REST API AUTHORIZATION

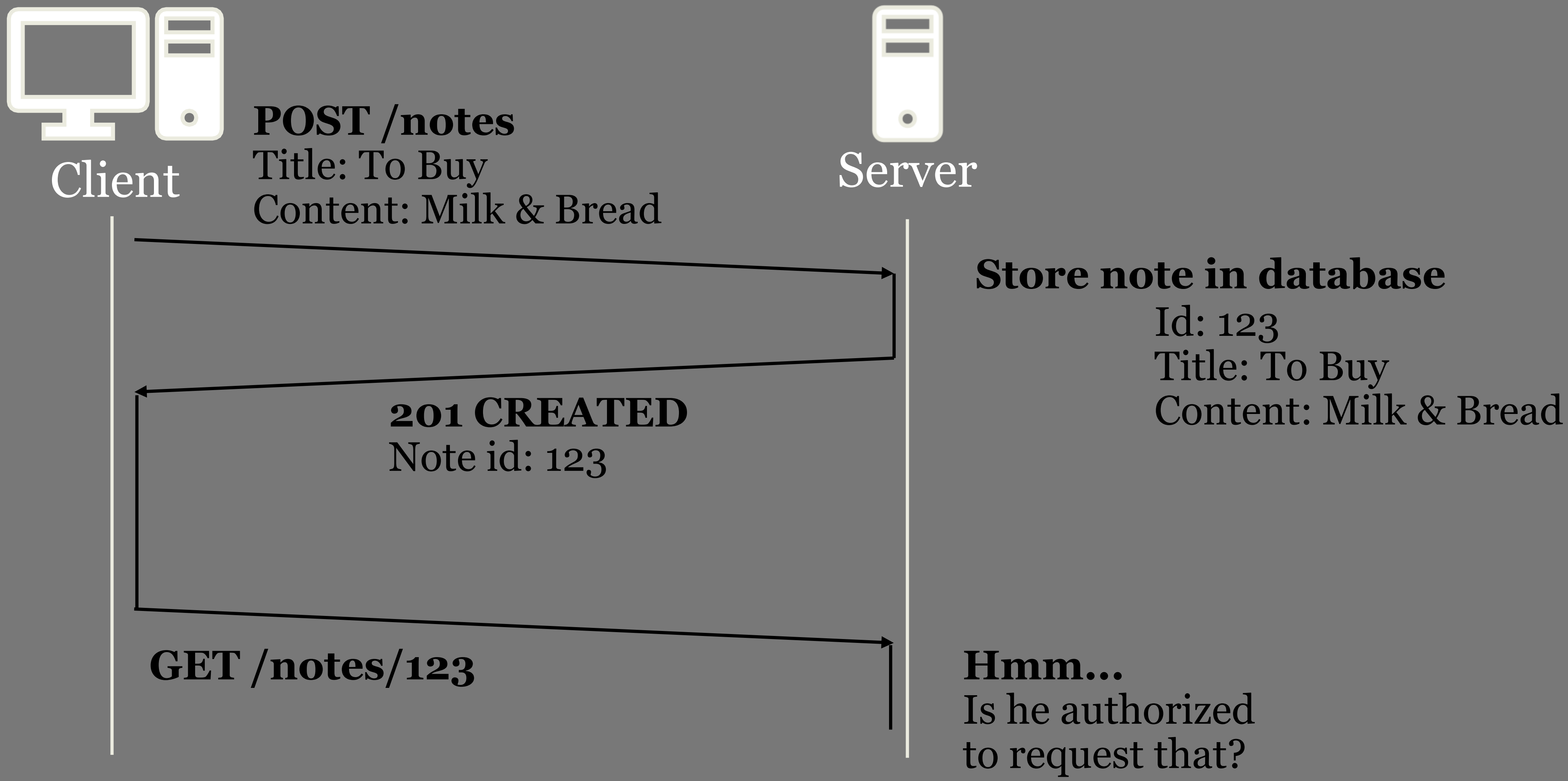
Peter Larsson-Green

Jönköping University

Autumn 2018

JÖNKÖPING UNIVERSITY
School of Engineering

AUTHORIZATION



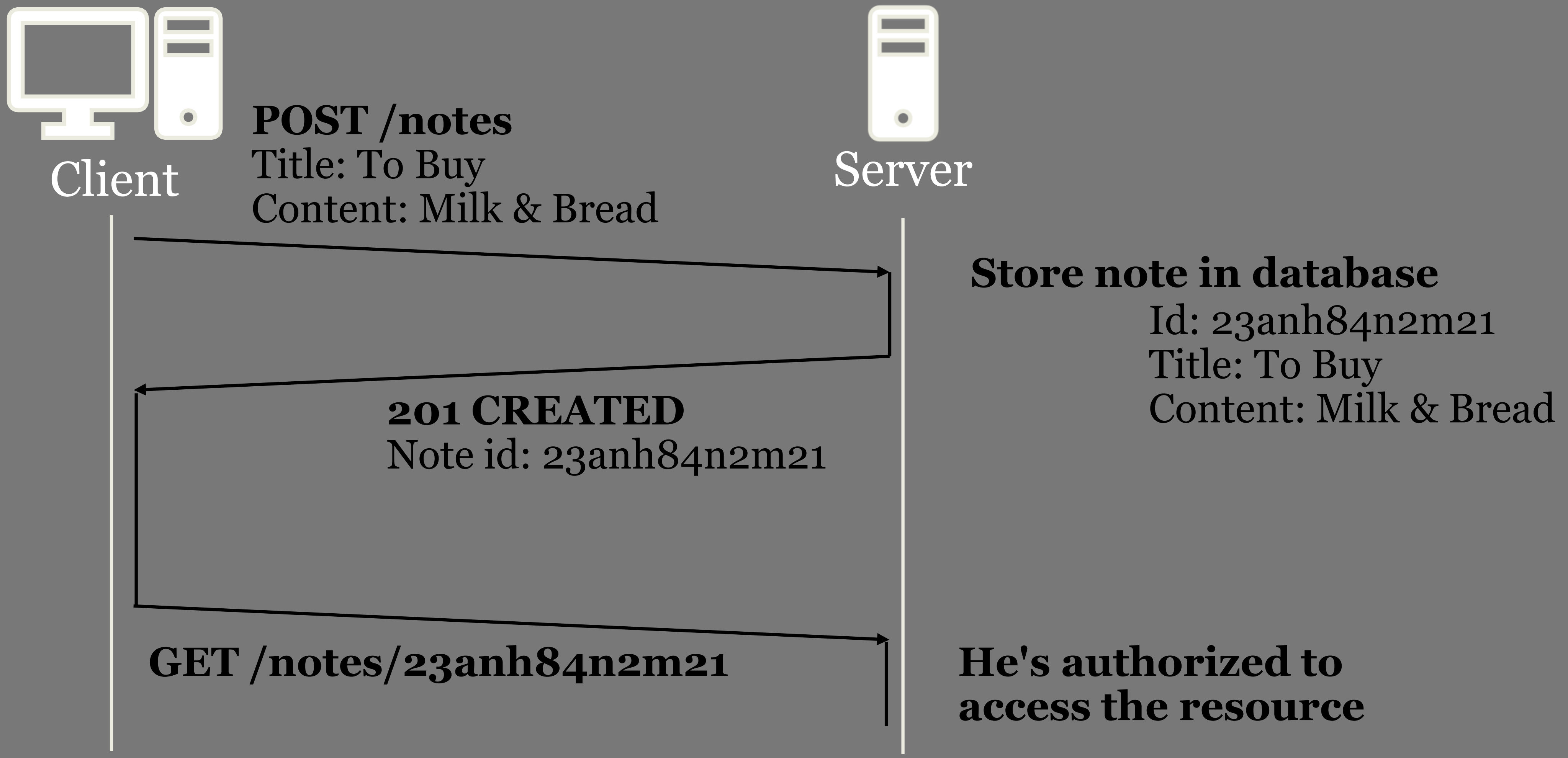
AUTHENTICATION VS AUTHORIZATION

Identity

Is the user really who he claims to be?

What is the user allowed to do?

