신뢰기반 디지털 ID 관리기술

한국과학기술정보연구원 조진용
jiny92@kisti.re.kr

2024년 6월 13일(목)

Audience

대상

¶ 표준 사용자 인증규약에 관심 있는 누구나

수준

¶ 중급+α
¶ IT 관련 기초지식 필요

참고

¶ 강의자료: https://edu.kafe.or.kr

KAFE

Agenda

1교시 (9:30AM - 10:20AM)

- 배경 지식

2교시 (10:30AM – 11:20AM)

- SAML 인증 규약

3교시 (11:30AM - 12:20AM)

- OIDC 인증 규약

1교시: 배경 지식

암호화

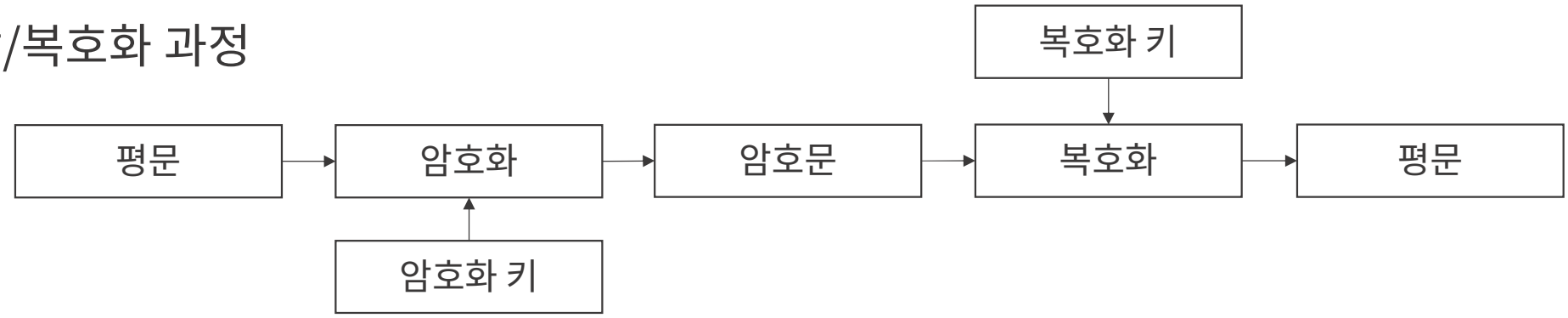
목적

¶ 평문(plain text)를 암호화된 문장(cipher text)로 만드는 과정

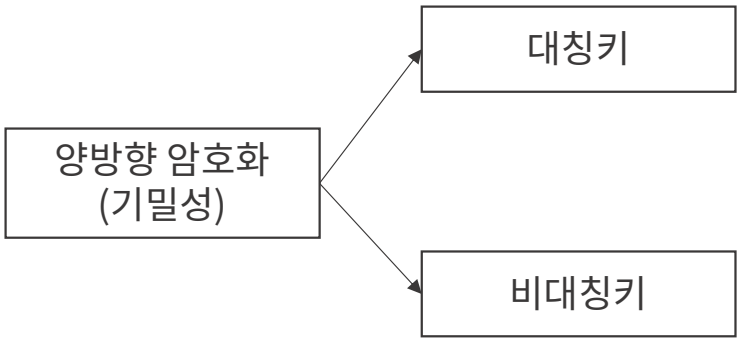
특징

① 기밀성(Confidentiality): 암/복호화
② 무결성(Integrity): 정보의 조작 여부 확인(Hash 함수)
③ 부인봉쇄(Non-repudiation): 송수신자의 송수신 사실 부인 봉쇄(전자서명)

암/복호화 과정

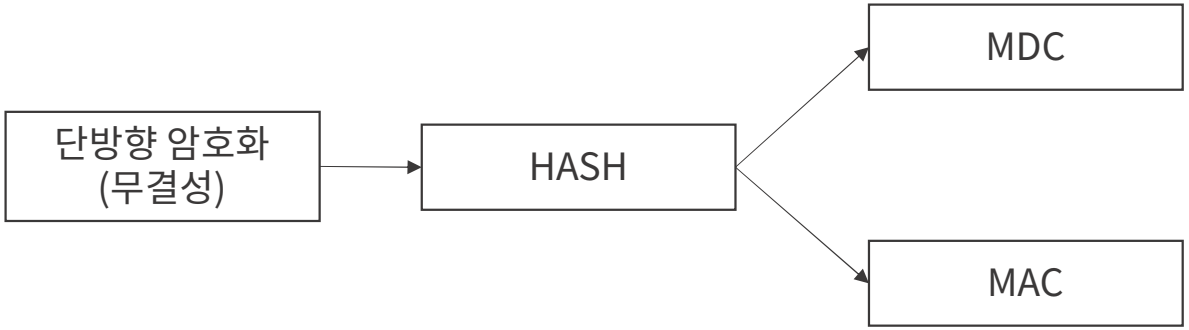


암호화 분류



Symmetric key, Shared key
암호화키 = 복호화키
암호화 목적
AES, DES/3DES, RC4, IDEA, etc.

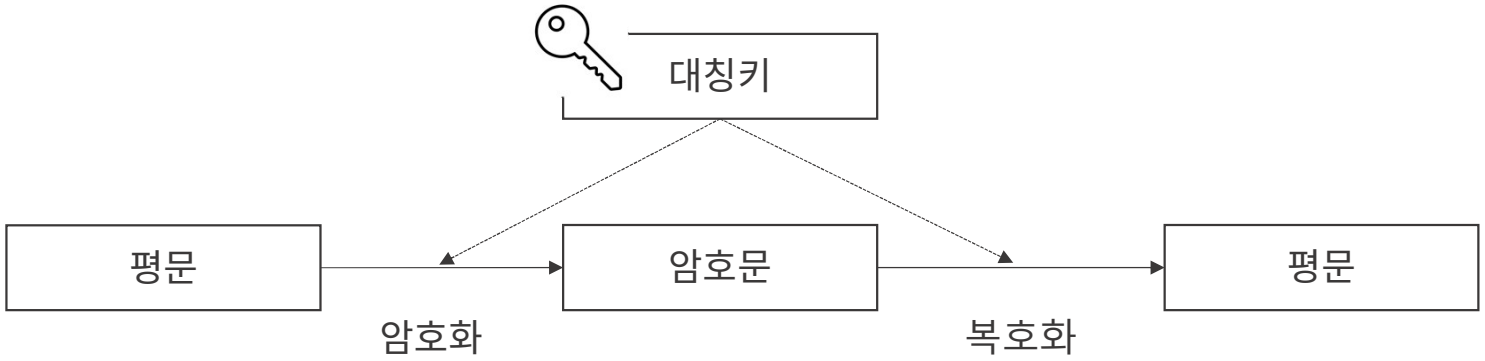
Public/private key
암호화키 ≠ 복호화키
부인방지, 메시지인증
RSA, DH, DSA, ECC



Modification Detection Code
Key 없음
비밀번호 암호화
SHA-1, MD5, etc

Message Authentication Code
Key 있음
OTP
HMAC, etc

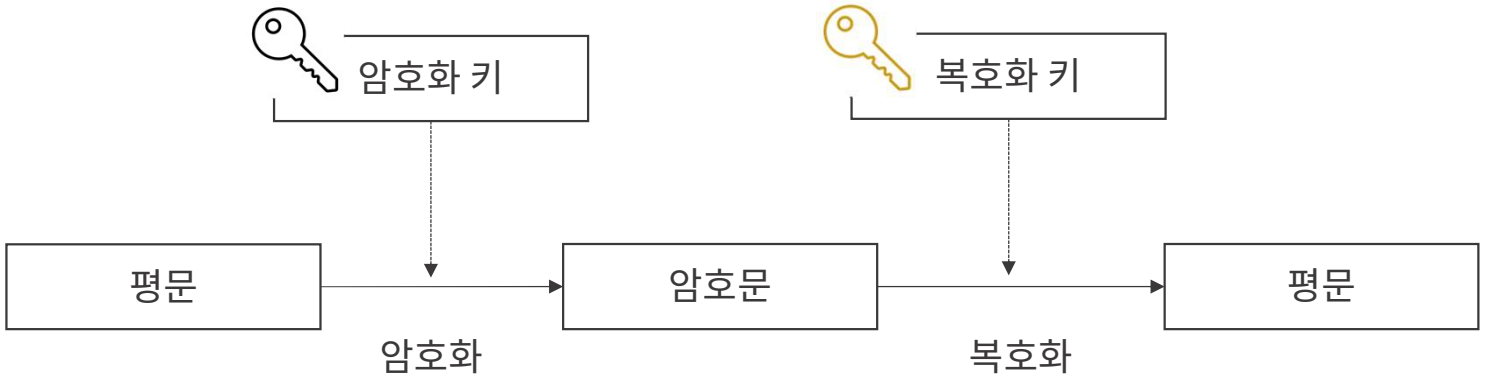
대칭키(Symmetric key)



특징

- 암호화 키와 복호화 키가 동일
- 모든 전송 당사자가 키를 공유해야 함
- 부인방지 불가능
- 구현이 용이하고 빠르지만, 키관리가 어려움
- AES, DES, SEED

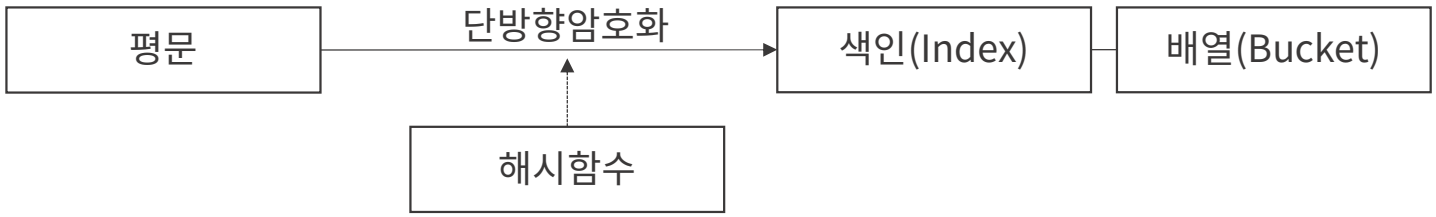
비대칭키(Asymmetric key)

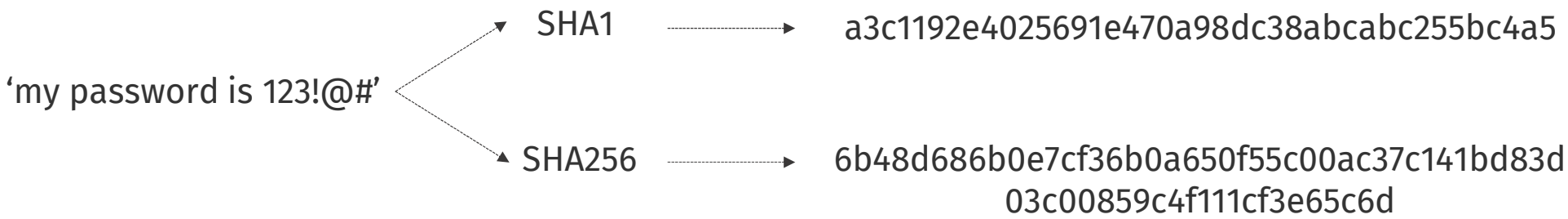


특징

- 암호화 키와 복호화 키가 다름
- 인증기관을 통해 전송 당사자 별로 개인키(Private key)와 공개키(Public key) 발급
- 공개키(Public key)만 공유
- 암호화: 공개키로 암호화하고 개인키로 복호화
- 서명: 개인키로 서명하고 공개키로 서명 검증
- 해독시간이 오래 걸리지만, 암호해독이 어렵고 전자서명이 가능하고
- RSA, ECC

해시(Hash)





