakukan dengan menyusun prototipe berdasarkan hasil perancangan pada Bab IV. Prototipe dibagi menjadi dua bagian, yaitu *Authentication Server* dan *Resource Server*.

V.1.1 *Authentication Server*

Pada *Authentication Server* terdapat dua aplikasi yaitu *authenticator* yang digunakan untuk proses autentikasi client dan aplikasi administrator untuk pengelolaan aplikasi client oleh *developer*.

V.1.1.1 *Authenticator*

Authenticator diimplementasikan menggunakan bahasa pemrograman PHP dengan *framework* Silex (Silex, 2016). Silex merupakan *microframework* PHP yang ditujukan untuk membangun aplikasi web yang ringan dan sederhana, seperti REST API.

Class utama pada *authenticator* adalah *Application.php*. *Class* ini memanggil *class* *AuthBuilder.php* yang di dalamnya terdapat *method* *mountProviderIntoApplication()*. *Method* ini berfungsi untuk melakukan *mounting route* yang didefinisikan pada *controller* dan memberikan awalan setiap *route* dengan *prefix* “/auth”. *Class* *Application.php* juga mendefinisikan *response header* dengan memberikan nilai “*” pada *Access-Control-Allow-Origin*. Hal ini berfungsi untuk mekanisme *Cross-Origin Resource Sharing* (CORS) (W3C, 2014) sehingga memungkinkan dilakukannya proses *request-response* antara *Authentication Server* dan client dengan domain yang berbeda. Berikut ini adalah kode program pada *Application.php* dan *AuthBuilder.php*.

```
class Application extends SilexApplication
{
    public function __construct(array $values = [])
    {
        parent::__construct($values);
    }
}
```

```

        AuthBuilder::mountProviderIntoApplication('/auth', $this);

        $this->after(function (Request $request, Response
$response) {
            $response->headers->set('Access-Control-Allow-Origin',
            '*');
        });
    }
}

class AuthBuilder
{
    public static function mountProviderIntoApplication($route,
Application $app)
    {
        $app->mount($route, (new AuthControllerProvider())-
>setBaseRoute($route));
    }
}

```

Controller didefinisikan pada *class* AuthControllerProvider.php. Pada *class* ini *route* untuk setiap proses didefinisikan sebagai berikut.

1. Tahap inialisasi: [http:// 192.168.182.136/authenticater/auth/inialisasi/](http://192.168.182.136/authenticater/auth/inialisasi/)
2. Tahap identifikasi: [http:// 192.168.182.136/authenticater/auth/identifikasi/](http://192.168.182.136/authenticater/auth/identifikasi/)
3. Tahap autentikasi: [http:// 192.168.182.136/authenticater/auth/autentikasi/](http://192.168.182.136/authenticater/auth/autentikasi/)
4. Tahap sinkronisasi: [http:// 192.168.182.136/authenticater/auth/sinkronisasi/](http://192.168.182.136/authenticater/auth/sinkronisasi/)

Pada *class* ini juga didefinisikan *method* setUpMiddlewares() yang berisi fungsi *before middleware* sehingga memungkinkan baris program dijalankan sebelum *controller* dieksekusi. *Middelware* tersebut menjalankan fungsi yang melakukan registrasi *class* AuthServiceProvider.php yang berisi *method-method* untuk proses autentikasi, seperti initialization(), getClient(), generateNextTokens(), generateSeqNumAndHashValue(), validateHashValue(), getNewTokenForUser(), dan sebagainya. Kode program selengkapnya untuk *class* AuthControllerProvider.php dan AuthServiceProvider.php dapat dilihat pada Lampiran B. Berikut ini merupakan petikan kode program pada *class* AuthControllerProvider.php untuk *method* extractControllers() dan setUpMiddlewares().

```

private function extractControllers(Application $app)
{
    $controllers = $app['controllers_factory'];

    // proses inisialisasi
    $controllers->post(self::INISIALISASI, function
(Request $request) use ($app) {
        $unmtoken = $request->get('unmtoken');
        $urltoken = $request->get('urltoken');
        $seqnum = $request->get('seqnum');
        $initkey = $request->get('initkey');

        return $app-
>json($app[AuthServiceProvider::AUTH_INITIALIZATION]($unmtoken,
$urltoken, $seqnum, $initkey));
    });

    // proses identifikasi
    $controllers-
>post(self::IDENTIFIKASI.'/{urltoken}/{unmtoken}/{seqnum}',
function (Request $request, $urltoken, $unmtoken, $seqnum) use
($app) {

        $client =
$app[AuthServiceProvider::AUTH_FIND_CLIENT]($urltoken, $unmtoken,
$seqnum);

        if(isset($client))
            $client2 =
$app[AuthServiceProvider::AUTH_NEXT_TOKENS]($client);

        return $app->json([
            'status' => true,
            'info' =>
['n'=>$app[AuthServiceProvider::AUTH_RAND_SEQ_NUM]($client2)]
        ]);
    });

    $controllers-
>post(self::AUTENTIKASI.'/{urltoken}/{unmtoken}/{seqnum}',
function (Request $request, $urltoken, $unmtoken, $seqnum) use
($app) {

        $hashvalue = $request->get('hashvalue');

        $cclient = new CClient($app['db']);

        $client =
$app[AuthServiceProvider::AUTH_FIND_CLIENT]($urltoken, $unmtoken,
$seqnum);

        $status =
$app[AuthServiceProvider::AUTH_VALIDATE_HASH_VALUE]($hashvalue,
$client);

        return $app->json([
            'status' => $status,
            'info' => $status ? ['token' =>
$app[AuthServiceProvider::AUTH_NEW_TOKEN]($client)] : []

```

```

        ));
    });

    // proses sinkronisasi
    $controllers->post(self::SINKRONISASI, function
(Request $request) use ($app) {
        $hashvalue = $request->get('hashvalue');
        $unmtoken = $request->get('unmtoken');
        $urltoken = $request->get('urltoken');
        $seqnum = $request->get('seqnum');
        $synckey = $request->get('synckey');

        if(isset($hashvalue))
        {
            $status =
$app[AuthServiceProvider::SYNC_VALIDATE_HASH_VALUE]($hashvalue,
$synckey);

            return $app->json($status);
        }
        else{
            $client =
$app[AuthServiceProvider::SYNC_FIND_CLIENT]($synckey);

            return $app->json([
                'status' => $client?true:false,
                'info' =>
$client?['n'=>$app[AuthServiceProvider::SYNC_RAND_SEQ_NUM]($client
)]:[]
            ]);
        }
    });

    return $controllers;
}

private function setUpMiddlewares(Application $app)
{
    $app->before(function (Request $request) use ($app) {
        $app->register(new
AuthServiceProvider($app['db']));
    });
}

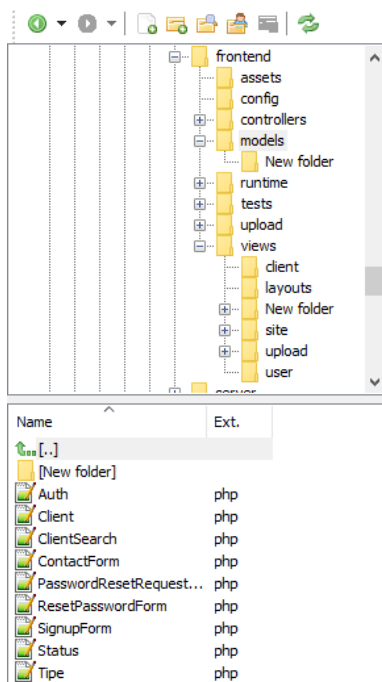
```

Authenticator memiliki *class* yang mewakili entitas eksternal client yaitu *class* *Client.php*. *Class* ini mendefinisikan atribut-atribut yang dimiliki oleh entitas client. Selain itu, terdapat *class* *CClient.php* yang mendefinisikan operasi-operasi yang dapat dilakukan terhadap data client, seperti pengambilan dan *update* data. Kode program untuk kedua *class* ini diberikan pada Lampiran C.

Implementasi memerlukan beberapa *library*, yaitu untuk melakukan pengekstrakan *seed* dari *root file*, pembangkitan token, dan penghitungan nilai *hash*. *Extract seed* dilakukan dengan menggunakan fungsi `fread()` pada PHP. Fungsi ini memiliki dua parameter yaitu parameter *file* dan *length*. Parameter *file* mendefinisikan *file* yang akan dibaca dan parameter *length* mendefinisikan banyaknya *byte* yang dibaca dari *file* tersebut. Pembangkitan token dalam proses autentikasi dilakukan dengan menggunakan fungsi `mt_rand()` pada PHP. Fungsi ini menghasilkan *random number* melalui Mersenne Twister *Random Number Generator* (Matsumoto dan Nishimura, 1998). Penghitungan nilai *hash* dilakukan dengan fungsi `md5()`. Fungsi ini menghitung MD5 *hash* dari sebuah *string* dengan algoritma MD5 *Message-Digest*. Keluaran dari fungsi ini adalah nilai *hash* berupa bilangan heksadesimal sebanyak 32 karakter. Selain itu, implementasi pada *authenticator* juga menggunakan fungsi yang terdapat pada Yii Framework, diantaranya fungsi `generateRandomString()` untuk menghasilkan *access token* dan fungsi `compareString()` untuk membandingkan dua *string*. *Source code* selengkapnya diberikan pada bagian lampiran.

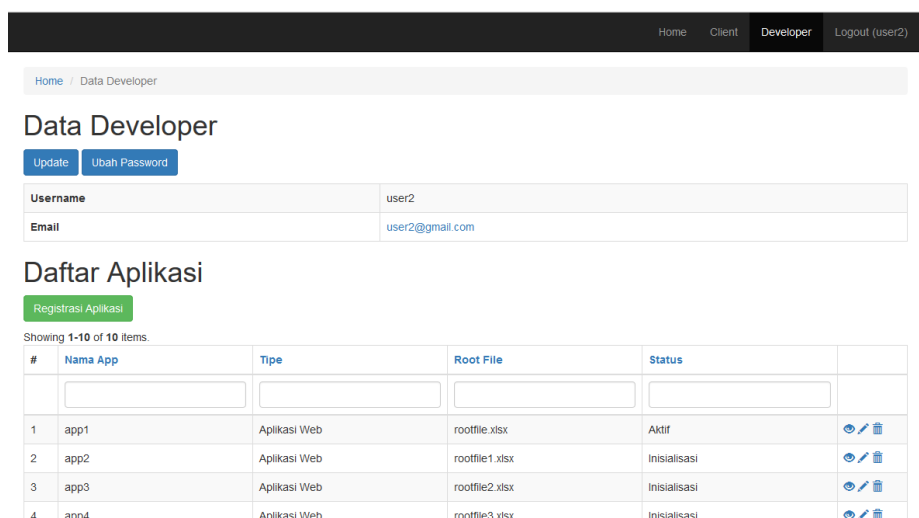
V.1.1.2 Aplikasi Administrator

Pada tahap inisialisasi terdapat interaksi antara *developer* dengan aplikasi administrator di *Authentication Server* untuk melakukan registrasi dan pengelolaan aplikasi client. Aplikasi administrator diimplementasikan menggunakan bahasa pemrograman PHP dengan framework Yii 2.0 (Yii, n.d.). Yii merupakan *framework* PHP berkinerja tinggi untuk pengembangan aplikasi web. Pengembangan web dengan framework Yii mengikuti pola *model-view-controller* (MVC) yang bertujuan untuk memisahkan *business logic* dengan antarmuka sehingga developer dapat dengan mudah melakukan perubahan tanpa mempengaruhi bagian lain. Struktur direktori aplikasi administrator dapat dilihat pada Gambar V.1.



Gambar V.1. Struktur direktori aplikasi administrator

Antarmuka utama aplikasi administrator dapat dilihat pada Gambar V.2. Pada aplikasi ini *developer* dapat melakukan pengelolaan aplikasinya seperti mendaftarkan aplikasi client, menghapus, melakukan *update*, dan sebagainya. Antarmuka lainnya diantaranya form registrasi *developer*, form *login*, form registrasi aplikasi, dan form *update* aplikasi. Antarmuka selengkapnya dapat dilihat pada Lampiran J.



Gambar V.2. Antarmuka utama aplikasi administrator

V.1.1.3 Database

Kedua aplikasi *authenticator* dan *administrator* mengakses database yang sama, yaitu *authdb*. Database *authdb* diimplementasikan dengan database MySQL. Struktur database *authdb* dapat dilihat pada Gambar V.3. SQL database dapat dilihat pada Lampiran I.

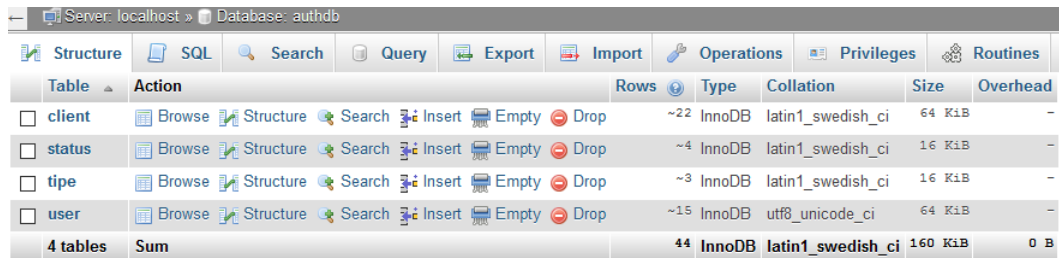


Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> client		~22	InnoDB	latin1_swedish_ci	64 KkB	-
<input type="checkbox"/> status		~4	InnoDB	latin1_swedish_ci	16 KkB	-
<input type="checkbox"/> tipe		~3	InnoDB	latin1_swedish_ci	16 KkB	-
<input type="checkbox"/> user		~15	InnoDB	utf8_unicode_ci	64 KkB	-
4 tables	Sum	44	InnoDB	latin1_swedish_ci	160 KkB	0 B

Gambar V.3. Struktur database *authdb*

V.1.2 Resource Server

Aplikasi pada *Resource Server* berfungsi dalam menyediakan API bagi client untuk mengakses *resource* pada server. Aplikasi ini diimplementasikan menggunakan pemrograman PHP dengan *framework* Silex. *Class* utama pada aplikasi ini adalah *Application.php*. *Class* ini memanggil *method* *mountProviderIntoApplication()* pada *class* *AuthBuilder.php*. *Method* ini berfungsi melakukan mounting *route* pada *controller* *AuthControllerProvider* dan menambahkan awalan untuk setiap *route* dengan *prefix* “/auth”.

Class *Application.php* juga mendefinisikan *response header* *Access-Control-Allow-Origin* dengan nilai “*” sehingga proses *request-response resource* dapat dilakukan oleh client yang berbeda host. Selain itu, *response header* *Access-Control-Allow-Headers* didefinisikan dengan ‘X-Token’ yang menunjukkan bahwa HTTP header X-Token tersedia bagi client untuk mengirim access token. Petikan kode program untuk *class* *Application* dan *AuthBuilder* adalah sebagai berikut.

```
class Application extends SilexApplication
{
    public function __construct(array $values = [])
    {
        parent::__construct($values);
    }
}
```

```

        AuthBuilder::mountProviderIntoApplication('/auth', $this);

        $this->after(function (Request $request, Response
$response) {
            $response->headers->set('Access-Control-Allow-Origin',
            '*');
            $response->headers->set('Access-Control-Allow-
Headers', 'X-Token');
        });
    }
}

class AuthBuilder
{
    public static function mountProviderIntoApplication($route,
Application $app)
    {
        $app->mount($route, (new AuthControllerProvider())-
>setBaseRoute($route));
    }
}

```

Controller AuthControllerProvider mendefinisikan *method* *setUpMiddlewares()* yang memiliki *before middleware* yang akan dijalankan sebelum *controller* dieksekusi. *Middleware* ini berfungsi melakukan registrasi *class AuthServiceProvider* yang didalamnya terdapat *method* *validateToken()* untuk memvalidasi *access token* yang dikirim oleh client. Selain itu, *controller AuthControllerProvider* memiliki *method* *getTokenFromRequest()* untuk mengambil *access token* yang dikirim client melalui HTTP *header* X-token dan *method* *isValidTokenForApplication()* untuk menentukan apakah *access token* tersebut valid atau tidak. Berikut ini adalah petikan kode program pada *class controller AuthControllerProvider*.

```

private function setUpMiddlewares(Application $app)
{
    $app->before(function (Request $request) use ($app) {

        $app->register(new
AuthServiceProvider($app['db']));

        $urltoken = $request->get('urltoken');
        $unmtoken = $request->get('unmtoken');
        $seqnum = $request->get('seqnum');

        if (!$this->isAuthRequiredForPath($request-
>getPathInfo()) && $request->getMethod() != 'OPTIONS') {

```

```

        if (!$this->isValidTokenForApplication($app,
        $this->getTokenFromRequest($request), $urltoken, $umtoken,
        $seqnum)) {
            throw new AccessDeniedHttpException('Access
Denied');
        }
    }
});
}

private function getTokenFromRequest(Request $request)
{
    // 1. ambil id client dan tokennya
    return $request->headers->get(self::TOKEN_HEADER_KEY,
$request->get(self::TOKEN_REQUEST_KEY));
}

private function isValidTokenForApplication(Application $app,
$token, $urltoken, $umtoken, $seqnum)
{
    return
$app[AuthServiceProvider::AUTH_VALIDATE_TOKEN]($token, $urltoken,
$umtoken, $seqnum);
}

```

Seperti halnya aplikasi *authenticater*, aplikasi ini juga memiliki Class Client.php yang mewakili aplikasi client yang mengakses *resource* ke *Resource Server*. Class ini mendefinisikan atribut dan operasi yang dapat dimiliki client. Terdapat juga class CClient.php yang mendefinisikan operasi untuk mengambil data client. Kode program untuk kedua class Client.php dan CClient.php dapat dilihat pada lampiran.

Aplikasi di *Resource Server* memiliki koneksi ke dua database, yaitu database autentikasi pada *Authentication Server* dan database *resource* pada *Resource Server*. Database *resource* diimplementasikan dengan database MySQL. Data *resource* menggunakan data *sample* dari MySQL. Struktur database dapat dilihat pada Gambar V.4.