AUTHORIZATION WITHOUT AUTHENTICATION



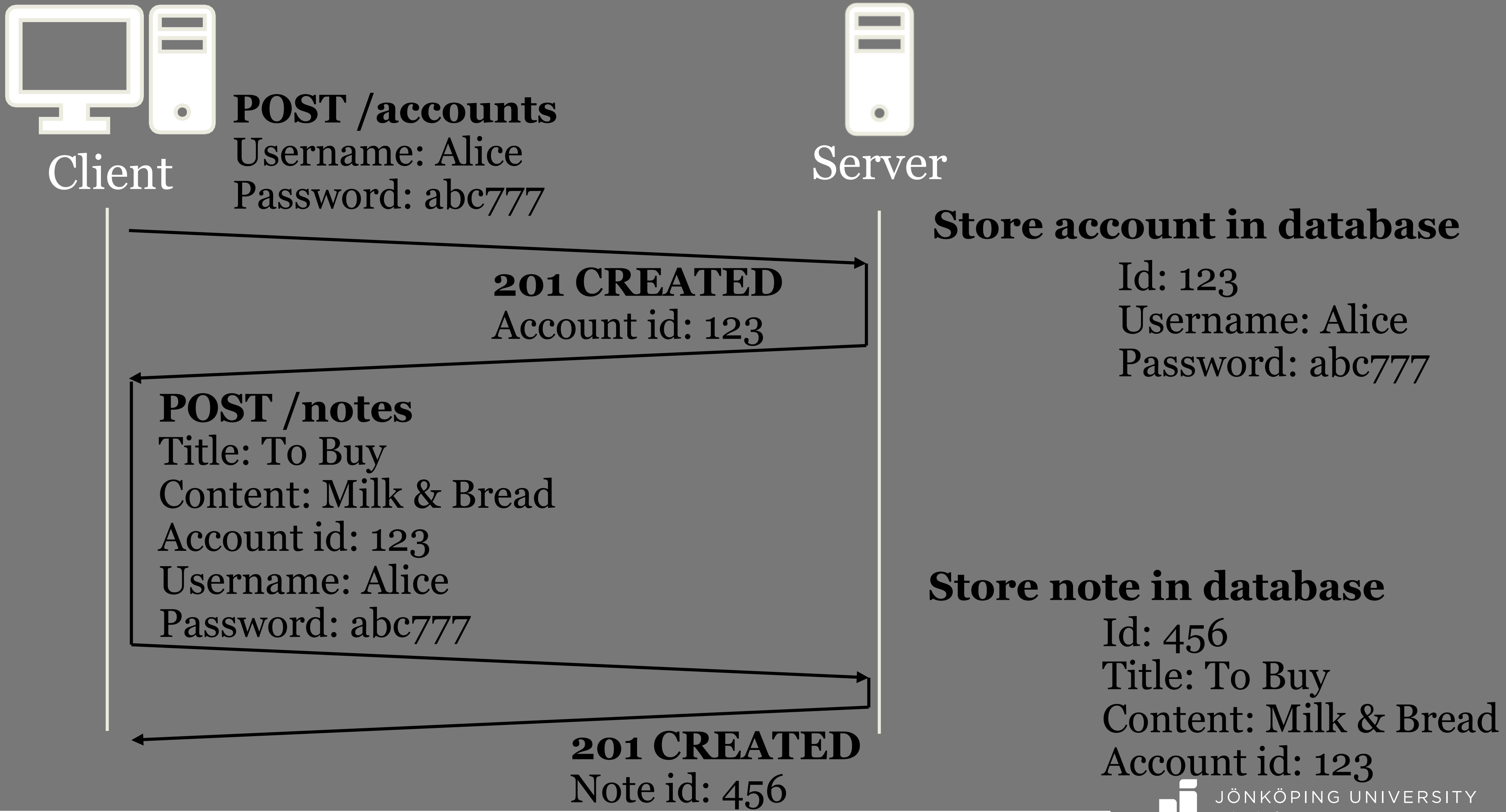
IMPLEMENTING AUTHENTICATION

1. Users needs to be uniquely identified.
   - Use account resources.
2. Users needs to be able to prove ownership of an account.