특징

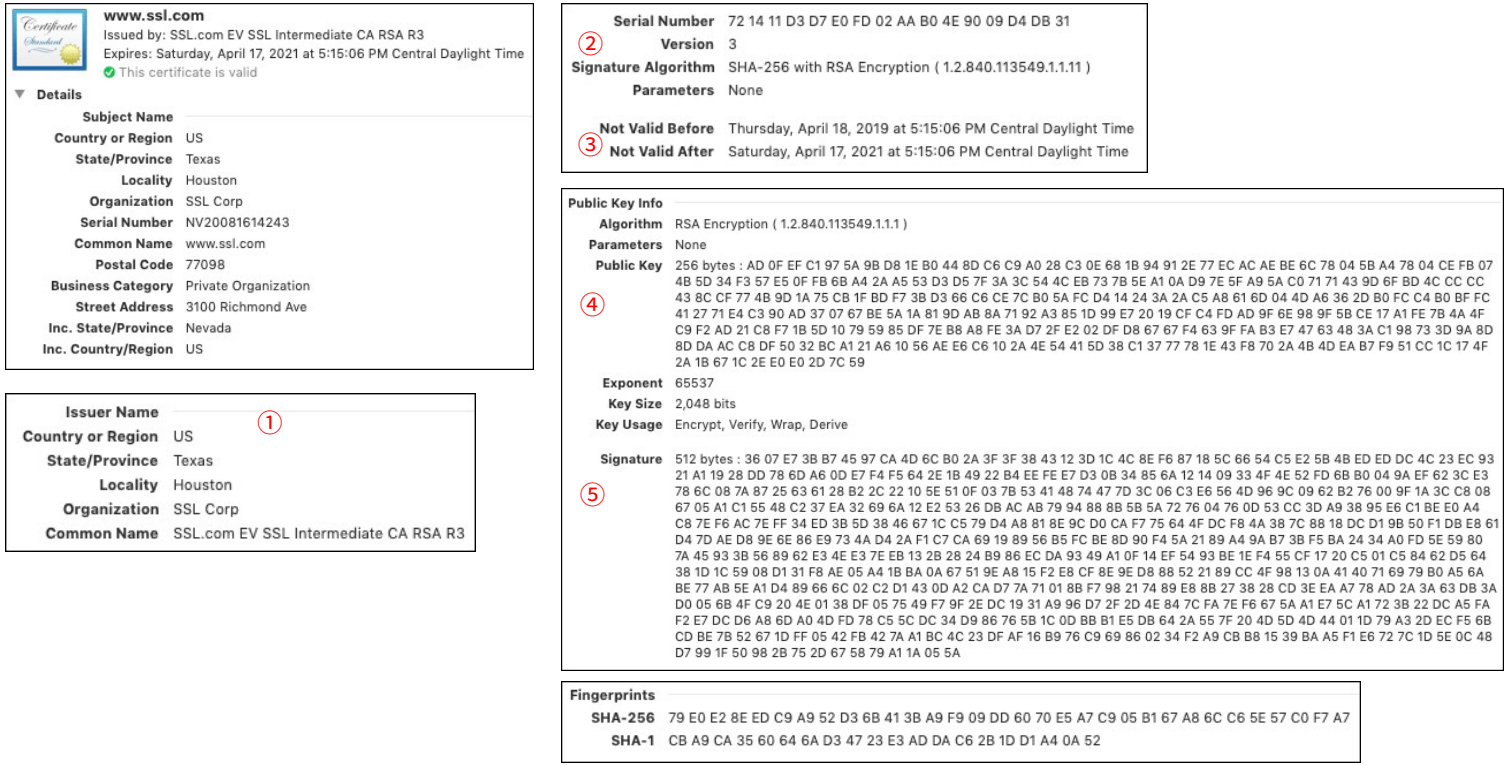
- 복호화 불가능
- 빠르지만, 색인이 충돌될 수 있음

X.509 인증서

특징

- PKI(Public Key Infrastructure)에서 사용되는 디지털 인증서 표준
- 공개키를 안전하게 배포하는데 사용

① 인증서 발급 자
② 서명 알고리즘
③ 유효 기간
④ 공개키 정보
⑤ 인증서 서명



Img. src.: ssl.com

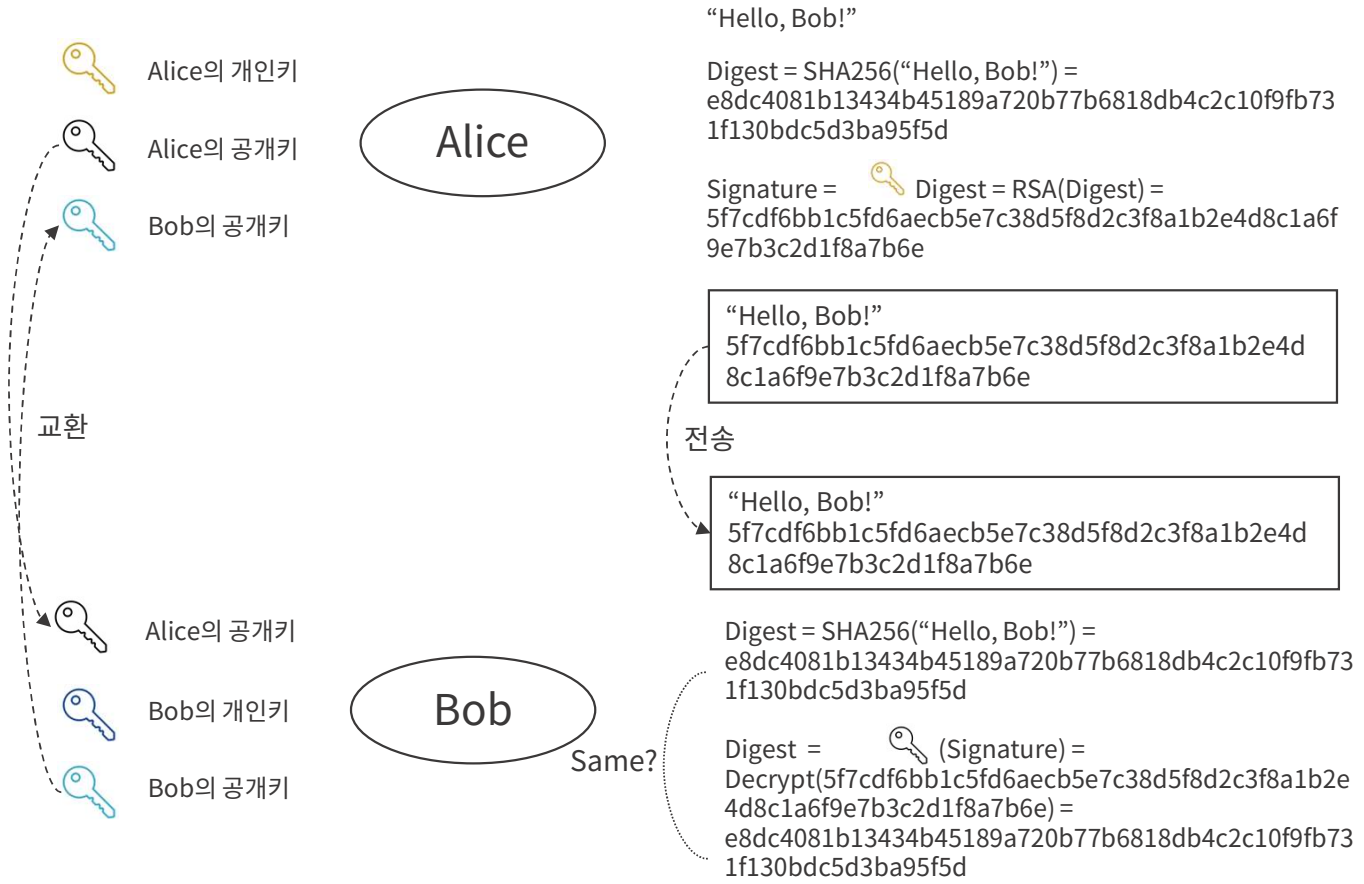
전자서명

Digest

- 메시지의 내용을 해싱한 값
- 메시지의 무결성을 확인

Signature

- Digest를 개인키로 암호화한 값
- 무결성과 출처를 확인



인증 및 인가

인증 (Authentication)

- 사용자의 신원을 확인하는 과정
  - 사용자 주장이 실제 사용자의 신원과 일치하는지 검증
- 크리덴셜(검증하는 수단): uid/pw, 생체 인식, x.509 인증서 등
- 예
  ‘사용자 A가 nametag를 찍고 건물 B에 출입 '

인가/권한부여 (Authorization)

- 인증된 사용자를 대상으로 리소스나 기능에 접근할 수 있는 권한을 확인하고 제어하는 과정
- 사용자의 역할, 책임, 권한 등에 따라 접근 권한을 부여
- 예
  ‘건물 B에 출입한 사용자 A는 사무실에는 출입할 수 있지만, 권한이 없어 전산실에는 출입이 불가능 '

KAFE

신원관리 모델(Identity management model)

Centralized identity

- 중앙 기관이 ID 정보를 관리하고 제어
- 사용자는 중앙 기관에 신원정보를 등록
- 관리 주체가 동일한 서비스에 대해 사용자 인증 가능

Federated identity

- 여러 기관이 ID 정보를 협력 관리
- 사용자는 소속 기관에 신원정보를 등록
- 소셜 로그인이 대표적인 예
- 연합에 참여한(관리주체가 다른) 서비스에 대해 사용자 인증 가능

Decentralized identity

- 사용자 자신이 직접 ID를 관리 및 제어
- 분산원장 기술을 활용
- 신원정보의 선택적 공개

KAFE

Open Standards for Federated Identity

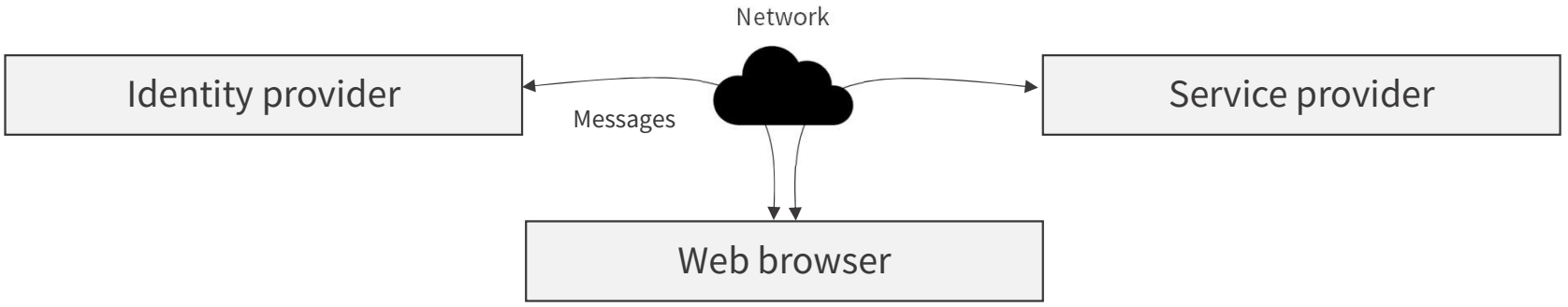
- SAML (Security Assertion Markup Language)
- OIDC/OAuth2 (OpenID Connect)

SAML	OIDC	OAuth2
Developed by OASIS in 2001	Developed by the OpenID Foundation in 2014	Developed by Twitter and Google in 2006
Primary for authentication	Authentication & Authorization	Authorization
XML-based assertions	JSON Web Tokens (JWTS)	JSON Web Tokens (JWTS)
HTTP redirects, GET, POST	Authorization code flow, ID token for authentication	Grant types: authorization code, implicit, password credential, client credentials
Browser, service provider, identity provider	Browser, OIDC provider, OIDC client	Browser, OAuth2 provider, OAuth2 client
Enterprise for web SSO	Web and Mobile SSO	Social SSO

KAFE

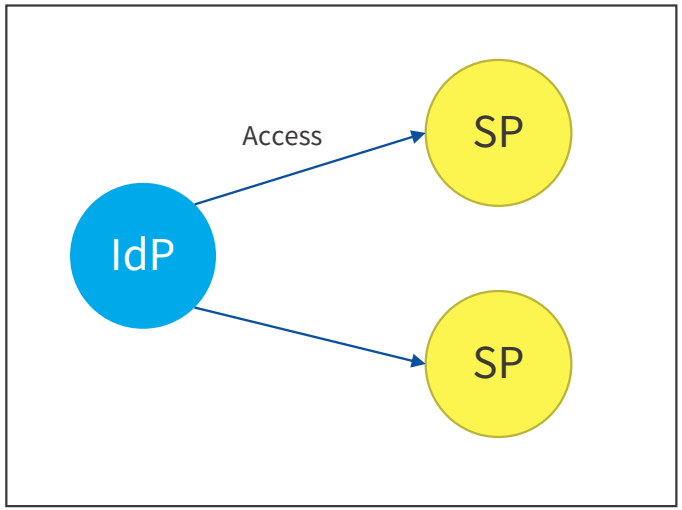
Entities in Federated Identity

Identity provider (IdP)	Service provider (SP)
OIDC provider	OIDC client
Manage user identities and associated attributes	Offers services or resources relying on IdP's identity information (a.k.a., relying party)
Providing identity information to service providers	Trust the IdP to authenticate users
Verifies the user's credentials and issues an assertion or token	Redirects a user to the IdP for authentication. Receives assertion or token and grant access to the user
e.g., social login provider (Google, Facebook, etc.)	Cloud services

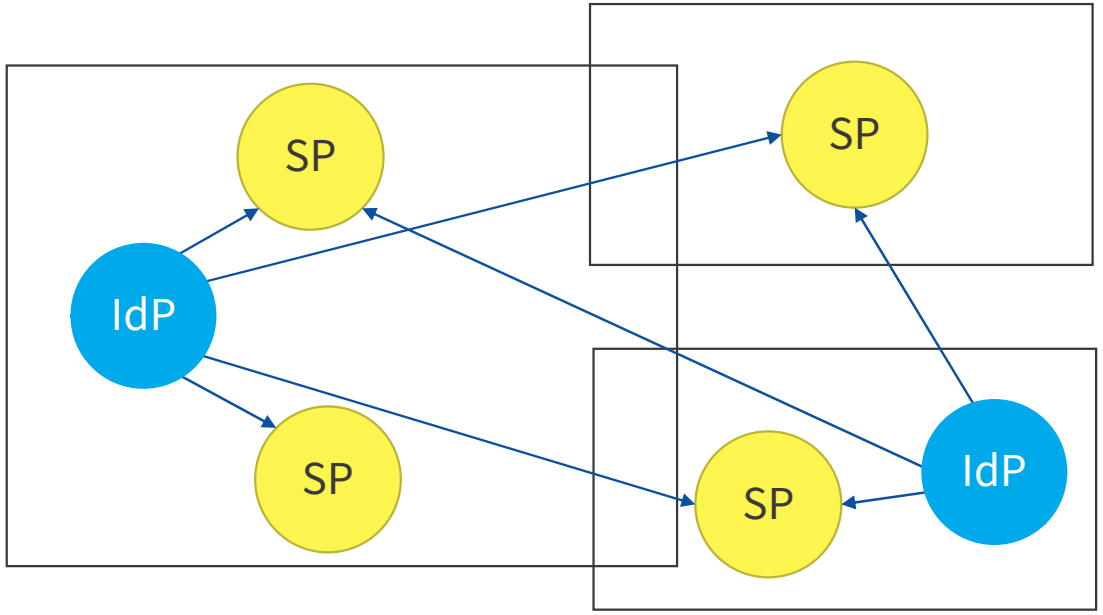


Why federated identity important?