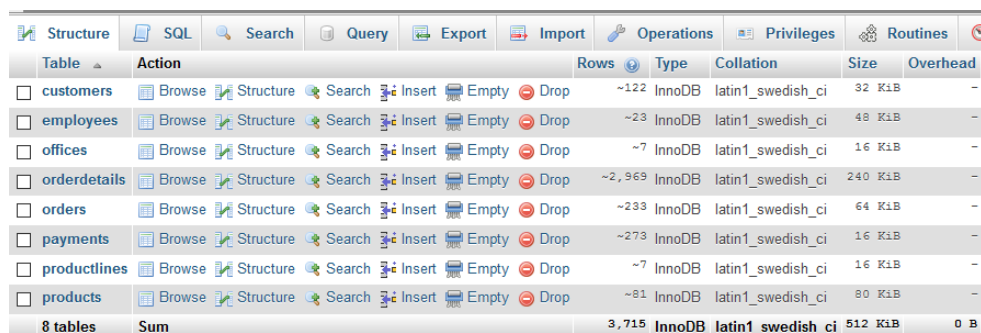


Table	Action	Rows	Type	Collation	Size	Overhead
customers	Browse Structure Search Insert Empty Drop	~122	InnoDB	latin1_swedish_ci	32 KiB	-
employees	Browse Structure Search Insert Empty Drop	~23	InnoDB	latin1_swedish_ci	48 KiB	-
offices	Browse Structure Search Insert Empty Drop	~7	InnoDB	latin1_swedish_ci	16 KiB	-
orderdetails	Browse Structure Search Insert Empty Drop	~2,969	InnoDB	latin1_swedish_ci	240 KiB	-
orders	Browse Structure Search Insert Empty Drop	~233	InnoDB	latin1_swedish_ci	64 KiB	-
payments	Browse Structure Search Insert Empty Drop	~273	InnoDB	latin1_swedish_ci	16 KiB	-
productlines	Browse Structure Search Insert Empty Drop	~7	InnoDB	latin1_swedish_ci	16 KiB	-
products	Browse Structure Search Insert Empty Drop	~81	InnoDB	latin1_swedish_ci	80 KiB	-
8 tables	Sum	3,715	InnoDB	latin1_swedish_ci	512 KiB	0 B

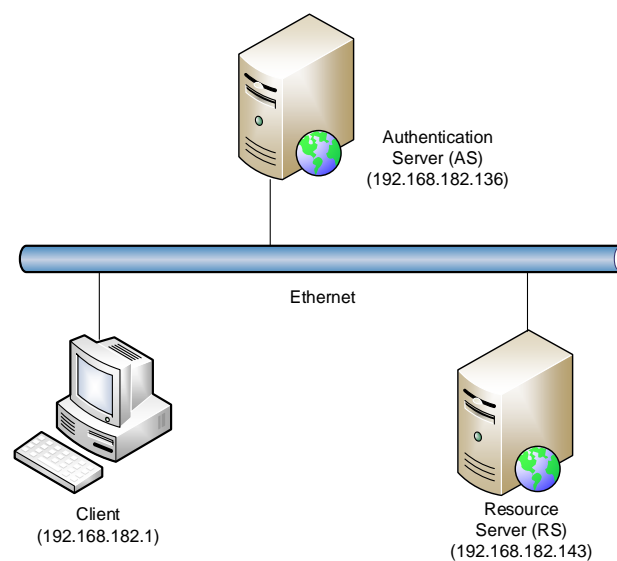
Gambar V.4. Struktur *database resource*, resourcedb

Koneksi aplikasi ke kedua *database* tersebut didefinisikan pada file `index.php` sebagai berikut.

```
$app->register(new Silex\Provider\DoctrineServiceProvider(),
array(
    'dbs.options' => array (
        'auth' => array(
            'driver' => 'pdo_mysql',
            'host' => '192.168.182.136',
            'dbname' => 'authdb',
            'user' => 'root',
            'password' => 'auliak',
        ),
        'resource' => array(
            'driver' => 'pdo_mysql',
            'host' => 'localhost',
            'dbname' => 'resourcedb',
            'user' => 'root',
            'password' => 'auliak',
        ),
    ),
));
```

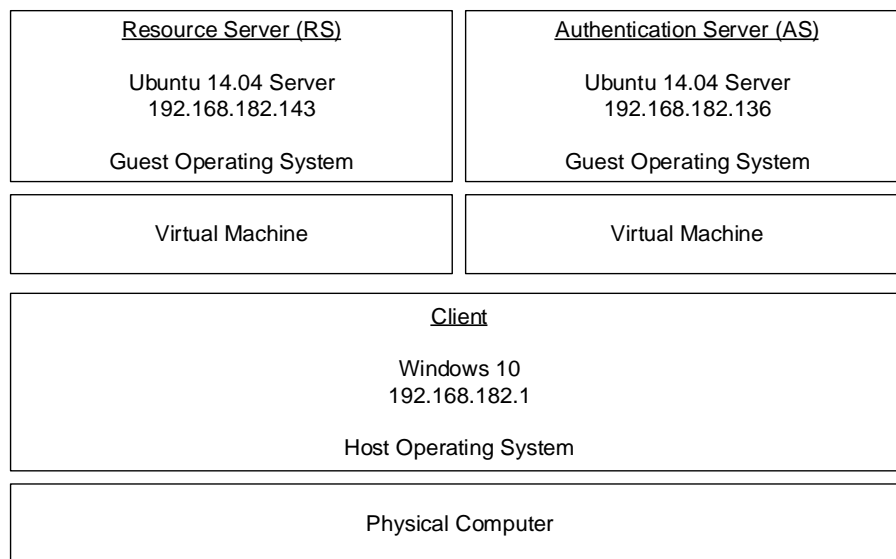
V.2 Pengujian Prototipe

Pengujian dilakukan terhadap prototipe yang telah dibangun. Arsitektur jaringan pengujian dirancang memiliki tiga *host* yaitu *Authentication Server*, *Resource Server*, dan *Client* yang terhubung dalam jaringan *Local Area Network* (LAN). Gambar V.5 menunjukkan arsitektur jaringan pengujian prototipe.



Gambar V.5. Gambar arsitektur pengujian

Arsitektur jaringan pengujian diimplementasikan dengan menggunakan aplikasi *virtual machine*, VMWare, yang diinstal dalam sistem operasi *host*. Pada aplikasi VMWare dibuat dua server, yaitu *Authentication Server* dan *Resource Server*, yang terhubung melalui jaringan LAN. Aplikasi client yang akan mengakses kedua server tersebut terdapat pada sistem operasi *host*. Gambar V.6 menunjukkan arsitektur pengujian pada VMWare. Spesifikasi *hardware* dan *software* untuk proses pengujian dapat dilihat pada Tabel V.1.



Gambar V.6. Gambar arsitektur jaringan pada *virtual machine* untuk pengujian

Tabel V.1. Tabel spesifikasi lingkungan pengujian

Spesifikasi	Client	<i>Authentication Server (AS)</i>	<i>Resource Server (RS)</i>
Sistem Operasi	Windows 10	Ubuntu 14.04	Ubuntu 14.04
RAM	4GB	1GB	1GB
IP Address	162.168.182.1	162.168.182.136	162.168.182.143

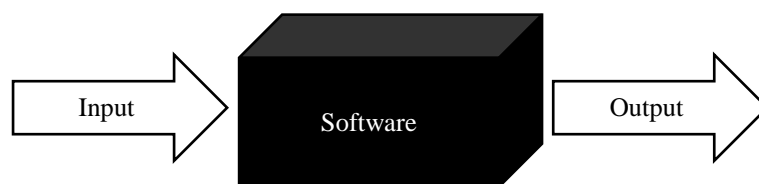
Untuk proses pengujian, aplikasi client dibuat untuk melakukan proses *request resource* ke *Resource Server* dan proses autentikasi ke *Authentication Server*. Aplikasi client disusun menggunakan bahasa pemrograman PHP dengan *framework* Yii 2.0. Proses *request-response* dilakukan dengan menggunakan ajax pada bahasa pemrograman javascript dengan *library* jQuery.

Aplikasi client memiliki *database* untuk menyimpan informasi autentikasi client seperti *access token*, *root file*, *sequence number*, *initializaion key*, dan *synchronization key*. Database tersebut diimplementasikan dengan MySQL.

V.2.1 *Black Box Testing*

Pada setiap pembangunan *software* terdapat spesifikasi yang dideskripsikan menggunakan kata-kata dan gambar, seperti pada bagian perancangan (Bab IV) yang menggambarkan fungsionalitas dari *software* tersebut. Pengujian *software* dilakukan dengan pendekatan *black box testing* untuk menguji fungsi-fungsi *software* sesuai dengan spesifikasinya tanpa melihat bagaimana operasi di dalam *software* tersebut. Jika suatu input dimasukkan ke dalam fungsi *software* maka akan mendapatkan *output* tertentu tanpa melihat bagaimana cara *software* tersebut menghasilkan *output* tersebut (Patton, 2001).

Uji ini dilakukan dengan menjalankan fungsi-fungsi pada aplikasi administrator dan proses autentikasi antara client dan *authenticator*. Dalam proses pengujian dilihat apakah *output* yang dihasilkan setiap fungsi sesuai dengan yang diharapkan tanpa melihat alur proses yang terjadi. Ceklis dilakukan untuk melihat apakah fungsi-fungsi tersebut lulus uji atau tidak. Ilustrasi *black box testing* ini dapat dilihat pada Gambar V.7.



Gambar V.7. Gambar ilustrasi *black box testing*

Test case pada pengujian dibuat berdasarkan nilai input ekstrim, misalnya nilai maksimum, nilai minimum, nilai yang terletak di dalam atau di luar batas, serta nilai *error*. Pengujian ini disebut juga dengan *boundary testing*. Penentuan batas nilai input dilakukan berdasarkan spesifikasi pada persyaratan sistem. Selain itu, batas tersebut juga dapat ditentukan dengan melihat keterkaitan dengan bagian lain pada

sistem, misalnya kolom pada database yang memiliki tipe data dan panjang karakter tertentu. Cara lainnya adalah dengan menggunakan pendekatan intuitif, yaitu dengan memasukkan berbagai macam nilai yang besar untuk menghasilkan *error* pada sistem.

Nilai yang terletak pada batas disebut dengan nilai batas valid, sedangkan nilai yang terletak di luar batas disebut dengan nilai batas tidak valid. *Test case* dirancang berdasarkan nilai batas valid dan nilai batas tidak valid. *Test case* dibuat dengan memasukkan nilai input yang terletak tepat pada batas, di bawah batas, dan di atas batas.

Pengujian dilakukan pada fungsi-fungsi yang ada pada aplikasi administrator dan proses autentikasi antara client dan *authenticator*. Pengujian aplikasi administrator dilakukan oleh peneliti dengan berperan seolah-olah adalah sebagai *developer* aplikasi. Pengujian proses autentikasi dilakukan antara aplikasi web, sebagai client, dengan *authenticator*. Hasil pengujian dapat dilihat pada Tabel V.3 untuk pengujian aplikasi administrator dan Tabel V.4 untuk pengujian proses autentikasi.

Fitur *forget password* pada aplikasi administrator berfungsi untuk mengirimkan email kepada pengguna yang berisi link untuk mengubah password pengguna. Pada *framework* Yii 2.0, pengiriman email diimplementasikan oleh kelas `yii\mail\BaseMailer`. Pada pengujian fungsi ini, properti `$useFileTransport` diberikan nilai *true* sehingga email tidak dikirim kepada penerima email sebenarnya, melainkan disimpan sebagai file di dalam direktori `runtime/mail`.

Pada fungsi registrasi aplikasi, salah satu pengujiannya dilakukan dengan melakukan *upload root file* dengan berbagai macam ukuran mulai dari 10 KB hingga 20 MB. Hasil dari pengujian menunjukkan bahwa file yang memiliki ukuran lebih dari 2 MB tidak dapat dilakukan *upload*. Hal ini disebabkan karena PHP memiliki konfigurasi untuk membatasi penggunaan *resource*.

Secara *default*, upload file dibatasi pada file yang berukuran maksimal 2 MB. Ukuran file maksimal yang dapat diupload dapat diubah pada file konfigurasi, `php.ini`, dengan mengganti nilai parameter `upload_max_filesize`. Selain itu, parameter `post_max_size` juga mempengaruhi *upload* file. Parameter ini menentukan ukuran maksimal data yang dapat dikirim. Secara umum, nilai `post_max_size` harus lebih besar dari `upload_max_filesize`.

Pengujian *upload root file* juga dilakukan dengan berbagai macam format yaitu `.jpg`, `.png`, `.gif`, `.doc`, `.docx`, `.pdf`, `.ppt`, `.pptx`, `.txt`, dan `.xls`. Hasil pengujian menunjukkan bahwa root file dengan format `.doc` dan `.ppt` menghasilkan UNMToken yang sama. Hal ini disebabkan karena *seed* untuk membangkitkan UNMToken diekstrak dari 12 byte terakhir dari *root file*. File dengan format `.doc` dan `.ppt` memiliki 12 byte terakhir yang sama sehingga menghasilkan UNMToken yang sama. Tabel V.2 menunjukkan token yang dihasilkan dari berbagai macam format *root file*.

Tabel V.2. Tabel token yang dihasilkan dari berbagai macam format *root file*

<i>Root File</i>	URLToken	UNMToken
rootfile.jpg	767384805	762780353
rootfile.png	436983878	2018655968
rootfile.gif	66834366	702945965
rootfile.doc	862996015	1898357091
rootfile.docx	579677213	5017013
rootfile.pdf	2099566235	911234134
rootfile.ppt	316345860	1898357091
rootfile.pptx	2005980749	61338231
rootfile.txt	1531591794	215655886
rootfile.xls	452988721	1456527122

Tabel V.3. Hasil pengujian pada sistem admin

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
Registrasi developer aplikasi	1. Klik menu sign up 2. Input data 3. Klik save	Username	1. String dengan panjang 1-255 karakter 2. Unik 3. Not null	V	Username dengan panjang karakter 1	Username tersimpan dalam database	Data tersimpan dalam database	✓
					Username dengan panjang karakter 255	Username tersimpan dalam database	Data tersimpan dalam database	✓
				T	Username dengan panjang karakter >255	Pesan error	Data tersimpan dalam database namun terpotong	✗
					Username Kosong	Pesan error	Pesan error	✓
					Username telah digunakan pengguna lain	Pesan error	Pesan error	✓
		Email	1. Format email 2. Panjang email maksimal 255 karakter 3. Unik	V	Format email sesuai	Email tersimpan dalam database	Email tersimpan dalam database	✓
				T	Format email tidak sesuai	Pesan error	Pesan error	✓
					Email telah digunakan pengguna lain	Pesan error	Pesan error	✓
		Password	Minimal 6 karakter	V	Password	Password hash tersimpan	Password hash tersimpan	✓
				T	Password <6 karakter	Pesan error	Pesan error	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
		Password repeat	Sama dengan password	V	Password repeat	Password hash tersimpan	Password hash tersimpan	✓
				T	Password repeat tidak sama dengan password	Pesan error	Pesan error	✓
Login	1. Klik menu login 2. Input data 3. Klik tombol login	Username	Username yang pernah didaftarkan	V	Username yang pernah didaftarkan	User berhasil login	User berhasil login	✓
				T	Username tidak sesuai	Pesan error	Pesan error	✓
		Password	Password yang pernah didaftarkan	V	Password yang pernah didaftarkan	User berhasil login	User berhasil login	✓
				T	Password tidak sesuai	Pesan error	Pesan error	✓
Reset password	1. Klik link “lupa password” 2. Masukkan email 3. Klik submit	Email	1. Email telah terdaftar 2. Format email	V	Email yang telah terdaftar dan sesuai format	Email yang berisi link dan password reset token	Email yang berisi link dan password reset token	✓
				T	Email tidak terdaftar	Pesan error	Pesan error	✓
					Format email tidak sesuai	Pesan error	Pesan error	✓
Form reset password	1. Buka email 2. Copy dan buka link pada browser	Token	String token yang dikirim pada email	V	Token sesuai	Menampilkan form reset	Menampilkan form reset	✓
				T	Token tidak sesuai	Pesan error	Pesan error	✓
		Password baru	Minimal 6 karakter	V	Password baru sesuai	Password hash baru tersimpan	Password hash baru tersimpan	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
	3. Masukkan password baru			T	Password baru <6 karakter	Pesan error	Pesan error	✓
	4. Klik submit	Password repeat	Sama dengan password	V	Password repeat sama	Password hash baru tersimpan	Password hash baru tersimpan	✓
				T	Password repeat tidak sama	Pesan error	Pesan error	✓
Update data developer	1. Klik menu data user 2. Klik tombol update data 3. Update data 4. Klik tombol update	Username	1. String dengan panjang 1-255 karakter 2. Unik 3. Not null	V	Username dengan panjang karakter 1	Username tersimpan dalam database	Data tersimpan dalam database	✓
					Username dengan panjang karakter 255	Username tersimpan dalam database	Data tersimpan dalam database	✓
				T	Username dengan panjang karakter >255	Pesan error	Data tersimpan dalam database namun terpotong	✗
					Username telah digunakan pengguna lain	Pesan error	Pesan error	✓
		Email	1. Format email 2. Panjang email maks 255 karakter 3. Unik	V	Email	Email tersimpan dalam database	Email tersimpan dalam database	✓
				T	Format email tidak sesuai	Pesan error	Pesan error	✓
					Email telah digunakan pengguna lain	Pesan error	Pesan error	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
Ganti password	1. Klik menu data user 2. Klik tombol ganti password 3. Masukkan password lama dan password baru 4. Klik tombol ganti password	Password	Passwor user yang bersangkutan	V	Password yang pernah didaftarkan	Password hash baru tersimpan	Password hash baru tersimpan	✓
				T	Password tidak sesuai	Pesan error	Pesan error	✓
		Password baru	Minimal 6 karakter	V	Password baru sesuai	Password hash baru tersimpan	Password hash baru tersimpan	✓
				T	Password baru <6 karakter	Pesan error	Pesan error	✓
		Password repeat	Sama dengan password baru	V	Password repeat sama	Password hash baru tersimpan	Password hash baru tersimpan	✓
				T	Password repeat tidak sama	Pesan error	Pesan error	✓
Registrasi aplikasi	1. Klik tombol registrasi aplikasi 2. Masukkan data 3. Pilih root file 4. Klik tombol registrasi aplikasi	Nama aplikasi	1. String dengan panjang 1-255 karakter 2. Not null	V	Nama aplikasi 1 karakter	Nama aplikasi tersimpan	Nama aplikasi tersimpan	✓
					Nama aplikasi 255 karakter	Nama aplikasi tersimpan	Nama aplikasi tersimpan	✓
				T	Nama aplikasi >255 karakter	Pesan error	Pesan error	✓
		Tipe aplikasi	Id tipe aplikasi yang dipilih dari dropdown list	V	Tipe aplikasi valid	Tipe aplikasi tersimpan	Tipe aplikasi tersimpan	✓
				T	Tipe aplikasi tidak dipilih	Pesan error	Pesan error	✓
		Root file	File dengan berbagai jenis	V	Root file berbagai format (.jpg, .png, .gif, .doc, .ppt, .xls,	Root file tersimpan	Root file tersimpan	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
			format dengan size > 24 bytes		.txt, .docx, .pptx, .pdf)			
					Root file berbagai size (1MB, 2MB, 3MB, 4MB, 5MB)	Root file tersimpan	Root file dengan size > 2M tidak tersimpan	✗
					Root file nama dan format yang sama	Root file tersimpan	Root file tersimpan	✓
				T	Root file size < 25 bytes	Pesan error	Pesan error	✓
Update aplikasi	1. Buka menu daftar aplikasi 2. Klik tombol update 3. Update data aplikasi 4. Klik tombol update	Nama aplikasi	1. String dengan panjang 1-255 karakter 2. Not null	V	Nama aplikasi 1 karakter	Nama aplikasi tersimpan	Nama aplikasi tersimpan	✓
					Nama aplikasi 255 karakter	Nama aplikasi tersimpan	Nama aplikasi tersimpan	✓
				T	Nama aplikasi >255 karakter	Pesan error	Pesan error	✓
		Tipe aplikasi	Id tipe aplikasi yang dipilih dari dropdown list	V	Tipe aplikasi valid	Tipe aplikasi tersimpan	Tipe aplikasi tersimpan	✓
				T	Tipe aplikasi tidak dipilih	Pesan error	Pesan error	✓
		Root file	File dengan berbagai jenis format dengan size > 24 bytes	V	Root file berbagai format (.jpg, .png, .gif, .doc, .ppt, .xls, .txt, .docx, .pptx, .pdf)	Root file tersimpan	Root file tersimpan	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
				T	Root file berbagai size (1MB, 2MB, 3MB, 4MB, 5MB)	Root file tersimpan	Root file dengan size > 2M tidak tersimpan	✗
					Root file nama dan format yang sama	Root file tersimpan	Root file tersimpan	✓
					Root file size < 25 bytes	Pesan error	Pesan error	✓
Delete aplikasi	1. Buka menu daftar aplikasi 2. Klik tombol delete 3. Klik tombol ok	Id aplikasi	1. Integer 2. Id aplikasi user yang bersangkutan 3. Id aplikasi yang pernah terdaftar	V	Id aplikasi	Data terhapus	Data terhapus	✓
						Root file terhapus (jika ada)	Root file terhapus (jika ada)	✓
View Detail Aplikasi	1. Buka menu daftar aplikasi 2. Klik view detail	Id aplikasi	1. Integer 2. Id aplikasi user yang bersangkutan 3. Id aplikasi yang pernah terdaftar	V	Id aplikasi	Data aplikasi (yang meliputi)	Data aplikasi (yang meliputi)	✓
				T	Id aplikasi tipe selain integer	Pesan error	Pesan error	✓
					Id aplikasi user lain	Pesan error	Pesan error	✓
					Id aplikasi yang belum terdaftar	Pesan error	Pesan error	✓
Request Initialization key	1. Buka menu daftar aplikasi	Id aplikasi	1. Integer 2. Id aplikasi user yang bersangkutan	V	Id aplikasi	Initializaiton key	Initializaiton key	✓
						Waktu dibuatnya initialization key	Waktu dibuatnya initialization key	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
	2. Klik view detail 3. Klik tombol request init key		3. Id aplikasi yang pernah terdaftar					
Request Synchronization key	1. Buka menu daftar aplikasi	Id aplikasi	1. Integer	V	Id aplikasi	Synchronization key	Synchronization key	✓
	2. Klik view detail		2. Id aplikasi user yang bersangkutan			Waktu dibuatnya synchronization key	Waktu dibuatnya synchronization key	✓
	3. Klik tombol request sync key		3. Id aplikasi yang pernah terdaftar			Synchronization key timeout	Synchronization key timeout	✓