   - Each user shares a secret with the server, e.g. a password.

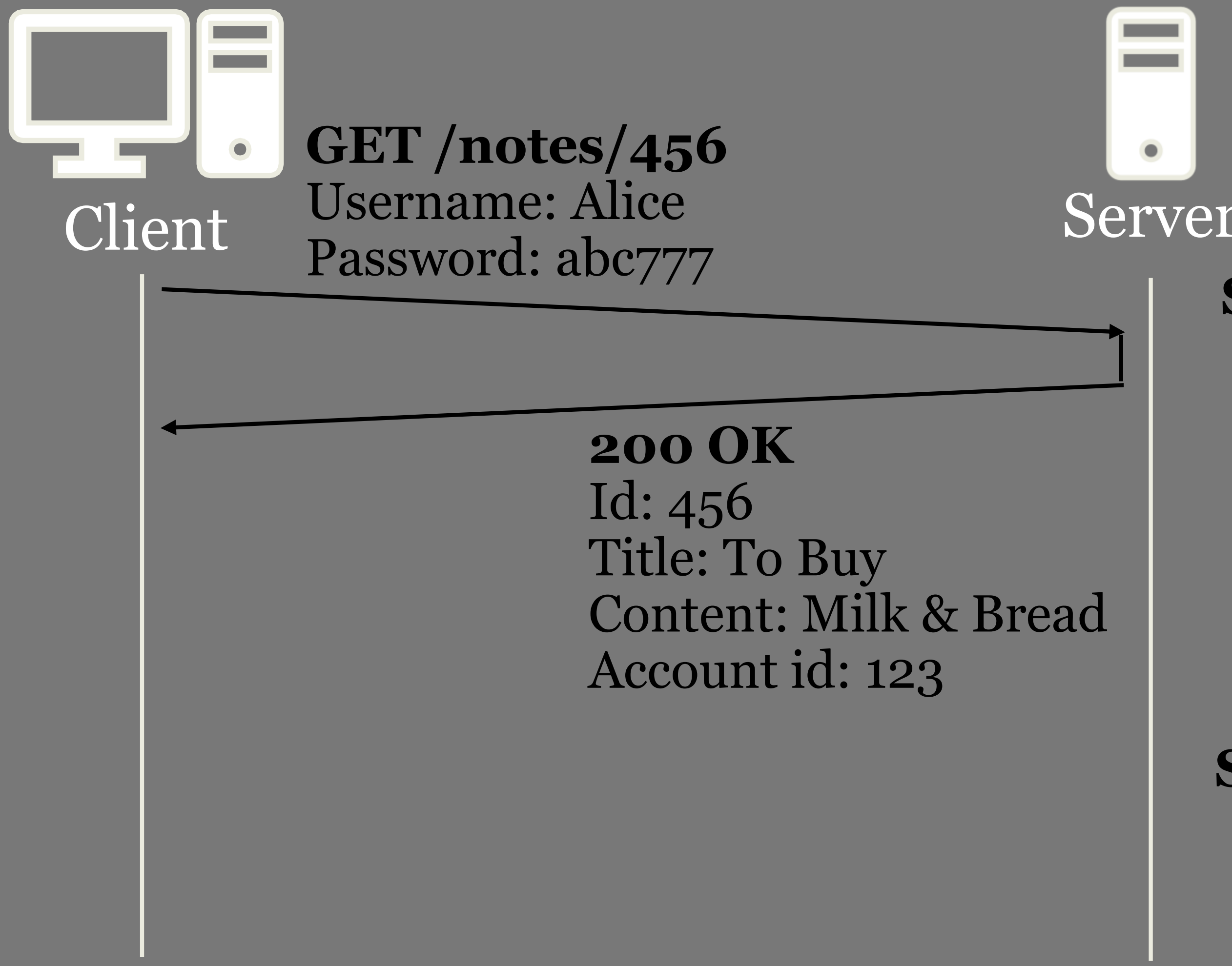
The accounts table

Id	Username	Password
1	User A	Password A
2	User B	Password B
3	User C	Password C
4	User D	Password D

