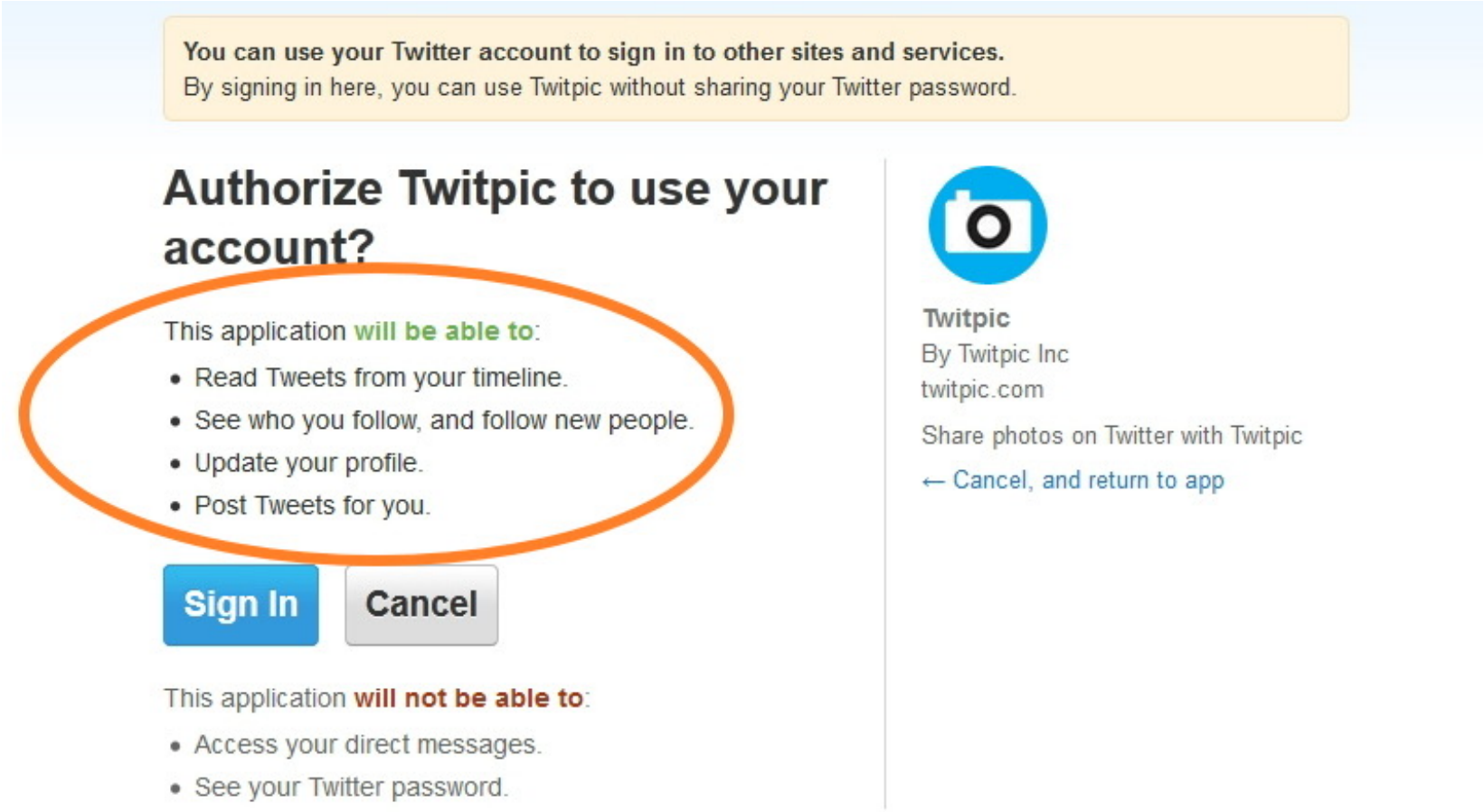
Network Security

<CH 7>

Youn Kyu Lee

Hongik University

Authorization



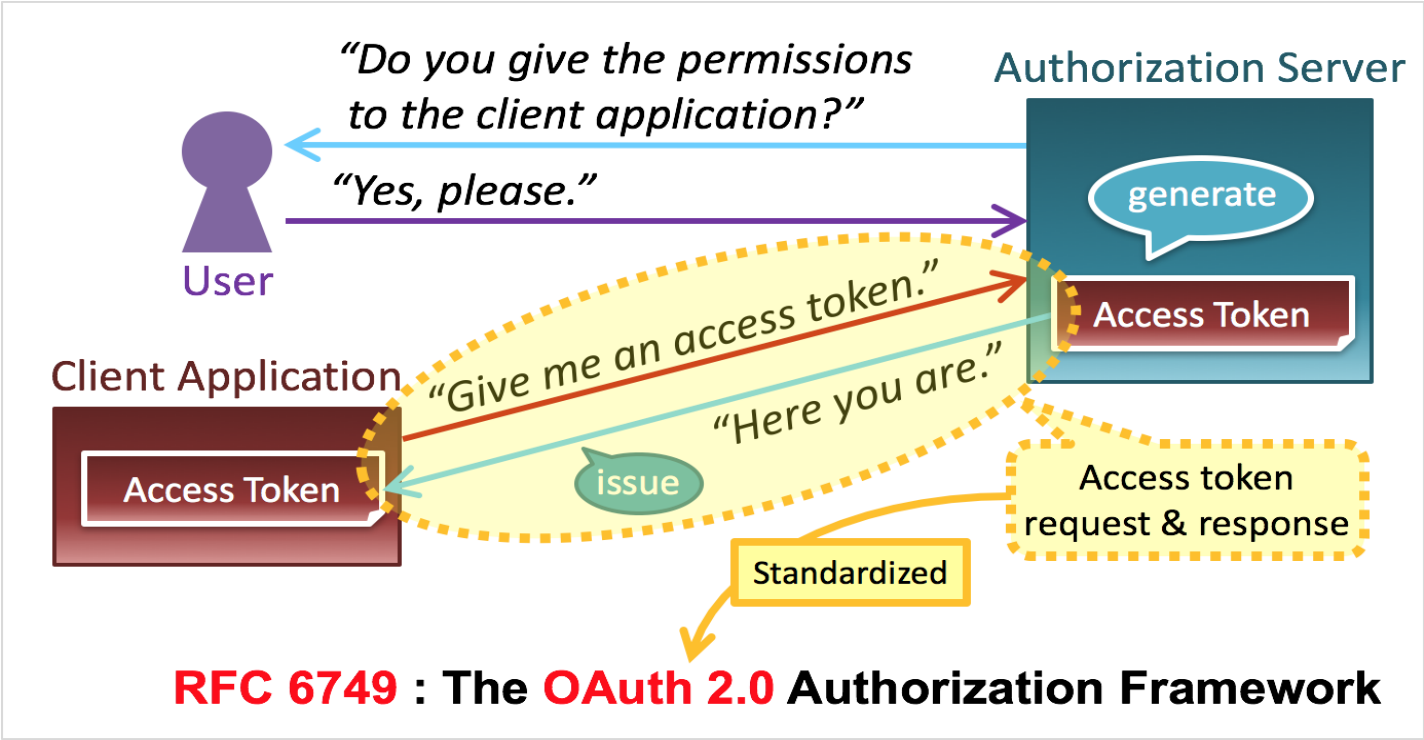
Source https://lifehacker.com/understanding-oauth-what-happens-when-you-log-into-a-s-5918086