Keterangan:

- V: Nilai input valid
- T: Nilai input tidak valid

Tabel V.4. Hasil pengujian proses autentikasi Client dengan Authentication Server

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
Request proses inisialisasi	1. Request init key pada sistem administrator 2. Update init key pada client 3. Klik sub menu inisialisasi pada menu uji coba fungsional	UrlToken, UnmToken, Sequence number, dan initialization key	UrlToken, UnmToken, Sequence number, dan initialization hasil generate oleh sistem	V	UrlToken, UnmToken, Sequence number, initialization key	Hasil perbandingan UrlToken, UnmToken, dan Sequence number	Hasil perbandingan UrlToken, UnmToken, dan Sequence number	✓
						UrlSeed	UrlSeed	✓
						UnmSeed	UnmSeed	✓
						Sequence number = 1	Sequence number = 1	✓
						UrlToken untuk n=1	UrlToken untuk n=1	✓
						UnmToken untuk n=1	UnmToken untuk n=1	✓
						Status aplikasi menjadi aktif	Status aplikasi menjadi aktif	✓
						Root file di sever terhapus	Root file di sever terhapus	✓
				T	UrlToken tidak sesuai	Pesan error Urltoken, Unmtoken, atau sequence number tidak sesuai.	Pesan error Urltoken, Unmtoken, atau sequence number tidak sesuai.	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
					UnmToken tidak sesuai	Pesan error	Pesan error	✓
					Sequence number tidak sesuai	Pesan error	Pesan error	✓
					Initialization key tidak sesuai	Pesan error (client tidak ditemukan)	Pesan error (client tidak ditemukan)	✓
					Initialization key expired	Pesan error (init key telah expired)	Pesan error (init key telah expired)	✓
					Root file tidak sesuai	Pesan error Urltoken, Unmtoken, atau sequence number tidak sesuai.	Pesan error Urltoken, Unmtoken, atau sequence number tidak sesuai.	✓
Request proses identifikasi	Klik sub menu identifikasi pada menu uji coba fungsional	UrlToken, UnmToken, sequence number	UrlToken, UnmToken, sequence number hasil generate oleh sistem	V	UrlToken, UnmToken, sequence number	Empat angka random	Empat angka random	✓
						Hash value dari token yang digenerate oleh keempat angka random	Hash value dari token yang digenerate oleh keempat angka random	✓
						Waktu digenerate hash value	Waktu digenerate hash value	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
						Hash value timeout	Hash value timeout	✓
						Sequence number + 1	Sequence number + 1	✓
						Next UrlToken	Next UrlToken	✓
						Next UnmToken	Next UnmToken	✓
				T	UrlToken tidak sesuai	Pesan error (client tidak ditemukan)	Pesan error (client tidak ditemukan)	✓
					UnmToken tidak sesuai	Pesan error (client tidak ditemukan)	Pesan error (client tidak ditemukan)	✓
					Sequence number tidak sesuai	Pesan error	Pesan error	✓
					Root file tidak sesuai	Pesan error (client tidak ditemukan)	Pesan error (client tidak ditemukan)	✓
Request proses autentikasi	Klik sub menu autentikasi pada menu uji coba fungsional	Hash value, UrlToken, UnmToken, sequence number	Hash value, UrlToken, UnmToken, dan sequence number hasil generate oleh sistem	V	Hash value, UrlToken, UnmToken, sequence number	Hasil perbandingan hash value	Hasil perbandingan hash value	✓
						Access token terkirim ke client	Access token terkirim ke client	✓
				T	Hash value tidak sesuai	Pesan error (Hash value tidak sesuai.)	Pesan error (Hash value tidak sesuai.)	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
					Hash value expired	Pesan error (Hash vlaue telah expired.)	Pesan error (Hash vlaue telah expired.)	✓
					UrlToken tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					UnmToken tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					Sequence number tidak sesuai	Pesan error	Pesan error	✓
Request data ke Resource Server	Klik sub menu request dadta pada menu uji coba fungsional	UrlToken, UnmToken, sequence number, Access token	UrlToken, UnmToken, sequence number, dan Access token hasil generate oleh sistem	V	UrlToken, UnmToken, sequence number, Access token	Data yang direquest	Data yang direquest	✓
				T	UrlToken tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					UnmToken tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
					Sequence number tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					Root file tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					Access token expired	Pesan error (access token telah timeout)	Pesan error (access token telah timeout)	✓
					Access token tidak sesuai	Pesan error (access Token tidak sesuai.)	Pesan error (access Token tidak sesuai.)	✓
Request proses sinkronisasi	1. Request sync key 2. Update sync key pada client 3. Klik sub menu identifikasi pada menu uji coba fungsional	Synchronization key	Synchronization key hasil generate oleh sistem	V	Synchronization key	Empat angka random	Empat angka random	✓
						Hash value dari token yang digenerate oleh keempat angka random	Hash value dari token yang digenerate oleh keempat angka random	✓
						Waktu digenerate hash value	Waktu digenerate hash value	✓
						Hash value timeout	Hash value timeout	✓

Fungsi	Skenario	Input	Batasan	Nilai input		Output yang diharapkan	Output yang dihasilkan	Lulus uji
					Hash value dari token yang digenerate oleh keempat angka random	Hasil perbandingan hash value	Hasil perbandingan hash value	✓
						Sequence number yang valid	Sequence number yang valid	✓
				T	Synchronization key tidak sesuai	Pesan error (Client tidak ditemukan)	Pesan error (Client tidak ditemukan)	✓
					Synchronization key expired	Pesan error (Synchronization key telah expired.)	Pesan error (Synchronization key telah expired.)	✓
					hash value tidak sesuai	Pesan error (Hash value tidak sesuai.)	Pesan error (Hash value tidak sesuai.)	✓
					hash value expired	Pesan error (Hash value telah expired.)	Pesan error (Hash value telah expired.)	✓

Keterangan:

- V: Nilai input valid
- T: Nilai input tidak valid

V.2.2 Uji Kinerja

Pengujian kinerja dilakukan dengan menghitung total waktu yang diperlukan pada setiap tahapan *Seed Based Authentication* untuk iterasi input yang bervariasi. Pengukuran waktu dilakukan dengan menggunakan fungsi `performance.now()` pada *javascript*. Fungsi ini mengembalikan nilai yang merepresentasikan waktu yang diukur sejak *unix time* yang memiliki satuan dalam *milliseconds* dengan akurasi hingga 5 *microseconds*. Contoh penggunaan fungsi ini dalam *source code* adalah sebagai berikut.

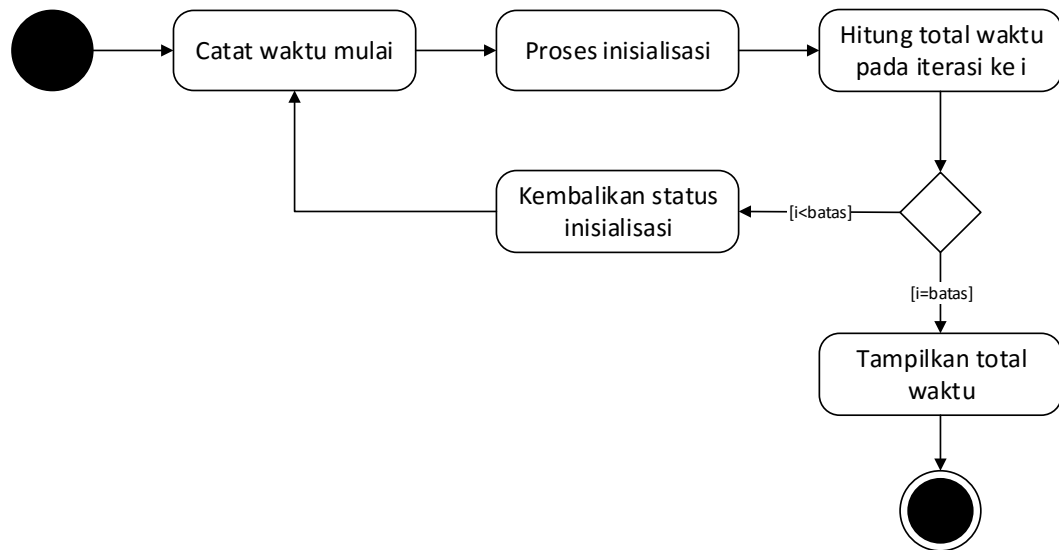
```
var t0 = performance.now();
doSomething();
var t1 = performance.now();
console.log("Fungsi doSomething membutuhkan " + (t1-t0) + "ms.");
```

V.2.2.1 Skenario Uji Kinerja

Skenario pengujian kinerja untuk setiap tahapan adalah sebagai berikut.

1. Inisialisasi

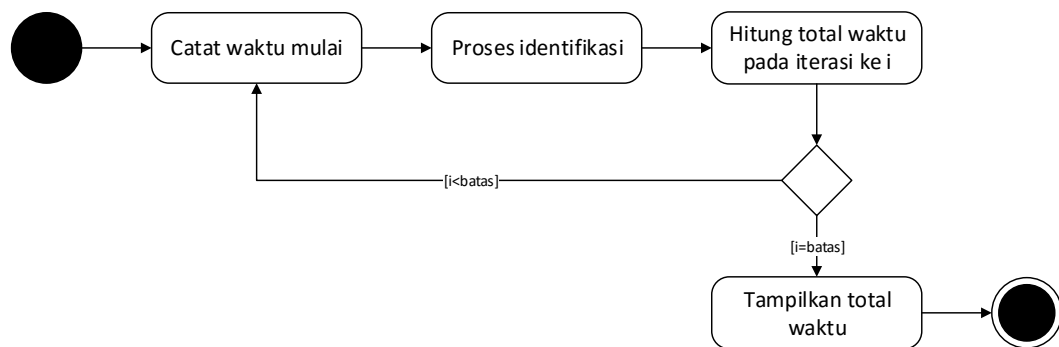
Uji kinerja pada tahap inisialisasi diawali dengan registrasi aplikasi dan *upload root file*. Setelah itu, *initialization* key disalin ke aplikasi client untuk proses inisialisasi. Selanjutnya, proses inisialisasi dilakukan antara aplikasi client dengan *authenticator* dengan skenario seperti pada Gambar V.8. Proses inisialisasi pada iterasi ke-*i* mengakibatkan status aplikasi menjadi aktif dan *root file* di *Authentication Server* dihapus sehingga proses inisialisasi tidak bisa dilakukan untuk iterasi berikutnya. Oleh karena itu, proses inisialisasi pada iterasi berikutnya memerlukan proses pengembalian status aplikasi menjadi inisialisasi dan penyalinan *root file* pada *Authentication Server*.



Gambar V.8. Skenario uji kinerja tahap inisialisasi

2. Identifikasi

Sebelum melakukan pengujian kinerja pada tahap identifikasi, penguji memasukkan jumlah iterasi. Selanjutnya, proses identifikasi dilakukan antara aplikasi client dan *authenticator* sebanyak jumlah iterasi. Uji kinerja pada tahap ini dilakukan dengan skenario yang ditunjukkan pada Gambar V.9.

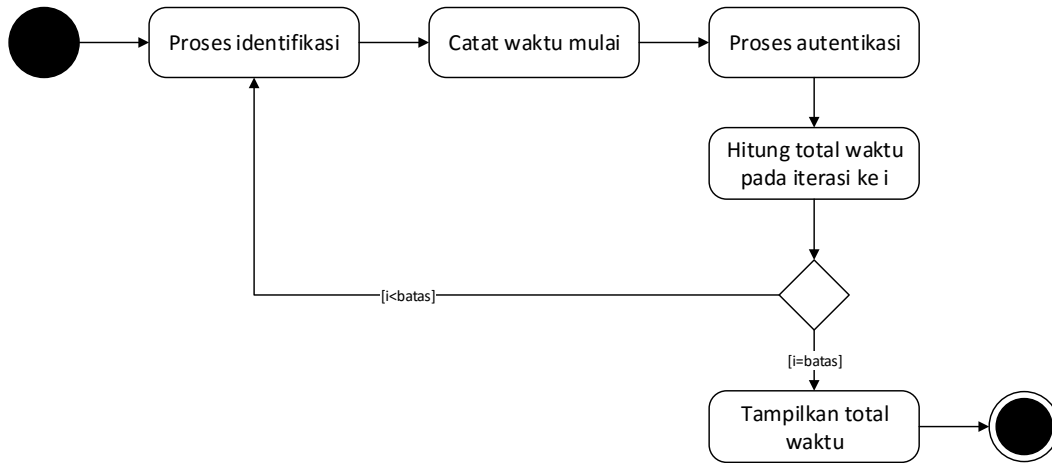


Gambar V.9. Skenario uji kinerja tahap identifikasi

3. Autentikasi

Sama seperti pada tahap identifikasi, sebelum melakukan uji kinerja untuk tahap autentikasi, penguji memasukkan jumlah iterasi dan kemudian proses autentikasi dijalankan sebanyak jumlah iterasi tersebut. Gambar V.10

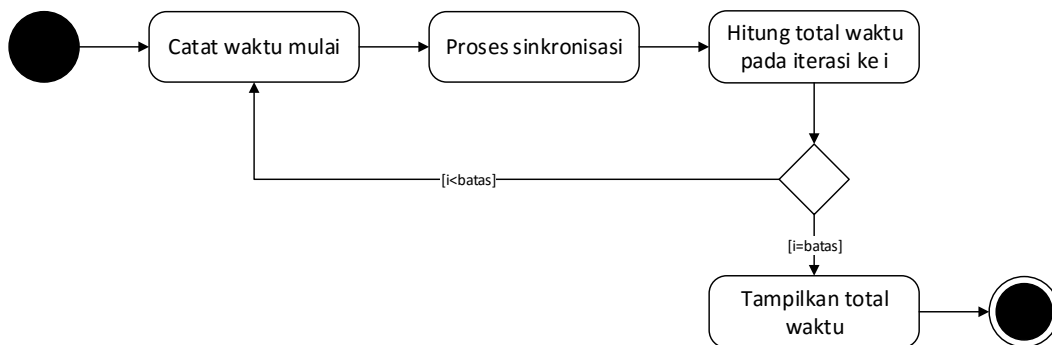
menunjukkan skenario uji kinerja untuk tahap autentikasi. Pengukuran total waktu pada tahap autentikasi selalu didahulukan dengan tahap identifikasi.



Gambar V.10. Skenario uji kinerja tahap autentikasi

4. Sinkronisasi

Uji kinerja pada tahap sinkronisasi diawali dengan *request synchronization key* melalui aplikasi administrator di *Authentication Server*. Setelah itu, informasi autentikasi pada client diperbaharui dengan *synchronization key* tersebut. Selanjutnya, uji kinerja dilakukan dengan menghitung total waktu proses sinkronisasi sesuai dengan jumlah iterasi dengan skenario seperti pada Gambar IV.7.



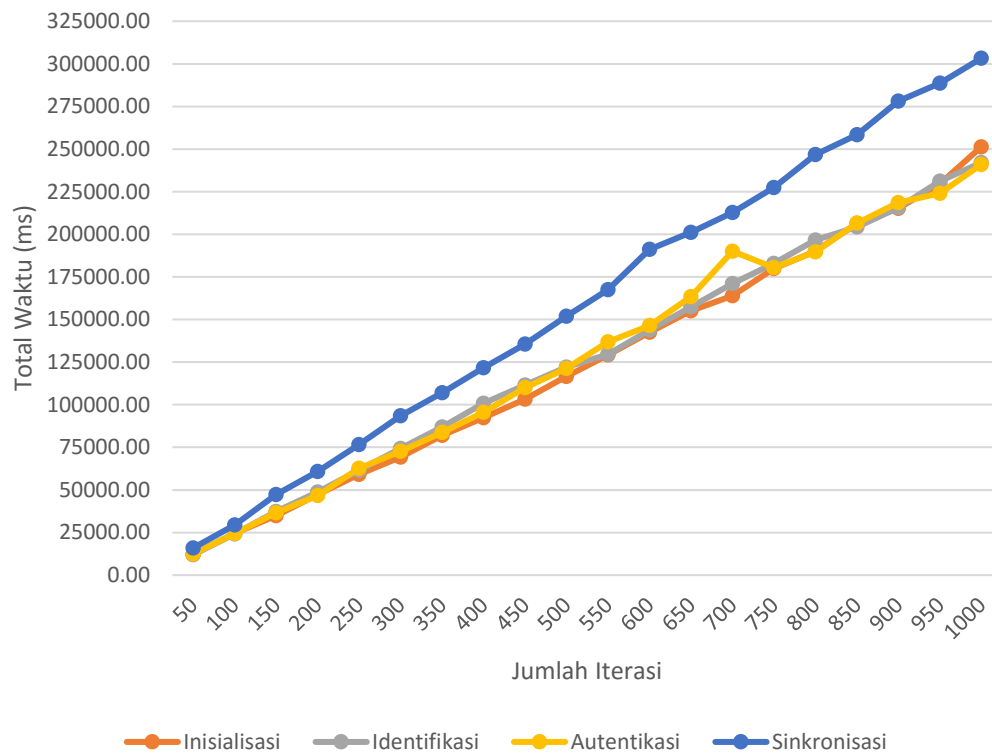
Gambar V.11. Skenario uji kinerja tahap sinkronisasi

V.2.2.2 Analisis Deskriptif

Hasil pengukuran total waktu untuk setiap tahapan diberikan pada Tabel V.5. dan Gambar V.12. Grafik menunjukkan bahwa total waktu berbanding lurus dengan jumlah iterasi. Tahap inisialisasi, identifikasi, dan autentikasi memiliki total waktu yang relatif sama. Tahap sinkronisasi memiliki total waktu yang lebih besar dibandingkan dengan yang lainnya karena pada dasarnya tahapan ini adalah gabungan antara tahap identifikasi dan autentikasi, namun memiliki tujuan yang berbeda, yaitu untuk menyamakan *sequence number* antara *Client* dan *Resource Server*.

