METODE *IMPERSONATION* PADA SERVER AUTORISASI MENGGUNAKAN PROTOKOL *CLIENT-INITIATED BACK-CHANNEL AUTHENTICATION*

Rizky Januar Akbar[1)], Nurul Fajrin Ariyani[2)], Muhammad Adistya Azhar[3)], dan Andika Andra[4)]
[1,2,3,4)]Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya 60111
e-mail: rja@its.ac.id[1)], nurulfajrin@if.its.ac.id[2)], aa@adisazhar.com[3)], andika16@mhs.if.its.ac.id[4)]

ABSTRAK

Pada beberapa aplikasi tertentu terdapat fitur impersonation (login as) yang dapat dilakukan oleh pengguna yang memiliki hak akses secara sah yaitu administrator sistem. Fitur ini bermanfaat agar tim pengembang atau tim perawat aplikasi yang memiliki hak akses sebagai administrator dapat mereproduksi error atau bug, memeriksa fitur-fitur spesifik pada aplikasi sesuai dengan konteks login pengguna tertentu. Selain manfaat yang didapat tersebut, terdapat pula kelemahan sistem dimana administrator dapat menyalahgunakan wewenang untuk mengakses data pribadi pengguna atau melakukan aktivitas-aktivitas tertentu di sistem tanpa sepengetahuan dan ijin dari pemilik akun.

Penelitian ini mengusulkan metode impersonation menggunakan protokol Client-Initiated Back-channel Authentication (CIBA) sehingga ketika administrator melakukan mekanisme impersonation harus didahului oleh proses persetujuan dari pemilik akun menggunakan fasilitas autentikator atau authentication device. Dengan memanfaatkan authentication device maka fitur impersonation harus mendapatkan ijin dari pemilik akun, dan pembuktian identitas pemilik akun juga dapat dilakukan tanpa harus berinteraksi langsung. Hasil menunjukkan bahwa implementasi protokol CIBA ini dapat digunakan untuk menunjang metode impersonation dan juga dapat berjalan diatas implementasi server autorisasi yang sudah berjalan menggunakan protokol OAuth 2.0 dan OpenID Connect 1.0. Uji coba sistem dilakukan dengan mengadopsi conformance testing dari FAPI CIBA.

Kata Kunci: *Authorization Server, Authentication Device, Client Initiated Backchannel Authentication, Consumption Device, Impersonation, Relying Party.*

IMPERSONATION METHOD ON AUTHORIZATION SERVER USING CLIENT-INITIATED BACK-CHANNEL AUTHENTICATION PROTOCOL

Rizky Januar Akbar[1)], Nurul Fajrin Ariyani[2)], Muhammad Adistya Azhar[3)], dan Andika Andra[4)]
[1,2,3,4)]Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya 60111
e-mail: rja@its.ac.id[1)], nurulfajrin@if.its.ac.id[2)], aa@adisazhar.com[3)], andika16@mhs.if.its.ac.id[4)]

ABSTRACT

There is an impersonation (login as) feature in several applications that can be used by system administrators who have special privileges. This feature can be utilized by development and maintenance teams that have administrator rights to reproduce errors or bugs, to check specific features in applications according to the specific users' login sessions. Beside its benefits, there is a security vulnerability that allows administrators to abuse the rights. They can access users' private data or execute some activities inside the system without account or resource owners' consents.

This research proposes an impersonation method on authorization server using Client-Initiated Back-channel Authentication (CIBA) protocol. This method prevents impersonation without account or resource owners' consent. The application will ask users' authentication and permission via authentication device possessed by resource owners before the administrator performs impersonation. By utilizing authentication device, the impersonation feature should be preceded by users' consent and there is no direct interaction needed between the administrator and resource owners to prove the users' identities. The result shows that the implementation of CIBA protocol can be used to complement the impersonation method and can also run on the authorization server that uses OAuth 2.0 and OpenID Connect 1.0 protocols. The system testing is done by adopting FAPI CIBA conformance testing.

***Keywords**: Authorization Server, Authentication Device, Client Initiated Backchannel Authentication, Consumption Device, Impersonation, Relying Party.*

I. PENDAHULUAN

Seiring perkembangan dunia digital dan tersimpannya data pribadi pengguna di sistem semakin meningkatkan kesadaran pengguna untuk mengendalikan informasi pribadinya. Hal ini dipicu oleh ketidaktahuan pengguna atas bagaimana, kapan dan siapa yang mengakses informasi mereka. Di dunia keamanan, terdapat sebuah tipe serangan peniruan (*impersonation*) dimana seorang penyerang dapat berperan atau berpura-pura sebagai pengguna sah tertentu [1]. Serangan ini dilakukan oleh penyerang yang tidak memiliki hak akses. Di lain sisi, pada sistem *single sign-on* terdapat sebuah fitur peniruan dimana seorang administrator sistem yang memiliki hak akses sah dapat masuk sebagai (*login as*) pengguna akun lain, mengakses data pribadi, dan melakukan aktivitas di sistem tanpa diketahui oleh pengguna pemilik akunnya. Hal ini dapat terjadi jika administrator menyalahgunakan wewenangnya. Tetapi fitur peniruan ini juga bermanfaat untuk mempermudah tim pengembangan dan tim perawatan aplikasi dalam mereproduksi bug/error pada program berdasarkan komplain pengguna yang spesifik. Administrator dapat melakukan aktivitas pada aplikasi secara spesifik berdasarkan hak akses pengguna tertentu ketika melakukan uji coba pada aplikasi, untuk bertindak atas nama pengguna (pemilik akun) melakukan suatu tindakan ke dalam sistem dengan persetujuan (*consent*) pemilik akun atau berdasarkan keinginan administrator sendiri tanpa persetujuan pemilik akun.

Di beberapa aplikasi dengan data sensitif atau rahasia seperti data pribadi dan keuangan, atau proses bisnis yang sensitif seperti fungsi persetujuan transaksi keuangan maka fitur peniruan ini seharusnya dilengkapi dengan mekanisme permintaan persetujuan kepada pemilik akun. Permintaan persetujuan secara manual bisa dilakukan secara verbal melalui sambungan telepon dan merekam percakapan. Akan tetapi dibutuhkan mekanisme pembuktian identitas seseorang dengan memanfaatkan *multi-factor authentication* yaitu menggunakan atribut yang dimiliki pemilik akun yaitu (1) *something you know: password*, (2) *something you have: smartcard, security token, authenticator* dan (3) *something you are: biometrik/fingerprint* [2]. Pada metode tradisional di *call center* biasanya administrator akan bertanya pada pemilik akun dengan pertanyaan-pertanyaan tertentu untuk membuktikan identitas pengguna. Setelah identitas terbukti baru kemudian administrator bisa meminta persetujuan untuk melakukan tindakan atau mengakses data pribadi pengguna atas seizin pengguna. Cara tradisional ini tentu memakan waktu dan tidak praktis. Waktu yang dipergunakan untuk membuktikan identitas pengguna dengan pertanyaan seharusnya dapat dipangkas dengan solusi tertentu.