OAuth 2.0

- OAuth 2.0 is for authorization

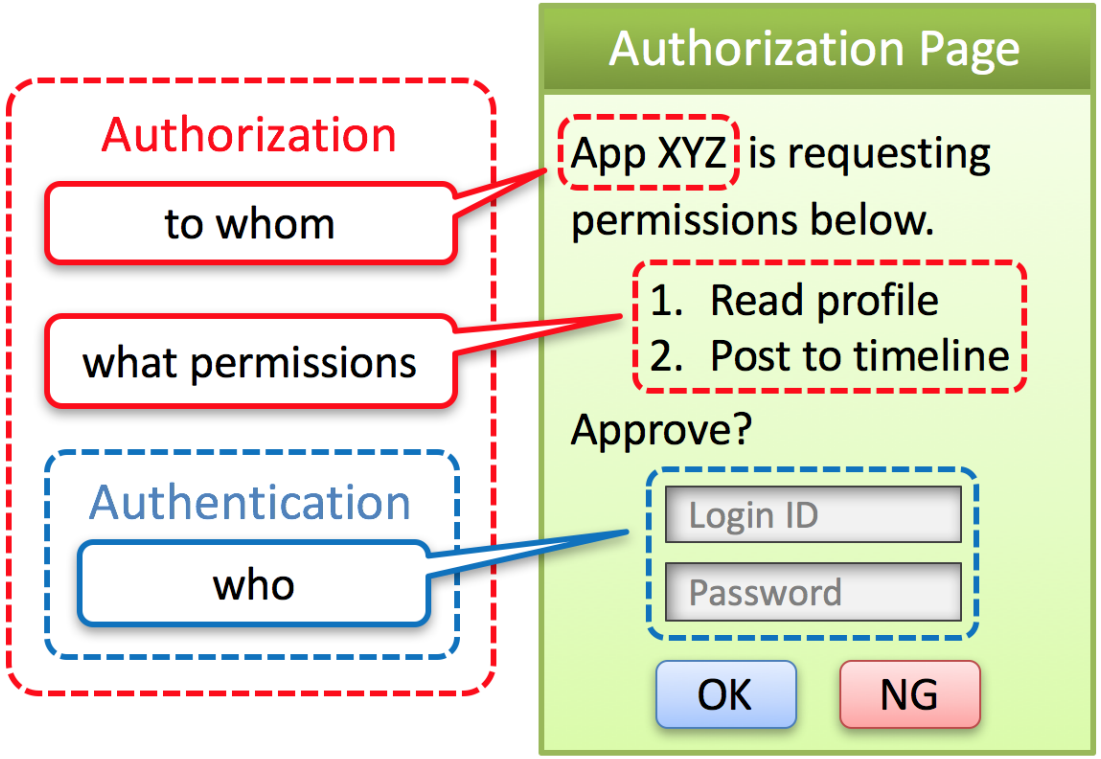
  (granting access to data and features *from one application to another*)

  - specification as to *how to issue "access tokens"* (RFC 6749)



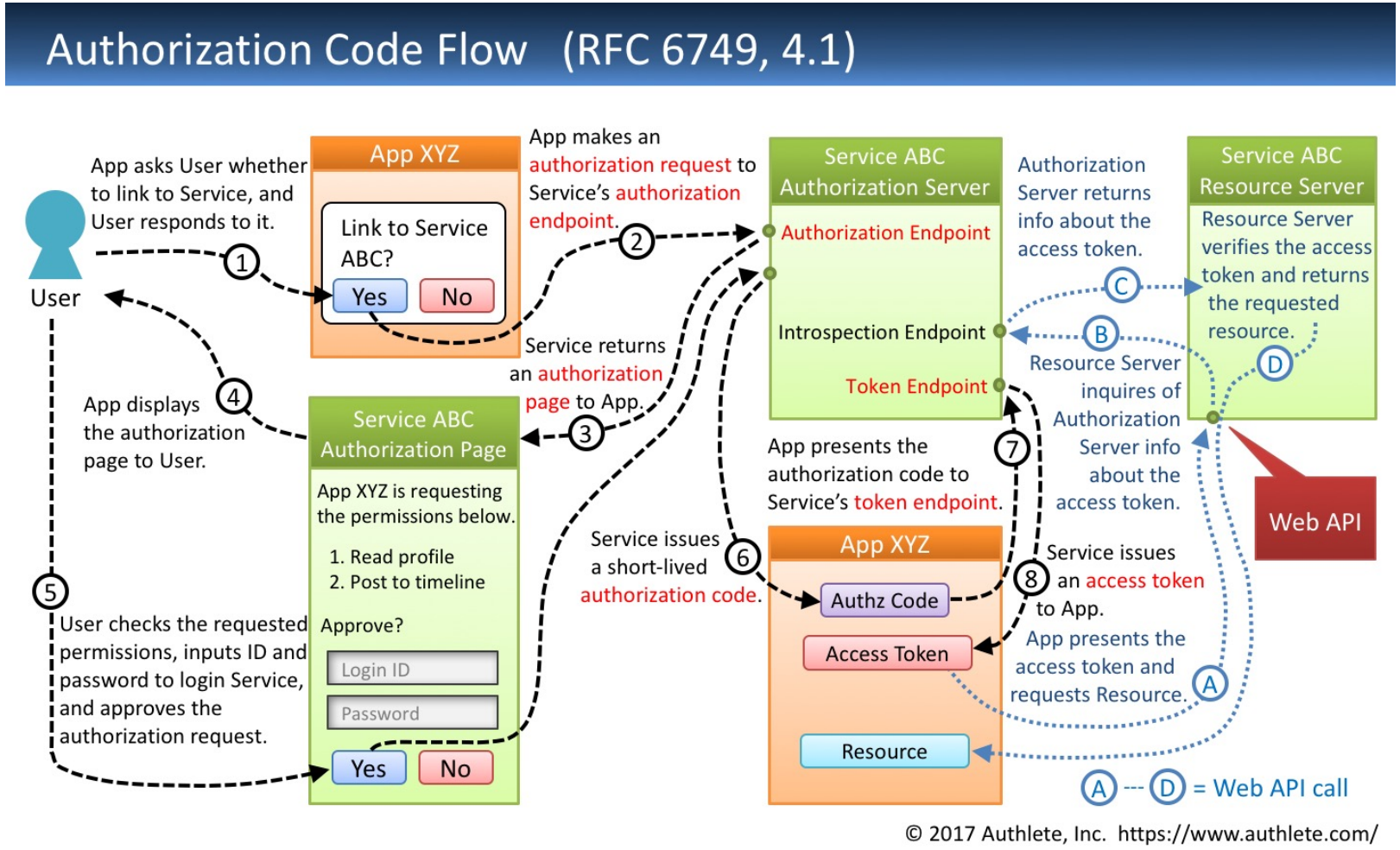
Source: ©2017 Authlete, Inc. https://www.authlete.com