AUTHORIZATION WITH AUTHENTICATION



AUTHORIZATION WITH AUTHENTICATION



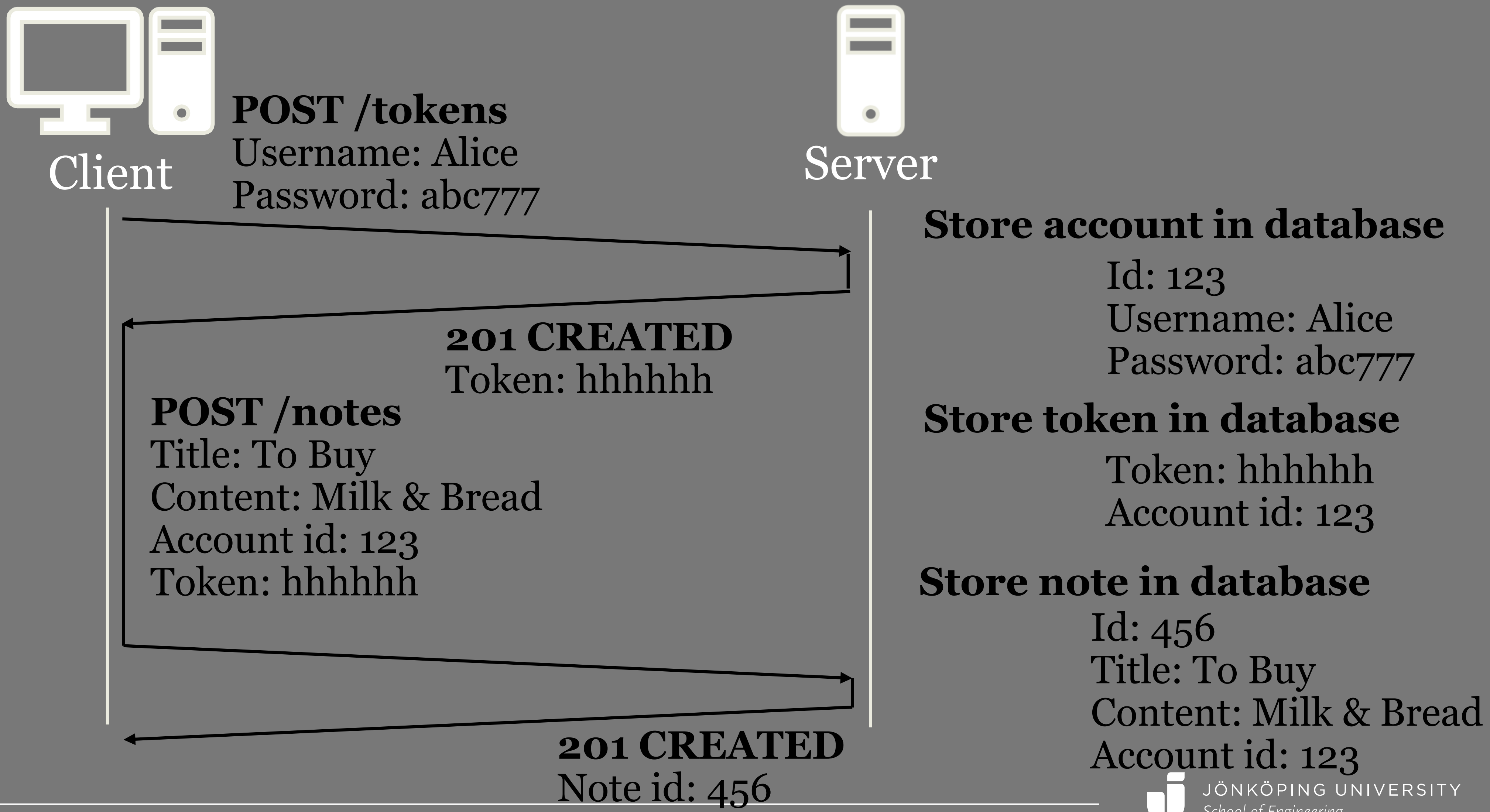
GET /notes/456
Username: Alice
Password: abc777

200 OK
Id: 456
Title: To Buy
Content: Milk & Bread
Account id: 123

Store account in database
Id: 123
Username: Alice
Password: abc777

Store note in database
Id: 456
Title: To Buy
Content: Milk & Bread
Account id: 123