- Increased Inter-operability and accessibility to the world-wide ICT resources



SSO(통합인증) in a single domain (e.g., a university)
Centralized identity management



SSO(통합인증) in multiple domains (e.g., universities)
Federated identity model

KAFE

Message format: XML vs JSON

- 데이터 구조화와 전송을 위한 형식
- XML (eXtensible Markup Language)
- JSON (JavaScript Object Notation)

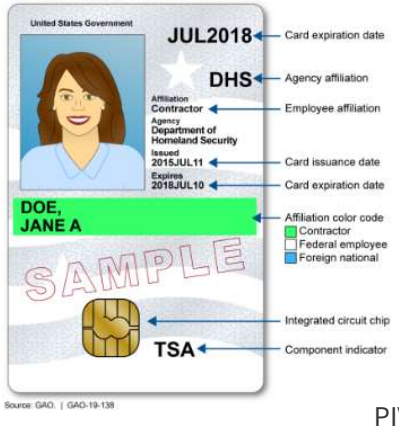
XML	JSON
Tag-based structure	Key-value pairs and arrays
Descriptive, human-readable	Concise, easy-to-use
Data types defined in Schemas or standardized DTD(Document Type Definition)s	Build-in data types: strings, numbers, Booleans, arrays, etc.
Highly extensible, complex	Extensible, simple and fast
Data exchange between systems, configuration files	RESTful API, mobile app data transmission

<config>
<database>
<host>localhost</host>
<port>5432</port>
<username>admin</username>
<password>secret</password>
</database>
</config>
	{
	"users": [
	{
	"id": 1,
	"name": "Alice",
	"email": "alice@example.com"
	}
	]
	}

Multi-factor authentication

- Something you know (지식): password or PIN
- Something you have (소유): token (smartphone, key, smart card)
- Something you are (생체): fingerprint, iris, face recognition

IVR: Interactive Voice Response



PIV(Personal Identity Verification) smart card

Image Source: https://surepassid.com/blog/2023/01/11/types-of-mfa-compared

Type of MFA		NIST-Compliant	Phishing-Resistant	% Targeted Phishing Attacks Resisted[1]	Notes
Other	Last Sign-In Location	No	No	0%	
	Knowledge Questions	No	No	40%	
	Email OTP	No	No	75%	
	Desktop OTP	Yes	No	85%	
Mobile Authenticators	SMS OTP	No	No	75%	Lower cost and complexity, but mobile devices not always allowed or available
	IVR OTP	No	No	75%	
	App OTP (i.e. Google Authenticator)	Yes	No	80%	
	Push Notification	Yes	No	90%	
	FIDO2/WebAuthn mobile token	Yes	Yes	100%	
Hardware Authenticators	OTP hardware token	Yes	No	100%	Higher cost and complexity, but sometimes required
	FIDO2/WebAuthn security key	Yes	Yes	100%	
	PIV smart card	Yes	Yes	100%	
	PIV-derived authenticator	Yes	Yes	100%	

KAFE

2교시: SAML 인증규약

Demo

Realized by Inter-federation (Federated Identity Management)

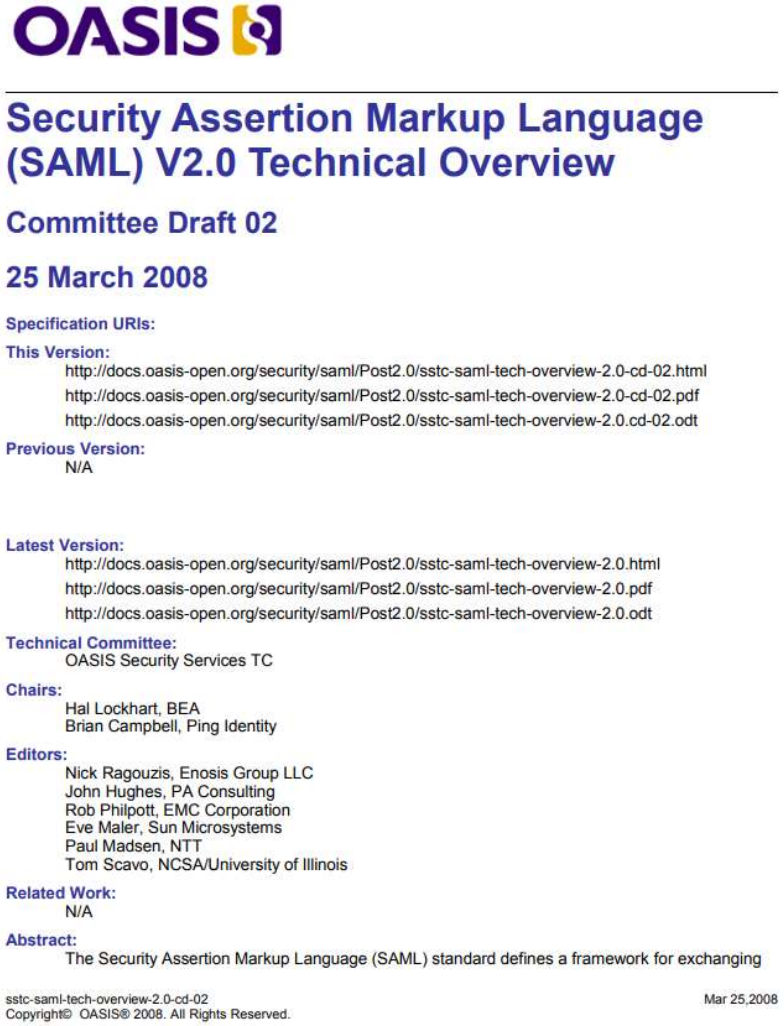
https://ieeexplore.ieee.org/Xplore/home.jsp

- Identity provider: KISTI (KR)
- Service provider: IEEExplore (US)
- Standard: SAML

SAML[샘엘]

SAML (Security Assertion Markup Language)
웹 통합인증(Single Sign On)을 위한 국제 공개표준
Federated authentication에 사용되는 주요 규약

- SAML 1.1(`03), SAML2.0(`05, Errata(정오표)/`19)
  - OASIS 보안서비스 기술위원회 제정
- 연구교육 분야 활용(`05~)
  - SWITCH(스위스 NREN)
- 강화된 보안 vs 적용 어려움
- 웹 환경 지원(모바일 환경 미지원)
  - OAuth2/OIDC 출현



OASIS

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

Committee Draft 02

25 March 2008

Specification URIs:

This Version:
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.cd-02.odt

Previous Version:
N/A

Latest Version:
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.pdf
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.odt

Technical Committee:
OASIS Security Services TC

Chairs:
Hal Lockhart, BEA
Brian Campbell, Ping Identity

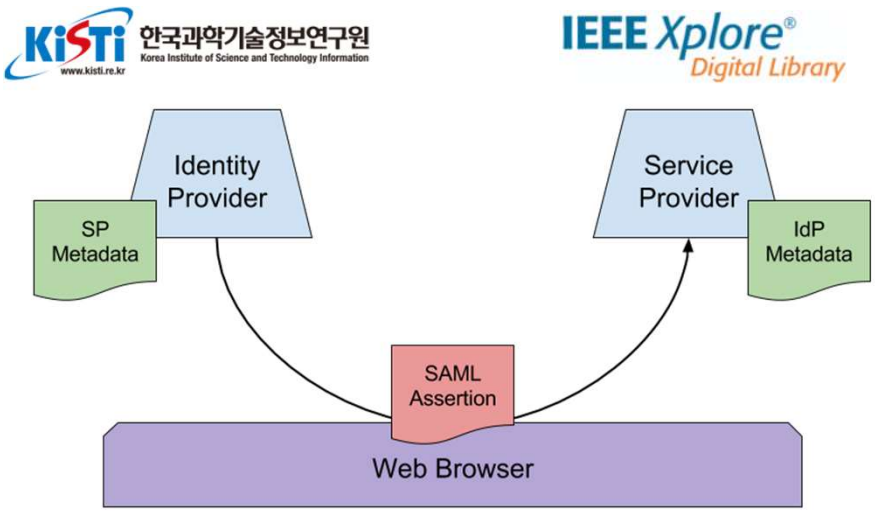
Editors:
Nick Ragouzis, Enosis Group LLC
John Hughes, PA Consulting
Rob Philpott, EMC Corporation
Eve Maler, Sun Microsystems
Paul Madsen, NTT
Tom Scavo, NCSA/University of Illinois

Related Work:
N/A

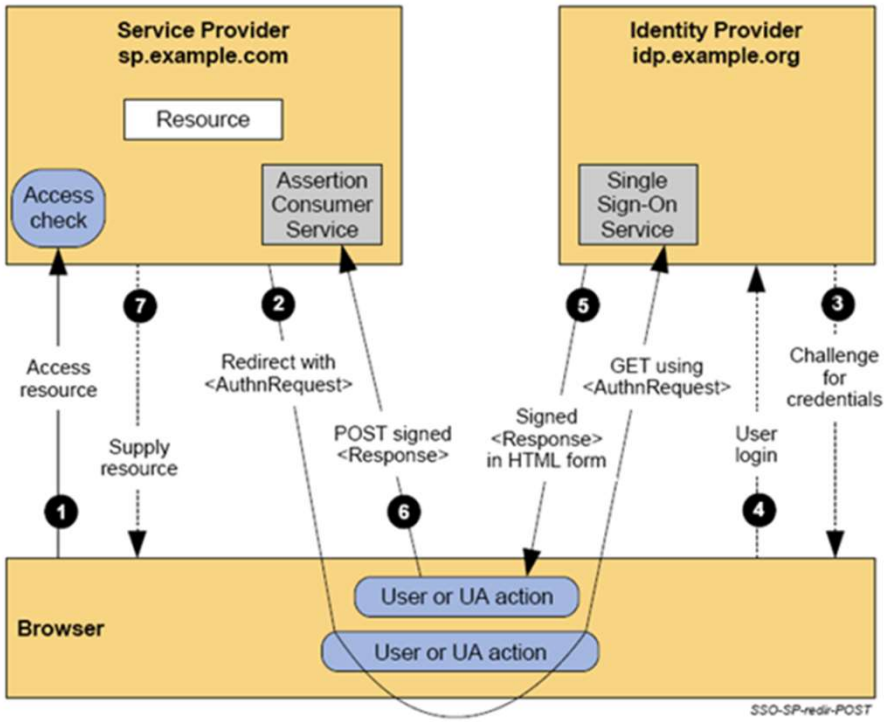
Abstract:
The Security Assertion Markup Language (SAML) standard defines a framework for exchanging

sstc-saml-tech-overview-2.0-cd-02 Mar 25,2008

Web SSO profile



(1) SAML 메타데이터 교환



(2) 인증메시지 교환

Source: Wikipedia, ECLIPSE Foundation

KAFE

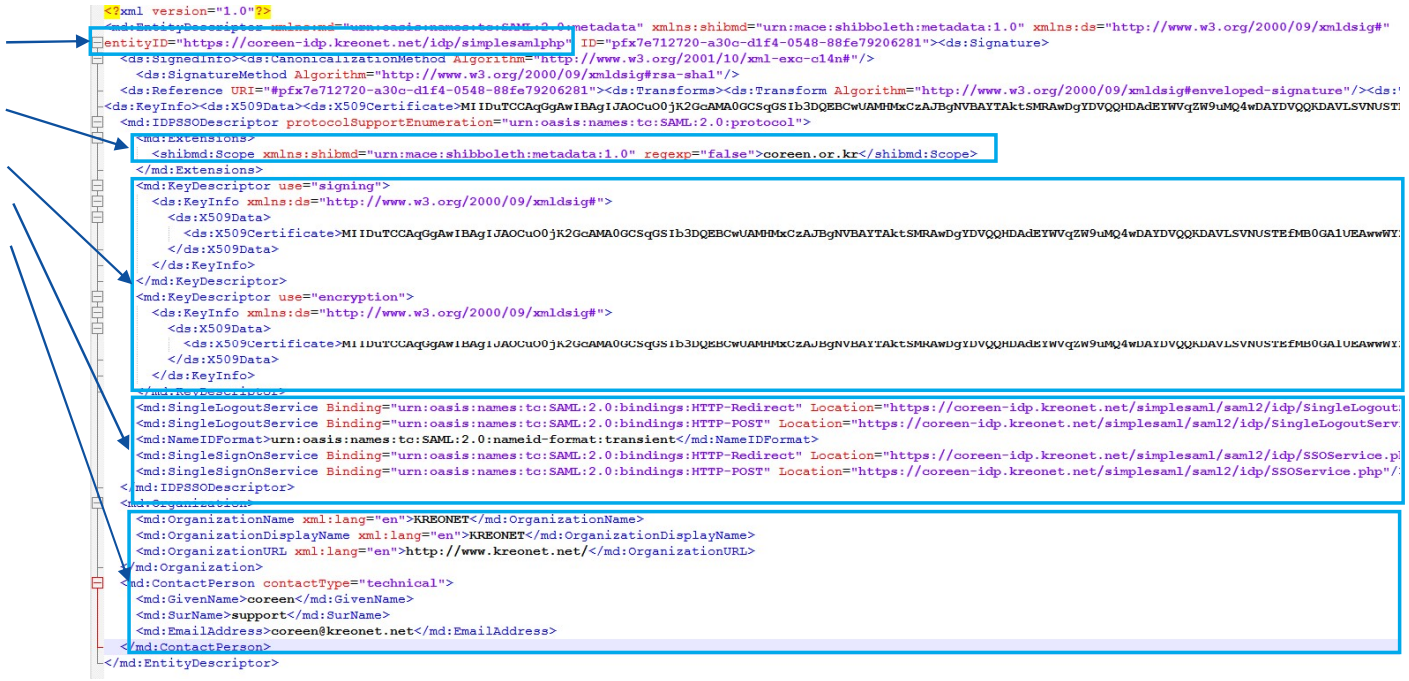
Other SAML profiles

- ECP (Enhanced Client or Proxy) Profile : server-to-server
  - ✓ SSO for non-browser clients; defined how SAML messages can be exchanged directly between IdP and SP without intervening of user's browser.
- Identity Provider Discovery Profile
- Single Logout Profile
- Assertion Query/Request Profile : server-to-server
  - ✓ Request specific SAML assertions from IdP
- Artifact Resolution Profile, and etc.