Makalah ini mengusulkan metode *impersonation* (*login as*) berdasarkan persetujuan pemilik akun tanpa melibatkan interaksi langsung dengan pemilik akun dengan menggunakan fasilitas autentikator atau *authentication device* yang dimiliki oleh pengguna. Pada metode ini administrator melakukan permintaan autentikasi di aplikasi klien sebagai akun pengguna lain dengan memilih fitur *impersonate user* kemudian pemilik akun memberikan persetujuan dengan terlebih dahulu melakukan autentikasi di *authentication device*, sebagai contoh *smartphone*. Jika pemilik akun terautentikasi dan menyetujui permintaan administrator, maka administrator dapat masuk ke aplikasi klien sebagai pemilik akun tersebut. Penelitian ini memanfaatkan protokol *Client-Initiated Backchannel Authentication* (*CIBA*)[3] yang diimplementasikan pada server autorisasi dimana menggunakan dua buah perangkat yang terpisah antara perangkat autentikasi oleh pengguna dan perangkat yang digunakan administrator, tetapi kedua perangkat tersebut akan diasumsikan oleh sistem menjadi satu perangkat yang sama. Dengan metode ini administrator juga dapat membuktikan identitas pengguna tanpa perlu memberikan pertanyaan-pertanyaan atau komunikasi secara langsung. Sehingga dengan cara ini administrator atau pengguna yang memiliki hak akses *impersonation* tidak dapat menyalahgunakan hak aksesnya.

II. PENELITIAN TERKAIT DAN KAJIAN LITERATUR

A. Penelitian Terkait

Terdapat beberapa penelitian yang telah membahas kelemahan-kelemahan dari keamanan di sistem single sign-on terutama pada serangan impersonation yaitu Sun dan Beznosov [4], Yang dan Manoharan [5], dan Li, dkk [6]. Serangan ini berfokus kepada serangan yang dilakukan oleh penyerang di luar sistem dan berpura-pura sebagai pengguna sah yang dapat masuk ke sistem sedangkan pada penelitian ini permasalahan terletak pada penyalahgunaan hak akses oleh administrator sistem yang sah untuk melakukan proses impersonation sebagai pemilik akun tertentu sehingga bisa mengakses data sensitif dan melakukan aktivitas tanpa diketahui oleh pengguna pemilik akun.

Selain itu terdapat beberapa penelitian yang membahas tentang pemberian hak akses dari seorang pengguna ke pengguna lainnya yang tidak memiliki hak administratif yaitu metode delegasi [7][8][9][10]. Metode delegasi ini serupa dengan metode impersonation oleh administrator tetapi hak akses diberikan oleh pengguna pemilik akun

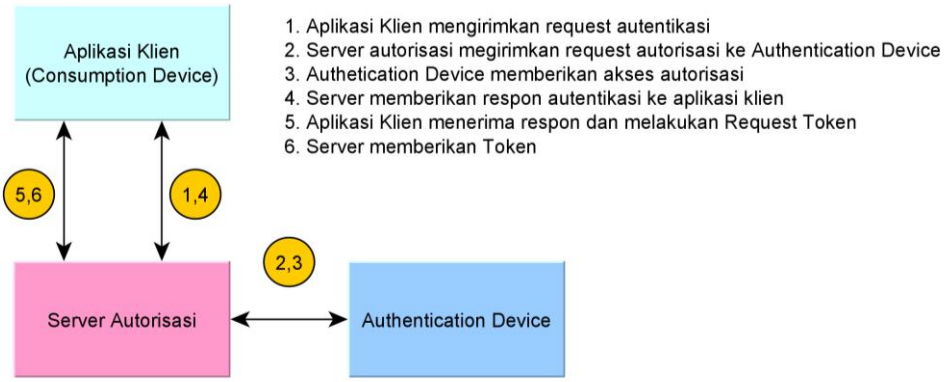
tertentu ke penguna lain secara temporer dan atas kesadaran pemilik akun pemberi hak. Metode delegasi tidak memerlukan mekanisme persetujuan oleh pemilik langsung karena proses delegasi diinisasi oleh pemilik akun dan sudah pasti diawali oleh persetujuan pemilik akun.

B. Client-Initiated Back-Channel Communication

Umumnya aplikasi klien memanfaatkan layanan autentikasi dan autorisasi yang disediakan oleh server autorisasi. Server autorisasi dapat berjalan menggunakan protokol OAuth 2.0 [11] dan OpenID Connect (OIDC) [12][13]. OIDC memungkinkan semua jenis klien yang membutuhkan layanan autentikasi, untuk melakukan autentikasi atau login dan menerima pernyataan yang dapat diverifikasi tentang identitas pengguna yang masuk. Sehingga setiap kali pengguna membutuhkan autentikasi pada suatu aplikasi klien, maka aplikasi klien akan mengalihkan pengguna ke server autorisasi dan dikembalikan ke aplikasi klien ketika autentikasi berhasil.

OIDC telah mengeluarkan alur autentikasi baru yang dinamakan *Client Initiated Backchannel Authentication (CIBA)* [3] dimana aplikasi klien/Relying Party (RP) dapat memperoleh identitas pengguna tanpa interaksi dengan pengguna di aplikasi klien. Alur ini melibatkan komunikasi langsung antara aplikasi klien dengan server autorisasi tanpa pengalihan (*redirect*) melalui browser pengguna seperti pada Gambar 1.

Terdapat beberapa penelitian yang menggunakan metode *back-channel authentication*. Penelitian Almeshekah, dkk memanfaatkan mekanisme komunikasi *back-channel* untuk memberikan tiga level hak akses yang bertingkat sehingga dapat memberikan fleksibilitas pada pengguna dalam mengakses data atau aplikasi [14]. Penelitian W.P. Payack menggunakan kunci elektronik sebagai kontrol akses yang dapat berkomunikasi secara wireless dengan smartphone untuk autentikasi multi-faktor [15]. Jenis autentikasi ini menggunakan fungsi kredensial digital di smartphone sehingga dapat memilih menggunakan teknologi atau komponen apa yang akan digunakan di dalam smartphone. Selain itu terdapat informasi biometrik serta fungsi perbandingan yang dapat digunakan untuk menetapkan parameter autentikasi untuk membuka kunci pintu. Selanjutnya pada penelitian Tonge dkk mengusulkan metode untuk mendapatkan token OAuth melalui autentikasi backchannel dengan cara yang aman [16]. Metode ini dikhususkan untuk mengakses data keuangan dan situasi yang serupa dimana memiliki resiko yang lebih tinggi. Token ini berguna untuk berinteraksi dengan data atau informasi rahasia melalui endpoint REST. Kemudian pada penelitian Chu, dkk mengusulkan metode dan sistem untuk autentikasi pengguna yang aman dengan menggunakan *one time password* (OTP) [17]. Metode ini terbagi menjadi 3 bagian yaitu pra-penyimpanan yang menggunakan aplikasi untuk menghasilkan OTP yang valid, saat autentikasi dimana tidak ada PIN yang disimpan di perangkat komputasi dan penyimpanan di server back-end yang menggunakan algoritma kriptografi jika nilai OTP sesuai dengan nilai OTP yang diterima. Penjelasan terminologi CIBA yang digunakan pada makalah ini dapat dilihat pada Tabel 1.