Tabel V.5. Tabel total waktu hasil pengukuran pada uji kinerja

Jumlah Iterasi	Total Waktu (ms)			
	Inisialisasi	Identifikasi	Autentikasi	Sinkronisasi
50	12227.32	11968.16	12422.58	15919.33
100	24572.49	24185.46	24396.52	29446.32
150	34951.56	37420.11	36568.15	47201.63
200	47247.24	48820.65	46638.21	60817.17
250	59185.63	61467.80	62656.31	76565.87
300	69169.60	74279.75	72673.23	93528.86
350	82057.02	86955.32	83793.09	107091.38
400	92363.48	100876.94	95634.68	121598
450	103258.06	111546.50	110021.1	135527.94
500	116540.51	122040.79	121383.03	151852.09
550	129164.48	129525.62	136885.21	167429.23
600	142431.57	143858.40	146504.4	191229.27
650	155054.67	157426.28	163284.38	201082.42
700	163859.43	171095.38	190036.08	212815.8
750	179845.68	182965.31	180367.91	227411.3
800	189984.64	196532.52	189665.77	246760.38
850	206372.31	204374.41	206549.74	258425.96
900	215165.98	215612.68	218524.64	278066.48
950	229590.27	231123.56	224020.18	288604.39
1000	251337.86	242056.52	240979.69	303292.18



Gambar V.12. Grafik total waktu menurut jumlah iterasi

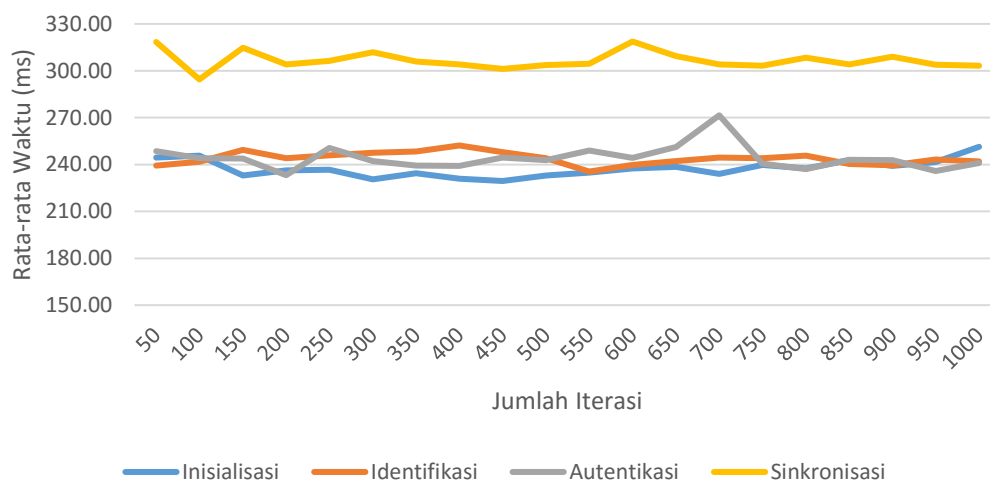
Rata-rata waktu eksekusi pada setiap tahapan diberikan pada Tabel V.6 dan Gambar V.13. Rata-rata dihitung berdasarkan persamaan (V.1).

$$Rata - rata\ waktu = \frac{Total\ waktu}{jumlah\ iterasi} \quad (V.1)$$

Tabel V.6. Tabel rata-rata waktu hasil pengukuran pada uji kinerja

Jumlah Iterasi	Rata-rata Waktu (ms)			
	Inisialisasi	Identifikasi	Autentikasi	Sinkronisasi
50	244.55	239.36	248.45	318.39
100	245.72	241.85	243.97	294.46
150	233.01	249.47	243.79	314.68
200	236.24	244.10	233.19	304.09
250	236.74	245.87	250.63	306.26
300	230.57	247.60	242.24	311.76
350	234.45	248.44	239.41	305.98
400	230.91	252.19	239.09	304.00
450	229.46	247.88	244.49	301.17

Jumlah Iterasi	Rata-rata Waktu (ms)			
	Inisialisasi	Identifikasi	Autentikasi	Sinkronisasi
500	233.08	244.08	242.77	303.70
550	234.84	235.50	248.88	304.42
600	237.39	239.76	244.17	318.72
650	238.55	242.19	251.21	309.36
700	234.08	244.42	271.48	304.02
750	239.79	243.95	240.49	303.22
800	237.48	245.67	237.08	308.45
850	242.79	240.44	243.00	304.03
900	239.07	239.57	242.81	308.96
950	241.67	243.29	235.81	303.79
1000	251.34	242.06	240.98	303.29



Gambar V.13. Grafik rata-rata waktu menurut jumlah iterasi

V.2.2.3 Analisis Regresi

Tabel V.7. Tabel output analisis regresi data uji kinerja pada tahap inisialisasi

Variables Entered/Removed ^a			
Model	Variables Entered	Variables Removed	Method
1	jumlah_iterasi ^b	.	Enter

a. Dependent Variable: total_waktu

b. All requested variables entered.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.999 ^a	.998	.998	3232.54477

a. Predictors: (Constant), jumlah_iterasi

ANOVA^a

Model	Sum of Squares	df	Mean Square	F	Sig.
1 Regression	99287939054.3	1	99287939054.3	9501.833	.000 ^b
Residual	52	18	52		
1 Total	188088221.962	19	10449345.665		
	99476027276.3	19			
	13				

a. Dependent Variable: total_waktu

b. Predictors: (Constant), jumlah_iterasi

Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	-3081.030	1501.619		-2.052	.055
	jumlah_iterasi	244.381	2.507	.999	97.477	.000

a. Dependent Variable: total_waktu

Output dari hasil analisis regresi menggunakan SPSS untuk data pengukuran kinerja pada tahap inisialisasi ditunjukkan pada tabel Tabel V.7. Output Model Summary menunjukkan bahwa nilai korelasi sebesar 0.999. Nilai koefisien determinasi (R^2) sebesar 0.998 yang berarti bahwa pengaruh variabel bebas (jumlah iterasi) terhadap variabel terikat (total waktu) sebesar 99.8%. Output Anova menunjukkan bahwa nilai F hitung sebesar 9501.833 dengan tingkat signifikansi sebesar 0.000 yang kurang dari 0.05 sehingga model regresi dapat digunakan untuk memprediksi variabel total waktu. Berdasarkan output Coefficients, persamaan regresi untuk data kinerja pada tahap inisialisasi ditunjukkan pada persamaan (V.2).

$$Y = a + bX = -3081.030 + 244.381X \quad (V.2)$$

Persamaan (V.2) menunjukkan bahwa nilai koefisien b sebesar 244.388 dan bertanda positif. Hal ini berarti bahwa setiap penambahan nilai variabel jumlah iterasi sebesar satu satuan maka nilai variabel total waktu bertambah sebesar 244.388.

Untuk mengetahui apakah ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) maka dilakukan uji signifikansi terhadap data uji kinerja pada tahap inisialisasi. Hipotesis pada uji signifikansi ini adalah sebagai berikut.

1. H_0 : Tidak ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap inisialisasi
2. H_1 : Ada pengaruh yang nyata (signifikan) dari variabel iterasi (X) terhadap variabel total_waktu (Y) pada tahap inisialisasi

Pada output analisis regresi diatas, pada bagian Coefficient, dapat diketahui bahwa nilai t hitung = 97.477 dengan nilai signifikansi $0.000 < 0.05$ sehingga dapat disimpulkan bahwa H_0 ditolak dan H_1 diterima, yang berarti bahwa ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap inisialisasi.

Tabel V.8. Tabel output analisis regresi data uji kinerja pada tahap identifikasi

Variables Entered/Removed ^a			
Model	Variables Entered	Variables Removed	Method
1	jumlah_iterasi ^b	.	Enter

a. Dependent Variable: total_waktu

b. All requested variables entered.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	1.000 ^a	.999	.999	1839.13228

a. Predictors: (Constant), jumlah_iterasi

ANOVA^a

Model	Sum of Squares	df	Mean Square	F	Sig.
1 Regression	96538498798.779	1	96538498798.779	28541.356	.000 ^b
Residual	60883336.071	18	3382407.559		
Total	96599382134.849	19			

a. Dependent Variable: total_waktu

b. Predictors: (Constant), jumlah_iterasi

Coefficients^a

Model	Unstandardized Coefficients		Standardized Coefficients	t	Sig.
	B	Std. Error	Beta		
1 (Constant)	1195.474	854.335		1.399	.179
jumlah_iterasi	240.974	1.426	1.000	168.942	.000

a. Dependent Variable: total_waktu

Tabel V.8 menunjukkan hasil output analisis regresi dari data hasil uji kinerja pada tahap identifikasi. Berdasarkan output tersebut, output Model Summary menunjukkan bahwa nilai korelasi (R) sebesar 1.000. Koefisien determinasi (R^2) menjelaskan persentase pengaruh variabel bebas, jumlah iterasi, terhadap variabel terikat, total waktu, yaitu sebesar 99.9 %. Output Anova menjelaskan tingkat signifikansi sebesar 0.000 kurang dari 0.05 yang berarti bahwa ada pengaruh yang signifikan variabel jumlah iterasi terhadap variabel total waktu sehingga model regresi dapat digunakan untuk memprediksi variabel total waktu. Berdasarkan output Coefficients, persamaan regresi dapat dibentuk seperti pada persamaan (V.3). Persamaan tersebut menunjukkan bahwa, pada tahap identifikasi, nilai konstanta sebesar 1195.474. Nilai koefisien regresi (b) sebesar 240.974 menjelaskan bahwa setiap penambahan nilai variabel jumlah iterasi sebesar satu satuan mengakibatkan penambahan nilai variabel total waktu sebesar 240.974 satuan.

$$Y = a + bX = 1195.474 + 240.974X \quad (V.3)$$

Uji signifikansi juga dilakukan terhadap data kinerja pada tahap identifikasi untuk mengetahui apakah ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y). Hipotesis pada uji signifikansi tersebut adalah sebagai berikut.

1. H_0 : Tidak ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap identifikasi
2. H_1 : Ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap identifikasi

Pada output analisis regresi diatas, pada bagian Coefficient, dapat diketahui bahwa nilai t hitung = 168.942 dengan nilai signifikansi $0.000 < 0.05$ sehingga dapat disimpulkan bahwa H_0 ditolak dan H_1 diterima, yang berarti bahwa ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap identifikasi.

Tabel V.9. Tabel output analisis regresi data uji kinerja pada tahap autentikasi

Variables Entered/Removed ^a			
Model	Variables Entered	Variables Removed	Method
1	jumlah_iterasi ^b	.	Enter

a. Dependent Variable: total_waktu

b. All requested variables entered.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.997 ^a	.995	.994	5366.71136

a. Predictors: (Constant), jumlah_iterasi

ANOVA ^a					
Model	Sum of Squares	df	Mean Square	F	Sig.

1	Regression	97661049389.7 11	1	97661049389.7 11	3390.821	.000 ^b
	Residual	518428634.814	18	28801590.823		
	Total	98179478024.5 25	19			

a. Dependent Variable: total_waktu

b. Predictors: (Constant), jumlah_iterasi

Coefficients ^a						
Model	Unstandardized Coefficients		Standardized Coefficients	t	Sig.	
	B	Std. Error	Beta			
1	(Constant)	905.701	2493.006		.363	.721
	jumlah iterasi	242.371	4.162	.997	58.231	.000

a. Dependent Variable: total_waktu

Output analisis regresi untuk data uji kinerja pada tahap autentikasi ditunjukkan pada Tabel V.9. Berdasarkan output tersebut, nilai korelasi (R) adalah sebesar 0.997. Koefisien determinasi (R²) sebesar 0.994 menunjukkan bahwa pengaruh variabel bebas, jumlah iterasi, terhadap variabel terikat, total waktu, pada tahap autentikasi adalah sebesar 99.4%. Nilai F hitung pada output Anova adalah sebesar 3390.821 dengan tingkat signifikansi sebesar 0.000 < 0.005 sehingga model regresi dapat digunakan untuk memprediksi variabel total waktu. Persamaan (V.4) adalah persamaan regresi yang dapat dibentuk berdasarkan output Coefficients.

$$Y = a + bX = 1195.474 + 240.974X \quad (V.4)$$

Persamaan (V.4) menunjukkan bahwa nilai koefisien regresi X adalah sebesar 240.974 dan bertanda positif. Hal ini menunjukkan bahwa setiap penambahan variabel bebas (jumlah iterasi) sebesar satu satuan maka nilai variabel total waktu bertambah sebesar 240.974 satuan.

Uji signifikansi dilakukan terhadap data kinerja pada tahap autentikasi dengan hipotesis sebagai berikut.

1. H_0 : Tidak ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap autentikasi
2. H_1 : Ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap autentikasi

Pada output analisis regresi diatas, dapat diketahui bahwa nilai t hitung = 58.231 dengan nilai signifikansi $0.000 < 0.05$ sehingga dapat disimpulkan bahwa H_0 ditolak dan H_1 diterima, yang berarti bahwa ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap autentikasi.

Tabel V.10. Tabel output analisis regresi data uji kinerja pada tahap sinkronisasi

Variables Entered/Removed ^a			
Model	Variables Entered	Variables Removed	Method
1	jumlah_iterasi ^b	.	Enter

a. Dependent Variable: total_waktu

b. All requested variables entered.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	1.000 ^a	.999	.999	2443.51835

a. Predictors: (Constant), jumlah_iterasi

ANOVA ^a					
Model	Sum of Squares	df	Mean Square	F	Sig.
1 Regression	154843435461.045	1	154843435461.045	25933.527	.000 ^b
Residual	107474075.078	18	5970781.949		
Total	154950909536.123	19			

a. Dependent Variable: total_waktu

b. Predictors: (Constant), jumlah_iterasi

Coefficients ^a					
Model	Unstandardized Coefficients		Standardized Coefficients	t	Sig.
	B	Std. Error	Beta		
1 (Constant)	510.294	1135.091		.450	.658
jumlah_iterasi	305.187	1.895	1.000	161.039	.000

a. Dependent Variable: total_waktu

Output analisis regresi diatas menunjukkan bahwa nilai korelasi adalah sebesar 1.000. Nilai koefisien determinasi (R^2) sebesar 0.999 berarti bahwa pengaruh variabel bebas terhadap variabel terikat adalah 99,9%. Nilai F hitung sebesar 25933.527 dengan tingkat signifikansi bernilai 0.000 yang kurang dari 0.005 menunjukkan bahwa model regresi dapat digunakan untuk memprediksi variabel total waktu. Persamaan (V.5) adalah persamaan yang dibentuk berdasar output Coefficients. Persamaan tersebut menunjukkan nilai koefisien regresi (b) sebesar 305.187 dan bertanda positif. Hal ini berarti setiap penambahan variabel total waktu sebesar satu satuan maka nilai variabel total waktu bertambah sebesar 305.187 satuan.

$$Y = a + bX = 510.294 + 305.187X \quad (V.5)$$

Uji signifikansi yang dilakukan terhadap data kinerja pada tahap sinkronisasi dilakukan untuk mengetahui apakah ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y). Hipotesis pada uji signifikansi tersebut adalah sebagai berikut.

1. H_0 : Tidak ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap sinkronisasi
2. H_1 : Ada pengaruh yang nyata (signifikan) dari variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap sinkronisasi

Pada output analisis regresi diatas, dapat diketahui bahwa nilai t hitung = 161.039 dengan nilai signifikansi $0.000 < 0.05$ sehingga dapat disimpulkan bahwa H_0 ditolak dan H_1 diterima. Hal ini berarti bahwa ada pengaruh yang nyata (signifikan) dari

variabel jumlah iterasi (X) terhadap variabel total waktu (Y) pada tahap sinkronisasi.

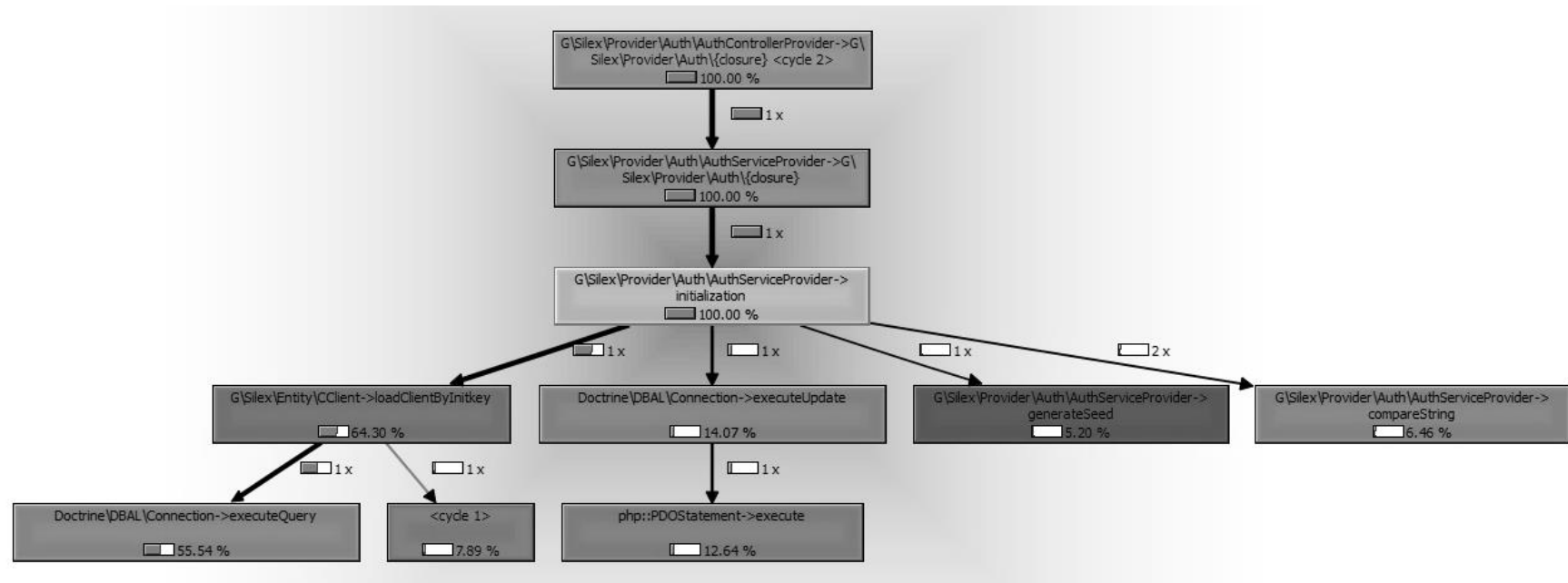
V.2.3 Xdebug

Xdebug merupakan ekstensi tambahan PHP yang menyediakan fungsi *debugging* dan *profiling*. Xdebug Profiler dapat menemukan *bottleneck* pada skrip PHP dan hasilnya dapat divisualisasikan dengan perangkat seperti KcacheGrind atau WinCacheGrind (Xdebug, n.d.).

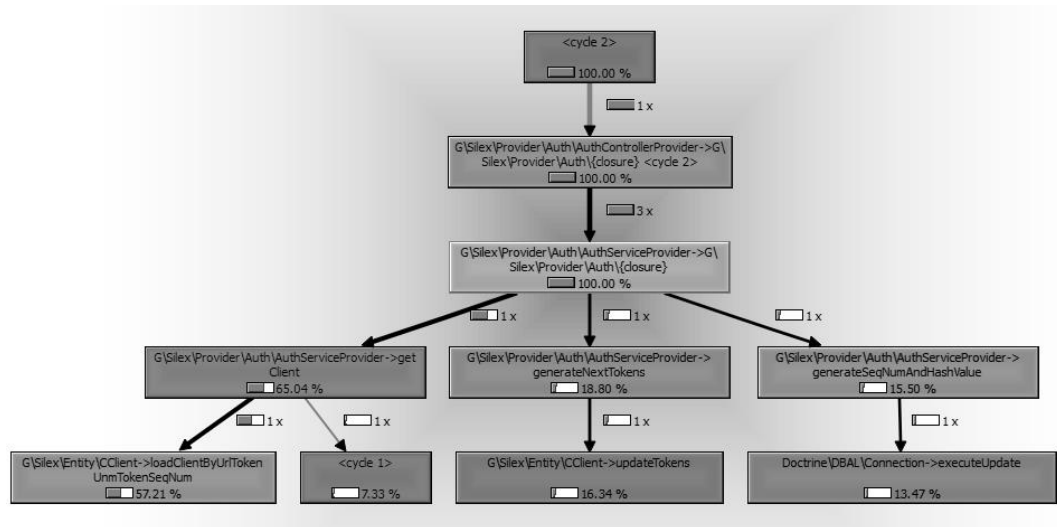
Output xdebug untuk proses inisialisasi ditunjukkan pada

Gambar V.14. Class AuthServiceProvider memiliki fungsi *initialization* yang memanggil fungsi loadClientByInitKey, compareString, executeUpdate, dan generateSeed. Dari keempat fungsi tersebut, fungsi loadClientByInitKey menghabiskan waktu terbanyak, yaitu sebesar 64.30%. Fungsi ini adalah fungsi yang mencari *client* berdasarkan *initialization key* sebelum proses inisialisasi dilakukan. Fungsi lainnya yaitu compareString sebesar 6.46%, executeUpdate sebesar 14.07%, dan generateSeed sebesar 5.20%.