Key reason for exchanging Metadata

- Exchange public keys for encryption and signature
- Exchange endpoint URLs

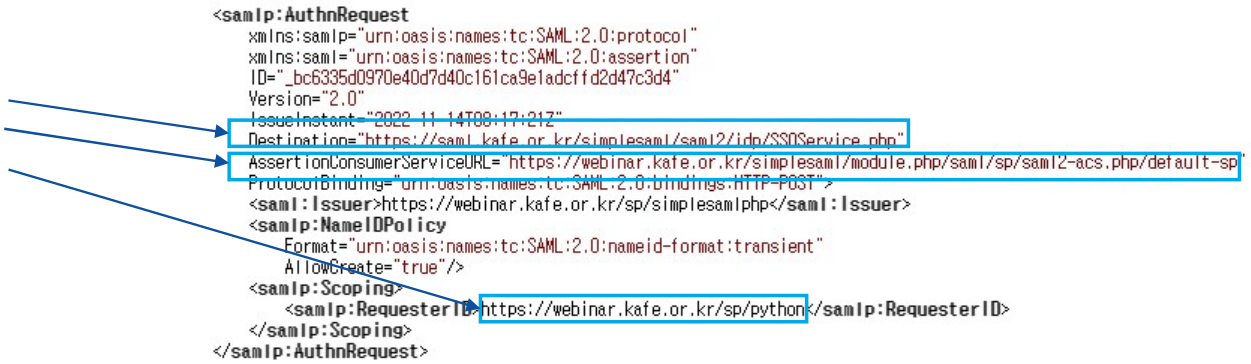
✓ 개체식별자(EntityID)
✓ 기관도메인(Scope)
✓ 공개키(Signature/encryption)
✓ 서비스 주소(Protocol endpoints)
✓ Contacts



SAML request message

- Issued by service providers for user authentication

✓ IdP의 SSO URL
✓ ACS(Assertion Consumer Service) 주소
✓ 서비스제공자 식별자



KAFE

SAML response message

- Issued by Identity provider to deliver user's authentication information
- Assertion: XML-based statement that contains information about a user's identity and attributes.

- ✓ 메시지 발행자(Issuer)
- ✓ Signature
  Digest를 개인키로 암호화
- ✓ X.509 인증서
- ✓ 인증정보: 누가, 언제, 어떤 서비스제공자를 위해;
  어떤 인증방식(예, 비밀번호)으로 로그인 했고;
  메시지가 언제까지 유효한가?
- ✓ 사용자 속성값

```
<samlp:ArtifactResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                        ID="s2d9926a42ce0f17629c6af30a6dd8a15fa7409415" InResponseTo="RkN2Yn1EM"
                        Version="2.0" IssueInstant="2007-01-02T20:48:35Z">
    <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        idp.ssocircle.com
    </saml:Issuer>
    <samlp:Status xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
        <samlp:StatusCode  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
            Value="urn:oasis:names:tc:SAML:2.0:status:Success">
        </samlp:StatusCode>
    </samlp:Status>
    <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                        ID="s276a3300f0e2b3d7f67a800d332156f0fdb99c5d0"
                        InResponseTo="NKRDzVK7e" Version="2.0" IssueInstant="2007-01-02T20:48:35Z">
        <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
            idp.ssocircle.com
        </saml:Issuer>
        <samlp:Status xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
            <samlp:StatusCode  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                Value="urn:oasis:names:tc:SAML:2.0:status:Success">
            </samlp:StatusCode>
        </samlp:Status>
        <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
                         ID="s25d1bbb23ed13c6da325a6f7637c6dc5d9e7f13a9" IssueInstant="2007-01-02T20:48:35Z">
            <saml:Issuer>
                idp.ssocircle.com
            </saml:Issuer>
            <saml:Subject>
                <saml:NameID NameQualifier="idp.ssocircle.com" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
                    amv9OywHQrEW2rmaRViMpWX4qyDA
                </saml:NameID>
                <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                    <saml:SubjectConfirmationData NotOnOrAfter="2007-01-02T20:58:35Z" InResponseTo="NKRDzVK7e"
                        Recipient="http://cgi.cohos.de:80/cgi-bin/zxid" ></saml:SubjectConfirmationData>
                </saml:SubjectConfirmation>
            </saml:Subject>
            <saml:Conditions NotBefore="2007-01-02T20:48:35Z" NotOnOrAfter="2007-01-02T20:58:35Z">
                <saml:AudienceRestriction>
                    <saml:Audience>
                        http://cgi.cohos.de:80/cgi-bin/zxid?o=B
                    </saml:Audience>
                </saml:AudienceRestriction>
            </saml:Conditions>
            <saml:AuthnStatement AuthnInstant="2007-01-02T20:48:35Z" SessionIndex="s24bfc21323ee9c117bf5769a074be1ff177262701">
                <saml:AuthnContext>
                    <saml:AuthnContextClassRef>
                        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
                    </saml:AuthnContextClassRef>
                </saml:AuthnContext>
            </saml:AuthnStatement>
        </saml:Assertion>
    </samlp:Response>
</samlp:ArtifactResponse>
```

Digital signature

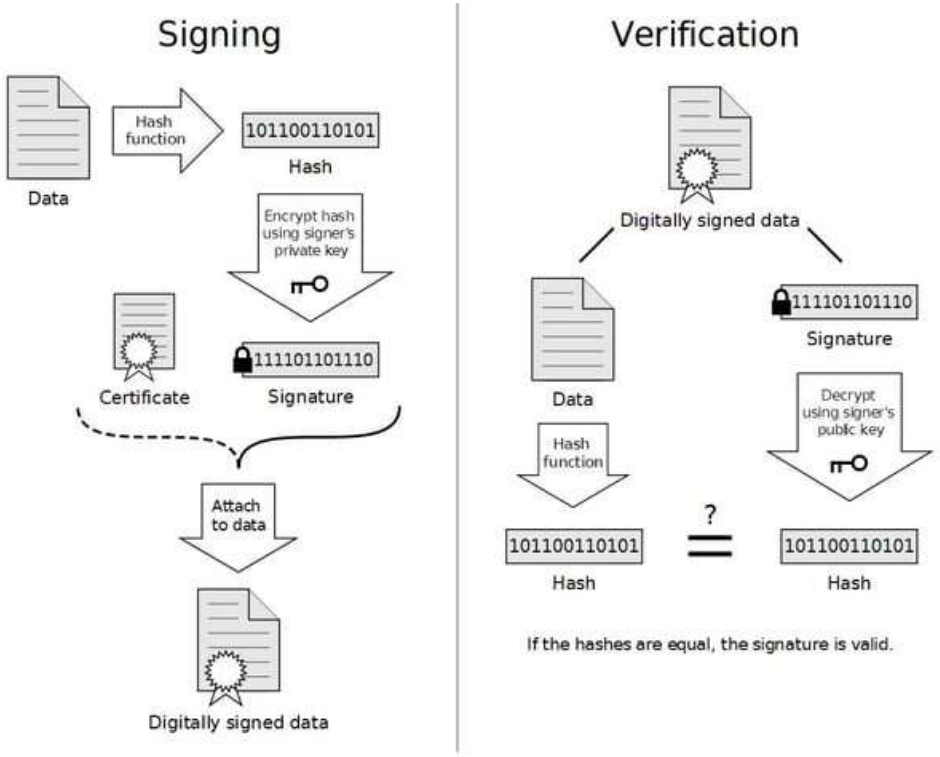


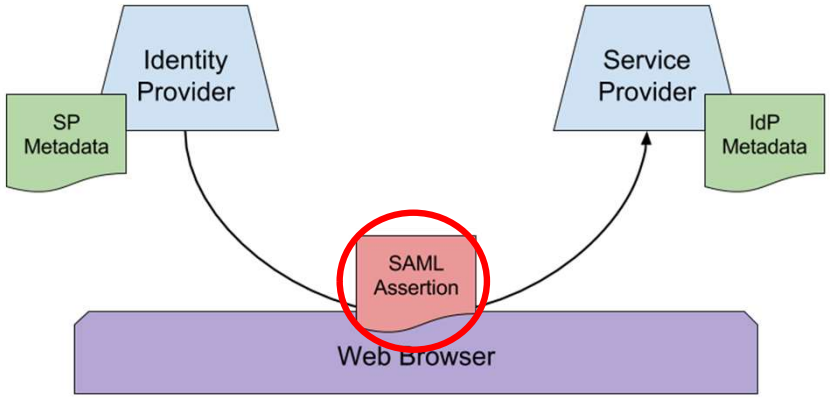
Image Source: https://www.identityfusion.com/

```
<saml:Assertion                                                                    ①
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    ID="_4ee1121c24771a9cf31ef208923f12f5d9e24d8722"
    Version="2.0"
    IssueInstant="2022-11-14T08:17:35Z">
    <saml:Issuer>https://saml.kafe.or.kr/idp/simplesamlphp</saml:Issuer>
    <ds:Signature                                                                  ②
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod
                Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
            <ds:Reference
                URI="#_4ee1121c24771a9cf31ef208923f12f5d9e24d8722">
                <ds:Transforms>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform
                        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
                    Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>
                    whUztIpvkU4o9p3kJbm3aVVylx1biAyZLfWzYyj6vK8=
                </ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>M3Hj+Vzab43p3JWIuUUIH1aVZNIDFfbZn+U32CmbAUAjCCVh4yJnMwkvsz6G8iGiOrUA+N1LO8N24oa1VjnpTsBUBCkhRW69UgdCf/YO
            ZpOAYuR1USWIQw==
        </ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>MIIDOTCCArmgAwIBAgIJAOMgLpVea1i4MA0GCSqGSIb3DQEBCwUAMH8xCzAJBgNVBAYTAktSMRAwDgYDVQQHDAdEYWVqZW9u
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
    <saml:Subject>
        <saml:NameID
            SPNameQualifier="https://webinar.kafe.or.kr/sp/simplesamlphp"
            Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">_9c25d611ebb697df5b96798d37f029a347b7586685
        </saml:NameID>
        <saml:SubjectConfirmation
            Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
            <saml:SubjectConfirmationData
```

Attributes in SAML Assertion

Assertion에 속성(Attributes) 포함

Friendly name/OID 형식 표기
¶ Friendly name
(예): cn/commonName, sn/surName,
¶ OID(Object Identifier)
(예): URN:OID:2.5.4.3



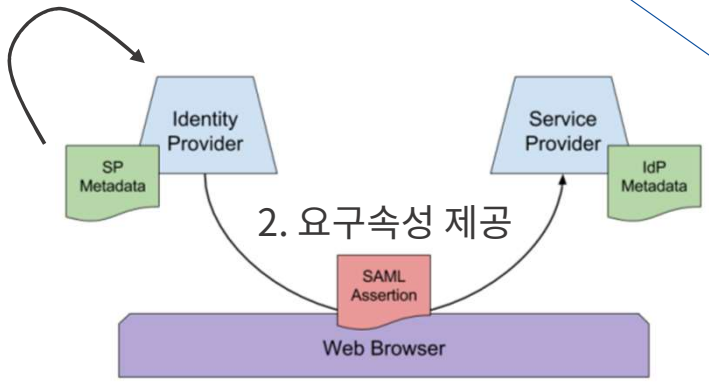
displayName
- SAML Name: urn:oid:2.16.840.1.113730.3.1.241
- LDAP source attribute: suDisplayname
- Example: Prof. John Doe

```
<saml:AttributeStatement>
    <saml:Attribute
        Name="urn:oid:2.16.840.1.113730.3.1.241"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">Jinyong JO
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:0.9.2342.19200300.100.1.3"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">jinyong.jo@gmail.com
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">staff
        </saml:AttributeValue>
        <saml:AttributeValue
            xsi:type="xs:string">member
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:2.5.4.10"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">KISTI
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">jiny92@coreen.or.kr
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:1.3.6.1.4.1.25178.1.2.9"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue
            xsi:type="xs:string">coreen.or.kr
        </saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10"
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml:AttributeValue>
            <saml:NameID
                NameQualifier="https://coreen-idp.kreonet.net/idp/simplesamlphp"
                SPNameQualifier="https://filesender.kreonet.net/sp/simplesamlphp"
                Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">d74f17c949df30493dc8cf7171959c01716f1d03
            </saml:NameID>
        </saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
```