AUTHORIZATION WITH TOKENS



POST /tokens
Username: Alice
Password: abc777

Store account in database
Id: 123
Username: Alice
Password: abc777

201 CREATED
Token: hhhhhh

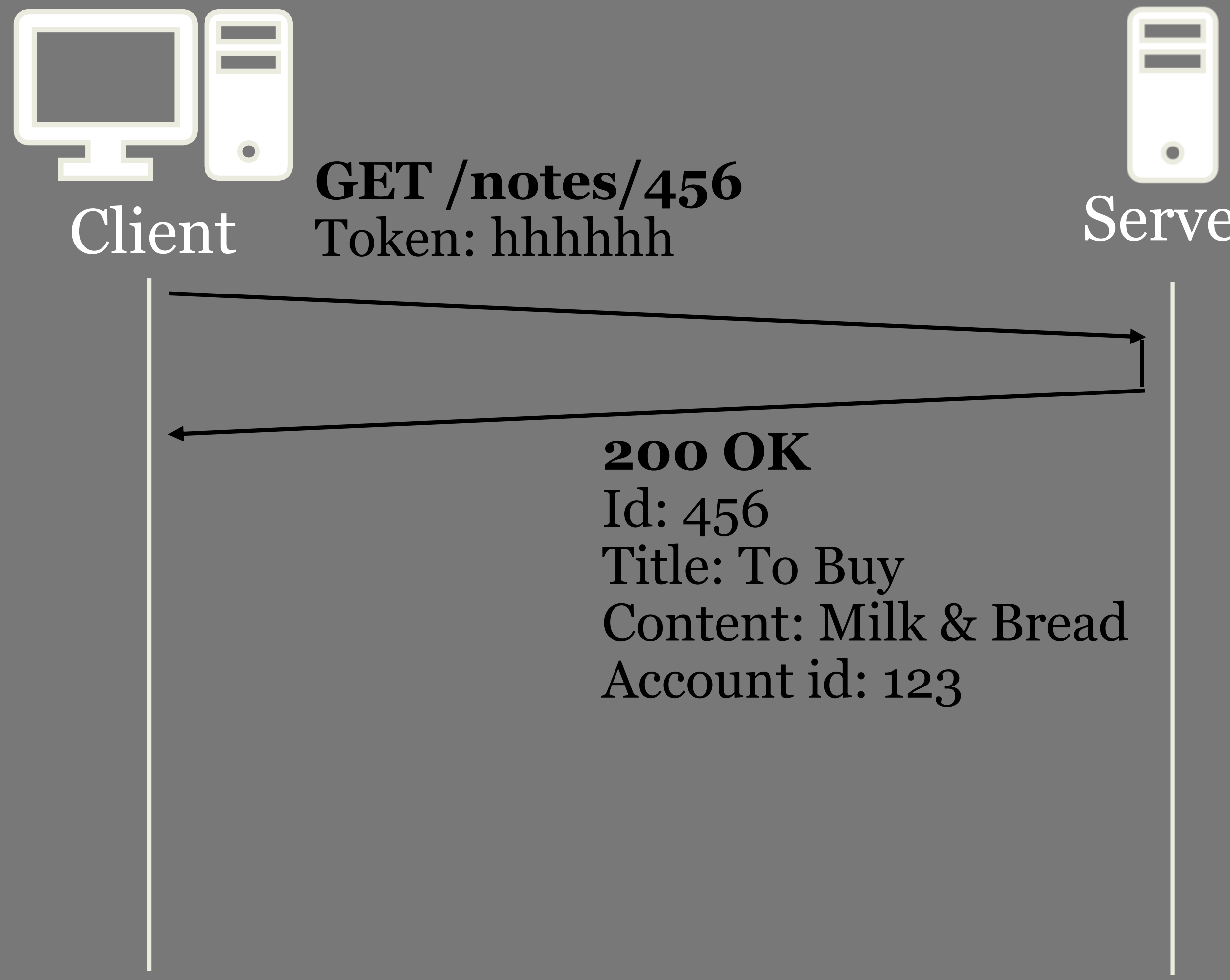
Store token in database
Token: hhhhhh
Account id: 123

POST /notes
Title: To Buy
Content: Milk & Bread
Account id: 123
Token: hhhhhh

Store note in database
Id: 456
Title: To Buy
Content: Milk & Bread
Account id: 123

201 CREATED
Note id: 456

AUTHORIZATION WITH TOKENS



Store account in database

Id: 123
Username: Alice
Password: abc777

Store token in database

Token: hhhhhh
Account id: 123

Store note in database

Id: 456
Title: To Buy
Content: Milk & Bread
Account id: 123

AUTHORIZATION

- Correctly implementing authorization is important.
- Proving that no security vulnerabilities exists is hard.