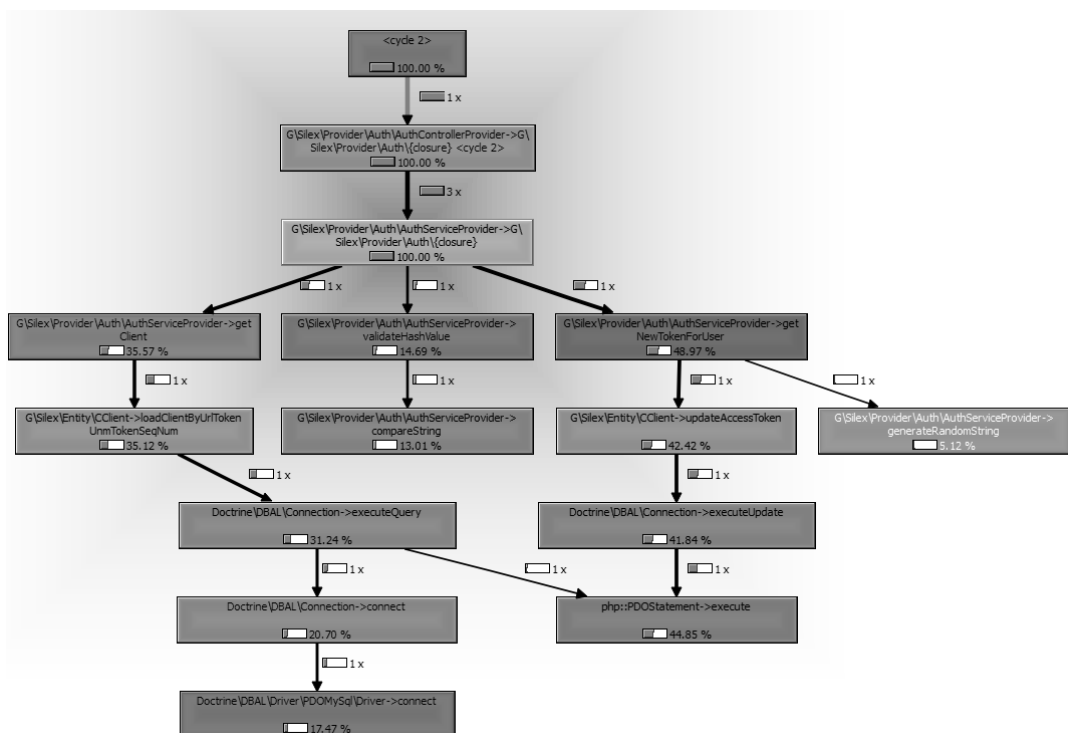
Gambar V.15 adalah *call graph* dari fungsi yang dipanggil pada proses identifikasi. Berdasarkan *call graph* tersebut, class AuthServiceProvider mengeksekusi fungsi getClient, generateSeqNumAndHashValue, dan generateNextTokens. Fungsi yang memiliki waktu eksekusi terbesar adalah fungsi getClient, yaitu sebesar 65.04%. Fungsi generateNextTokens memiliki waktu eksekusi terbesar kedua, yaitu sebesar 18.80%, dan yang terkecil adalah fungsi generateSeqNumAndHashValue, sebesar 15.50%.



Gambar V.14. *Call Graph* dari *output* xdebug pada proses inisialisasi



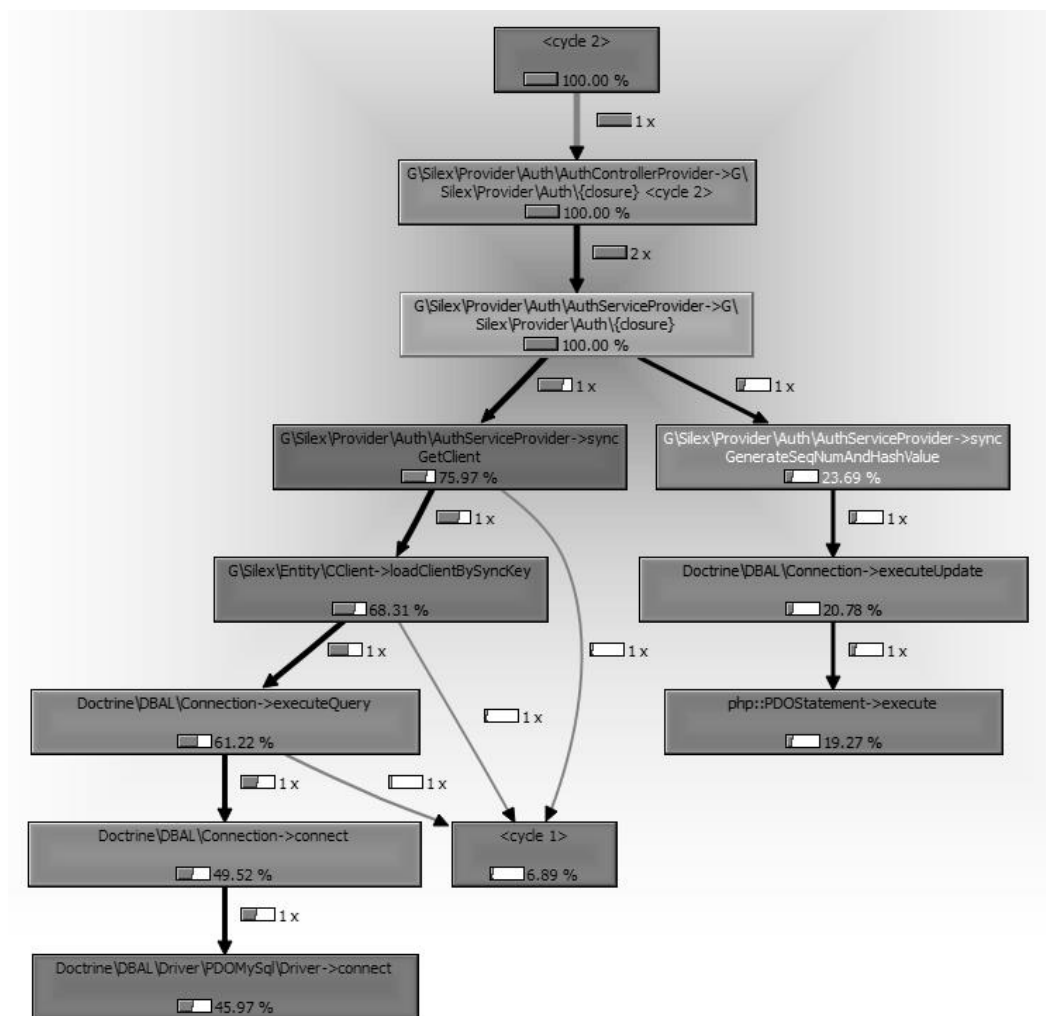
Gambar V.15. *Call Graph* dari output xdebug pada proses identifikasi



Gambar V.16. *Call Graph* dari output xdebug pada proses autentikasi

Gambar V.16 menunjukkan *call graph* dari setiap fungsi yang dipanggil dalam proses autentikasi. Berdasarkan *call graph* tersebut, pada proses autentikasi, *class* AuthControllerProvider memanggil *class* AuthServiceProvider. Terdapat tiga fungsi yang dipanggil dalam *class* AuthServiceProvider yaitu validateHashValue,

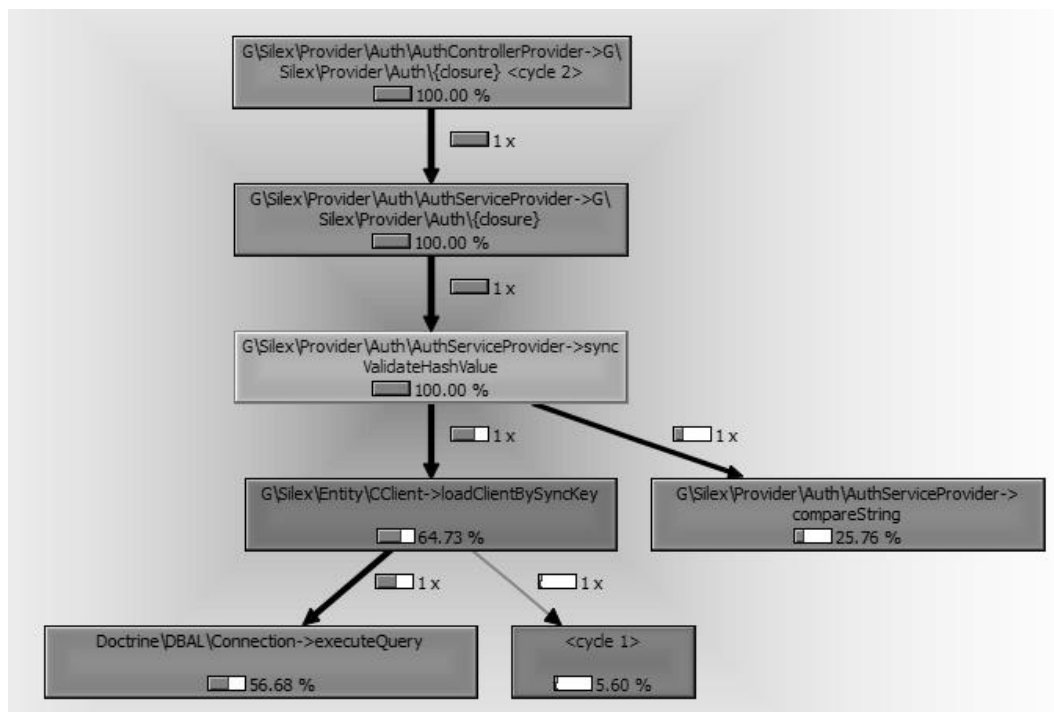
getClient, dan getNewTokenForUser. Waktu terbanyak terdapat pada fungsi getNewTokenForUser, yaitu sebesar 48.97%, kemudian diikuti oleh fungsi getClient sebesar 35.57% dan fungsi validateHashValue sebesar 14.69%. Fungsi yang menggunakan waktu terbanyak, yaitu fungsi getNewTokenForUser, memanggil dua fungsi, generateRandomString dan updateAccessToken. Dari kedua fungsi tersebut, fungsi yang menghabiskan waktu terbanyak adalah fungsi updateAccessToken, yaitu sebesar 42.42%, sedangkan generateRandomString menghabiskan waktu sebesar 5.12%.



Gambar V.17. *Call Graph* dari output xdebug pada proses sinkronisasi untuk request pertama

Tahapan sinkronisasi oleh aplikasi dilakukan dengan dua kali *request*. *Request* pertama adalah proses identifikasi client yang akan melakukan proses sinkronisasi, sedangkan *request* kedua dilakukan untuk mengirimkan *sequence number* yang valid jika autentikasi berhasil dilakukan. *Call graph* dari tahap sinkronisasi untuk *request* yang pertama ditunjukkan pada Gambar V.17. Pada *request* pertama, *class* *AuthControllerProvider* memanggil dua fungsi pada *class* *AuthServiceProvider*, yaitu fungsi *syncGetClient* dan *syncGenerateSeqNumAndHashValue*. Fungsi *syncGetClient* menghabiskan waktu eksekusi paling besar, yaitu sebesar 75.97%, sedangkan fungsi *syncGenerateSeqNumAndHashValue* sebesar 23.69%.

Gambar V.18 menunjukkan *call graph* dari proses sinkronisasi untuk *request* yang kedua. Berdasarkan *call graph* tersebut, *request* kedua memanggil fungsi *validateHashValue* pada *class* *AuthServiceProvider*. Fungsi tersebut memanggil dua fungsi yaitu fungsi *compareString* pada *class* yang sama dan fungsi *loadClientBySyncKey* pada *class* *CClient*. Fungsi *loadClientBySyncKey* menggunakan waktu eksekusi sebesar 64.73%, lebih besar dibandingkan dengan fungsi *compareString*, yaitu sebesar 25.76%.



Gambar V.18. *Call Graph* dari output xdebug pada proses sinkronisasi untuk request kedua

V.2.4 Evaluasi Ketahanan Serangan

Sebagai pengganti dari *username* dan *password* statis, *Seed Based Authentication* menggunakan token, sebagai *first factor*, yang dibangkitkan secara dinamis setiap kali proses autentikasi dimulai. Hal ini menjadikan metode ini menyediakan tingkat keamanan yang lebih tinggi dibandingkan *HTTP Basic Authentication* and *HTTP Digest Authentication* (Nassar dan Chen, 2015).

Metode ini baik dalam menghadapi serangan keamanan *dictionary attack* karena dalam proses autentikasi token selalu berubah-ubah dan tidak ada kata di dalam kamus yang sesuai dengan token tersebut. Selain itu, *brute force attack* menjadi lebih sulit dilakukan pada metode ini dibandingkan dengan metode autentikasi yang menggunakan *password* statis.

Implementasi PRNG untuk membangkitkan token dalam prototipe menggunakan fungsi `mt_rand()` dalam PHP. Fungsi `mt_rand()` merupakan fungsi PRNG untuk membangkitkan *random value* melalui Mersenne Twister *Random Number Generator* (Matsumoto dan Nishimura, 1998). Fungsi `mt_rand()` merupakan pengganti fungsi yang lebih lama, yaitu `rand()`, yang memiliki karakteristik yang tidak jelas, meragukan, dan lambat. Fungsi ini dapat menghasilkan *random number* empat kali lebih cepat dibandingkan dengan fungsi `rand()`.

PHP mengimplemetasikan versi 32 bit dari Mersenne Twister sehingga memiliki 2^{31} kombinasi yang mungkin, yaitu sebanyak 2.147.483.648 angka acak. Peretasan metode *Seed Based Authentication* dilakukan terhadap kombinasi dua token, yaitu URLToken dan UNMToken, dan *sequence number*. Oleh karena itu, kombinasi yang dihasilkan oleh dua token dan 1 digit *sequence number*, yaitu sebanyak $2^{31 \times 2} \times 10^1 \approx 4.6 \times 10^{16}$. Jika dibandingkan dengan metode autentikasi konvensional yang menggunakan *password* statis sebanyak 6 karakter, kombinasi yang mungkin adalah sebanyak $94^6 \approx 6.9 \times 10^{11}$. Hal ini menjadikan metode yang diusulkan memiliki kombinasi 6.7×10^7 kali lebih banyak sehingga lebih aman. Disamping itu, *attacker* memiliki waktu yang terbatas untuk melakukan peretasan

sebelum token dan *sequence number* berubah secara dinamis pada proses autentikasi berikutnya.

V.2.5 Perbandingan

Lee dkk. (2015) mengusulkan penggunaan *ID-Based Authentication* (IBA) untuk mengatasi keterbatasan *RESTful web service* dalam sistem autentikasi. *ID-Based Authentication* merupakan metode autentikasi yang didasari oleh *Boneh-Franklin ID-Based Encryption*. *ID-Based Encryption* merujuk pada algoritma *key asymmetric*. *Private key* dihasilkan oleh *Private Key Generator* (PKG). PKG merupakan pihak ketiga yang dapat melakukan komputasi untuk menghasilkan *private key* client dari ID unik seperti alamat email. Proses enkripsi menggunakan *public key* yang dibangkitkan dari *master public key* dan email penerima.

Setiap *resource* dalam *RESTful web service* dapat direpresentasikan oleh URI yang unik. *ID-Based Authentication* pada *RESTful web service* menggunakan *ID-Based Encryption* dengan REST URI sebagai ID unik. Proses autentikasi antara client dan *Resource Server* diawali dengan membuat *private key* masing-masing dan *public key* yang lainnya. Setelah itu, client melakukan enkripsi pesan dengan *public key* *Resource Server*. Pesan tersebut dienkrip kembali dengan *private key* client untuk *digital signature*. Setelah pesan diterima, *Resource Server* memverifikasi pesan dengan *public key* client dan kemudian melakukan dekripsi dengan *private key* *Resource Server*.

Metode *ID-Based Authentication* dapat mengatasi permasalahan konsep *stateless* pada *RESTful web service* dan isu keamanan dalam proses autentikasinya. Kelebihan metode ini secara rinci adalah sebagai berikut.

1. Selama proses autentikasi metode ini tidak menyimpan *state* dari client sehingga REST tetap memiliki atribut yang *statelessness*
2. Proses autentikasi tidak memerlukan *username* dan *password*
3. Metode ini lebih aman dibandingkan dengan *HTTP Basic Authentication* dan *HTTP Digest Authentication*

4. Tidak seperti halnya dengan sistem autentikasi saat ini yang menyimpan status client dan memiliki kerentanan terhadap terbukanya informasi client, metode ini diklaim dapat menyelesaikan isu tersebut.
5. Proses enkripsi pesan dilakukan dengan dua tahap yaitu enkripsi dengan *public key* penerima dan penandatanganan dengan *private key* pengirim pesan.

Secara umum, kriptografi asimetris memiliki kelemahan yaitu sebagai berikut.

1. Skema *public key* rentan terhadap *brute force attack*
2. Secara komputasi, metode ini lebih berat dibandingkan dengan kunci simetris. Namun, hal ini dapat ditangani dengan penggunaan *session key* selama komunikasi berlangsung. Proses *sharing session key* dilakukan dengan skema *public key*, sedangkan proses enkripsi dan dekripsi pesan dilakukan dengan skema kunci simetris, menggunakan *session key* tersebut.

ID-based Encryption tetap memiliki beberapa kelemahan diantaranya sebagai berikut.

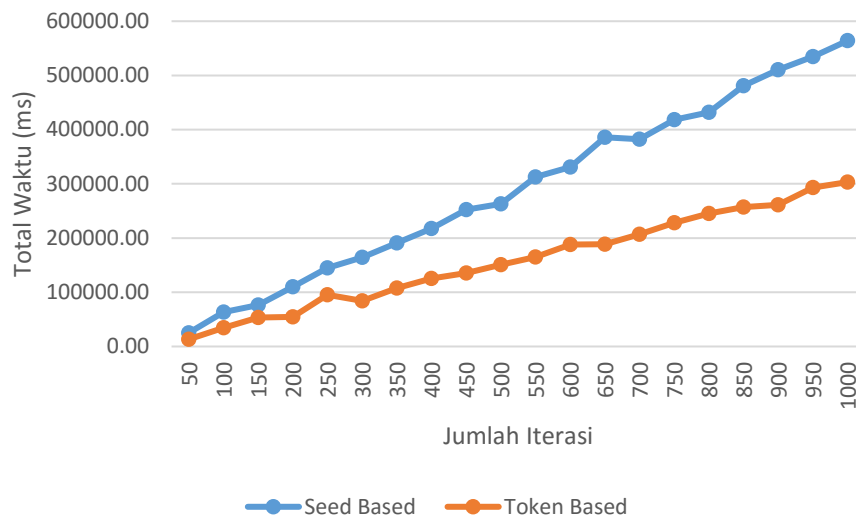
1. Jika PKG diserang maka semua pesan yang dienkripsi dengan pasangan *public-private key* dari server PKG menjadi tidak aman. Untuk mengatasinya, master *private-public key* dapat diperbaharui dengan pasangan kunci yang baru. Namun demikian, hal ini akan menjadikan masalah baru, yaitu permasalahan *key management*, karena setiap pengguna harus memiliki *public key* terbaru.
2. PKG dapat mendekripsi semua pesan tanpa otorisasi dari pengguna karena PKG yang membuat *private key*. Hal ini berarti IBE tidak dapat digunakan untuk membuktikan permasalahan *non-repudiation*.
3. Saluran yang aman diperlukan antara pengguna dan PKG untuk mentransmisikan *private key*. Selain itu, diperlukan autentikasi tambahan antara pengguna dan PKG.
4. IBE tidak aman terhadap serangan *code breaking quantum*

Pada subbab sebelumnya telah dijelaskan bagaimana metode *Seed Based Authenticon* memiliki keamanan lebih dibandingkan dengan metode *HTTP Basic Authenticon* dan *HTTP Basic Authenticon*. Begitu juga dengan metode *Token*

Based Authentication dikarenakan metode ini tetap menggunakan *username* dan *password* statis sebagai *first factor* dalam proses autentikasi. Berikutnya, simulasi dilakukan untuk membandingkan kinerja antara metode usulan dengan metode autentikasi *Token Based Authentication*. Pengukuran kinerja pada metode autentikasi *Token Based Authentication* dilakukan dengan membuat aplikasi client dan *service provider* dengan proses autentikasi menggunakan *Token Based Authentication*. Tabel V.11 menunjukkan hasil pengukuran kinerja pada *web service* yang menggunakan metode *Token Based Authentication* dan *Seed Based Authenticon* dalam proses autentikasinya.