Gambar 1. Alur autentikasi dan autorisasi pada perangkat yang berbeda.

TABEL 1
TERMINOLOGI PADA PROTOKOL CIBA.

Terminologi	Deskripsi
OpenID Provider (OP)	Server autorisasi yang mengimplementasi OIDC dan CIBA.
Relying Party (RP)	Aplikasi pengguna yang menggunakan layanan CIBA.
Consumption Device (CD)	Perangkat yang digunakan untuk mengoperasikan dan menjalankan aplikasi pengguna.
Authentication Device (AD)	Perangkat yang digunakan untuk melakukan autentikasi dan autorisasi terhadap permintaan akses data yang terproteksi oleh RP. Pada umumnya, perangkat ini berupa *smartphone* dan dipegang oleh pengguna.

III. METODOLOGI

Penelitian ini dilakukan dengan beberapa tahapan yaitu analisis domain permasalahan, perancangan perangkat lunak, implementasi protokol CIBA, dan uji coba kesesuaian implementasi protokol CIBA.

A. Analisis Domain Permasalahan

Domain permasalahan yang diangkat dalam pengerjaan penelitian ini meliputi bagaimana mengimplementasi metode *impersonation* pada server menggunakan protokol CIBA, serta secara khusus pengimplementasian pendaftaran aplikasi klien agar memungkinkan untuk menggunakan alur CIBA, layanan pengambilan dan pemberian *access token* berdasarkan mode *token poll*, *ping*, dan *push*, dan *session binding* antara *Consumption Device (CD)* dan *Authentication Device (AD)*.

Di lingkungan dimana administrator dapat dengan bebas melakukan fitur impersonate tanpa persetujuan pemilik akun maka administrator melakukan langkah-langkah sebagai berikut:

1. mengakses aplikasi klien menggunakan username dan password miliknya,
2. memilih akun yang akan di-impersonate,
3. sistem meminta konfirmasi password administrator untuk verifikasi hak akses,
4. administrator berhasil masuk ke aplikasi klien sebagai akun atau pengguna yang dipilih sebelumnya.

Pada metode ini terdapat titik kelemahan keamanan dimana seorang administrator dengan hak akses *impersonate* dapat masuk ke aplikasi klien sebagai pengguna tertentu sesuai dengan kehendaknya tanpa seijin pemilik akun. Administrator dapat mengakses data pribadi yang sensitif dan melakukan aktivitas di sistem tanpa diketahui pemilik akun. Untuk mengatasi kelemahan sistem tersebut, maka aplikasi klien harus dapat meminta ijin kepada pemilik akun. Metode yang diusulkan adalah sebagai berikut:

1. administrator mengakses aplikasi klien (CD) kemudian CD meminta autentikasi ke server autorisasi (OP),
2. administrator melakukan login di OP menggunakan username dan password,
3. administrator memilih akun yang akan di-impersonate di CD,
4. CD mengirimkan permintaan autentikasi ke OP dengan mekanisme CIBA,
5. OP mengirimkan permintaan autentikasi ke AD milik pemilik akun dengan mekanisme CIBA,
6. pemilik akun melakukan autentikasi di AD (smartphone) dan mengijinkan penggunaan akun miliknya oleh administrator,
7. jika pemilik akun memberikan persetujuan, maka CD mengijinkan administrator untuk masuk ke sistem sebagai pengguna tersebut,
8. jika pemilik akun tidak memberikan persetujuan, maka CD tidak mengijinkan administrator untuk masuk ke sistem sebagai pengguna tersebut.

Analisis keuntungan dari metode *impersonation* dengan CIBA adalah sebagai berikut:

1. dapat memastikan administrator tidak menyalahgunakan kewenangan dalam mengakses akun karena harus meminta ijin kepada pemilik akun,
2. administrator perlu berinteraksi secara langsung dengan pemilik akun untuk dapat meminta ijin dan memverifikasi kebenaran pemilik akun,
3. kebenaran identitas pemilik akun dapat dipastikan di sisi AD, dimana pemilik akun harus melakukan autentikasi. Metode autentikasi bisa menggunakan satu faktor atau multi faktor sesuai dengan tingkat keamanan yang diinginkan oleh pemilik akun.

B. Perancangan Perangkat Lunak

Pada penelitian ini dibangun sebuah mekanisme autentikasi dan autorisasi pada perangkat yang berbeda. Penelitian ini diimplementasikan pada sebuah server yang bertugas sebagai perangkat autorisasi. Pada umumnya, keluhan yang terjadi dapat berupa error yang muncul ketika menggunakan sebuah aplikasi. Agar keluhan tersebut dapat ditangani dengan baik, maka administrator sebaiknya dapat melihat secara langsung error yang terjadi. Satu-satunya cara agar error dapat dilihat secara langsung adalah dengan mengoperasikan akun pelapor dengan melakukan login atas nama yang dipilih dan menggunakan fitur impersonate sebagai akun pelapor dan aplikasi menginisiasi protocol CIBA. Dikarenakan akun pengguna yang bersifat sensitif, diperlukan prosedur agar pemilik akun dapat memberi akses yang dilakukan oleh administrator. Akan tetapi pada cara ini, administrator masih membutuhkan hak akses dari pengguna. Ketika permintaan hak akses muncul pada AD dan diberi hak akses oleh pelapor, maka administrator dapat melanjutkan *impersonation*. Sebaliknya jika pelapor tidak memberi hak akses, maka administrator tidak dapat melanjutkan *impersonation*. Pemberian hak akses oleh pengguna akan memberi tanda bahwa proses CIBA dapat diselesaikan, sehingga aplikasi klien memungkinkan untuk mendapatkan token dari server autorisasi.

Entitas yang berjalan pada lingkungan CIBA ada empat, yaitu *OpenID Provider* atau server autorisasi (OP), *Authentication Device* (AD), Relying Party (RP), dan *Consumption Device* (CD). RP dan CD dalam konteks penelitian ini adalah aplikasi klien yang sama.

Implementasi CIBA dilakukan diatas pustaka Bshaffer OAuth 2.0 Server PHP [18]. Tabel 2 menjelaskan struktur pustaka Bshaffer yang menjadi basis dari pengembangan protokol CIBA. Pustaka Bshaffer memiliki fungsi-fungsi yang dapat mengakomodir alur autorisasi seperti *Authorization Cod*e, *Client Credentials*, *Implicit*, dan *User Credentials*. Protokol CIBA belum didukung oleh pustaka ini. Oleh karena itu terdapat beberapa fungsionalitas CIBA yang harus ditambahkan di *pustaka* Bshaffer meliputi:

1. Inisiasi request autentikasi yang menentukan batasan hak akses serta identifikasi pengguna yang dituju (*authentication request*).
2. Pengiriman dan penerimaan izin hak akses dari pengguna yang dituju (*obtaining end-user consent*).
3. Pembuatan token menggunakan mode *push, ping*, dan *poll* (*issuing token*).
4. Pemberian akses data yang terproteksi menggunakan token.