Attributes provided by IdP

- Service provider records the required attributes in the SP metadata
- IdP only delivers the attributes that can be provided among the required attributes in the SP metadata

1. 서비스제공자의 요구속성 확인




```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md="urn:oasis:n
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:saml="urn:oasis:names:tc:SAML:2.0:a
xmlns:shibmd="urn:mace:shibboleth:metadata:1.0" xmlns:mdui="urn:oasis:names:tc:SAML:met
xmlns:mdattr="urn:oasis:names:tc:SAML:metadata:attribute" xmlns:mdrpi="urn:oasis:names:
xmlns:idpdisc="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol" xmlns:init=
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xi="http://www.w3.org/2001/
entityID="https://snu.bookcube.biz/sp/shibboleth">
  <md:Extensions>
   <mdrpi:RegistrationInfo registrationAuthority="http://kafe.kreonet.net" registrationI
  </md:Extensions>
```

…,

```
<md:AttributeConsumingService index="0">
 <md:ServiceName xml:lang="en">Bookcube for Seoul National University</md:ServiceName>
 <md:ServiceDescription xml:lang="en">It is a service that allows you to use e-books online. Users can a
 <md:RequestedAttribute FriendlyName="eduPersonPrincipalName" Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" Na
 <md:RequestedAttribute FriendlyName="eduPersonAffiliation" Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1" Name
 <md:RequestedAttribute FriendlyName="email" Name="urn:oid:0.9.2342.19200300.100.1.3" NameFormat="urn:oa
 <md:RequestedAttribute FriendlyName="displayName" Name="urn:oid:2.16.840.1.113730.3.1.241" NameFormat="
</md:AttributeConsumingService>
```

SAML SP Metadata

Attribute-based Access Control

- application service utilizes attributes for user authorization
- Ex.
  kisti.re.kr에 소속된 학생만 서비스 이용을 허용하라!

```
<saml:Attribute
        Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9"
       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue
          xsi:type="xs:string">student@kisti.re.kr
        </saml:AttributeValue>
</saml:Attribute>
```

User attributes in SAML Assertion from an IdP

<참고>
urn:oid:1.3.6.1.4.1.5923.1.1.1.9 == eduPersonScopedAffiliation

KAFE

REFEDS Specifications

- Standards, and policies for identity management and authentication within the research and education community (단체 표준).

Specification	Easy-to-remember description	Example
Entity Category	Spec. assigned to SAML metadata	Research and Scholarship
Entity Attribute	Spec. assigned to SAML metadata	https://macedir.ogr/entity-category
Profile	Spec. assigned to SAML messages	MFA Profile
Metadata Extension	Spec. extended to SAML metadata	MDUI (Metadata UI) extension, MDPRI (Metadata Registration Practice Statement) extension
Framework	Collection of spec., profiles, and etc.	Assurance Framework

KAFE

Research and Scholarship Category

- Minimize the use of user attributes for privacy preservation.
- SP declares itself as a member of the R&S category → IdP releases a specific set of user attributes to the SPs in this category.
  - eduPersonPrincipalName (ePPN), mail, displayName, givenName, sn, eduPersonScopedAffilation

SP metadata

```
<md:Extensions>
 <mdattr:EntityAttributes>
  <saml:Attribute Name="http://macedir.org/entity-category" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
   <saml:AttributeValue>http://refeds.org/category/research-and-scholarship</saml:AttributeValue>
  </saml:Attribute>
 </mdattr:EntityAttributes>
</md:Extensions>
```

IdP metadata

```
<md:Extensions>
 <mdattr:EntityAttributes>
  <saml:Attribute Name="http://macedir.org/entity-category-support" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri">
   <saml:AttributeValue>http://refeds.org/category/research-and-scholarship</saml:AttributeValue>
  </saml:Attribute>
 </mdattr:EntityAttributes>
</md:Extensions>
```

KAFE

MFA profile

- Defined in Authentication Context Class Reference (ACR) profile → Determine the level of assurance.

Request message from SP to IdP

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
                    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                    ID="abc123" Version="2.0" IssueInstant="2023-06-10T12:00:00Z"
                    AssertionConsumerServiceURL="https://sp.example.org/acs">
 <saml:Issuer>https://sp.example.org</saml:Issuer>
 <samlp:RequestedAuthnContext Comparison="exact">
  <saml:AuthnContextClassRef>https://refeds.org/profile/mfa</saml:AuthnContextClassRef>
 </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

Response message from IdP to SP

```
<samlp:Response xmlns:samlp=…,>
 <saml:Issuer>https://idp.example.org</saml:Issuer>
…,
  <saml:AuthnStatement AuthnInstant="2023-06-10T12:00:30Z">
   <saml:AuthnContext>
    <saml:AuthnContextClassRef>https://refeds.org/profile/mfa</saml:AuthnContextClassRef>
    <saml:AuthenticatingAuthority>https://idp.example.org</saml:AuthenticatingAuthority>
   </saml:AuthnContext>
  </saml:AuthnStatement>
 </saml:Assertion>
</samlp:Response>
```

KAFE

3교시: OIDC/OAuth2 인증규약

OpenID Connect

- OAuth2에 Identity layer를 추가한 규약
  - ✓ Use of the ID token and UserInfo endpoint
- Next generation federated identity를 위한 규약

KAFE

Discovery and Registration

- OpenID Connect Discovery: A mechanism for RP(Relying Party)s to discovery the OP(OIDC Provider)'s configuration including endpoints, supported scopes, and public keys
- Dynamic client registration
  - ✓ A protocol for RPs to dynamically register with the OP and obtain client credentials (Client ID and Secret)

Discovery URL

```
https://[openid-provider.example.com]/.well-known/openid-configuration
```

Example of an OIDC provider configuration response

```
{
 "issuer": "https://openid-provider.example.com",
 "authorization_endpoint": "https://openid-provider.example.com/authorize",
 "token_endpoint": "https://openid-provider.example.com/token",
 "userinfo_endpoint": "https://openid-provider.example.com/userinfo",
 "jwks_uri": "https://openid-provider.example.com/jwks",
 "scopes_supported": ["openid", "profile", "email"],
 "response_types_supported": ["code", "id_token", "token id_token"],
 "subject_types_supported": ["public"],
 "id_token_signing_alg_values_supported": ["RS256"]
}
```

참고) SP (Service Provider) == RP (Relying Party) == OIDC Client

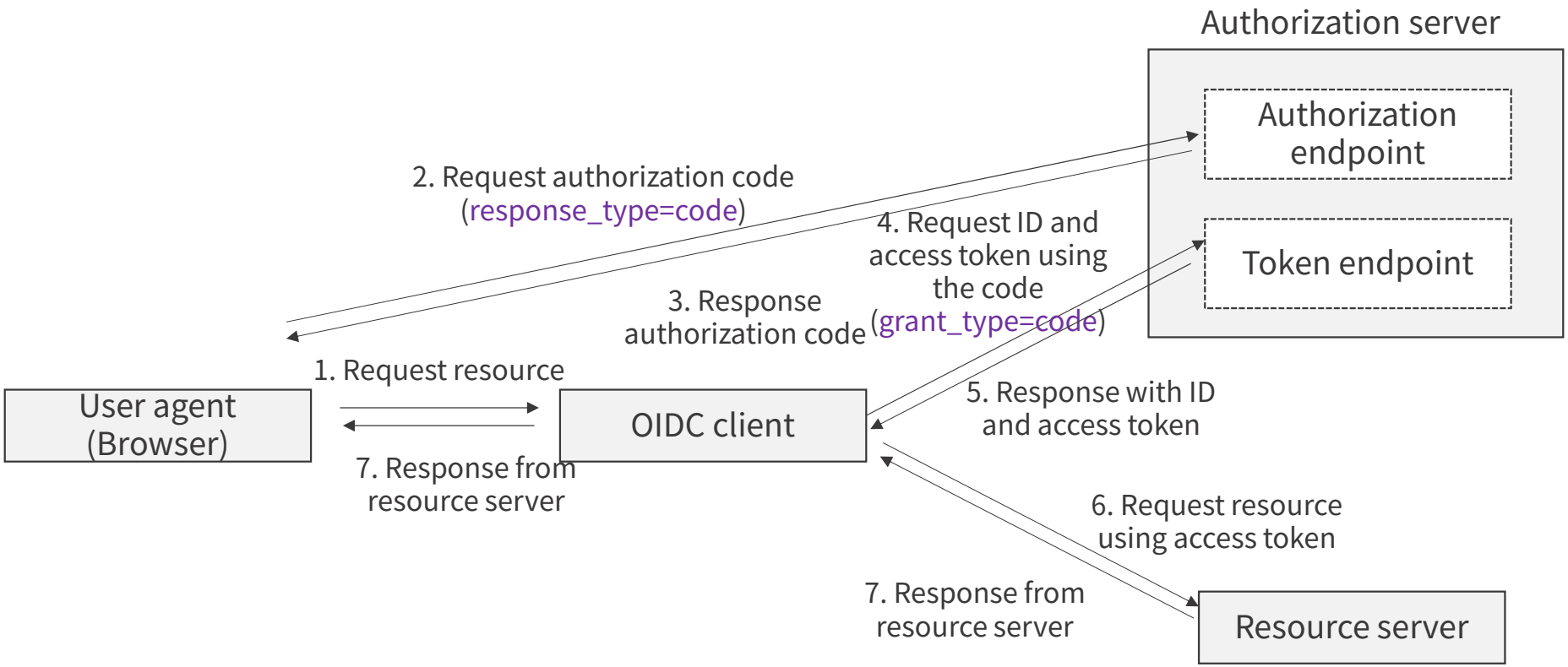
KAFE

Endpoints

- Authorization Endpoint: User is redirected for authentication and consent.
- Token Endpoint: RP (OIDC client) exchanges the authorization code for tokens
- UserInfo Endpoint: Optional endpoint on the OP that returns additional claims about the authenticated user.

KAFE

Overview of OIDC code flow



KAFE

OIDC Authentication flows

- Authorization Code Flow: OP(OIDC provider) issues an authorization code, which the RP(OIDC client) exchanges for an ID token and access token
- Implicit Flow: OP directly returns the ID token and access token to RP. Less secure.
- Hybrid Flow: OP returns both and authorization code and tokens to the RP

Property	Code flow	Implicit flow	Hybrid flow
Authorization endpoint issues all tokens	No	Yes	No
Token endpoint issues all tokens	Yes	No	No
Tokens not revealed to User agent (Browser)	Yes	No	No
Client can be authenticated	Yes	No	Yes
Refresh token possible	Yes	No	Yes
Token relayed via web browser?	No	Yes	No
Server-to-server communication for tokens?	Yes	No	Varies

KAFE

response_type

- Specify desired response

```
GET /authorize?
 response_type=code
 &scope=openid%20profile%20email
 &client_id=123456
 &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
 &state=abc123
 &claims={"id_token":{"name":null,"email":null}}
```

response_type values	Flow
code	Authorization code
id_token	implicit
id_token token	Implicit
code id_token	Hybrid
code token	Hybrid
code id_token token	Hybrid

token == access_token

PKCE('pixie') extension

- Proof Key for Code Exchange
- To prevent 'code interception attacks'

hello ea09ae9cc6768c50fcee903ed054556e5bfc8347907f12598aa24193

① Client는 unique secret(i.e., code verifier)과 code challenge(i.e., hashed version of the code verifier)를 생성

ea09ae9cc6768c50fcee903ed054556e5bfc8347907f12598aa24193

② Authorization request 시, code challenge를 함께 보냄; response 시, authorization code 반환

③ Token request 시, 전달받은 authorization code와 code verifier를 보냄

④ Authorization server는 code verifier로 code challenge를 생성하고 ②의 code challenge와 일치하는지 확인

hash(hello) == ea09ae9cc6768c50fcee903ed054556e5bfc8347907f12598aa24193

※ code challenge method(hash 알고리즘)는 사전에 정의: SHA256 또는 plain

KAFE

PKCE(‘pixie’) extension (contd.)