Tabel V.11. Tabel total dan rata-rata waktu eksekusi *Token Based Authentication* dan *Seed Based Authenticon*

Iterasi	Total waktu (ms)		Rata-rata (ms)	
	Seed Based	Token Based	Seed Based	Token Based
50	25246.32	13303.16	504.93	266.06
100	63182.29	34350.09	631.82	343.50
150	76522.02	53205.44	510.15	354.70
200	110329.52	54614.45	551.65	273.07
250	144640.93	95192.19	578.56	380.77
300	164435.33	84198.81	548.12	280.66
350	190697.67	107677.41	544.85	307.65
400	217358.47	125449.43	543.40	313.62
450	252468.73	135405.31	561.04	300.90
500	262977.22	150830.74	525.95	301.66
550	312518.27	165240.65	568.22	300.44
600	330778.38	188069.97	551.30	313.45
650	386065.47	188741.33	593.95	290.37
700	382324.14	206962.20	546.18	295.66
750	418234.97	228130.09	557.65	304.17
800	432140.15	245400.92	540.18	306.75
850	480830.74	257334.46	565.68	302.75
900	510569.51	261051.42	567.30	290.06
950	534728.00	293326.23	562.87	308.76
1000	564072.73	303243.50	564.07	303.24



Gambar V.19. Grafik perbandingan total waktu eksekusi *Token Based Authentication* dan *Seed Based Authentication*

Gambar V.19 menunjukkan grafik perbandingan total waktu proses autentikasi *web service* dengan *Token Based Authentication* dan *Seed Based Authentication*. Dapat dilihat pada grafik, metode autentikasi dengan *Seed Based Authentication* lebih lama dalam proses autentikasi. Total waktu bisa mencapai dua kali lipat dibandingkan dengan metode *Token Based Authentication*. Sehingga dapat disimpulkan bahwa walaupun metode *Seed Based Authentication* lebih aman, namun proses autentikasi memakan waktu yang lebih lama dibandingkan dengan metode *Token Based Authentication*, metode yang saat ini banyak digunakan pada *web service* dalam proses autentikasi.

Bab VI Kesimpulan dan Saran

VI.1 Kesimpulan

Setelah proses perancangan, implementasi, dan pengujian dilakukan maka beberapa kesimpulan dapat diambil, antara lain sebagai berikut.

1. Perancangan proses autentikasi pada *RESTful web service* mengombinasikan metode *Token Based Authentication* dan *Seed Based Authentication*. Hal ini dilakukan karena kelebihan *Token Based Authentication* dalam menjaga konsep *stateless* pada *RESTful web service* dan kelebihan *Seed Based Authentication* dalam mengatasi permasalahan autentikasi yang dilakukan dengan *password* dan *username* yang statis.
2. Implementasi hasil perancangan dilakukan dengan menyusun prototipe yang terbagi menjadi dua bagian yaitu *Authentication Server* dan *Resource Server*. *Authentication Server* memiliki dua aplikasi yaitu aplikasi administrator untuk pengelolaan aplikasi oleh developer dan *authenticater* untuk proses autentikasi antara client dan *service provider*. *Resource Server* berfungsi sebagai penyedia API bagi client untuk mengakses *resource*.
3. Secara umum, *black box testing* menunjukkan bahwa fungsi-fungsi dalam prototipe telah sesuai dengan spesifikasi pada perancangan dan berfungsi dengan baik. Permasalahan pada sistem terjadi ketika *upload root file* yang melebihi batas maksimal ukuran *file* yang terdapat dalam *file* konfigurasi *php.ini*. Selain itu, dari berbagai macam format *root file* yang dilakukan *upload*, format *.doc* dan *.ppt* menghasilkan *UNMToken* yang sama.
4. Uji kinerja menunjukkan bahwa tahap inisialisasi, identifikasi, dan autentikasi memiliki total waktu dan rata-rata waktu yang relatif sama. Tahap sinkronisasi memiliki total waktu dan rata-rata waktu yang lebih besar dibandingkan dengan tahapan lainnya. Analisis regresi menunjukkan bahwa jumlah iterasi berpengaruh signifikan terhadap total waktu untuk keempat tahapan tersebut.
5. Pengujian kinerja dengan Xdebug menunjukkan bahwa fungsi-fungsi dalam kode program bekerja dengan baik, tidak ada *bottleneck* pada skrip PHP. Pada setiap tahapan autentikasi, distribusi terbesar waktu eksekusi fungsi terdapat pada fungsi yang berhubungan dengan koneksi ke *database*, yaitu fungsi

executeUpdate(), getClient(), getNewTokenForUser(), syncGetClient(), dan loadClientBySyncKey().

6. Sebagai pengganti *username* dan *password*, *Seed Based Authentication* menggunakan token yang dibangkitkan setiap kali proses autentikasi dilakukan. Hal ini menjadikan metode ini lebih aman dibandingkan *HTTP Basic Authentication* dan *HTTP Digest Authentication*. Selain itu, metode ini baik dalam menghadapi serangan keamanan *dictionary attack* karena token selalu berubah-ubah dan tidak ada kata di dalam kamus yang sesuai dengan token tersebut. *Brute force attack* juga menjadi lebih sulit dilakukan pada metode ini dibandingkan dengan metode autentikasi yang menggunakan password statis.
7. Jika *Seed Based Authentication* dibandingkan dengan metode *ID-Based Authentication*, masing-masing memiliki kelebihan dan kekurangan. *ID-Based Authentication* melakukan enkripsi dengan dua tahap yaitu enkripsi dengan *public key* penerima dan penandatanganan dengan *private key* pengirim pesan. Namun demikian, metode ini memiliki kelemahan karena melibatkan pihak ketiga, yaitu *Private Key Generator* (PKG), sehingga diperlukan autentikasi tambahan. Selain itu, metode ini masih rentan terhadap *brute force attack*.
8. Meskipun lebih baik dibandingkan *Token Based Authentication* dari sisi keamanan, *Seed Based Authentication* memiliki waktu proses yang lebih lama. Perbedaanannya bisa mencapai dua kali lipat dibandingkan dengan waktu proses autentikasi pada *Token Based Authentication*.

VI.2 Saran

Saran untuk penelitian selanjutnya adalah penerapan *Seed Based Authentication* pada *RESTful web service* yang digunakan pada aplikasi IoT, misalnya untuk komunikasi antara beberapa sensor.

DAFTAR PUSTAKA

- Aihkisalo, T., dan Paaso, T. (2012): Latencies of service invocation and processing of the rest and soap web service interfaces, *Services (SERVICES), 2012 IEEE Eighth World Congress on*, 100–107.
- Alghamdi, T. A., Lasebae, A., dan Aiash, M. (2013): Security analysis of the constrained application protocol in the Internet of Things, *Future Generation Communication Technology (FGCT), 2013 second international conference on*, 163–168.
- Blessing, L. T. M., Chakrabarti, A., dan Blessing, L. T. M. (2009): *DRM, a design research methodology*, London: Springer.
- Canvel, B., Hiltgen, A., Vaudenay, S., dan Vuagnoux, M. (2003): Password interception in a SSL/TLS channel, *Annual International Cryptology Conference*, 583–599.
- Cassola, A., Robertson, W. K., Kirda, E., dan Noubir, G. (2013): A Practical, Targeted, and Stealthy Attack Against WPA Enterprise Authentication., NDSS.
- Dell’Amico, M., Michiardi, P., dan Roudier, Y. (2010): Password strength: An empirical analysis, *INFOCOM, 2010 Proceedings IEEE*, 1–9.
- Feng, X., Shen, J., dan Fan, Y. (2009): REST: An alternative to RPC for Web services architecture, *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, 7–10.
- Fielding, R. T. (2000): *Architectural styles and the design of network-based software architectures*, Disertasi Program Doktor, University of California, Irvine.
- Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., dan Stewart, L. (1999): HTTP authentication: Basic and digest access authentication.
- Inglesant, P. G., dan Sasse, M. A. (2010): Studying password use in the wild: practical problems and possible solutions.
- Inglesant, P. G., dan Sasse, M. A. (2010): The true cost of unusable password policies: password use in the wild, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 383–392.
- Jo, J., Kim, Y., dan Lee, S. (2014): Mindmetrics: Identifying users without their login IDs, *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, 2121–2126.
- Joshi, Y., Das, D., dan Saha, S. (2009): Mitigating man in the middle attack over secure sockets layer, *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, 1–5.
- Kumar, D., Ryu, Y., dan Kwon, D. (2008): A survey on biometric fingerprints: The cardless payment system, *2008 International Symposium on Biometrics and Security Technologies*, 1–6.
- Lee, S., Jo, J.-Y., dan Kim, Y. (2015): Method for secure RESTful web service, *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*, 77–81.

- Matsumoto, M., dan Nishimura, T. (1998): Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation*, 3-30.
- Miles, R., dan Hamilton, K. (2006): *Learning UML 2.0*, 1st ed, Beijing ; Sebastopol, CA: O'Reilly.
- Nassar, N., dan Chen, L.-C. (2015): Seed-based authentication, *Collaboration Technologies and Systems (CTS), 2015 International Conference on*, 345–350.
- Peng, D., Li, C., dan Huo, H. (2009): An extended usernametoken-based approach for REST-style web service security authentication, *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, 582–586.
- Patton, R. (2001): *Software Testing*, Sams Publishing.
- Salmon, J. K., Moraes, M. A., Dror, R. O., dan Shaw, D. E. (2011): Parallel random numbers: as easy as 1, 2, 3, *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, 1–12.
- Stallings, W. (2011): *Cryptography and Network Security Principles and Practice*, 5th ed, New York: Pearson Education, Inc.
- Silex (2016): *The Silex Book*, diambil dari <https://silex.sensiolabs.org/doc/2.0/>.
- Vapen, A., Byers, D., dan Shahmehri, N. (2010): 2-clickAuth Optical Challenge-Response Authentication, *2010 International Conference on Availability, Reliability and Security*, 79–86.
- W3C (2004): Web Services Architecture, diambil 12 Oktober 2016, dari <https://www.w3.org/TR/ws-arch/>.
- Xdebug (n.d.): Xdebug: Documentation, diambil 19 April 2017, dari <https://xdebug.org/docs/>.
- Yun, J., Abowd, G., Woo, W., dan Ryu, J. (2007): Biometric User Identification with Dynamic Footprint, 225–230.
- Yii (n.d.): Yii PHP Framework: Best for Web 2.0 Development, diambil 17 Maret 2017, dari <http://www.yiiframework.com/>.
- Zhang, F., Kondoro, A., dan Muftic, S. (2012): Location-Based Authentication and Authorization Using Smart Phones, *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 1285–1292.

LAMPIRAN

Halaman ini sengaja dikosongkan

Lampiran A Source Code AuthControllerProvider.php Aplikasi
Authenticator pada Authenticater Server

```
<?php

namespace G\Silex\Provider\Auth;

use
Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;
use Symfony\Component\HttpFoundation\Request;
use Silex\ControllerProviderInterface;
use Silex\Application;
use G\Silex\Entity\Client;
use G\Silex\Entity\CClient;

class AuthControllerProvider implements
ControllerProviderInterface
{
    const INISIALISASI = '/inisialisasi';
    const IDENTIFIKASI = '/identifikasi';
    const AUTENTIKASI = '/autentikasi';
    const SINKRONISASI = '/sinkronisasi';

    private $baseRoute;

    public function setBaseRoute($baseRoute)
    {
        $this->baseRoute = $baseRoute;
        return $this;
    }

    public function connect(Application $app)
    {
        $this->setUpMiddlewares($app);
        return $this->extractControllers($app);
    }

    private function extractControllers(Application $app)
    {
        $controllers = $app['controllers_factory'];

        // proses inisialisasi
        $controllers->post(self::INISIALISASI, function
(Request $request) use ($app) {
            $unmtoken = $request->get('unmtoken');
            $urltoken = $request->get('urltoken');
            $seqnum = $request->get('seqnum');
            $initkey = $request->get('initkey');

            return $app-
>json($app[AuthServiceProvider::AUTH_INITIALIZATION]($unmtoken,
$urltoken, $seqnum, $initkey));
        });

        // proses identifikasi dan autentikasi
        $controllers-
>post(self::IDENTIFIKASI.'/{urltoken}/{unmtoken}/{seqnum}',
```

```

function (Request $request, $urltoken, $unmtoken, $seqnum) use
($app) {

    $client =
$app[AuthServiceProvider::AUTH_FIND_CLIENT]($urltoken, $unmtoken,
$seqnum);

    if(isset($client))
        $client2 =
$app[AuthServiceProvider::AUTH_NEXT_TOKENS]($client);

    return $app->json([
        'status' => true,
        'info' =>
['n'=>$app[AuthServiceProvider::AUTH_RAND_SEQ_NUM]($client2)]
    ]);
});

$controllers-
>post(self::AUTENTIKASI.'/{urltoken}/{unmtoken}/{seqnum}',
function (Request $request, $urltoken, $unmtoken, $seqnum) use
($app) {

    $hashvalue = $request->get('hashvalue');
    $cclient = new CClient($app['db']);
    $client =
$app[AuthServiceProvider::AUTH_FIND_CLIENT]($urltoken, $unmtoken,
$seqnum);

    $status =
$app[AuthServiceProvider::AUTH_VALIDATE_HASH_VALUE]($hashvalue,
$client);

    return $app->json([
        'status' => $status,
        'info' => $status ? ['token' =>
$app[AuthServiceProvider::AUTH_NEW_TOKEN]($client)] : []
    ]);
});

// proses sinkronisasi
$controllers->post(self::SINKRONISASI, function
(Request $request) use ($app) {
    $hashvalue = $request->get('hashvalue');
    $unmtoken = $request->get('unmtoken');
    $urltoken = $request->get('urltoken');
    $seqnum = $request->get('seqnum');
    $synckey = $request->get('synckey');

    if(isset($hashvalue))
    {
        $status =
$app[AuthServiceProvider::SYNC_VALIDATE_HASH_VALUE]($hashvalue,
$synckey);

        return $app->json($status);
    }
    else{
        $client =
$app[AuthServiceProvider::SYNC_FIND_CLIENT]($synckey);
        return $app->json([
            'status' => $client?true:false,

```

```

                                'info'    =>
$client?['n']==>$app[AuthServiceProvider::SYNC RAND_SEQ_NUM] ($client
):[]
                                []);
                                }
                                });

    return $controllers;
}

private function setUpMiddlewares(Application $app)
{
    $app->before(function (Request $request) use ($app) {
        $app->register(new
AuthServiceProvider($app['db']));
    });
}
}

```

Lampiran B *Source Code AuthServiceProvider.php Aplikasi Authenticater pada Authenticater Server*

```
<?php

namespace G\Silex\Provider\Auth;

use Silex\Application;
use Silex\ServiceProviderInterface;
use G\Silex\Entity\CCClient;
use Doctrine\DBAL\Connection;
use
Symfony\Component\HttpFoundation\Exception\AccessDeniedHttpException;

class AuthServiceProvider implements ServiceProviderInterface
{
    const AUTH_FIND_CLIENT = 'auth.find.client';
    const AUTH_NEXT_TOKENS = 'auth.next.tokens';
    const AUTH_VALIDATE_HASH_VALUE = 'auth.validate.hash.value';
    const AUTH_NEW_TOKEN = 'auth.new.token';
    const AUTH_INITIALIZATION = 'auth.initialization';
    const AUTH_RAND_SEQ_NUM = 'auth.rand.seq.num';
    const SYNC_FIND_CLIENT = 'sync.find.client';
    const SYNC_VALIDATE_HASH_VALUE = 'sync.validate.hash.value';
    const SYNC_RAND_SEQ_NUM = 'sync.rand.seq.num';

    private $conn;

    public function __construct(Connection $conn)
    {
        $this->conn = $conn;
    }

    public function register(Application $app)
    {
        $app[self::AUTH_FIND_CLIENT] = $app->protect(function
($urltoken, $unmtoken, $n) {
            return $this->getClient($urltoken, $unmtoken, $n);
        });

        $app[self::AUTH_NEXT_TOKENS] = $app->protect(function
($client) {
            return $this->generateNextTokens($client);
        });

        $app[self::AUTH_VALIDATE_HASH_VALUE] = $app-
>protect(function ($hashvalue, $client) {
            return $this->validateHashValue($hashvalue, $client);
        });

        $app[self::AUTH_NEW_TOKEN] = $app->protect(function
($client) {
            return $this->getNewTokenForUser($client);
        });

        $app[self::AUTH_INITIALIZATION] = $app->protect(function
($unmtoken, $urltoken, $seqnum, $initkey) {
```



```

        return $this->initialization($unmtoken, $urltoken,
$seqnum, $initkey);
    });

    $app[self::AUTH RAND_SEQ_NUM] = $app->protect(function
($client) {
        return $this->generateSeqNumAndHashValue($client);
    });

    $app[self::SYNC_FIND_CLIENT] = $app->protect(function
($synckey) {
        return $this->syncGetClient($synckey);
    });

    $app[self::SYNC_VALIDATE_HASH_VALUE] = $app-
>protect(function ($hashvalue, $synckey) {
        return $this->syncValidateHashValue($hashvalue,
$synckey);
    });

    $app[self::SYNC RAND_SEQ_NUM] = $app->protect(function
($client) {
        return $this->syncGenerateSeqNumAndHashValue($client);
    });
}

public function boot(Application $app)
{
}

private function initialization($unmtoken, $urltoken,
$seqnum, $initkey)
{
    $cclient = new CClient($this->conn);
    $client = $cclient->loadClientByInitkey($initkey);
    $created_at = $client->getInitKeyT();
    $timeout = $client->getInitKeyTo();

    if($client->getStatusId()==1)
        throw new AccessDeniedHttpException('Proses
inisialisasi telah dilakukan. Client berstatus aktif.');
```

else if(time() - \$created_at >= \$timeout)
 throw new
AccessDeniedHttpException('Initialization key telah expired.');

else
 {
 \$root_file = \$client->getRootFile();
 \$path = 'file/temp/'.\$root_file;

if(file_exists(\$path))
 {
 // Generate seed dan token untuk proses
inisialisasi

\$seed = \$this->generateSeed(\$root_file);
 \$seqnum_s = 0;
 \$unmtoken_s = \$this-
>generateToken(\$seed['unmseed'],\$seqnum_s);
 \$urltoken_s = \$this-
>generateToken(\$seed['urlseed'],\$seqnum_s);

```

        // Membandingkan token client dan server
        if($this->compareString($unmtoken,
$unmtoken_s) && $this->compareString($urltoken, $urltoken_s) &&
$seqnum==$seqnum_s)
        {
            // Hapus root file
            unlink($path);

            // Generate next token
            $next_seqnum = 1;
            $next_urltoken = $this-
>generateToken($seed['urlseed'],$next_seqnum);
            $next_unmtoken = $this-
>generateToken($seed['unmseed'],$next_seqnum);

            // Simpan next token dan sequence
            number
            $cclient = new CClient($this->conn);
            $cclient-
>updateTokenSeqNumAndStatus($seed['unmseed'], $seed['urlseed'],
$next_seqnum, $next_unmtoken, $next_urltoken, $initkey);

            return true;
        }
        else
            throw new
AccessDeniedHttpException('Urltoken, Unmtoken, atau sequence
number tidak sesuai.');
```

```

        }
        throw new AccessDeniedHttpException('Root file
tidak ditemukan.');
```

```

    }

    private function getClient($urltoken, $unmtoken, $seqnum)
    {
        $cclient = new CClient($this->conn);
        $client = $cclient-
>loadClientByUrlTokenUnmTokenSeqNum($urltoken, $unmtoken,
$seqnum);

        return $client;
    }

    private function generateNextTokens($client)
    {
        $unmtoken = $client->getUnmToken();
        $urltoken = $client->getUrlToken();
        $seqnum = $client->getSeqNum();

        $next_unmtoken = $this->generateToken($client-
>getUnmSeed(), $client->getSeqNum() + 1);
        $next_urltoken = $this->generateToken($client-
>getUrlSeed(), $client->getSeqNum() + 1);

        // update sequence number dan username token and url
        token
        $cclient = new CClient($this->conn);