Aplikasi server autorisasi CIBA merupakan aplikasi yang menggunakan *library* server autorisasi CIBA. Aplikasi ini memanggil fungsi-fungsi yang terdapat pada *library* server autorisasi CIBA. Aplikasi ini tidak menyimpan logika kode untuk memenuhi protokol CIBA, melainkan *library* server autorisasi CIBA yang melapisi logika CIBA. Alasan menggunakan pendekatan ini adalah:

1. Agar tidak terjadi *hard coupling* atau ketergantungan secara langsung antara kerangka kerja dan *library*.
2. Agar dapat menggunakan *library* server autorisasi pada kerangka kerja yang berbeda.

Aplikasi server autorisasi berada dalam proyek yang dinamakan dengan *php-oidc*. *Php-oidc* adalah proyek yang dibangun menggunakan kerangka kerja Phalcon. Proyek ini terdiri dari beberapa modul, yaitu modul *authenticator, oidc, secman,* dan *sso* yang bisa dilihat pada Tabel 2. *Php-oidc* merupakan proyek yang memiliki struktur *multi module*, yang bisa diartikan sebagai proyek besar dengan proyek-proyek spesifik didalamnya. Aplikasi server autorisasi berada pada modul *oidc*.

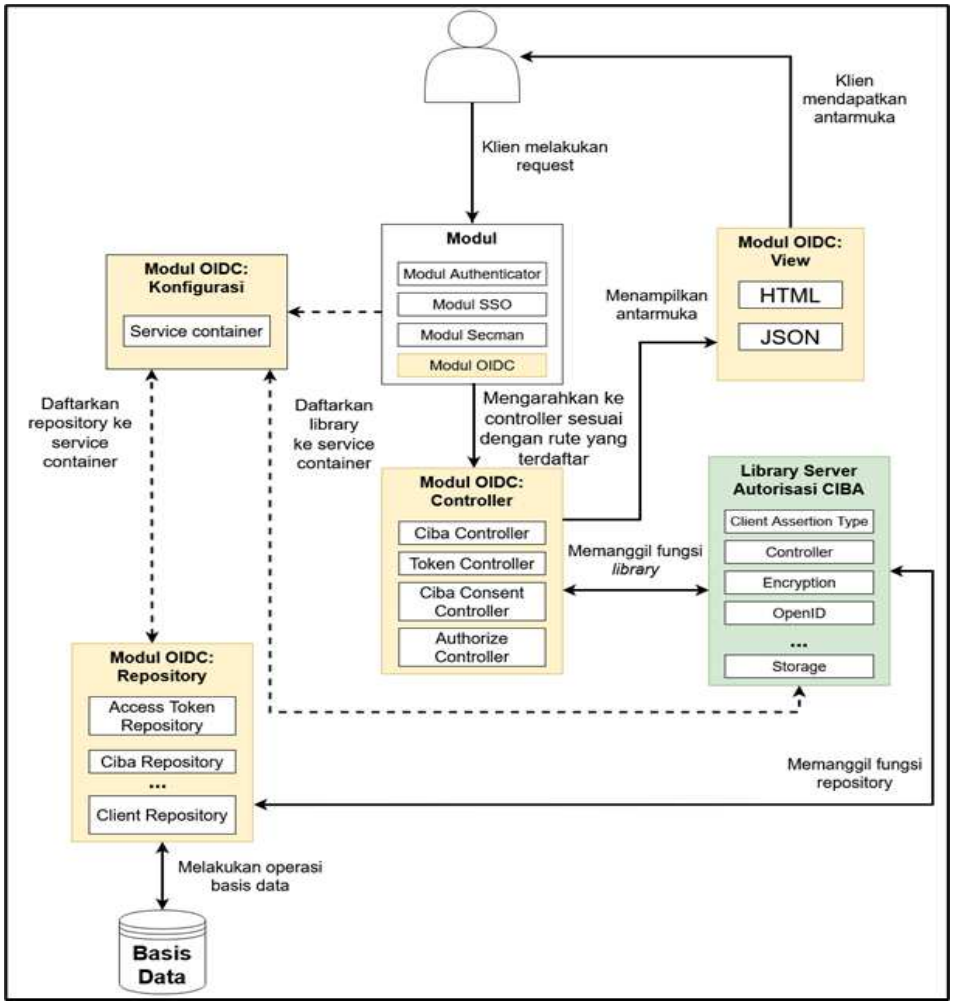
Cara kerja modul *oidc* dapat dilihat pada Gambar 2. Klien (RP/CD) akan melakukan *request* yang ditujukan ke modul *oidc*. Modul *oidc* lalu akan meneruskan *request* yang masuk ke *controller* yang bersangkutan. Pada *controller* terdapat fungsi-fungsi yang akan menggunakan *library* server autorisasi CIBA. Untuk memudahkan pembangunan obyek *library*, modul *oidc* dapat memanfaatkan fitur *service container* dan *dependency injection*. Fungsi *controller* akan memanggil fungsi pada *library* server autorisasi CIBA. Pada *library* tersebut, sudah dilakukan *dependency injection* sehingga kelas *repository* yang terdapat pada modul *oidc* dapat digunakan oleh *library*. *Library* dapat melakukan proses bisnis CIBA lebih lanjut seperti validasi autentikasi, penyusunan *authentication request id* dan penyusunan *access token*. Lalu jika tugas pada *library* sudah selesai, *controller oidc* dapat mengembalikan pesan ke klien dalam bentuk HTML atau JSON.

TABEL 2
STRUKTUR PUSTAKA BSHAFFER (LIBRARY SERVER AUTORISASI).

Nama Direktori	Deskripsi
ClientAssertionType	*Direktori* yang mengelompokkan tipe autentikasi klien.
Controller	*Direktori* yang mengelompokkan layanan protokol OAuth 2.0 dan OIDC. Dilakukan validasi *request* yang masuk sebelum didelegasikan ke lapisan selanjutnya.
Encryption	*Direktori* yang mengelompokkan algoritma enkripsi dan *digital signing*.
GrantType	*Direktori* yang mengelompokkan alur protokol.
OpenID	*Direktori* yang mengelompokkan komponen-komponen yang spesifik ke OpenID.
ResponseType	*Direktori* yang mengelompokkan bentuk obyek *response*.
Storage	*Direktori* yang mengelompokkan tanggung jawab penyimpanan dan pengambilan data.
TokenType	*Direktori* yang mengelompokkan tipe token yang digunakan pada HTTP *request*.
Transport	*Direktori* yang mengelompokkan tanggung jawab pengiriman data ke sistem eksternal melalui protokol pilihan.

TABEL 3
MODUL PADA SERVER OTORISASI.

Nama Modul	Deskripsi
Authenticator	Modul untuk mendukung *Authentication Device.*
Oidc	Modul yang berperan sebagai server autorisasi. Aplikasi ini memberi layanan autentikasi dan autorisasi dengan alur *Authorization Code* dan CIBA.
Secman	Module klien *Security Manager* yang mengatur keamanan server autorisasi.
Sso	Module klien yang memberi antarmuka server autorisasi.



Gambar 2. Aplikasi server autorisasi CIBA .