- Authorization code request

```
/authorize?
 response_type=code
&client_id={yourClientId}
&code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
&redirect_uri={yourCallbackUrl}
&scope=appointments%20contacts
&audience=appointments:api
&state=xyzABC123
```

- Token request

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code={authorizationCode}
&client_id={yourClientId}
&redirect_uri={yourCallbackUrl}
&code_verifier={yourCodeVerifier}
```

KAFE

grant_type

- Used in token requests : e.g., I am sending grant_type (authorization_code), please response it!

grant_type	Description	OIDC/OAuth2 Flows
authorization_code	Request tokens using the code	Authorization code flow
refresh_token	Request new access token using a refresh token	-
client_credentials	Request an access token using client credentials	Client Credential Flow
password	Request an access token using User's password	Resource Owner Password Credentials Flow
Implicit	Request an access token without authorization code	(deprecated!!)

KAFE

Token endpoint authentication method

- Way a client authenticates itself to the authorization server when making a request to the token endpoint to obtain an access token or refresh an access token

  ✓ Client secret basic authentication:
  base64_encode(s6BhdRkqt3:gXAYYki8jW5asTgFi6SpEKjoa6VxQn)

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWEFZWWtpOGpXNWFzVGdGaTZTcEVLam9hNlZ4UW4=
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=xyz...&redirect_uri=https://client.example.com/cb
```