OAuth 2.0



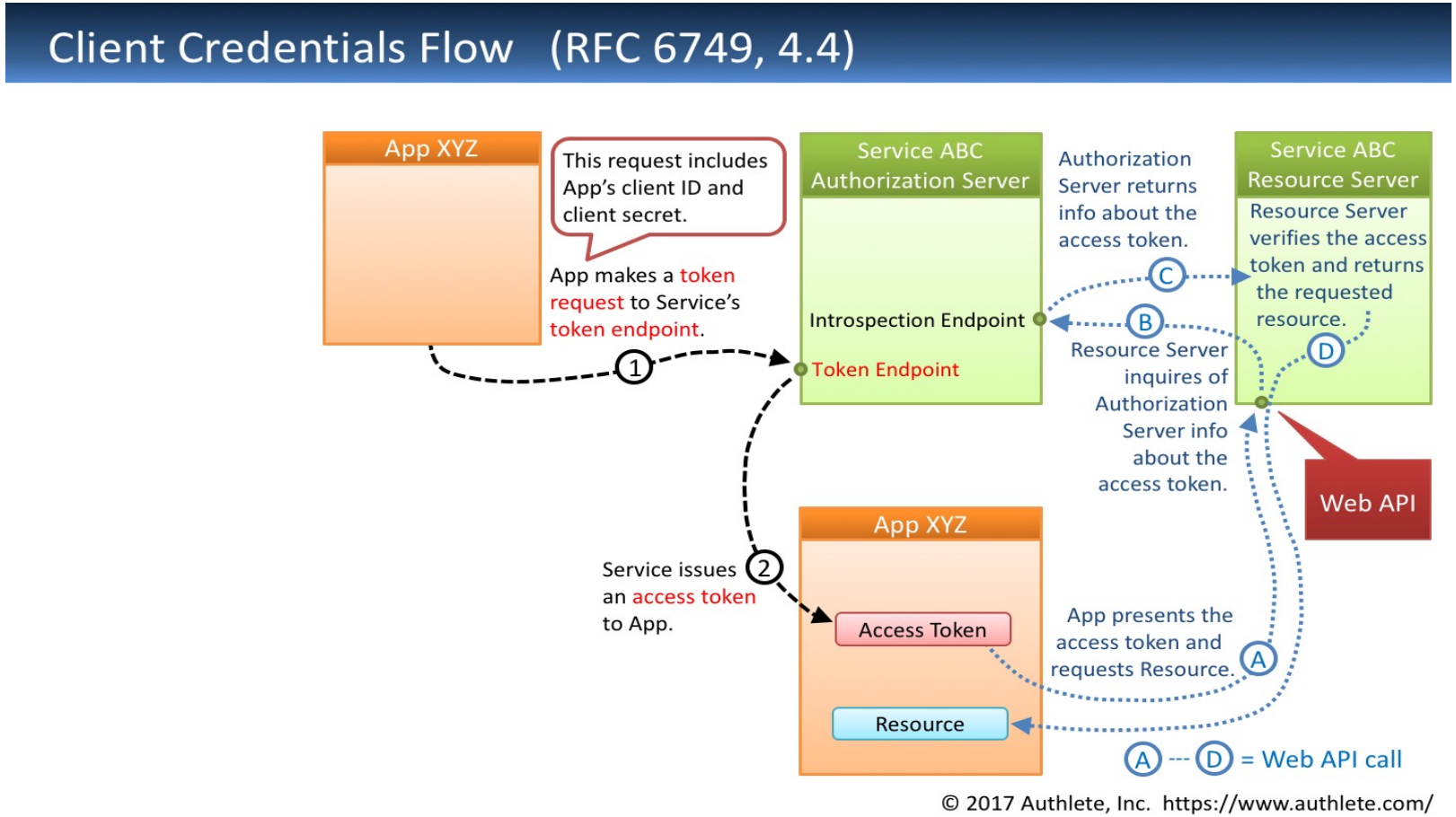
Authentication	Who one is.
Authorization	Who grants what permissions to whom.

OAuth 2.0



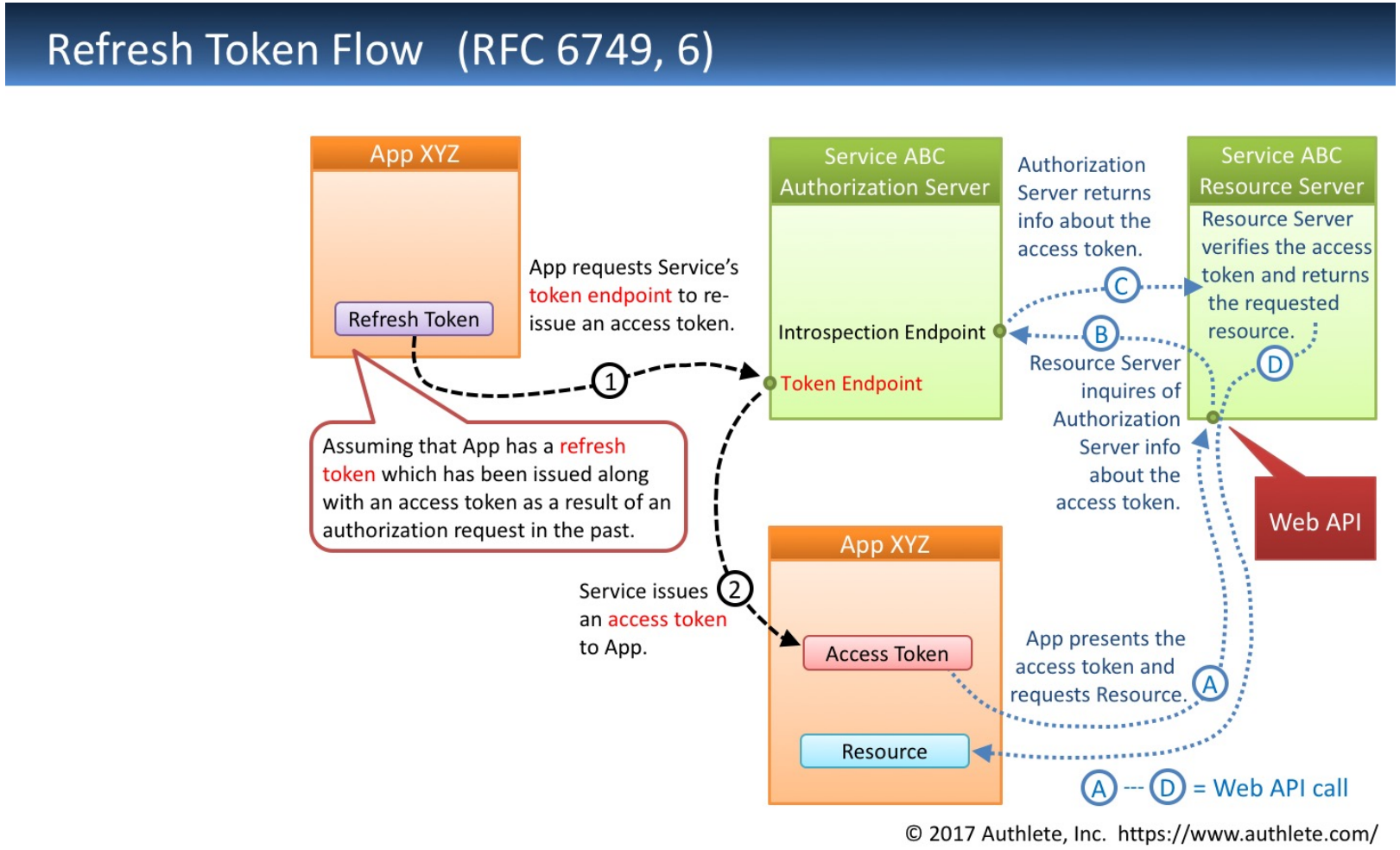
client application (a) makes an authorization request to an authorization endpoint, (b) receives a short-lived authorization code, (c) makes a token request to a token endpoint with the authorization code, and (d) gets an access token