C. Implementasi Protokol CIBA

Bagian ini menjelaskan alur sistem yang diusulkan mulai dari pendaftaran aplikasi klien, inisiasi request autentikasi, validasi request autentikasi, pemberian hak akses ke pengguna dan pengambilan token oleh pengguna.

Pendaftaran Aplikasi Klien

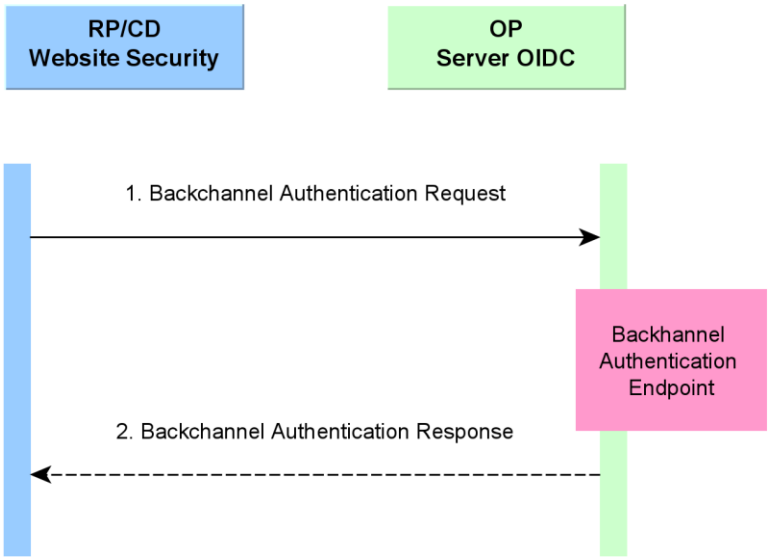
Sebelum memulai proses autentikasi, aplikasi klien harus didaftarkan pada server agar dapat menggunakan protocol CIBA. Pada server terdapat proses pencatatan identitas berupa metadata agar server dapat mengidentifikasi apakah aplikasi klien merupakan aplikasi yang legal atau ilegal. Metadata yang harus diberikan oleh aplikasi klien dapat dilihat pada Tabel 4.

Inisiasi *Request* Autentikasi

Pada tahap pertama, *Relying Party* (RP) akan mengirim *backchannel authentication request* ke *backchannel authentication endpoint* yang dimiliki oleh *OpenID Provider* (OP). Pada Gambar 3, dapat dilihat bahwa ada aplikasi klien yang mengirim *request* autentikasi ke Server OIDC.

TABEL 4
METADATA PENDAFTARAN APLIKASI KLIEN [16].

Nama Metadata	Deskripsi
backchannel_token_delivery_mode	Mode token yang ingin digunakan. Salah satu nilai dari: *poll, ping,* atau *push.*
backchannel_client_notification_endpoint	*Endpoint* aplikasi klien untuk menerima notifikasi dan token.
backchannel_authentication_request_signing_alg	Algoritma yang digunakan oleh aplikasi klien untuk melakukan *digital signing* pada *request* autentikasi.
backchannel_user_code_parameter	Menentukan penggunaan *user code* pada aplikasi klien.



Gambar 3. Aplikasi klien mengirim *request* autentikasi.

Terdapat dua proses pada Gambar 3 yaitu *backchannel authentication request* dan *backchannel authentication response*. Pada proses *backchannel authentication request*, aplikasi klien akan mengirim *request* ke *backchannel authentication endpoint* pada server OIDC, selanjutnya server akan memberi respon melalui *backchannel authentication response*. *Request* autentikasi memiliki beberapa mode token antara lain sebagai berikut.

Request Autentikasi Dengan Aplikasi Klien Yang Terdaftar Menggunakan Mode Token Poll

Pada *request* ini terdapat tujuh paramater yang bisa dikirim yang dapat dilihat pada Tabel 3.

Request Autentikasi Dengan Aplikasi Klien Yang Terdaftar Menggunakan Mode Ping dan Push

Pada *request* ini terdapat delapan parameter yang bisa dikirim. Delapan parameter ini berupa gabungan dari parameter yang telah dijelaskan pada Tabel 3 dan *client_notification_token* yang dapat dilihat pada Tabel 4. Aplikasi klien yang menggunakan *mode* token *ping* dan *push* harus menyertakan *client notification token* sebagai bentuk keamanan. Aturan tersebut bermanfaat untuk mencegah komunikasi yang berasal dari server autorisasi asing.

TABEL 3
PARAMETER REQUEST OTENTIKASI MENGGUNAKAN MODE POLL [16].

Nama Parameter	Deskripsi Parameter	Keterangan
scope	Hak akses yang akan dimiliki oleh RP.	Harus ada
login_hint_token	Berisi *token* yang mengidentifikasi *end-user* yang ingin diautentikasi.	Harus ada (Pilih salah satu antara login_hint_token, id_token_hint, login_hint).
id_token_hint	Berisi *token* yang sebelumnya didapatkan dari OP yang mengidentifikasi *end-user* yang ingin diautentikasi.	Harus ada (Pilih salah satu antara login_hint_token, id_token_hint, login_hint).
login_hint	Berisi identifikasi *end-user* yang ingin diautentikasi. Nilai dapat berupa *email address*, *phone number*, atau *account ID*.	Harus ada (Pilih salah satu antara login_hint_token, id_token_hint, login_hint).
binding_message	Pesan atau kode yang akan dimunculkan pada CD dan AD untuk mengikat *session*	Tidak harus ada
user_code	Kode rahasia yang dimiliki oleh pengguna.	Tidak harus ada
requested_expiry	Nilai untuk memberi jangka kedaluwarsa penggunaan *authentication request id*.	Tidak harus ada

TABEL 4
PARAMETER REQUEST OTENTIKASI MENGGUNAKAN MODE PING DAN PUSH.

Nama Parameter	Deskripsi Parameter	Keterangan
client_notification_token	Berisi *token* yang disusun oleh RP yang akan digunakan oleh OP sebagai *bearer token* ketika OP mengeksekusi *notification endpoint*.	Harus ada

Validasi *Request* Autentikasi

Request autentikasi yang berasal dari aplikasi klien harus divalidasi oleh server autorisasi. Langkah validasi yang dilakukan oleh server autorisasi adalah sebagai berikut:

1. Server autorisasi melakukan autentikasi aplikasi klien menggunakan metode autentikasi yang sudah terdaftar.
2. Jika merupakan *signed request* maka JWT yang terdapat pada parameter *request* harus divalidasi.
3. Memastikan bahwa setiap parameter yang wajib terpenuhi.
4. Memastikan bahwa akun pengguna yang disertakan pada parameter *hint* dapat teridentifikasi.
5. Jika *hint* tidak valid, maka harus mengembalikan response dengan error *unknown_user_id*.
6. Memastikan bahwa setiap parameter memiliki format yang sesuai.
7. Mengabaikan parameter yang tidak diketahui atau tidak sesuai dengan spesifikasi.

Jika ditemukan *error* saat proses validasi, maka server autorisasi akan mengembalikan *response* dengan format yang ditunjukkan pada Tabel 5 dan mengirimkan kode *error* yang digunakan seperti pada Tabel 6. Jika tidak terjadi *error* pada proses validasi, maka server autorisasi mengembalikan *response* dengan format yang dapat dilihat pada Tabel 7.