  ✓ Client secret post authentication

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=xyz...&redirect_uri=https://client.example.com/cb&client_id=s6BhdRkqt3&client_secret=gXAYYki8jW5asTgFi6SpEKjoa6VxQn
```

  ✓ Client secret JWT authentication
  ✓ Private key JWT authentication

KAFE

When need the client secret?

- In other words, what cases do we need the ‘token endpoint authentication’?

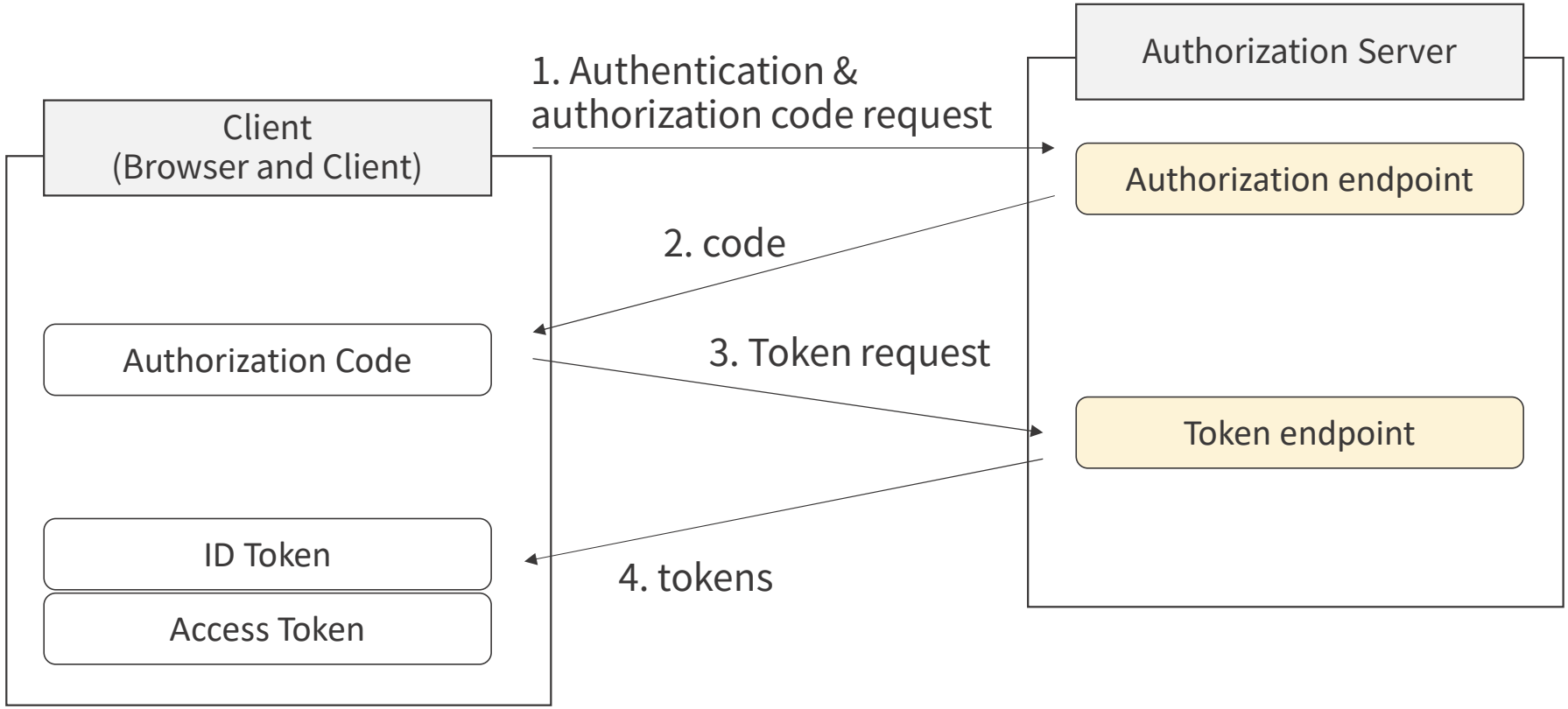
- Require
  - Authorization code flow: Token request, Refresh token request
  - Client credential flow: Client credential grant
  - Resource owner password credential flow: Resource owner password credential grant

- Do not require
  - Implicit flow
  - Authorization code flow with PKCE

KAFE

OIDC flows based on response_type

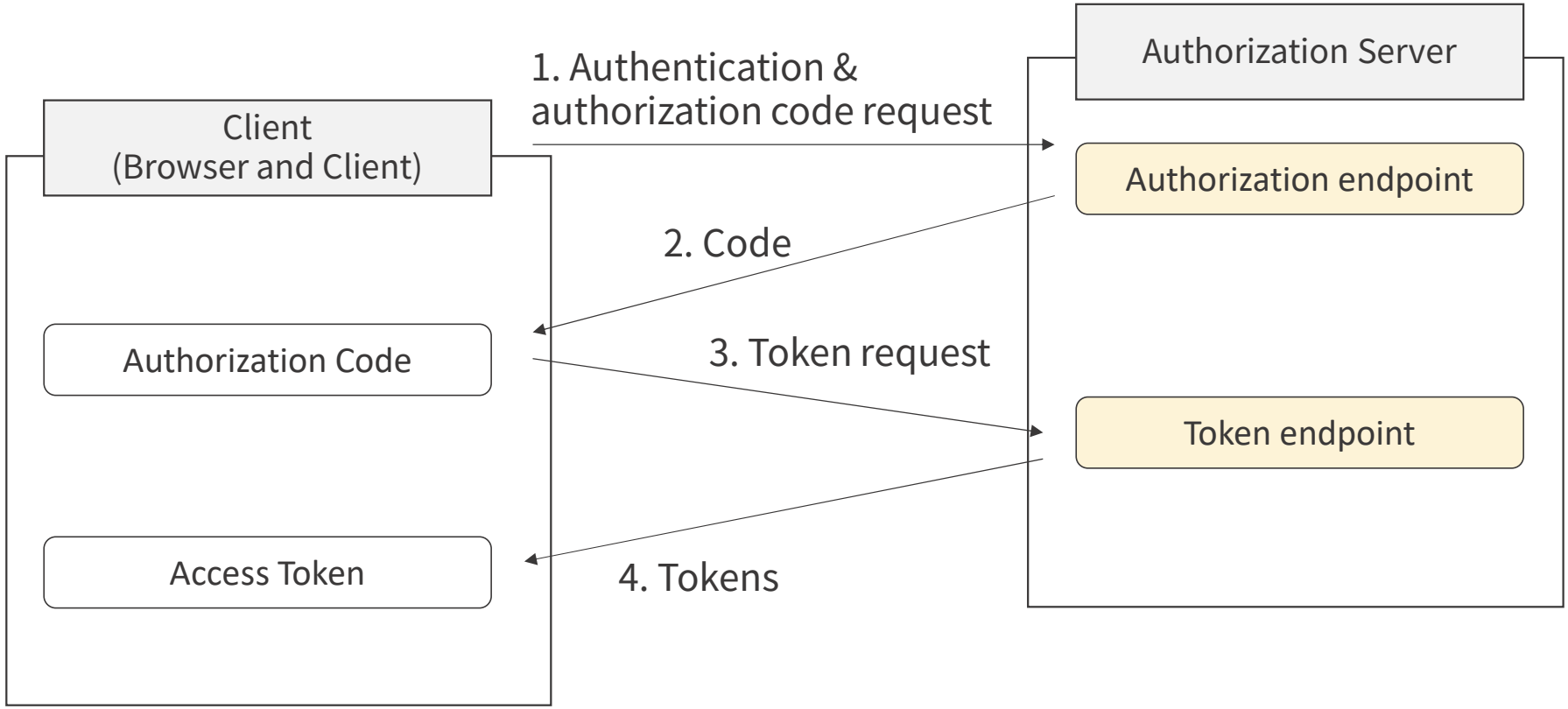
- response_type = code, scope includes ‘openid’



KAFE

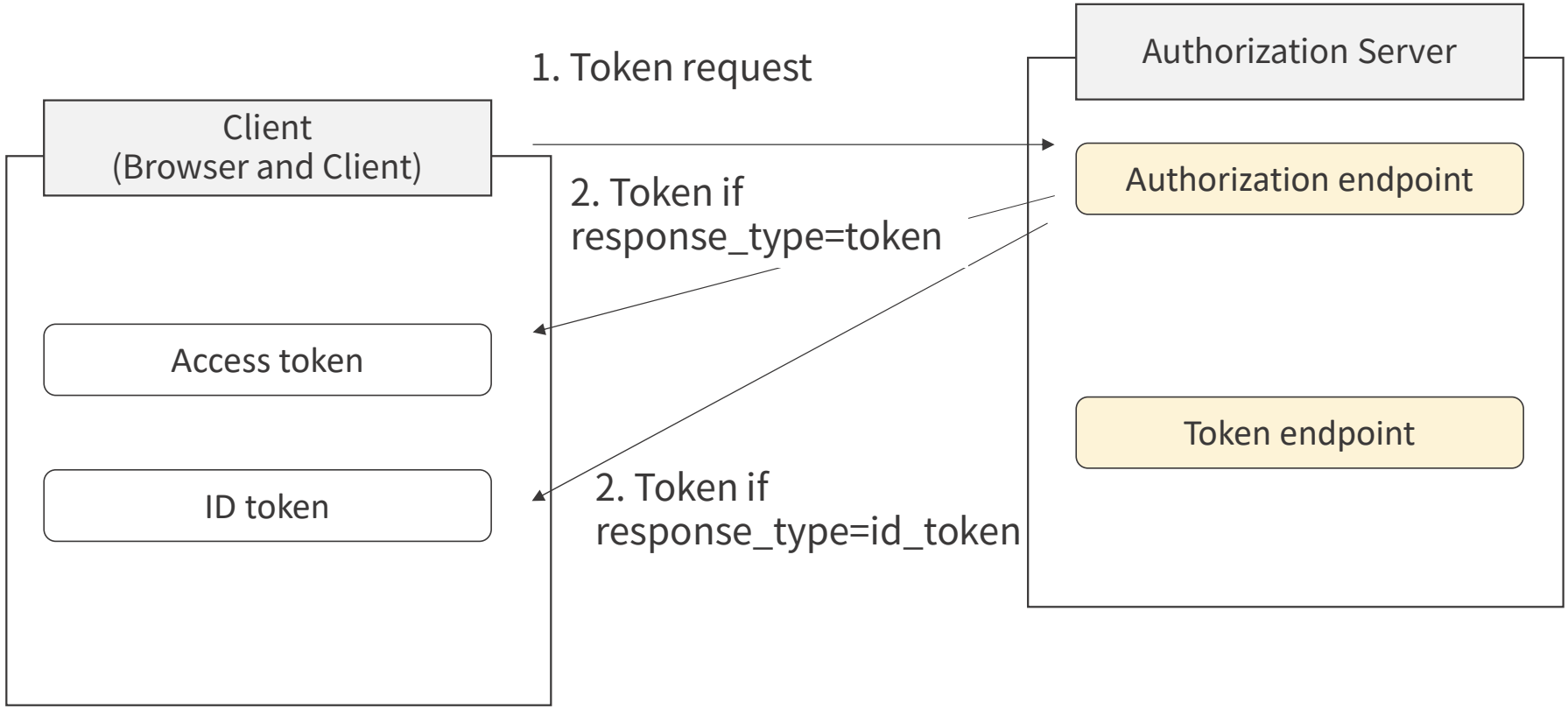
OIDC flows based on response_type

- response_type = code, scope does not include ‘openid’



OIDC flows based on response_type

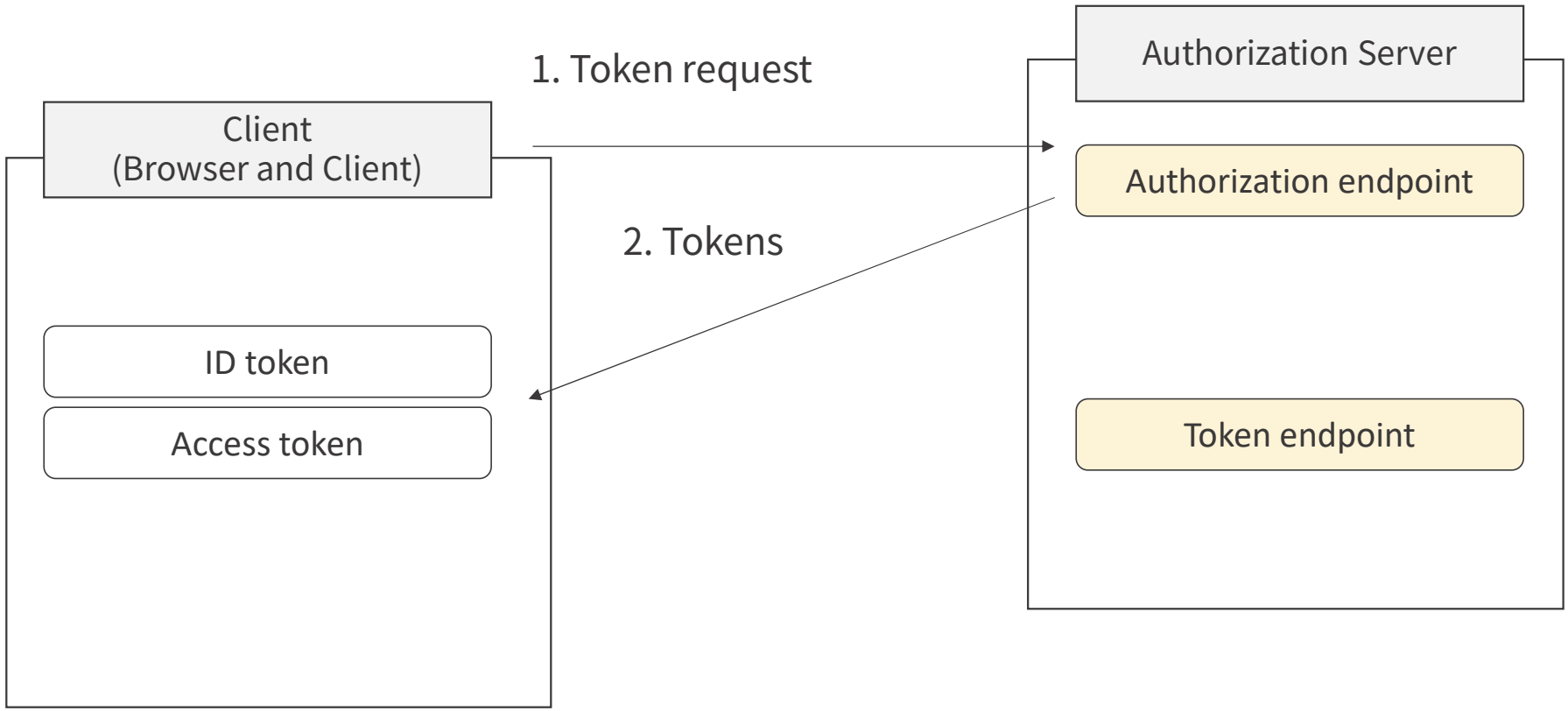
- response_type = token (implicit flow) or response_type=id_token



KAFE

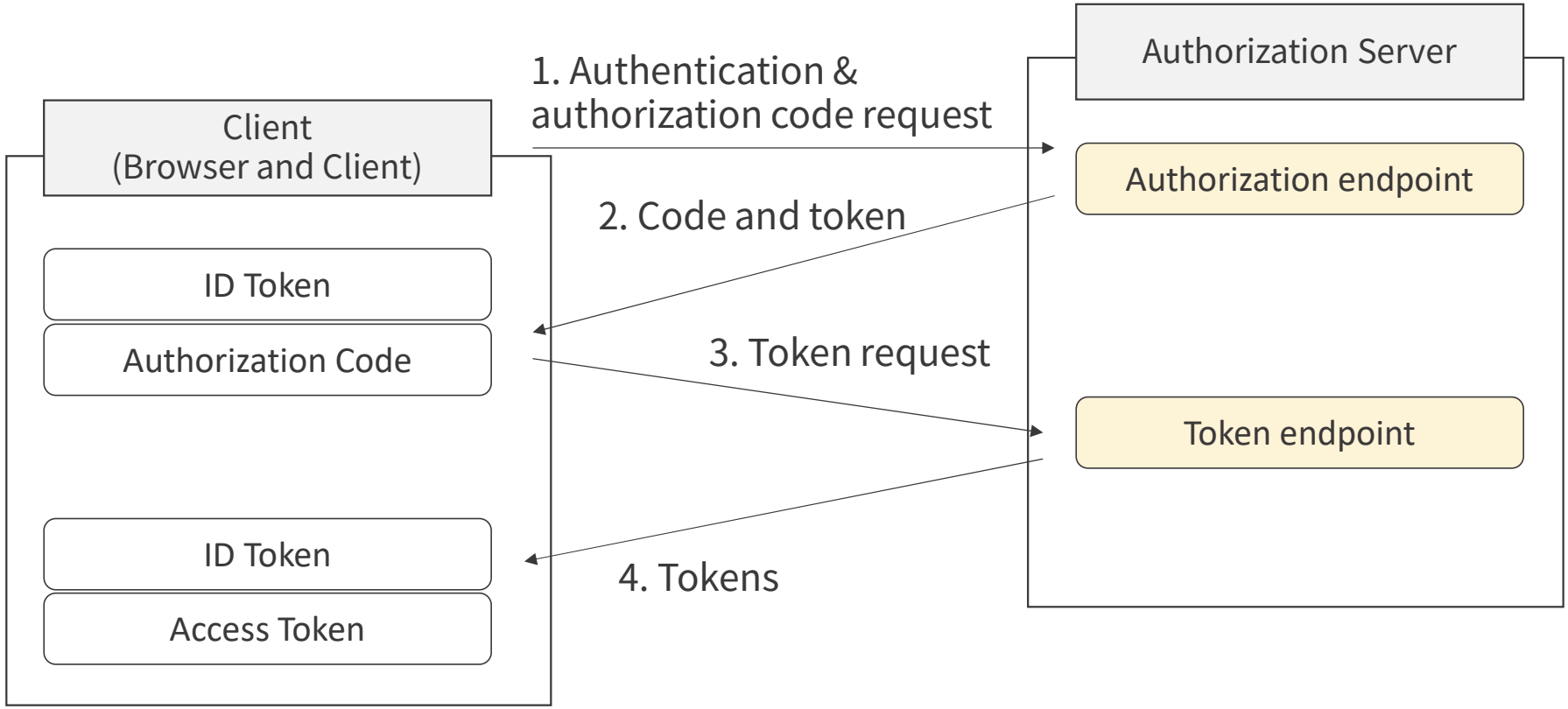
OIDC flows based on response_type

- response_type = id_token token (implicit flow)



OIDC flows based on response_type

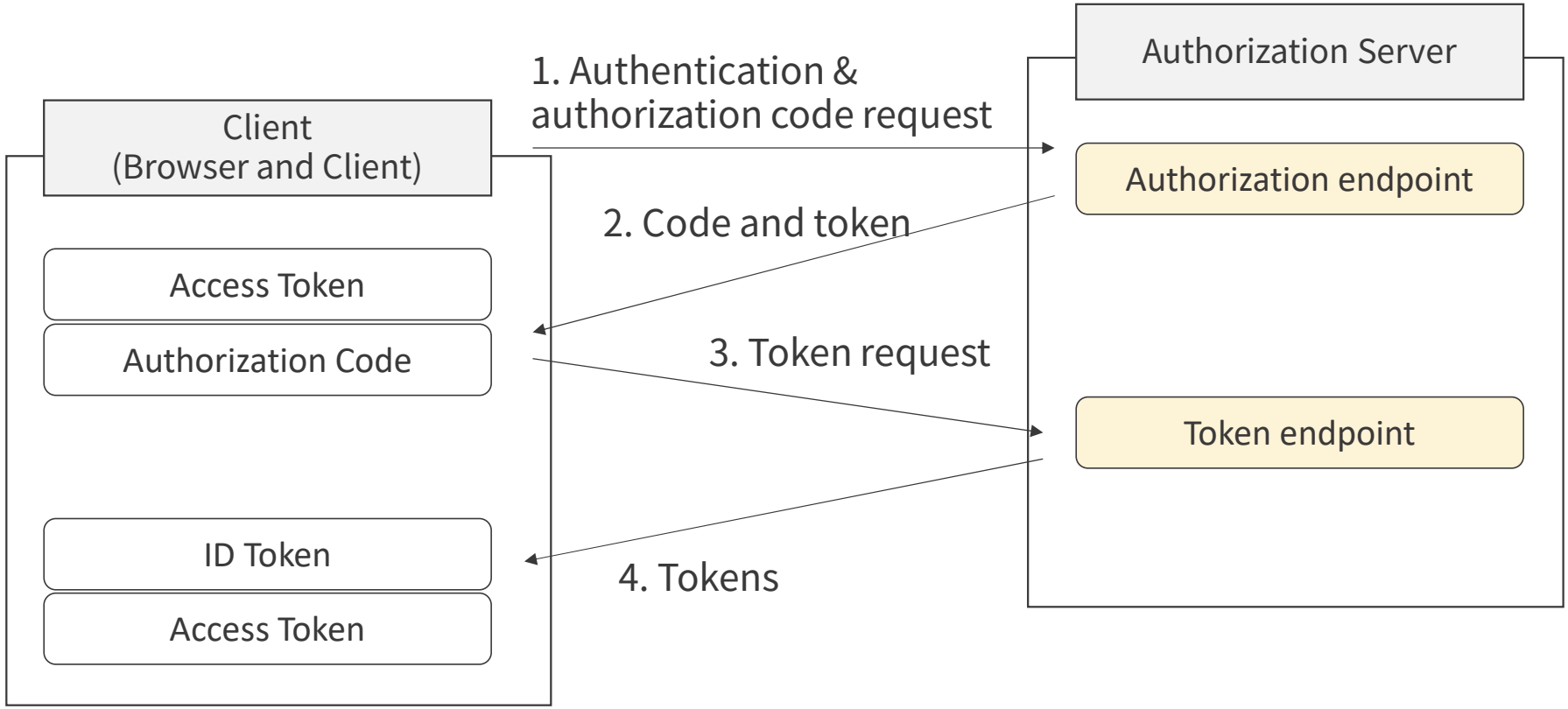
- response_type = code id_token (hybrid flow)



KAFE

OIDC flows based on response_type

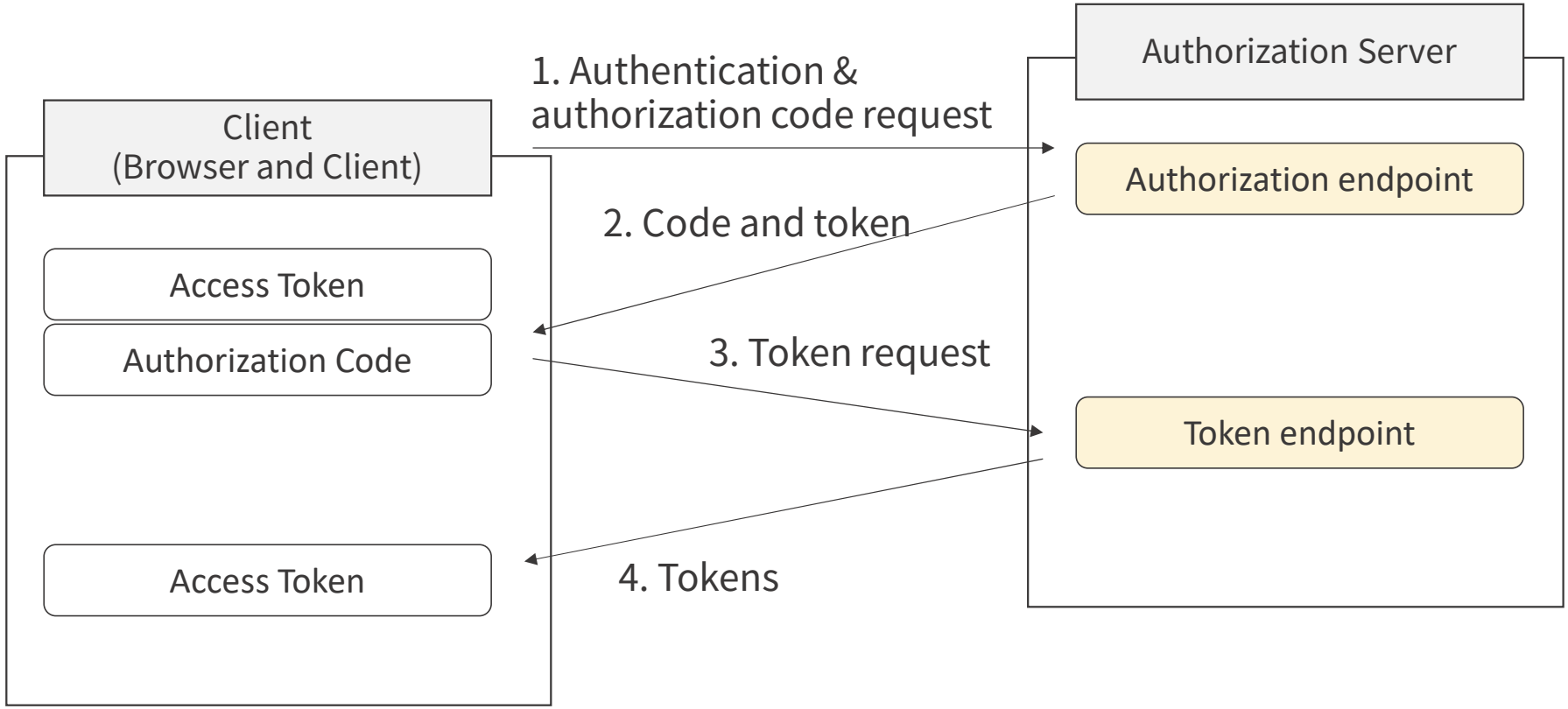
- response_type = code token (hybrid flow), and openid scope is included



KAFE

OIDC flows based on response_type

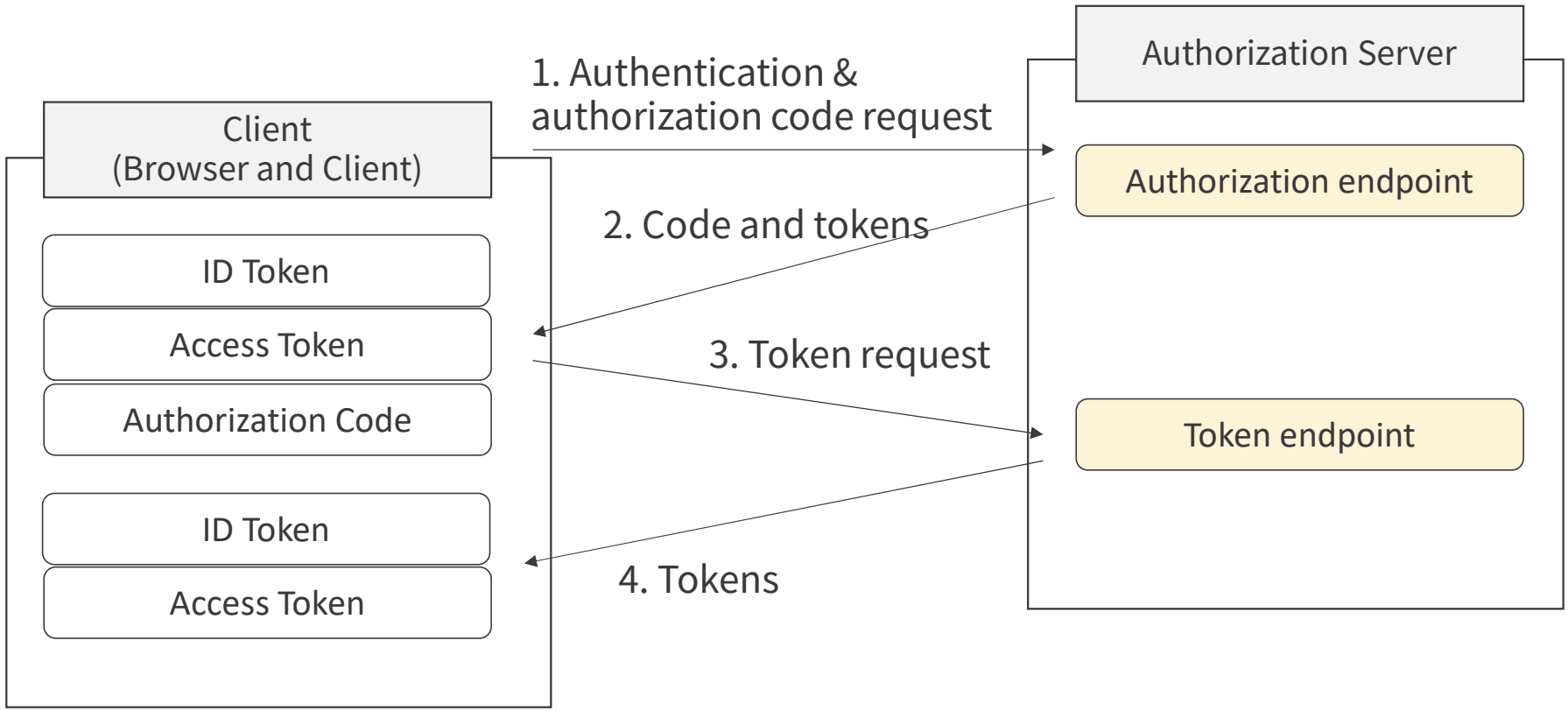
- response_type = code token (hybrid flow), and openid scope is not included



KAFE

OIDC flows based on response_type

- response_type = code id_token token



KAFE

OIDC Token

- Tokens

Tokens	Description	Example

ID token	JWT contains claims about authenticated user (user ID, name, email, etc.)	Payload only!
		{
		"iss": "https://example.com",
		"sub": "user123",
		"aud": "client456",
		"exp": 1623456789,
		"iat": 1623456489,
		"auth_time": 1623456478,
		"nonce": "abc123",
		"name": "John Doe",
		"email": "john@example.com",
		"email_verified": true
		}

Access token	Short-lived token to access protected resources on behalf of the user	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNjIzNDU2NDg5LCJleHAiOjE2MjM0NTY3ODksInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwifQ.SgVOLWW36a3p6J1Awpkq2k4Nnzc1Iq_ZK45Zmn8EJ5s
Refresh token	Long-lived token used to obtain new access token, without requiring the user to reauthenticate.	9LTvx7HINRjfvOtgixcTbDIVmkQvTXoA

KAFE

Access Token

- Base64 encoded (pervious page)

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNjIzNDU2NDg5LCJleHAiOjE2MjM0NTY3ODksInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwifQ.SgVOLWW36a3p6J1Awpkq2k4Nnzc1Iq_ZK45Zmn8EJ5s

- Decoded (Header and payload)