OAuth 2.0



- client application (a) makes a token request to a token endpoint and (b) gets an access token. (user authentication not performed)
- allowed only for confidential clients (client_id and client_secret parameters are required when requesting to Token Endpoint)

OAuth 2.0



client application (a) presents a refresh token to a token endpoint and (b) gets a new access token.

OpenID Connect(OIDC)

- For “secure delegated access”
- A thin layer on top on OAuth 2.0 that adds login and profile information about the logged one
  - an OAuth 2.0 extension
  - specification as to *how to issue “ID tokens”*
  - same flow as OAuth but with additional ID token to identify your ID, name, login time, ID token’s expiration
  - you can use a trusted external provider(IdP(Identity Provider) or Auth. Server) to prove to a given application that ‘you are who you say you are’, without ever having to grant that application access to your credentials

OpenID Connect(OIDC)

- Web API called “authorization endpoint” requires `response_type` as a mandatory request parameter

- The value of `response_type` is either `code` or `token`. OIDC added a new value `id_token`

- A request for an ID token has to include `openid` in the `scope` request parameter (if `openid` not included, ID token not be issued)

OIDC ID Token

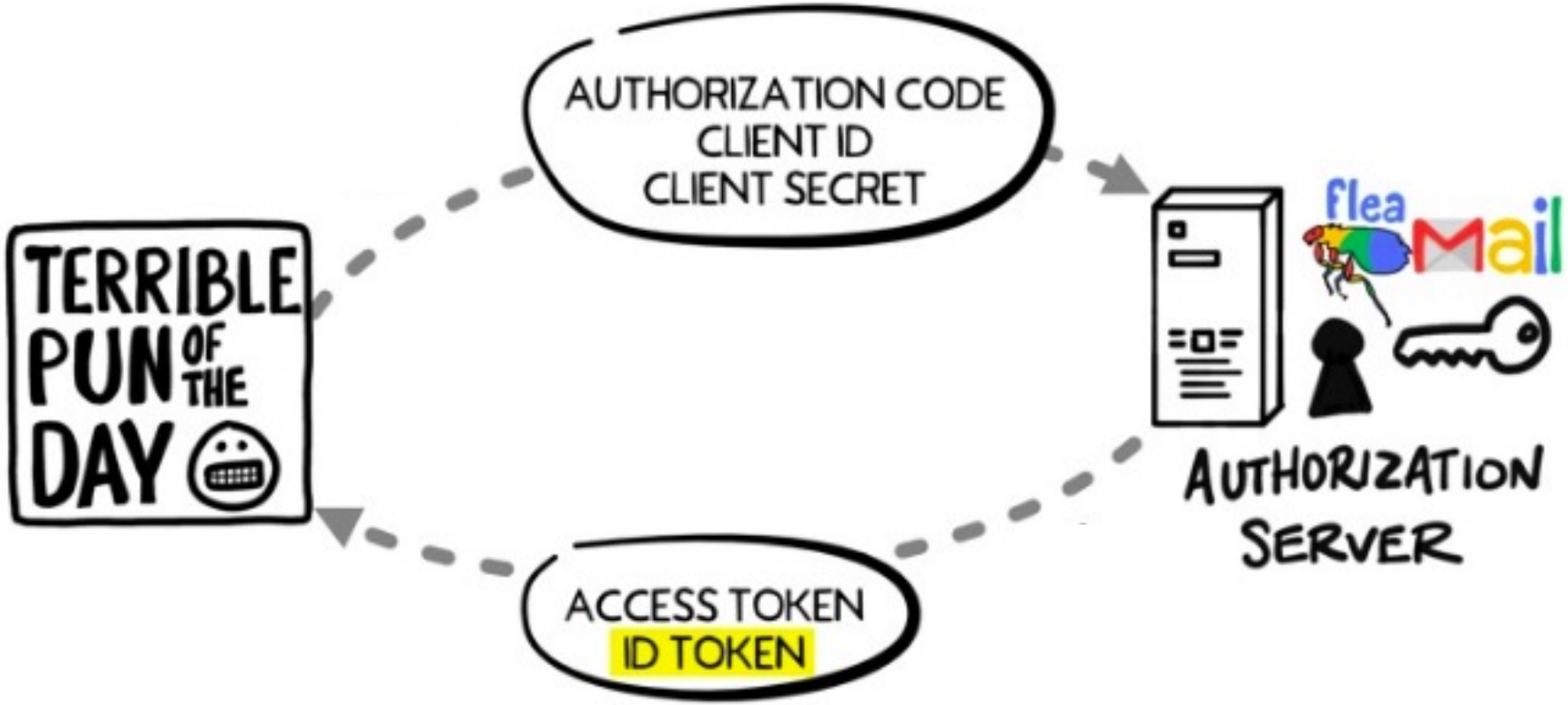
- the primary extension that OIDC makes to OAuth 2.0 to enable End-User to be Authenticated