Pemberian Hak Akses Oleh Pengguna

Jika validasi *request* autentikasi berhasil, maka OP akan mengirim permintaan izin hak akses ke AD [3]. Proses autorisasi akan didelegasikan ke AD. Pengguna melakukan autorisasi di AD dan hasil autorisasi dikirimkan ke OP. Pengguna dapat pilihan untuk mengizinkan atau tidak mengizinkan aplikasi klien. Proses autorisasi dapat dilihat pada Gambar 4.

TABEL 5
FORMAT RESPONSE ERROR OTENTIKASI [16].

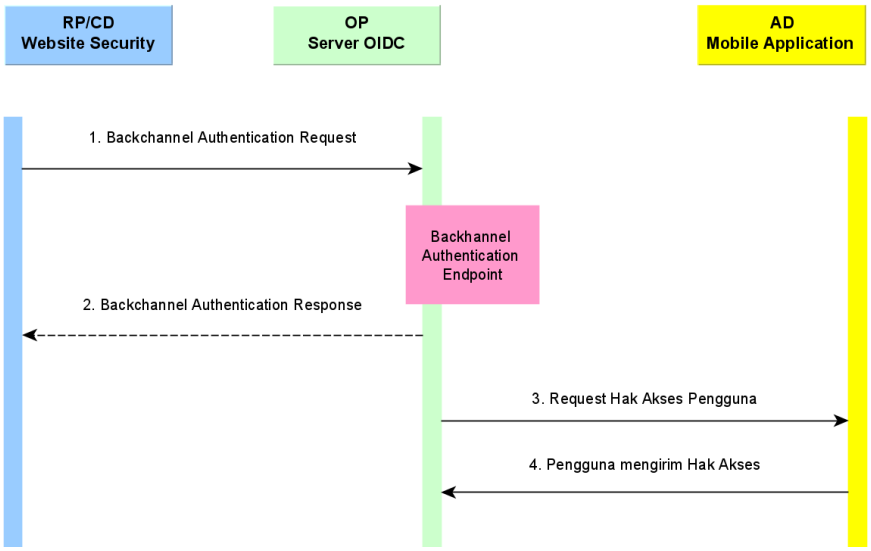
Nama Parameter	Deskripsi	Keterangan
error	Kode *error* yang mendefinisikan *error* yang terjadi.	Harus ada
error_description	Deskripsi *error* yang terjadi.	Tidak harus ada
error_uri	*Link* yang dapat digunakan oleh *developer* untuk melihat penjelasan *error* agar dapat menentukan langkah untuk memperbaiki *error* yang terjadi.	Tidak harus ada

TABEL 6
KODE ERROR OTENTIKASI [16].

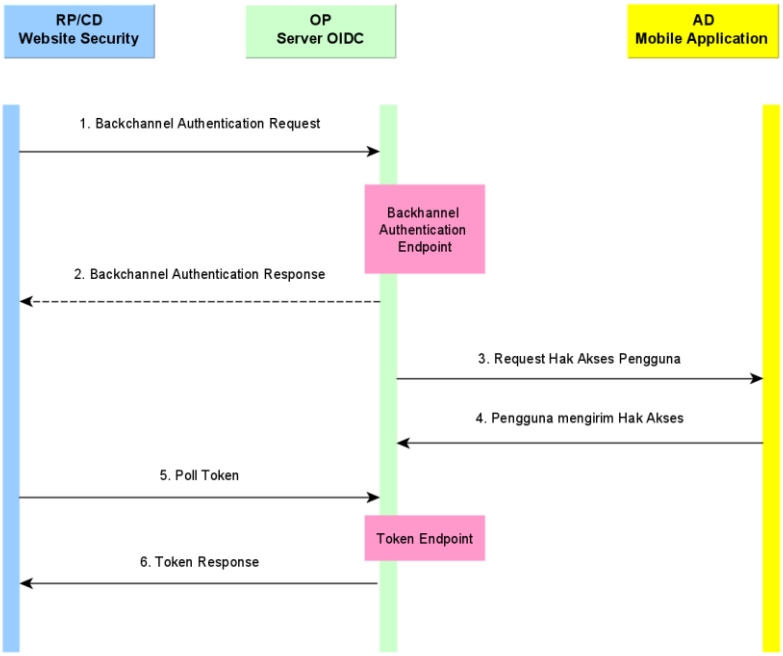
Kode	Deskripsi	Status HTTP
invalid_request	Ada parameter yang tidak lengkap.	400
invalid_scope	*Scope* yang diminta tidak valid.	400
expired_login_hint_token	Token telah kedaluwarsa.	400
unknown_user_id	Akun pengguna tidak ditemukan.	400
unauthorized_client	Aplikasi klien tidak berhak menggunakan protokol tersebut.	400
missing_user_code	Parameter *user code* tidak lengkap.	400
invalid_user_code	Parameter *user code* salah.	400
invalid_binding_message	Parameter *binding message* salah.	400
invalid_client	Autentikasi aplikasi klien tidak berhasil.	401
access_denied	Server autorisasi tidak mengizinkan aplikasi klien untuk menggunakan protokol tersebut.	403

TABEL 7
FORMAT RESPONSE OTENTIKASI YANG BERHASIL [16].

Nama Parameter	Deskripsi	Keterangan
auth_req_id	Identifikasi yang bersifat unik untuk merepresentasikan sesi *request* autentikasi.	Harus ada
expires_in	Waktu kedaluwarsa *auth_req_id*.	Harus ada
interval	Minimal jangka waktu dalam detik untuk melakukan token *polling.* Khusus mode token *poll.*	Tidak harus ada



Gambar 4. Autorisasi pada *authentication device* (*mobile application*).



Gambar 5. Proses mode *token poll*.

Pengambilan Token oleh Aplikasi Klien

Terdapat tiga mode pada pengambilan token oleh aplikasi klien (RP) antara lain (1) mode *token poll*, (2) mode *token ping* dan (3) mode *token push*.

(1) Mode Token Poll

Polling adalah proses melakukan HTTP *request* secara terus menerus dalam interval yang sudah ditentukan. Setelah RP mendapatkan *response* dari *backchannel authentication endpoint* yang dimiliki oleh OP, RP akan melakukan *polling* terhadap token *endpoint*.

Saat proses *polling,* dapat terjadi kemungkinan pengguna belum mengirim izin hak akses. Pada skenario tersebut, aplikasi klien dapat melakukan *polling* berkali-kali sampai pengguna memberi atau menolak hak akses. Proses *polling* dapat dilihat pada Gambar 5.