```

```

        $cclient->updateTokens($urltoken, $unmtoken, $seqnum,
$next_unmtoken, $next_urltoken);

        $client->setUnmToken($next_unmtoken);
        $client->setUrlToken($next_urltoken);
        $client->setSeqNum($seqnum + 1);

        return $client;
    }

    private function generateSeqNumAndHashValue($client)
    {
        $client = $client;
        $unm_seed = $client->getUnmSeed();
        $url_seed = $client->getUrlSeed();
        $unmtoken = $client->getUnmToken();
        $urltoken = $client->getUrlToken();
        $seqnum = $client->getSeqNum();

        $n1 = rand();
        $n2 = rand();
        $n3 = rand();
        $n4 = rand();

        $hashvalue_s = $this->generateTokenHash($unm_seed,
$url_seed, $n1,$n2,$n3,$n4);
        $token_hash_t = time();
        $token_hash_to = 1000;

        $cclient = new CClient($this->conn);
        $cclient->updateAuthTokenHash($hashvalue_s,
$token_hash_t, $token_hash_to, $unmtoken, $urltoken, $seqnum);

        $seq_num = array(
            'n1'=>$n1,
            'n2'=>$n2,
            'n3'=>$n3,
            'n4'=>$n4
        );

        return $seq_num;
    }

    private function validateHashValue($hashvalue, $client)
    {
        $hashvalue_s = $client->getTokenHash();

        $created_at = $client->getTokenHashT();
        $timeout = $client->getTokenHashTo();

        if(time() - $created_at >= $timeout)
            throw new AccessDeniedHttpException('Hash value
telah expired.');
```

```

        else
        {
            $status = $this->compareString($hashvalue,
$hashvalue_s);

            if($status==false)

```

```

        throw new AccessDeniedHttpException('Hash
value tidak sesuai.');
```

```

        else
            return $status;
    }
}

private function getNewTokenForUser($client)
{
    $new_token = $this->generateRandomString(32);

    $unmtoken = $client->getUnmToken();
    $urltoken = $client->getUrlToken();
    $seqnum = $client->getSeqNum();

    // update access token
    $cclient = new CClient($this->conn);
    $cclient->updateAccessToken($urltoken, $unmtoken,
$seqnum, $new_token);

    return $new_token;
}

private function syncGetClient($synckey)
{
    $cclient = new CClient($this->conn);
    $client = $cclient->loadClientBySyncKey($synckey);
    $created_at = $client->getSyncKeyT();
    $timeout = $client->getSyncKeyTo();

    if(time() - $created_at >= $timeout)
        throw new
AccessDeniedHttpException('Synchronization key telah expired.');
```

```

    else
        return $client;
}

private function syncValidateHashValue($hashvalue, $synckey)
{
    $cclient = new CClient($this->conn);
    $client = $cclient->loadClientBySyncKey($synckey);
    $hashvalue_s = $client->getSyncTokenHash();

    $created_at = $client->getSyncTokenHashT();
    $timeout = $client->getSyncTokenHashTo();

    if(time() - $created_at >= $timeout)
        throw new AccessDeniedHttpException('Hash value
telah expired.');
```

```

    else
    {
        $status = $this->compareString($hashvalue,
$hashvalue_s);

        if($status==false)
            throw new AccessDeniedHttpException('Hash
value tidak sesuai.');
```

```

        return [

```

```

        'status' => $status,
        'info'   => $status ? ['seqnum' =>
$client->getSeqNum()] : []
    ];
    }
}

private function syncGenerateSeqNumAndHashValue($client)
{
    $unm_seed = $client->getUnmSeed();
    $url_seed = $client->getUrlSeed();
    $synckey = $client->getSyncKey();

    $n1 = rand();
    $n2 = rand();
    $n3 = rand();
    $n4 = rand();

    $hashvalue_s = $this->generateTokenHash($unm_seed,
$url_seed, $n1,$n2,$n3,$n4);
    $sync_token_hash_t = time();
    $sync_token_hash_to = 1000;

    $cclient = new CClient($this->conn);
    $cclient->updateSyncTokenHash($hashvalue_s,
$sync_token_hash_t, $sync_token_hash_to, $synckey);

    $seq_num = array(
        'n1'=>$n1,
        'n2'=>$n2,
        'n3'=>$n3,
        'n4'=>$n4
    );

    return $seq_num;
}

private function compareString($expected, $actual)
{
    $expected .= "\0";
    $actual .= "\0";
    $expectedLength = mb_strlen($expected, '8bit');
    $actualLength = mb_strlen($actual, '8bit');
    $diff = $expectedLength - $actualLength;
    for ($i = 0; $i < $actualLength; $i++) {
        $diff |= (ord($actual[$i]) ^ ord($expected[$i %
$expectedLength]));
    }
    return $diff === 0;
}

public function onAuthenticationSuccess(Request $request,
TokenInterface $token, $providerKey)
{
    // on success, let the request continue
    return;
}

```

```

        public function onAuthenticationFailure(Request $request,
        AuthenticationException $exception)
        {
            $data = array(
                'message' => strtr($exception->getMessageKey(),
        $exception->getMessageData()),
            );

            return new JsonResponse($data, 403);
        }

        private function generateTokenHash($unm_seed, $url_seed,
        $n1,$n2,$n3,$n4)
        {
            $urltoken1 = $this->generateToken($url_seed,$n1);
            $urltoken2 = $this->generateToken($url_seed,$n2);

            $unmtoken1 = $this->generateToken($unm_seed,$n3);
            $unmtoken2 = $this->generateToken($unm_seed,$n4);

            return
        md5($urltoken1.$urltoken2.$unmtoken1.$unmtoken2);
        }

        private function generateToken($seed, $seqnum){
            $seed = crc32($seed.$seqnum);
            mt_srand($seed);
            $token = mt_rand();

            return $token;
        }

        private function generateSeed($root_file){
            $file = WEB_DIRECTORY.'/file/temp/'.$root_file;
            $fp = fopen($file, 'r');
            $fsize = filesize($file);
            $seed = fread($fp, $fsize);
            $urlseed = substr($seed, $fsize/2, 12);
            $unmseed = substr($seed, -12);
            fclose($fp);

            $seed = array(
                'urlseed'=>$urlseed,
                'unmseed'=>$unmseed,
            );

            return $seed;
        }
    }
}

```

Lampiran C *Source Code* CClient.php Aplikasi *Authenticator* pada *Authenticator Server*

```
<?php

namespace G\Silex\Entity;

use Doctrine\DBAL\Connection;
use
Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;

class CClient
{
    private $conn;

    public function __construct(Connection $conn)
    {
        $this->conn = $conn;
    }

    public function loadClientById($id)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE id = ?', array($id));

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client tidak
ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unm_token'], $data['url_token'], $data['root_file'],
$data['unm_seed'], $data['url_seed'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
$data['sync_token_hash'], $data['sync_token_hash_t'],
$data['sync_token_hash_to'], $data['seq_num'],
$data['access_token_to'], $data['init_key_to'],
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
$data['init_key'], $data['init_key_t'], $data['status_id']);
    }

    public function loadClientByUnmToken($unm_token)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE unm_token = ?', array($unm_token));

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client tidak
ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unm_token'], $data['url_token'], $data['root_file'],
$data['unm_seed'], $data['url_seed'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
$data['sync_token_hash'], $data['sync_token_hash_t'],
```

```

$data['sync_token_hash_to'], $data['seq_num'],
$data['access_token_to'], $data['init_key_to'],
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
$data['init_key'], $data['init_key_t'], $data['status_id']);
    }

    public function
loadClientByUrlTokenUnmTokenSeqNum($urltoken, $unmtoken, $seqnum)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE url_token = ? AND unmtoken = ? AND seq_num = ?',
array($urltoken, $unmtoken, $seqnum));

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client
tidak ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unmtoken'], $data['url_token'], $data['root_file'],
$data['unmtoken'], $data['url_token'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
$data['sync_token_hash'], $data['sync_token_hash_t'],
$data['sync_token_hash_to'], $data['seq_num'],
$data['access_token_to'], $data['init_key_to'],
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
$data['init_key'], $data['init_key_t'], $data['status_id']);
    }

    public function loadClientBySyncKey($synckey)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE sync_key = ?', array($synckey));

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client tidak
ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unmtoken'], $data['url_token'], $data['root_file'],
$data['unmtoken'], $data['url_token'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
$data['sync_token_hash'], $data['sync_token_hash_t'],
$data['sync_token_hash_to'], $data['seq_num'],
$data['access_token_to'], $data['init_key_to'],
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
$data['init_key'], $data['init_key_t'], $data['status_id']);
    }

    public function updateAccessToken($url_token, $unmtoken,
$seq_num, $new_token)
    {
        $sql = "UPDATE client SET access_token_to = 1000,
access_token = ? , access_token_t = ? WHERE unmtoken = ? AND
url_token = ? AND seq_num = ?";

```



```

        $this->conn->executeUpdate($sql, array($new_token,
time(), $unm_token, $url_token, $seq_num));

        return true;
    }

    public function updateTokens($url_token, $unm_token,
$seq_num, $next_unmtoken, $next_urltoken)
    {
        $sql = "UPDATE client SET seq_num = seq_num+1,
unm_token = ?, url_token=? WHERE unm_token = ? AND url_token = ?
AND seq_num = ?";

        $this->conn->executeUpdate($sql, array($next_unmtoken,
$next_urltoken, $unm_token, $url_token, $seq_num));

        return true;
    }

    public function updateTokenSeqNumAndStatus($unm_seed,
$url_seed, $next_seqnum, $next_unmtoken, $next_urltoken, $initkey)
    {
        $sql = "UPDATE client SET status_id = 1, unm_seed = ?,
url_seed = ?, seq_num = ?, unm_token = ?, url_token = ? WHERE
init_key = ?";
        $this->conn->executeUpdate($sql, array($unm_seed,
$url_seed, $next_seqnum, $next_unmtoken, $next_urltoken,
$initkey));
        return true;
    }

    public function updateAuthTokenHash($hashvalue_s,
$token_hash_t, $token_hash_to, $unmtoken, $urltoken, $seqnum)
    {
        $sql = "UPDATE client SET token_hash = ?, token_hash_t
= ?, token_hash_to = ? WHERE unm_token = ? AND url_token = ? AND
seq_num = ?";
        $this->conn->executeUpdate($sql, array($hashvalue_s,
$token_hash_t, $token_hash_to, $unmtoken, $urltoken, $seqnum));
        return true;
    }

    public function updateSyncTokenHash($hashvalue_s,
$sync_token_hash_t, $sync_token_hash_to, $synckey)
    {
        $sql = "UPDATE client SET sync_token_hash = ? ,
sync_token_hash_t = ?, sync_token_hash_to = ? WHERE sync_key =
?";
        $this->conn->executeUpdate($sql, array($hashvalue_s,
$sync_token_hash_t, $sync_token_hash_to, $synckey));

        return true;
    }

    public function loadClientByInitkey($initkey)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM
client WHERE init_key = ?', array($initkey));

```

```

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client
tidak ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
        $data['access_token'], $data['access_token_t'],
        $data['unm_token'], $data['url_token'], $data['root_file'],
        $data['unm_seed'], $data['url_seed'], $data['token_hash'],
        $data['token_hash_t'], $data['token_hash_to'],
        $data['sync_token_hash'], $data['sync_token_hash_t'],
        $data['sync_token_hash_to'], $data['seq_num'],
        $data['access_token_to'], $data['init_key_to'],
        $data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
        $data['init_key'], $data['init_key_t'], $data['status_id']);
    }
}

?>
```

Lampiran D Source Code Client.php Aplikasi Authenticater pada Authenticater Server

```
<?php

namespace G\Silex\Entity;

class Client
{
    private $id;
    private $nama_app;
    private $access_token;
    private $access_token_t;
    private $unm_token;
    private $url_token;
    private $root_file;
    private $unm_seed;
    private $url_seed;
    private $token_hash;
    private $token_hash_t;
    private $token_hash_to;
    private $sync_token_hash;
    private $sync_token_hash_t;
    private $sync_token_hash_to;
    private $seq_num;
    private $access_token_to;
    private $init_key_to;
    private $sync_key_to;
    private $sync_key;
    private $sync_key_t;
    private $init_key;
    private $init_key_t;
    private $status_id;

    public function __construct($id, $nama_app, $access_token,
    $access_token_t, $unm_token, $url_token, $root_file, $unm_seed,
    $url_seed, $token_hash, $token_hash_t, $token_hash_to,
    $sync_token_hash, $sync_token_hash_t, $sync_token_hash_to,
    $seq_num, $access_token_to, $init_key_to, $sync_key_to, $sync_key,
    $sync_key_t, $init_key, $init_key_t, $status_id)
    {
        $this->id = $id;
        $this->nama_app = $nama_app;
        $this->access_token = $access_token;
        $this->access_token_t = $access_token_t;
        $this->unm_token = $unm_token;
        $this->url_token = $url_token;
        $this->root_file = $root_file;
        $this->unm_seed = $unm_seed;
        $this->url_seed = $url_seed;
        $this->token_hash = $token_hash;
        $this->token_hash_t = $token_hash_t;
        $this->token_hash_to = $token_hash_to;
        $this->sync_token_hash = $sync_token_hash;
        $this->sync_token_hash_t = $sync_token_hash_t;
        $this->sync_token_hash_to = $sync_token_hash_to;
    }
}
```

```

        $this->seq_num = $seq_num;
        $this->access_token_to = $access_token_to;
        $this->init_key_to = $init_key_to;
        $this->sync_key_to = $sync_key_to;
        $this->sync_key = $sync_key;
        $this->sync_key_t = $sync_key_t;
        $this->init_key = $init_key;
        $this->init_key_t = $init_key_t;
        $this->status_id = $status_id;
    }

    public function getId()      {
        return $this->id;
    }

    public function getNamaapp()
    {
        return $this->nama_app;
    }

    public function getAccessToken()
    {
        return $this->access_token;
    }

    public function getAccessTokenT()
    {
        return $this->access_token_t;
    }

    public function getUnmToken()
    {
        return $this->unm_token;
    }

    public function getUrlToken()
    {
        return $this->url_token;
    }

    public function getRootFile()
    {
        return $this->root_file;
    }

    public function getUnmSeed()
    {
        return $this->unm_seed;
    }

    public function getUrlSeed()
    {
        return $this->url_seed;
    }

    public function getTokenHash()
    {
        return $this->token_hash;
    }

```

```

public function getTokenHashT()
{
    return $this->token_hash_t;
}

public function getTokenHashTo()
{
    return $this->token_hash_to;
}

public function getSyncTokenHash()
{
    return $this->sync_token_hash;
}

public function getSyncTokenHashT()
{
    return $this->sync_token_hash_t;
}

public function getSyncTokenHashTo()
{
    return $this->sync_token_hash_to;
}

public function getSeqNum()
{
    return $this->seq_num;
}

public function getAccessTokenTo()
{
    return $this->access_token_to;
}

public function getInitKeyTo()
{
    return $this->init_key_to;
}

public function getSyncKeyTo()
{
    return $this->sync_key_to;
}

public function getSyncKey()
{
    return $this->sync_key;
}

public function getSyncKeyT()
{
    return $this->sync_key_t;
}

public function getInitKey()
{
    return $this->init_key;
}

```

```
}

public function getInitKeyT()
{
    return $this->init_key_t;
}

public function getStatusId()
{
    return $this->status_id;
}

public function setUnmToken($unm_token)
{
    $this->unm_token = $unm_token;
}

public function setUrlToken($url_token)
{
    $this->url_token = $url_token;
}

public function setSeqNum($seq_num)
{
    $this->seq_num = $seq_num;
}
}

?>
```

Lampiran E *Source Code* AuthControllerProvider.php pada *Resource Server*

```
<?php

namespace G\Silex\Provider\Auth;

use
Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;
use Symfony\Component\HttpFoundation\Request;
use Sillex\ControllerProviderInterface;
use Sillex\Application;
use G\Silex\Entity\Client;
use G\Silex\Entity\CClient;

class AuthControllerProvider implements
ControllerProviderInterface
{
    const TOKEN_HEADER_KEY = 'X-Token';
    const TOKEN_REQUEST_KEY = '_token';

    private $baseRoute;

    public function setBaseRoute($baseRoute)
    {
        $this->baseRoute = $baseRoute;

        return $this;
    }

    public function connect(Application $app)
    {
        $this->setUpMiddlewares($app);

        return $this->extractControllers($app);
    }

    private function extractControllers(Application $app)
    {
        $controllers = $app['controllers_factory'];

        return $controllers;
    }

    private function setUpMiddlewares(Application $app)
    {
        $app->before(function (Request $request) use ($app) {

            $app->register(new
AuthServiceProvider($app['db']));

            $urltoken = $request->get('urltoken');
            $unmtoken = $request->get('unmtoken');
            $seqnum = $request->get('seqnum');

            if (!$this->isAuthRequiredForPath($request-
>getPathInfo()) && $request->getMethod() != 'OPTIONS') {
```

```

        if (!$this->isValidTokenForApplication($app,
$this->getTokenFromRequest($request), $urltoken, $unmtoken,
$seqnum)) {
            throw new AccessDeniedHttpException('Access
Denied');
        }
    }
});
}

private function getTokenFromRequest(Request $request)
{
    return $request->headers->get(self::TOKEN_HEADER_KEY,
$request->get(self::TOKEN_REQUEST_KEY));
}

// route yang tidak perlu autentikasi
private function isAuthRequiredForPath($path)
{
    $path_arr = explode('/', $path);
    $path_new = implode('/', array_slice($path_arr, 0,
3));

    return in_array($path_new, [
    ]);
}

private function isValidTokenForApplication(Application $app,
$token, $urltoken, $unmtoken, $seqnum)
{
    return
$app[AuthServiceProvider::AUTH_VALIDATE_TOKEN]($token, $urltoken,
$unmtoken, $seqnum);
}
}