  (by specifying the usage and data structure of *id_token*)

- a security token that contains Claims about the Authentication of an End-User by an Authorization Server

- is represented as a JSON Web Token (JWT)

OIDC ID Token



Source: *An Illustrated Guide to OAuth and OpenID Connect , by David Neal*

OIDC ID Token

- Looks like:

eyJraWQiOiIxZTlnZGs3IiwiYWxnIjoiUlMyNTYifQ.ewogImlz
cyI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4
Mjg5NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAi
bi0wUzZfV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEz
MTEyODA5NzAsCiAibmFtZSI6ICJKYW5lIERvZSIsCiAiZ2l2ZW5fbmFtZSI6
ICJKYW5lIiwKICJmYW1pbHlfbmFtZSI6ICJEb2UiLAogImdlbmRlciI6ICJm
ZW1hbGUiLAogImJpcnRoZGF0ZSI6ICIwMDAwLTEwLTMxIiwKICJlbWFpbCI6
ICJqYW5lZG9lQGV4YW1wbGUuY29tIiwKICJwaWN0dXJlIjogImh0dHA6Ly9l
eGFtcGxlLmNvbS9qYW5lZG9lL21lLmpwZyIKfQ.rHQjEmBqn9Jre0OLykYNn
spA10Qql2rvx4FsD00jwlB0Sym4NzpgvPKsDjn_wMkHxcp6CilPcoKrWHcip
R2iAjzLvDNAReF97zoJqq880ZD1bwY82JDauCXELVR9O6_B0w3K-E7yM2mac
AAgNCUwtik6SjoSUZRcf-O5lygIyLENx882p6MtmwaL1hd6qn5RZOQ0TLrOY
u0532g9Exxcm-ChymrB4xLykpDj3lUivJt63eEGGN6DH5K6o33TcxkIjNrCD
4XB1CKKumZvCedgHHF3IAK4dVEDSUoGlH9z4pP_eWYNXvqQOjGs-rDaQzUHl
6cQQWNiDpWOl_lxXjQEvQ

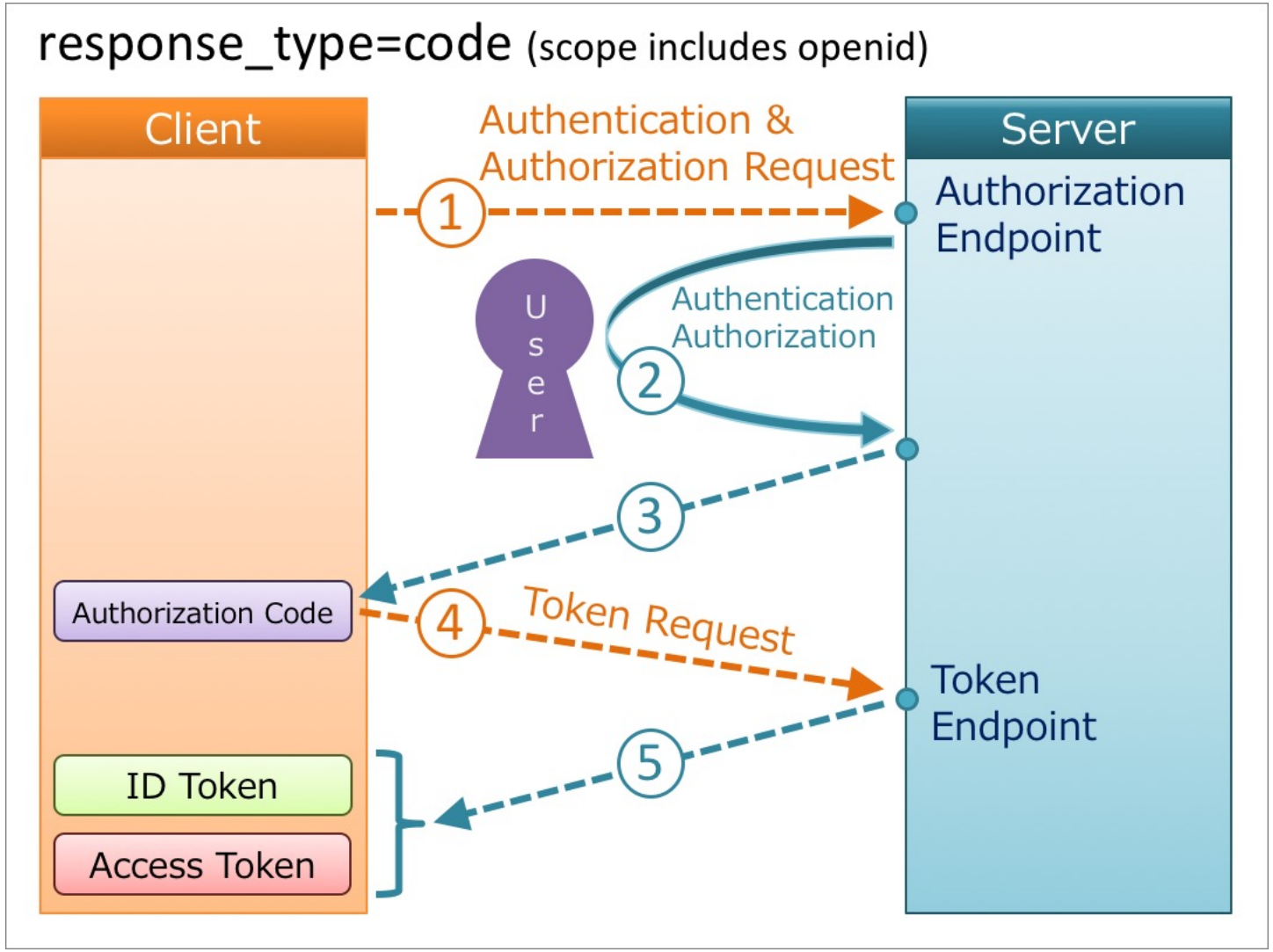
- Header + Payload + Signature

OIDC ID Token

- Decoded Payload field (as a JWT(JSON Web Token))