```
{
 "alg": "HS256",
 "typ": "JWT"
}
.
{
 "sub": "user123",
 "name": "John Doe",
 "iat": 1623456489,
 "exp": 1623456789,
 "scope": "openid profile email"
}
```

- Signature: SgVOLWW36a3p6J1Awpkq2k4Nnzc1Iq_ZK45Zmn8EJ5s

KAFE

Scope and claim

- Claims: pieces of user information → to provide user attributes
- Scopes: broader categories of access and permissions → to provide permissions

Scopes	Claims
What information should be included in the tokens and what permissions client should be granted (Not returned by the Userinfo endpoint)	User's identity and permissions. Returned in the ID token or from the Userinfo endpoint
openid, profile, email, address, phone	sub, name, given_name, family_name, email, email_verified, picture
Request format: scope=openid profile email	Request format: claims={"id_token":{"name":null,"email":null}}

```
GET /authorize?
 response_type=code
 &scope=openid%20profile%20email
 &client_id=123456
 &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
 &state=abc123
 &claims={"id_token":{"name":null,"email":null}}
```

KAFE

offline_access scope

- scope used to request a refresh token

```
GET /authorize?
  response_type=code
  &scope=openid%20profile%20email%20offline_access
  &client_id=123456
  &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
  &state=abc123
```

Response (skipped several steps)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNjIzNDU2NDg5LCJleHAiOjE2MjM0NTY3ODksInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgb2ZmbGluZV9hY2Nlc3MifQ.K5t1_CmRo7jcnXYFDWcNDFQXWcVwp7IkKfTW5T1qQjA",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "9LTvx7HINRjfvOtgixcTbDIVmkQvTXoA",
  "scope": "openid profile email offline_access"
}
```

KAFE

Claims defined for the ID token

- Without requesting the 'openid' scope → no return of the ID token.
- Other claims can be provided through UserInfo endpoint or added to the ID token.

Scope	Claim	Description	Claim	Description
openid (ID token)	iss	Issuer Identifier	nonce	
	sub	Subject Identifier	auth_time	Authentication time (optional)
	aud	Audience	acr	Authentication context class reference (optional)
	exp	Expiration time	amr	Authentication methods reference (optional)
	iat	Issued at		

aud: ID token을 수신할 권한이 있는 클라이언트 또는 API 리소스

KAFE

Brief overview of OIDC code flow - messages

1. Code request

```
GET /authorize?
  response_type=code
  &scope=openid%20profile%20email
  &client_id=123456
  &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
  &state=abc123
  &nonce=xyz789
```

2. Code response

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?
  code=AUTH_CODE_123
  &state=abc123
```

3. Token request (HTTP request body)

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=AUTH_CODE_123
&redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
&client_id=123456
&client_secret=abcdef
```

KAFE

Brief overview of OIDC code flow - messages

4. Token response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNjIzNDU2NDg5LCJleHAiOjE2MjM0NTY3ODksInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwifQ.K5t1_CmRo7jcnXYFDWcNDFQXWcVwp7IkKfTW5T1qQjA",
  "token_type": "Bearer",
  "expires_in": 3600,
  "id_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiZW1haWwiOiJqb2huQGV4YW1wbGUuY29tIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsImlhdCI6MTYyMzQ1NjQ4OSwiZXhwIjoxNjIzNDU3MDg5LCJpc3MiOiJodHRwczovL3NlcnZlci5leGFtcGxlLmNvbSIsImF1ZCI6IjEyMzQ1NiIsIm5vbmNlIjoieHl6Nzg5In0.8nXwJq3gQzC5D-ZfJvQ4x0UYv0YfQbJ8P4fE2Ym0FzM"
}
```

※ Bearer : ‘Carrying’ or ‘Possessing’ the permission to access resource
→ does not require re-authentication; possession of the bearer token is sufficient for authentication and authorization.

KAFE

userinfo endpoint

6. UserInfo request (in authorization header) to UserInfo endpoint: optional

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyMTIzIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNjIzNDU2NDg5LCJl
eHAiOjE2MjM0NTY3ODksInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgZWR1cGVyc29uX2VudGl0bGVtZW50IGlzbWVtYmV
yb2YifQ.K5t1_CmRo7jcnXYFDWcNDFQXWcVwp7IkKfTW5T1qQjA
```

KAFE

Other endpoints

- Token revocation endpoint
  - ✓ Can revoke access token and refresh token

https://{domain name}/revoke

- Introspection endpoint:
  - ✓ Can check the validity of access token (common) and refresh token (less common)

https://{domain name}/introspect

acr for MFA profile

- Corresponding to SAML Authentication Context Class Reference (acr)

Authentication request with acr_values

```
GET /authorize?
   response_type=code
   &client_id=s6BhdRkqt3
   &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcb
   &scope=openid%20profile%20email
   &state=af0ifjsldkj
   &nonce=n-0S6_WzA2Mj
   &acr_values=https%3A%2F%2Frefeds.org%2Fprofile%2Fmfa
```

Response: payload in ID token

```
{
 "iss": "https://server.example.com",
 "sub": "24400320",
 "aud": "s6BhdRkqt3",
 "exp": 1311281970,
 "iat": 1311280970,
 "auth_time": 1311280969,
 "nonce": "n-0S6_WzA2Mj",
 "acr": "https://refeds.org/profile/mfa",
 "amr": [
  "pwd",
  "otp"
 ]
}
```

AMR: Authentication Methods Reference

Demo

- Check the OIDC message generation
  - ✓ OIDC provider: INDIGO IAM
  - ✓ OIDC client: https://oidcdebugger.com/
  - ✓ Standard: OIDC

- OIDC discovery
  - ✓ https://indigo.kafe.or.kr/.well-known/openid-configuration

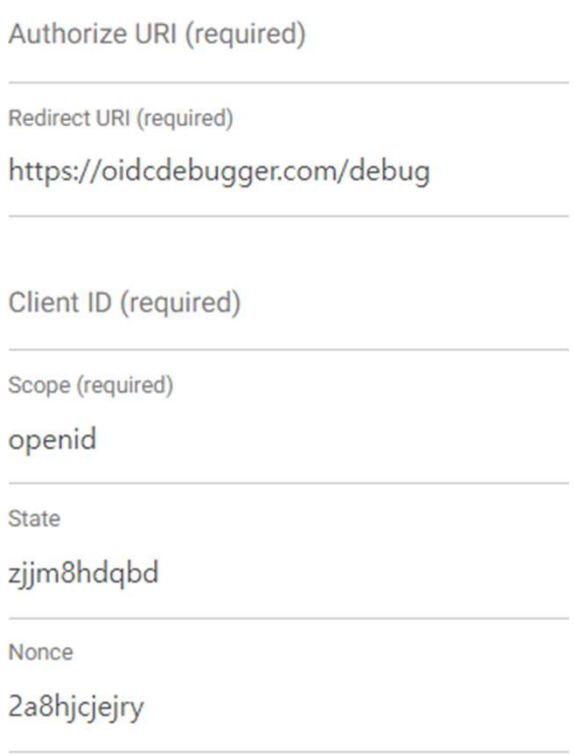
Property	Flow
grant_type	authorization_code, implicit, refresh_token, client_credentials, password, urn:ietf:params:oauth:grant-type:token-exchange, rn:ietf:params:oauth:grant-type:device_code
request_type	code, token

- Credential
  - ✓ client_id: 6d5f20f3-84d6-4baf-b937-fe232601b606
  - ✓ secret:

- Scope: openid, email, profile

Demo (contd.)

- https://oidcdebugger.com/



https://indigo.kafe.or.kr/authorize

6d5f20f3-84d6-4baf-b937-fe232601b606

- Response mode (response_mode): how the response should be sent from Op to RP
  - ✓ query: query parameters
  - ✓ form_post: HTTP POST method
  - ✓ fragment: hash fragment (e.g., https://client.example.com/cb#access_token=abc123&state=xyz)

참고 자료

Korean Access Federation

https://www.kafe.or.kr/

support@kafe.or.kr