```


Lampiran F *Source Code* AuthServiceProvider.php pada *Resource Server*

```
<?php

namespace G\Silex\Provider\Auth;

use Silex\Application;
use Silex\ServiceProviderInterface;
use G\Silex\Entity\CClient;
use Doctrine\DBAL\Connection;
use
Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;

class AuthServiceProvider implements ServiceProviderInterface
{
    const AUTH_VALIDATE_TOKEN      = 'auth.validate.token';

    private $conn;

    public function __construct(Connection $conn)
    {
        $this->conn = $conn;
    }

    public function register(Application $app)
    {
        $app[self::AUTH_VALIDATE_TOKEN] = $app->protect(function
($token, $urltoken, $unmtoken, $seqnum) {
            return $this->validateToken($token, $urltoken,
$unmtoken, $seqnum);
        });
    }

    public function boot(Application $app)
    {
    }

    private function getClient($urltoken, $unmtoken, $seqnum)
    {
        $cclient = new CClient($this->conn);
        $client = $cclient->loadClientByUrlTokenUnmTokenSeqNum($urltoken, $unmtoken,
$seqnum);

        return $client;
    }

    private function validateToken($token, $urltoken, $unmtoken,
$seqnum)
    {
        $client = $this->getClient($urltoken, $unmtoken,
$seqnum);
        $access_token = $client->getAccessToken();
        $created_at = $client->getAccessTokenT();
        $timeout = $client->getAccessTokenTo();

        if(time() - $created_at >= $timeout)
            throw new AccessDeniedHttpException('Access
Token telah expired.');
```

```

        else if($this->compareString($token, $access_token))
            return true;
        else
            throw new AccessDeniedHttpException('Access
Token tidak sesuai.');
```

```

    }

    private function compareString($expected, $actual)
    {
        $expected .= "\0";
        $actual .= "\0";
        $expectedLength = mb_strlen($expected, '8bit');
        $actualLength = mb_strlen($actual, '8bit');
        $diff = $expectedLength - $actualLength;
        for ($i = 0; $i < $actualLength; $i++) {
            $diff |= (ord($actual[$i]) ^ ord($expected[$i %
$expectedLength]));
        }
        return $diff === 0;
    }

    public function onAuthenticationSuccess(Request $request,
TokenInterface $token, $providerKey)
    {
        // on success, let the request continue
        return;
    }

    public function onAuthenticationFailure(Request $request,
AuthenticationException $exception)
    {
        $data = array(
            'message' => strtr($exception->getMessageKey(),
$exception->getMessageData()),
        );

        return new JsonResponse($data, 403);
    }
}

```

Lampiran G *Source Code CClient.php* pada *Resource Server*

```
<?php

namespace G\Silex\Entity;

use Doctrine\DBAL\Connection;
use
Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;

class CClient
{
    private $conn;

    public function __construct(Connection $conn)
    {
        $this->conn = $conn;
    }

    public function loadClientById($id)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE id = ?', array($id));

        if (!$data = $stmt->fetch()) {
            throw new AccessDeniedHttpException('Client tidak
ditemukan.');
```

```
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unm_token'], $data['url_token'], $data['root_file'],
$data['unm_seed'], $data['url_seed'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
$data['sync_token_hash'], $data['sync_token_hash_t'],
$data['sync_token_hash_to'], $data['seq_num'],
$data['access_token_to'], $data['init_key_to'],
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],
$data['init_key'], $data['init_key_t']);
    }

    public function
loadClientByUrlTokenUnmTokenSeqNum($urltoken, $unmtoken, $seqnum)
    {
        $stmt = $this->conn->executeQuery('SELECT * FROM client
WHERE url_token = ? AND unm_token = ? AND seq_num = ?',
array($urltoken, $unmtoken, $seqnum));

        if (!$data = $stmt->fetch()) {
            throw new
AccessDeniedHttpException(sprintf('Client tidak ditemukan.',
$unmtoken));
        }

        return new Client($data['id'], $data['nama_app'],
$data['access_token'], $data['access_token_t'],
$data['unm_token'], $data['url_token'], $data['root_file'],
$data['unm_seed'], $data['url_seed'], $data['token_hash'],
$data['token_hash_t'], $data['token_hash_to'],
```

```
$data['sync_token_hash'], $data['sync_token_hash_t'],  
$data['sync_token_hash_to'], $data['seq_num'],  
$data['access_token_to'], $data['init_key_to'],  
$data['sync_key_to'], $data['sync_key'], $data['sync_key_t'],  
$data['init_key'], $data['init_key_t']);  
    }  
}  
  
?>
```

Lampiran H *Source Code Client.php* pada *Resource Server*

```
<?php
namespace G\Silex\Entity;
class Client
{
    private $id;
    private $nama_app;
    private $access_token;
    private $access_token_t;
    private $unm_token;
    private $url_token;
    private $root_file;
    private $unm_seed;
    private $url_seed;
    private $token_hash;
    private $token_hash_t;
    private $token_hash_to;
    private $sync_token_hash;
    private $sync_token_hash_t;
    private $sync_token_hash_to;
    private $seq_num;
    private $access_token_to;
    private $init_key_to;
    private $sync_key_to;
    private $sync_key;
    private $sync_key_t;
    private $init_key;
    private $init_key_t;

    public function __construct($id, $nama_app, $access_token,
        $access_token_t, $unm_token, $url_token, $root_file, $unm_seed,
        $url_seed, $token_hash, $token_hash_t, $token_hash_to,
        $sync_token_hash, $sync_token_hash_t, $sync_token_hash_to,
        $seq_num, $access_token_to, $init_key_to, $sync_key_to, $sync_key,
        $sync_key_t, $init_key, $init_key_t)
    {
        $this->id = $id;
        $this->nama_app = $nama_app;
        $this->access_token = $access_token;
        $this->access_token_t = $access_token_t;
        $this->unm_token = $unm_token;
        $this->url_token = $url_token;
        $this->root_file = $root_file;
        $this->unm_seed = $unm_seed;
        $this->url_seed = $url_seed;
        $this->token_hash = $token_hash;
        $this->token_hash_t = $token_hash_t;
        $this->token_hash_to = $token_hash_to;
        $this->sync_token_hash = $sync_token_hash;
        $this->sync_token_hash_t = $sync_token_hash_t;
        $this->sync_token_hash_to = $sync_token_hash_to;
        $this->seq_num = $seq_num;
        $this->access_token_to = $access_token_to;
        $this->init_key_to = $init_key_to;
        $this->sync_key_to = $sync_key_to;
        $this->sync_key = $sync_key;
        $this->sync_key_t = $sync_key_t;
        $this->init_key = $init_key;
    }
}
```

```

        $this->init_key_t = $init_key_t;
    }

    public function getId()    {
        return $this->id;
    }

    public function getNamaapp()
    {
        return $this->nama_app;
    }

    public function getAccessToken()
    {
        return $this->access_token;
    }

    public function getAccessTokenT()
    {
        return $this->access_token_t;
    }

    public function getUnmToken()
    {
        return $this->unm_token;
    }

    public function getUrlToken()
    {
        return $this->url_token;
    }

    public function getRootFile()
    {
        return $this->root_file;
    }

    public function getUnmSeed()
    {
        return $this->unm_seed;
    }

    public function getUrlSeed()
    {
        return $this->url_seed;
    }

    public function getTokenHash()
    {
        return $this->token_hash;
    }

    public function getTokenHashT()
    {
        return $this->token_hash_t;
    }

    public function getTokenHashTo()
    {

```

```

        return $this->token_hash_to;
    }

    public function getSyncTokenHash()
    {
        return $this->sync_token_hash;
    }

    public function getSyncTokenHashT()
    {
        return $this->sync_token_hash_t;
    }

    public function getSyncTokenHashTo()
    {
        return $this->sync_token_hash_to;
    }

    public function getSeqNum()
    {
        return $this->seq_num;
    }

    public function getAccessTokenTo()
    {
        return $this->access_token_to;
    }

    public function getInitKeyTo()
    {
        return $this->init_key_to;
    }

    public function getSyncKeyTo()
    {
        return $this->sync_key_to;
    }

    public function getSyncKey()
    {
        return $this->sync_key;
    }

    public function getSyncKeyT()
    {
        return $this->sync_key_t;
    }

    public function getInitKey()
    {
        return $this->init_key;
    }

    public function getInitKeyT()
    {
        return $this->init_key_t;
    }
}
?>

```

Lampiran I SQL database authdb pada Authentication Server

```
--
-- Database: `authdb`
--

-- -----

--
-- Table structure for table `client`
--

CREATE TABLE IF NOT EXISTS `client` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nama_app` varchar(255) NOT NULL,
  `tipe_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `status_id` int(11) NOT NULL,
  `root_file` varchar(255) NOT NULL,
  `url_seed` varbinary(255) DEFAULT NULL,
  `unm_seed` varbinary(255) DEFAULT NULL,
  `seq_num` int(11) DEFAULT NULL,
  `url_token` varchar(255) DEFAULT NULL,
  `unm_token` varchar(255) DEFAULT NULL,
  `access_token` varchar(255) DEFAULT NULL,
  `access_token_t` int(11) DEFAULT NULL,
  `access_token_to` int(11) DEFAULT NULL,
  `init_key` varchar(255) DEFAULT NULL,
  `init_key_t` int(11) DEFAULT NULL,
  `init_key_to` int(11) DEFAULT NULL,
  `sync_key` varchar(255) DEFAULT NULL,
  `sync_key_t` int(11) DEFAULT NULL,
  `sync_key_to` int(11) DEFAULT NULL,
  `token_hash` varchar(255) DEFAULT NULL,
  `token_hash_t` int(11) DEFAULT NULL,
  `token_hash_to` int(11) DEFAULT NULL,
  `sync_token_hash` varchar(255) DEFAULT NULL,
  `sync_token_hash_t` int(11) DEFAULT NULL,
  `sync_token_hash_to` int(11) DEFAULT NULL,
  `created_at` int(11) DEFAULT NULL,
  `updated_at` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `tipe_id` (`tipe_id`),
  KEY `project_ibfk_3` (`status_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=45 ;

-- -----

--
-- Table structure for table `status`
--

CREATE TABLE IF NOT EXISTS `status` (
  `id` int(11) NOT NULL,
  `status` varchar(255) DEFAULT NULL,
  `ket` varchar(1000) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```



```

-- -----
--
-- Table structure for table `tipe`
--

CREATE TABLE IF NOT EXISTS `tipe` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tipe_app` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

-- -----
--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `auth_key` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  `password_hash` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password_reset_token` varchar(255) COLLATE utf8_unicode_ci
  DEFAULT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `status` smallint(6) NOT NULL DEFAULT '10',
  `created_at` int(11) NOT NULL,
  `updated_at` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`),
  UNIQUE KEY `password_reset_token` (`password_reset_token`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
  AUTO_INCREMENT=22 ;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `client`
--
ALTER TABLE `client`
  ADD CONSTRAINT `client_ibfk_1` FOREIGN KEY (`user_id`)
  REFERENCES `user` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE,
  ADD CONSTRAINT `client_ibfk_2` FOREIGN KEY (`tipe_id`)
  REFERENCES `tipe` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE,
  ADD CONSTRAINT `client_ibfk_3` FOREIGN KEY (`status_id`)
  REFERENCES `status` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

Lampiran J Antarmuka Aplikasi Administrator

The screenshot shows the 'Signup' page of an application administrator interface. At the top, there is a dark navigation bar with links for 'Home', 'Signup', and 'Login'. Below this, a breadcrumb trail shows 'Home / Signup'. The main heading is 'Signup'. A message states: 'Please fill out the following fields to signup:'. The form includes four input fields: 'Username' (with a red border and error message 'Username cannot be blank.'), 'Email', 'Password', and 'Password Repeat'. A blue 'Signup' button is at the bottom. The footer contains copyright information '© Auliak Amri 2017' and the institution 'Rekayasa dan Manajemen Keamanan Informasi - ITB'.

Gambar J.1. *Form pendaftaran developer aplikasi*

The screenshot shows the 'Login' page of the application administrator interface. The navigation bar at the top has links for 'Home', 'Signup', and 'Login'. The breadcrumb trail shows 'Home / Login'. The main heading is 'Login'. A message states: 'Please fill out the following fields to login:'. The form includes two input fields: 'Username' (with a red border and error message 'Username cannot be blank.') and 'Password'. Below the password field is a checkbox for 'Remember Me' and a link 'If you forgot your password you can [reset it](#)'. A blue 'Login' button is at the bottom. The footer contains copyright information '© Auliak Amri 2017' and the institution 'Rekayasa dan Manajemen Keamanan Informasi - ITB'.

Gambar J.2. *Form login*

The screenshot shows the 'Request password reset' page of the application administrator interface. The navigation bar at the top has links for 'Home', 'Signup', and 'Login'. The breadcrumb trail shows 'Home / Request password reset'. The main heading is 'Request password reset'. A message states: 'Please fill out your email. A link to reset password will be sent there.'. The form includes one input field for 'Email' (with a red border and error message 'Email cannot be blank.'). A blue 'Send' button is at the bottom.

Gambar J.3. *Form request password reset*

The screenshot shows a web application interface with a dark header containing links: Home, Client, Developer, and Logout (user2). Below the header is a breadcrumb trail: Home / Data Developer / Update. The main heading is 'Update Data Developer'. The form contains two text input fields: 'Username' with the value 'user2' and 'Email' with the value 'user2@gmail.com'. Below these fields is a blue 'Update' button. At the bottom of the page, there is a footer with the copyright notice '© Auliak Amri 2017' and the text 'Rekayasa dan Manajemen Keamanan Informasi - ITB'.

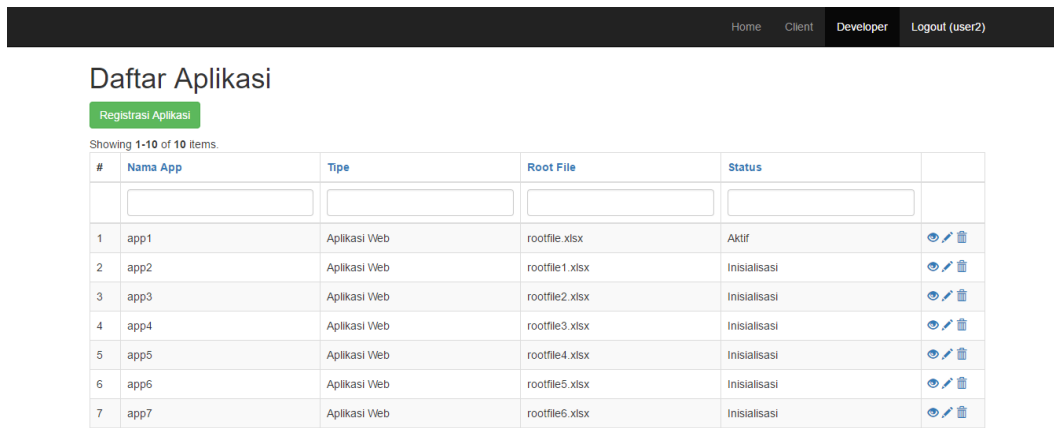
Gambar J.4. *Form update data developer aplikasi*

The screenshot shows a web application interface with a dark header containing links: Home, Client, Developer, and Logout (user2). Below the header is a breadcrumb trail: Home / Data Developer / Ubah password. The main heading is 'Ubah Password'. The form contains three text input fields: 'Password', 'New Password', and 'New Password Repeat'. Below these fields is a blue 'Ubah password' button. At the bottom of the page, there is a footer with the copyright notice '© Auliak Amri 2017' and the text 'Rekayasa dan Manajemen Keamanan Informasi - ITB'.

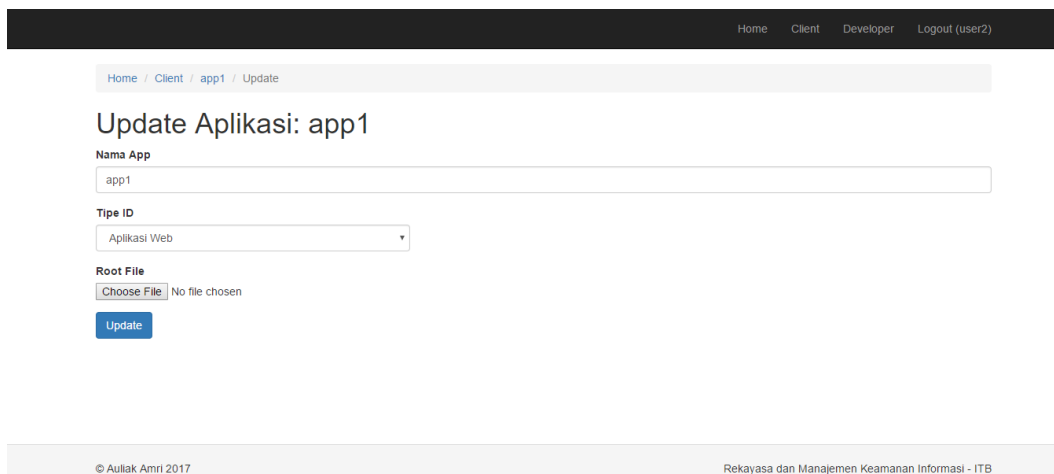
Gambar J.5. *Form ubah password developer aplikasi*

The screenshot shows a web application interface with a dark header containing links: Home, Client, Developer, and Logout (user2). Below the header is a breadcrumb trail: Home / Data User / Registrasi Aplikasi. The main heading is 'Registrasi Aplikasi'. The form contains three input fields: 'Nama App' (text), 'Tipe ID' (dropdown menu with the value '-- Pilih Jenis Aplikasi --'), and 'Root File' (file upload button labeled 'Choose File' with the text 'No file chosen'). Below these fields is a green 'Create' button. At the bottom of the page, there is a footer with the copyright notice '© Auliak Amri 2017' and the text 'Rekayasa dan Manajemen Keamanan Informasi - ITB'.

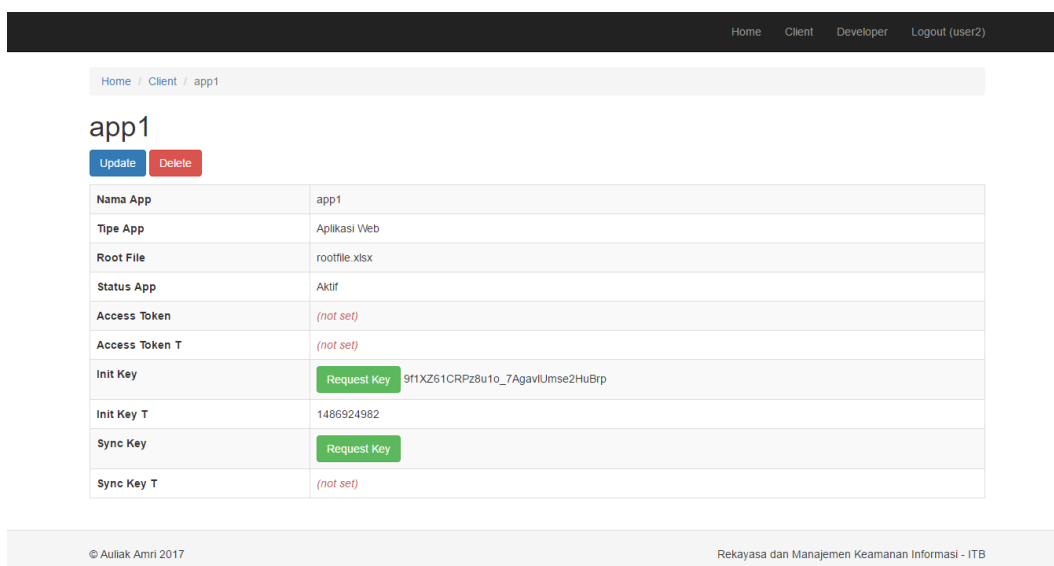
Gambar J.6. *Form registrasi aplikasi*



Gambar J.7. Halaman daftar aplikasi



Gambar J.8. *Form update* data aplikasi



Gambar J.9. Halaman detail aplikasi