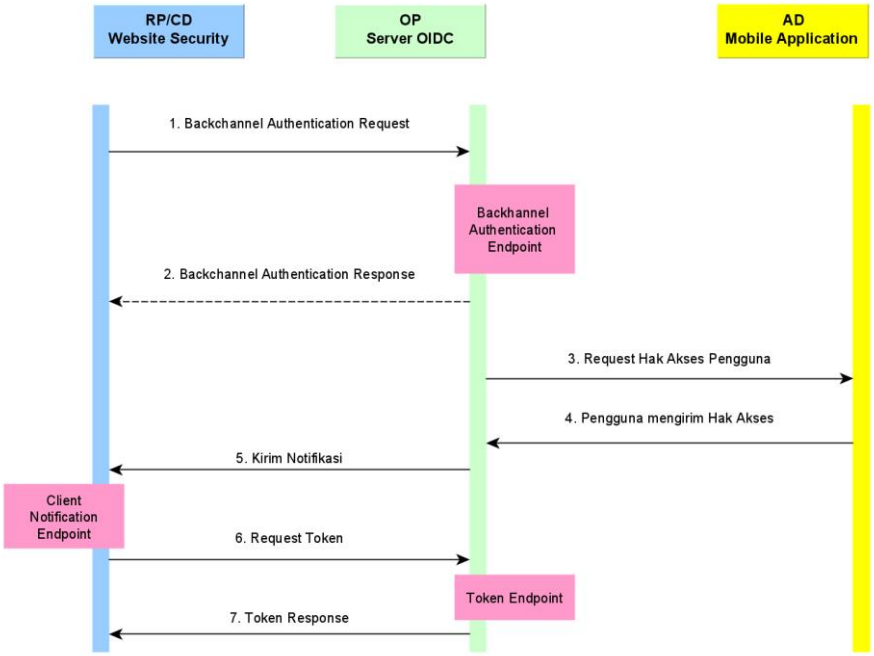
(2) Mode Token Ping

Pada mode *ping*, setelah proses autorisasi sukses dilakukan pada AD, OP akan mengirim notifikasi ke *client notification endpoint* yang telah dibuat ketika mendaftarkan aplikasi klien. Pengiriman notifikasi oleh OP akan disertakan *client notification token* pada *header request*. RP akan memeriksa *client notification token* untuk

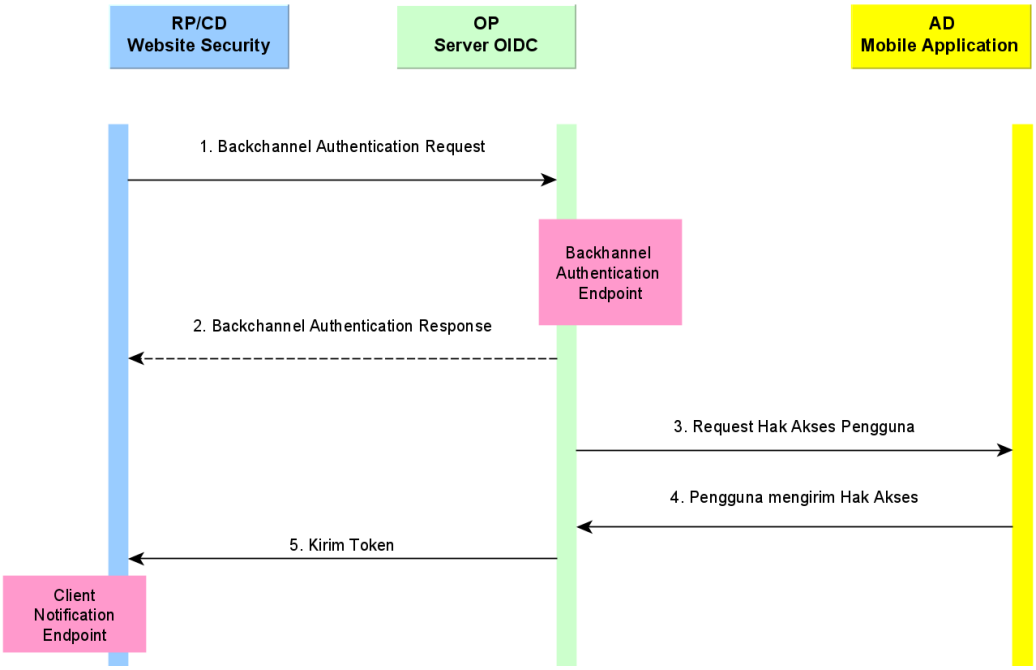
memastikan bahwa OP yang mengirim notifikasi adalah server autorisasi yang terpercaya. RP harus memiliki mekanisme untuk menyimpan *client notification token* yang dibuat ketika melakukan *request* autentikasi agar bisa dicocokkan. Notifikasi yang dikirim bertujuan untuk memberitahu RP bahwa token siap untuk diambil. RP akan mengeksekusi token *endpoint* untuk mengambil token. Proses pengambilan token menggunakan mode *ping* dapat dilihat pada Gambar 6.

(3) Mode Token Push

Pada mode *push,* setelah proses autorisasi sukses dilakukan pada AD, OP akan mengirim token ke RP menggunakan client *notification endpoint* yang telah dibuat ketika mendaftarkan aplikasi klien. Sama halnya seperti mode token *ping*, OP akan menyertakan *client notification token* yang harus divalidasi oleh RP. Selain melakukan validasi *client notification token,* mode token *push* harus mencocokkan hash dari *access token* dan *auth_req_id* dengan atribut *at_hash* dan *urn:openid:params:jwt:claim:auth_req_id* yang terdapat didalam *ID Token*. Proses pengambilan token pada mode token *push* dapat dilihat pada Gambar 7.



Gambar 6. Proses mode *token ping*.



Gambar 7. Proses mode *token push.*

TABEL 8
EVALUASI KASUS PENGUJIAN FUNGSIONALITAS SISTEM.

No.	Kode Kasus Pengujian	Terpenuhi
1	Kasus Pengujian CIBA Menggunakan Mode Token *Push*	YA
2	Kasus Pengujian CIBA Menggunakan Mode Token *Poll*	YA
3	Kasus Pengujian CIBA Menggunakan Mode Token *Ping*	YA
4	Kasus Pengujian Mendapatkan *Protected Resource*	YA
5	Kasus Pengujian Autentikasi Berperan Sebagai Akun Yang Berbeda	YA

IV. UJICOBA DAN EVALUASI SISTEM

Bagian ini menjelaskan proses uji coba yang digunakan untuk menguji fungsionalitas server autorisasi CIBA. Pengujian dilakukan dengan metode *black box* untuk menguji masing-masing fungsionalitas yang sudah dirancang pada sistem. Metode *black box* merupakan metode pengujian perangkat lunak yang memeriksa fungsionalitas perangkat lunak tanpa memandang struktur internalnya.

Server autorisasi CIBA dikembangkan dengan cara mengikuti spesifikasi yang telah dikeluarkan oleh organisasi yang bernama OpenID. Organisasi OpenID menyediakan *conformance testing* yang dapat digunakan untuk melakukan uji coba implementasi server autorisasi yang telah dikembangkan. *Conformance testing* ini bertujuan untuk mengetahui apakah sistem yang dikembangkan sudah sesuai standar spesifikasi atau tidak.