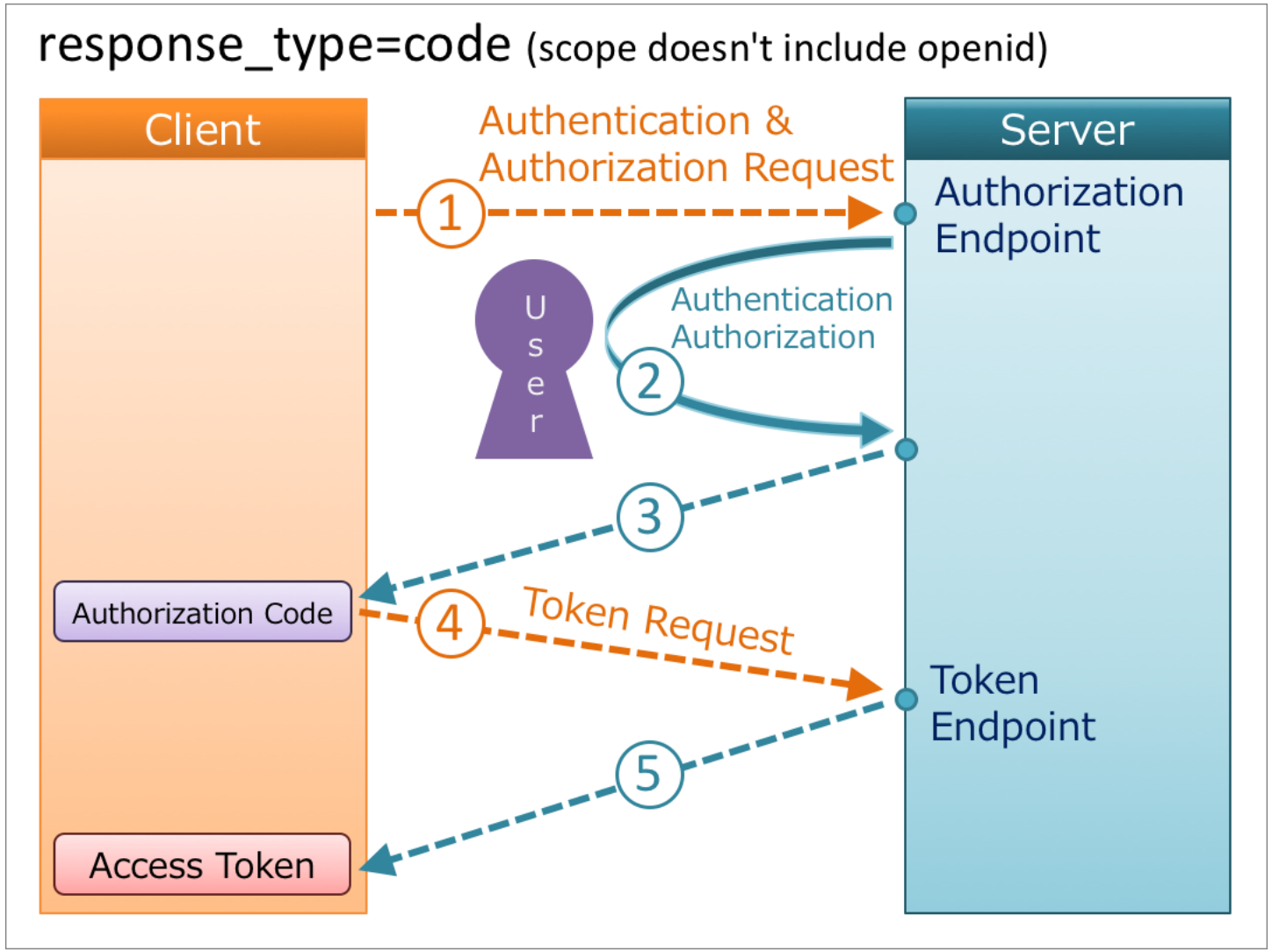

```
{
 "iss": "http://server.example.com",
 "sub": "248289761001",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "gender": "female",
 "birthdate": "0000-10-31",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

response_type=code



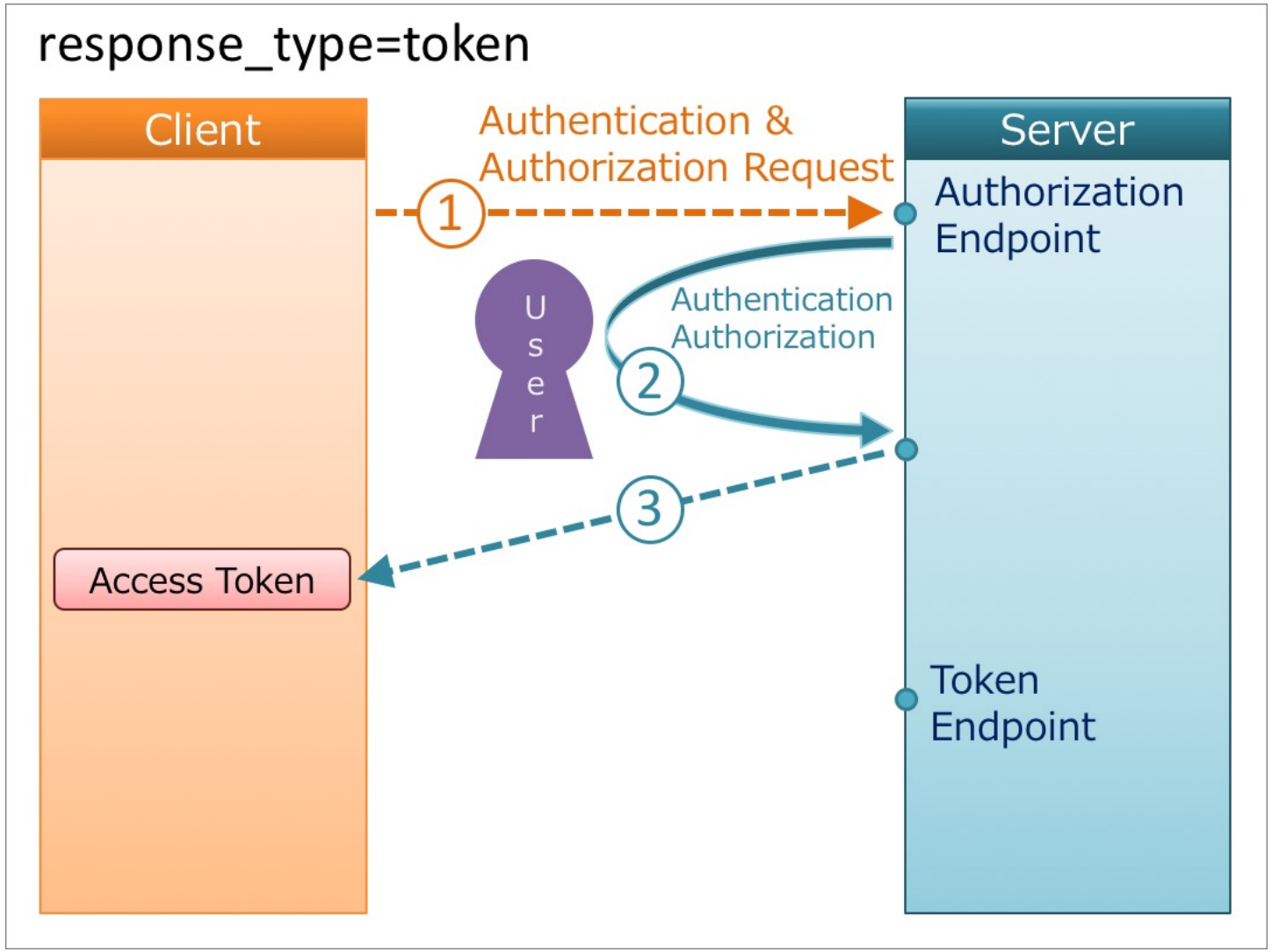
Source: ©2017 Authlete, Inc. https://www.authlete.com

response_type=code



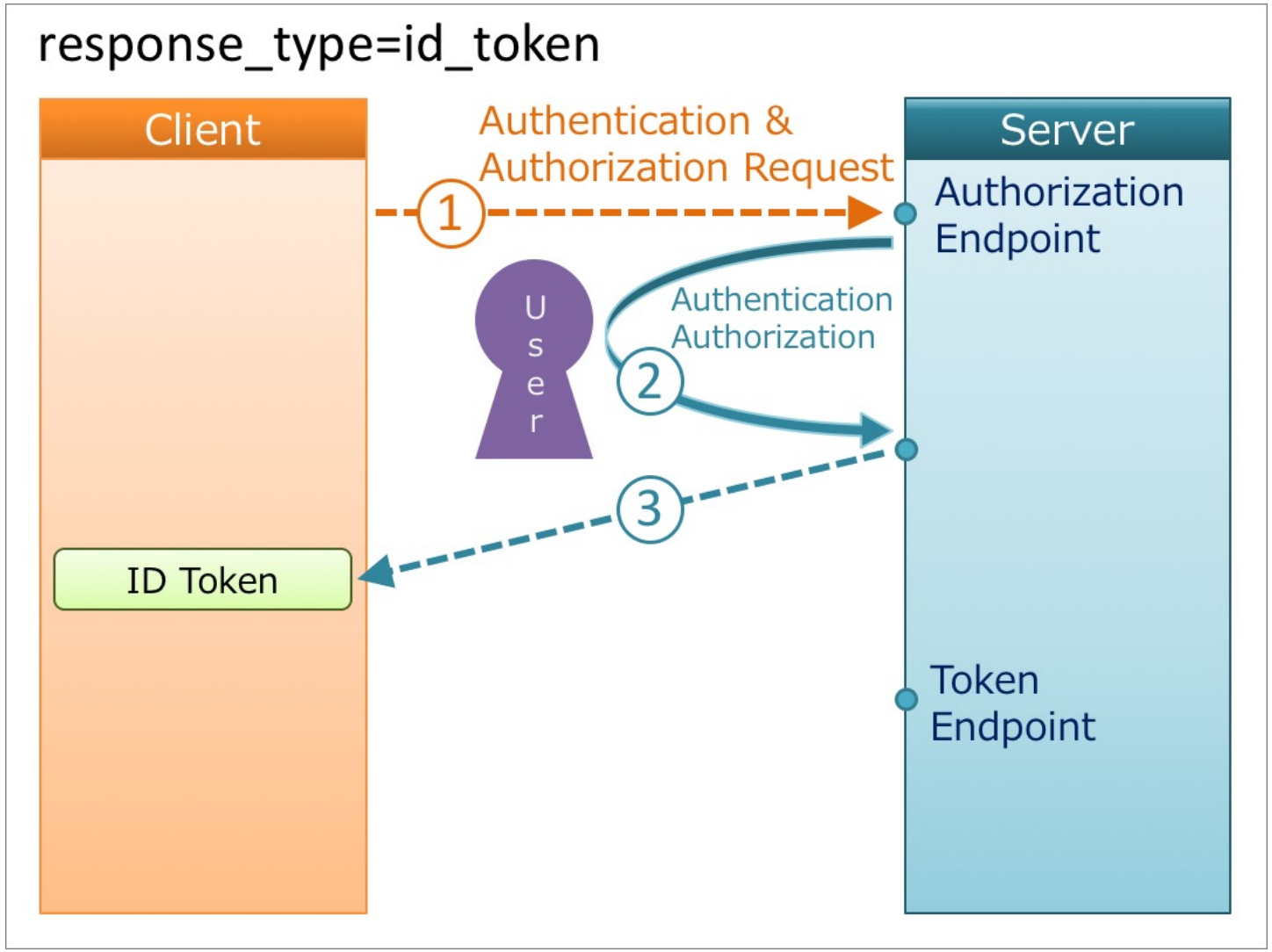
Source: ©2017 Authlete, Inc. https://www.authlete.com

response_type=token



Source: ©2017 Authlete, Inc. https://www.authlete.com

response_type=id_token



Source: ©2017 Authlete, Inc. https://www.authlete.com

Authentication vs Authorization

- Authentication — Are you who you say you are?
  - Restrictions on who (or what) can access system
- **Authorization** — Are you allowed to do that?
  - Restrictions on actions of authenticated users
- Authorization is a form of **access control**
- *OS authorization* traditionally enforced by
  - Access Control Lists (ACLs)
  - Capabilities

OS Access Control

- Authenticate principal(persons, processes, …) by passwords or security protocols or fingerprints …

  : are you who you say you are?

- and Authorize access to files, communication ports and other system resources

  : are you allowed to do that?

Access Control Matrix: matrix of access permissions

- **Subjects** (users) index the rows
- **Objects** (resources) index the columns

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	---	---
Alice	x	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Are You Allowed to Do That?

- Access control matrix has all relevant information
- Could be 1000's of users, 1000's of resources
  then matrix with 1,000,000's of entries
- How to manage such a large matrix?
- Need to check this matrix before access to any resource is allowed
- How to make this efficient?
- Groups and roles for large organizations
  - group: a list of principals
  - role: a set of access permissions that one or more principals may assume for a period of time

Access Control Lists (ACLs)

- ACL: store access control matrix by **column**
- Example: ACL for **insurance data** is in **blue**

	OS	Accounting program	Accounting data	**Insurance data**	Payroll data
Bob	rx	rx	r	---	---
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Access Control Lists (ACLs)

- Basic access control mechanism in Unix-based systems such as Linux, macOS, as well as in derivatives such as Android and iOS (cf. Windows access controls also based on ACLs but more complex)
- ACLs are for data-oriented protection
- ACLs are less suited where
  - user population is large and constantly changing
  - users want to delegate their authority to run a particular program to another user for a period

Access Control Lists (ACLs)

- ACLs are simple to implement, but are not efficient for security checking at runtime, since typical OS knows which user is running a program, rather than what files it has been authorized to access

- With ACLs, it's NOT straightforward to
  - find all the files to which a user has access
  - verify no files have been left world-readable or world-writable
  - revoke access of an employee who has just been fired

Unix security

- `rwx` used for file to be read, written and executed

  eg. `drwxrwxrwx Alice Accounts`

  `-rw-r----- Alice Accounts`

- Kernel runs as supervisor and all other programs run as users (access decision are made on the basis of the userid associated with the program)

  : userid zero → `root` (can do whatever it likes)

- so, it's hard to implement an audit trail as a file root cannot modify (so attacker with root privilege can remove all evidence of his intrusion!)

  : common way to protect logs against root compromise is to send it to another machine or even to a third-party service

Unix security

- No straightforward way of implementing access triples of (user, program, file) with ACLs since ACLs contain names of user, but not of programs
  : Unix provides indirect method: set-user-id(`suid`)
  : owner of a program can mark the file representing the program as `suid` , which enables it to run with the privilege of its owner rather than the privilege of the user who invoked it
  → lazy programmer can make an application `suid root`, so it can do anything! (a security hole)

Capabilities

- Store access control matrix by **row**
- Example: Capability for **Alice** is in **red**

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	---	---
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Capabilities

- Runtime security checking is more efficient

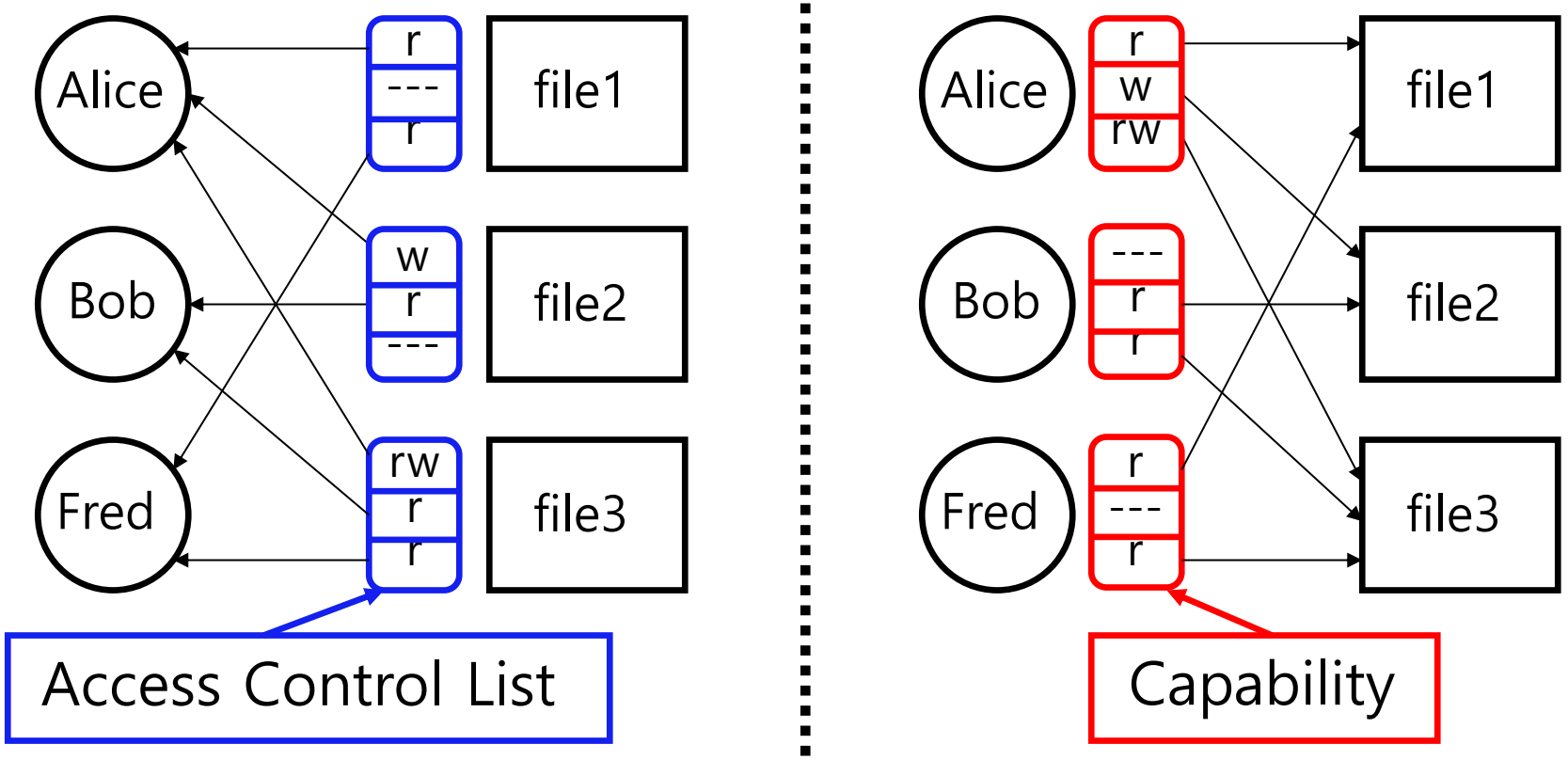
- Delegating a right is more easy (eg, Bob says,' here is my capability and I delegate to David the right to read file4 from 9am to 1pm' authorized by Bob)