Namun pada saat pengembangan penelitian ini, belum ada *conformance testing* yang dikembangkan OpenID untuk alur CIBA. Salah satu *conformance testing* yang telah disediakan oleh OpenID adalah untuk *Financial Grade API CIBA* (FAPI CIBA) [19]. Sedangkan alur CIBA dan FAPI CIBA memiliki perbedaan yang signifikan [16]. FAPI CIBA menerapkan mekanisme pengamanan yang berbeda, sebagai contoh pada FAPI CIBA tidak mendukung autentikasi klien dengan skema *Basic Authentication,* FAPI CIBA menggunakan skema *private key jwt* dan *mutual transport layer security* (MTLS)*,* namun penerapan CIBA pada penelitian ini menggunakan skema *Basic Authentication*. Selain itu FAPI CIBA harus menggunakan algoritma PS256 atau ES256 untuk menyusun JWT yang menggunakan serialisasi JWS, sedangkan pada penerapan CIBA pada penelitian ini menggunakan algoritma RS256. Sehingga jika menggunakan platform *conformance testing* FAPI CIBA yang disediakan oleh OpenID, hasil *conformance testing* akan gagal.

Hasil evaluasi yang ditunjukkan pada Tabel 8 terpenuhi semua sesuai skenario uji coba. Ketika ada kondisi yang ganjil, sistem akan menampilkan pesan dalam bentuk *error* sehingga pengembang sistem dapat mengetahui alasan terjadinya *error* tersebut.

V. KESIMPULAN

Pada penelitian ini, kami mengusulkan metode *impersonation* dengan menggunakan fasilitas autentikator yang dimiliki oleh pengguna. Metode menerapkan protokol *Client Initiated Backchannel Authentication* (CIBA) yang diimplementasikan pada server dimana menggunakan perangkat yang terpisah antara perangkat autentikasi oleh pengguna dan perangkat yang digunakan administrator, tetapi kedua perangkat tersebut akan diasumsikan oleh sistem menjadi satu perangkat yang sama. Berdasarkan hasil yang didapatkan pada tahap uji coba aplikasi, dapat diambil beberapa kesimpulan antara lain sebagai berikut:

1. Metode *impersonation* dengan persetujuan pemilik akun dapat dilakukan dengan bantuan protokol CIBA sehingga administrator tidak perlu berinteraksi secara langsung dengan pemilik akun serta identitas pemilik akun dapat terverifikasi dengan metode autentikasi di *authentication device*.
2. CIBA server yang dibangun pada penelitian ini dapat mengakomodasi proses bisnis yang terdiri dari pendaftaran aplikasi klien, alur autentikasi *backchannel* pada server, pengambilan *access token* serta *session binding* antar perangkat.
3. CIBA server yang dibangun pada penelitian ini telah memenuhi spesifikasi protokol CIBA dengan melayani permintaan autentikasi dengan parameter yang sesuai, serta tidak melanjutkan permintaan autentikasi jika parameter tidak sesuai yang didukung oleh pesan *error* yang deskriptif.
4. Pustaka CIBA server dapat digunakan pada server autorisasi tanpa menghambat protokol OpenID Connect 1.0 yang sudah tertanam salah satu diantaranya adalah *Authorization Code*.
5. *Test case* yang diambil dari *conformance testing* FAPI CIBA dapat diaplikasikan pada protokol CIBA karena berasal dari pondasi yang sama.

DAFTAR PUSTAKA

[1] P. Hu, R. Yang, Y. Li, dan W. Cheong Lau, "Application impersonation: problems of OAuth and API design in online social networks," dalam *Proc. ACM Conf. Online Social Networks*, hal. 271–278, 2014.

[2] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, dan Y. Koucheryavy, "Multi-Factor Authentication: A Survey," *Cryptography*, 2018.

[3] D. Tonge, J. Heenan, Authlete, T. Lodderstedt, dan B. Campbell, "Financial-grade API: Client Initiated Backchannel Authentication Profile," 15 Agustus 2019. [Online]. Tersedia di https://openid.net/specs/openid-financial-api-ciba.html. [Diakses pada 10 Oktober 2020].

[4] S.-T. Sun dan K. Beznosov, "The devil is in the (implementation) details: an empirical analysis of oauth sso systems," dalam *Proc. ACM Conference on Computer and Communications Security*, hal. 378-390, 2012.

[5] F. Yang dan S. Manoharan, "A security analysis of the OAuth protocol," dalam *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, hal. 271-276, 2013.

[6] Wanpeng Li, Chris J. Mitchell, dan Thomas Chen, "OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenID Connect," dalam *Proc. ACM Workshop on Security Standardisation Research Workshop*, hal. 35–44, 2019.

[7] Ben-Ghorbel-Talbi, Meriam, et al. "A delegation model for extended RBAC," *International Journal of Information Security*, vol. 9, no. 3, hal. 209-236, 2010.

[8] X. Zhang, S. Oh, dan R. Sandhu, "PBDM: a flexible delegation model in RBAC," dalam *Proc. ACM Symposium on Access Control Models and Technologies*, hal. 149-157, 2003.

[9] M. Li dan H. Wang, "ABDM: An extended flexible delegation model in RBAC," dalam *Proc. IEEE International Conference on Computer and Information Technology*, hal. 390-395, 2008.

[10] K. Hasebe, M. Mabuchi, dan A. Matsushita, "Capability-based delegation model in RBAC," dalam *Proc. ACM Symposium on Access Control Models and Technologies*, hal. 109-118, 2010.

[11] Tools.ietf.org. 2021. RFC 6749 - The OAuth 2.0 Authorization Framework. [Online] Tersedia di: <https://tools.ietf.org/html/rfc6749> [Diakses pada 10 Oktober 2020].

[12] R.H. Khan, J. Ylitalo, dan A.B. Ahmed, "OpenID authentication as a service in OpenStack," dalam *Proc. IEEE International Conference on Information Assurance and Security*, hal. 372-377, 2011.

[13] N. Sakimura, N. J. Bradley, P. Identity, M. Jones, M. B. d. Medeiros, dan G. C. Mortimore, "OpenID Connect Core 1.0," OpenID Connect, 2014. [Online]. Tersedia di: https://openid.net/specs/openid-connect-core-1_0.html. [Diakses pada 10 Oktober 2020].

[14] M.H. Almeshekah, M.J. Atallah, dan E.H. Spafford, "Back Channels Can Be Useful! – Layering Authentication Channels to Provide Covert Communication", *Security Protocols XXI*, vol. 8263, hal. 189-195, 2013.

[15] W.P. Payack JR., "Back Channel Authentication Using Smartphones". U.S. Patent Application No 15/383,952, 2017.

[16] D. Tonge, Moneyhub, J. Heenan, Authlete, T. Lodderstedt, dan B. Campbell, "Financial-grade API: Client Initiated Backchannel Authentication Profile," 15 Agustus 2019. [Online]. Tersedia di: https://openid.net/specs/openid-financial-api-ciba-ID1.html. [Diakses pada 10 Oktober 2020].

[17] CHU, Ronald King-Hang, et al. "Methods and systems for secure user authentication". U.S. Patent No 9,768,963, 2017.

[18] https://bshaffer.github.io/oauth2-server-php-docs/ . [Diakses pada 10 Oktober 2020].

[19] https://openid.net/certification/fapi_op_testing/ . [Diakses pada 10 Oktober 2020].