- Changing file's status becomes more tricky as it is hard to find out which users have access

Capabilities

- Usage case: ‘a nurse shall have access to all patients who are on his/her ward, or who have been there in the last 90 days’
  - In ACL-based solution, each access control decision requires a reference to administrative system to find out which nurses and which patients were on which ward, when
  - Capabilities allow to give nurses certificates that entitle them to access the files associated with a number of wards
- Public key certificates in public key cryptography are in effect capabilities
- Capabilities started to supplement ACLs in OSes, including recent Windows, FreeBSD, and iOS

ACLs vs Capabilities



Database wrt Access Control

- handle various web pages, fronted by web servers that pass transactions to the databases directly
- contain data that matter to our lives – bank accounts, patient records, employment records, .. (sometimes be exposed to random online users)
- Oracle, MySQL … have their own access control mechanisms with privileges available for both users and objects (so mechanisms are a mixture of ACLs and Capabilities)
- modern databases are intrinsically complex, as they support various domains of business processes
- modern databases may deal with possible statistical inference rather than simple yes/no access rules

Browsers wrt Access Control

- the place on which you run code written by people you don't want to trust and who will occasionally be malicious

  → treat browsers as access control devices

- need to deal with the situation: a user wants to run some code from a website, but he is concerned that the code might do something nasty, such as stealing his address book and mailing it off to a marketing company

Database wrt Access Control

- ‘same origin policy’: Javascript or other active content on a web page is ONLY allowed to comm. with the domain it originally came from
- ‘sandbox’(w/ Java by Sun Microsystems -> Oracle): a restricted environment in which the code
  - enforced by having the code executed by JVM w/ only limited access rights : idea adopted by Javascript and other active content
    1) has no access to the local hard drive(or only temporary access to a restricted directory)
    2) is only allowed to comm. with the domain it came from(same origin policy)
       ← to prevent the code from altering the host

Q & A

aiclasshongik@gmail.com