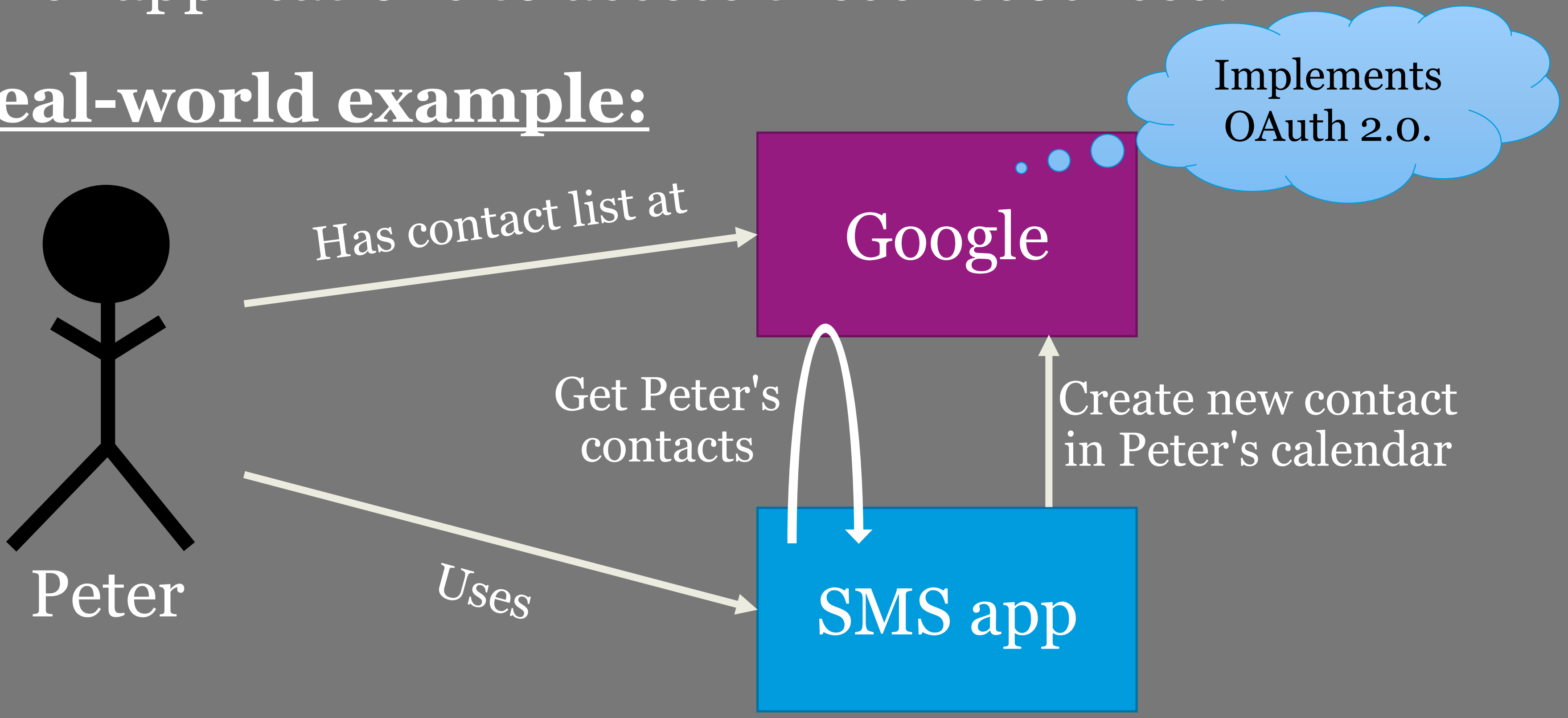
Authorization frameworks:

- Proved to work good.
- Everybody do it the same way.

OAUTH 2.0 - WHAT IS IT?

A framework for an application with user resources that allows other applications to access these resources.

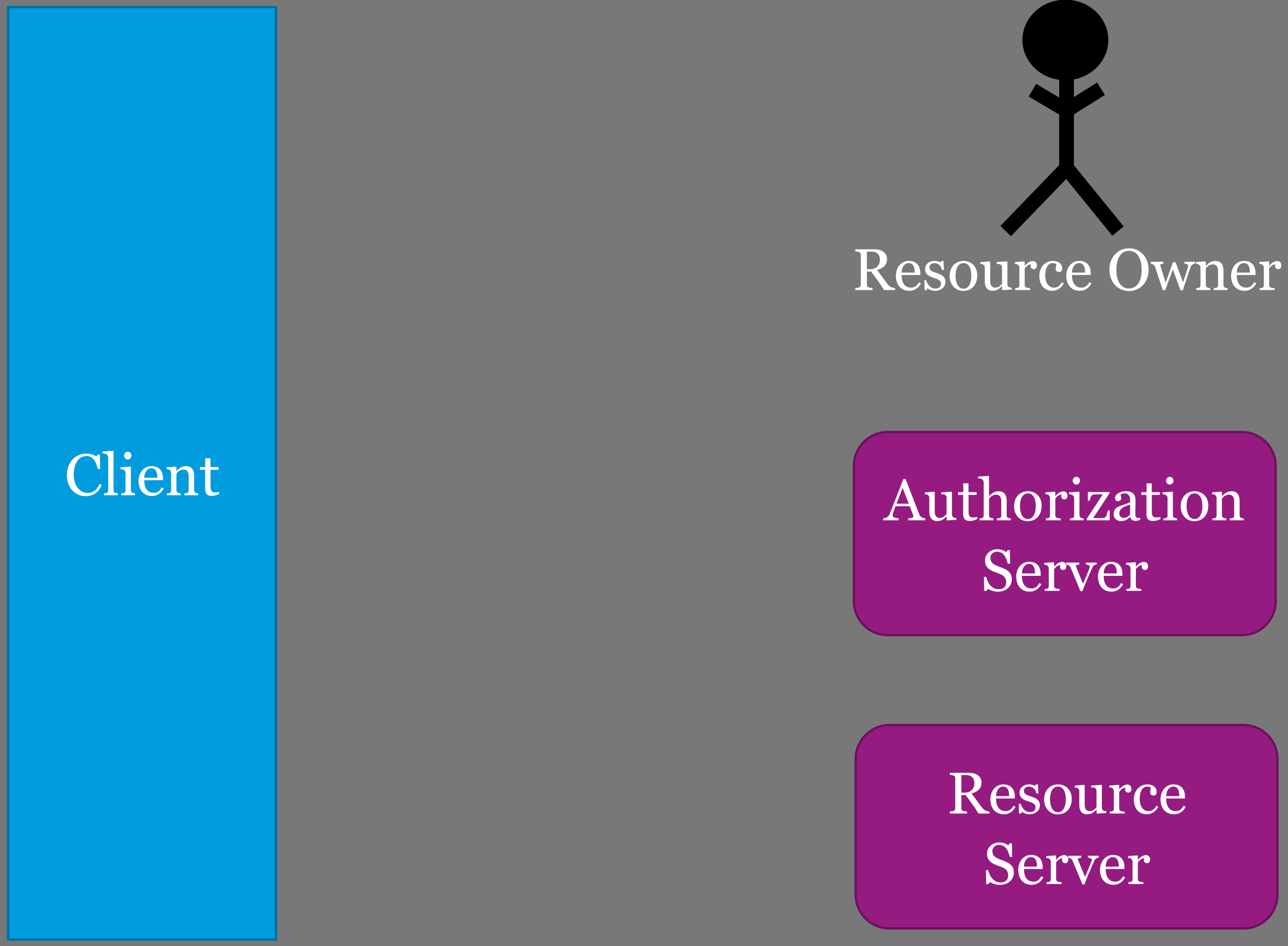
Real-world example:



OAUTH 2.0 - HOW DOES IT WORK?

1. The SMS app pre-register itself as an client at Google.
2. Peter starts using the SMS app.
3. The SMS app tells Peter it would like to access Peter's contact list at Google.
   - The SMS app redirects Peter to Google.
4. Peter tells Google that the SMS app may access his contact list.
   - Peter receives a token with permission to access his contact list.
5. Google redirects Peter back to the SMS app.
   - Peter gives the token to the SMS app.
6. The SMS app uses the token to prove to Google that it has permission to access Peter's contact list.

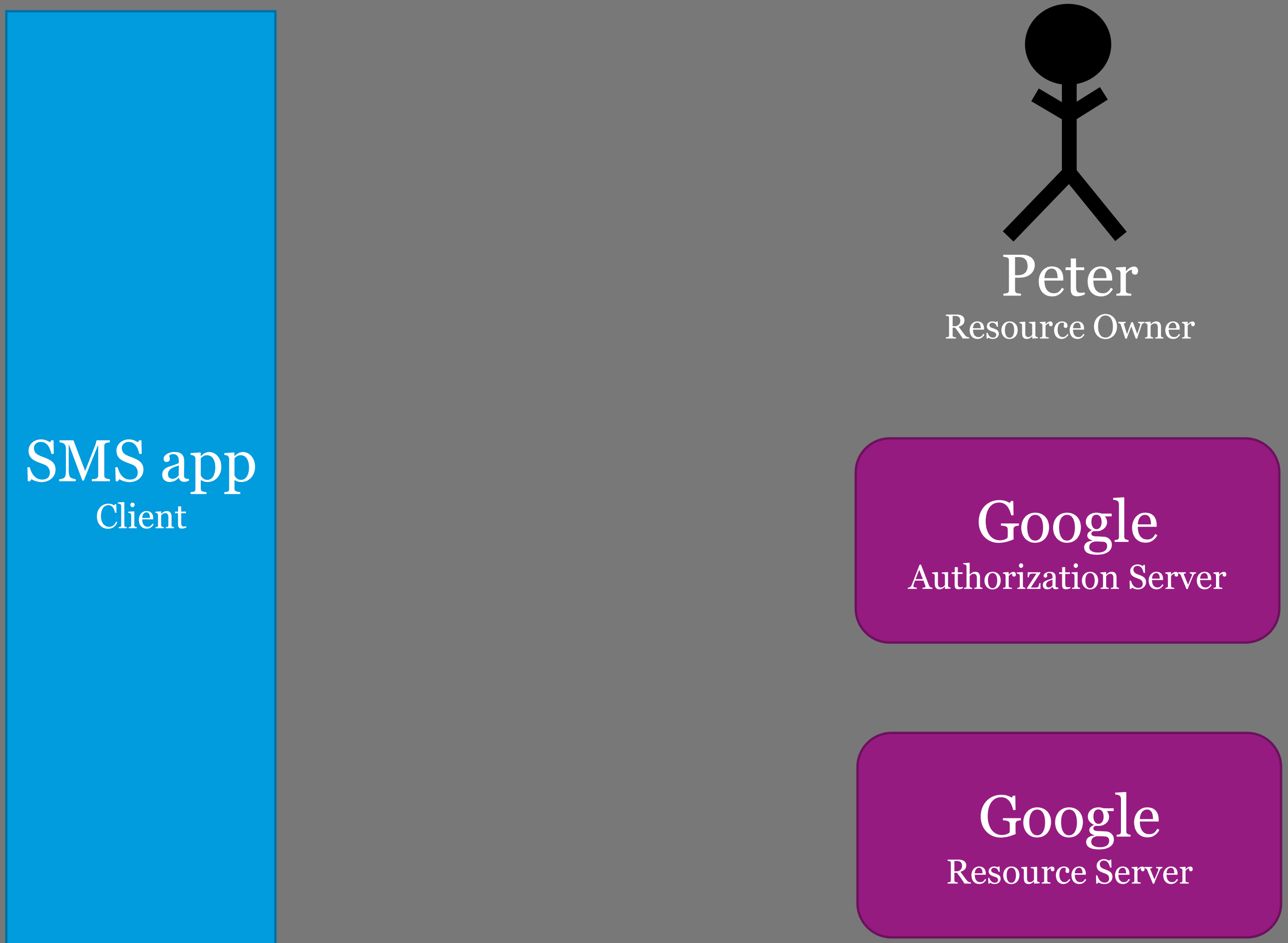
OAUTH 2.0 - ROLES



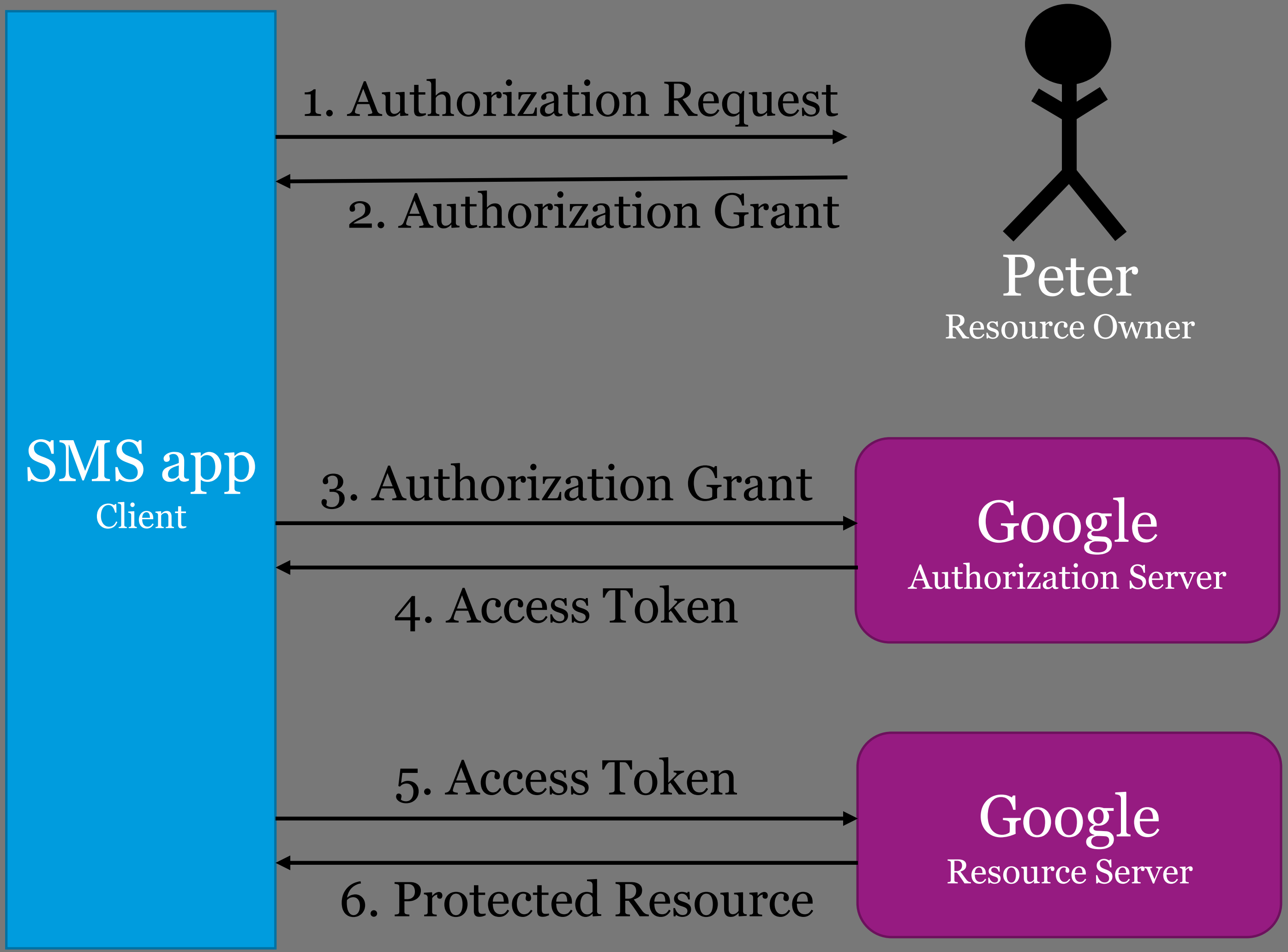
OAUTH 2.0 - ROLES

The client needs to register itself at the server first. Retrieves:

- `client_id`
- `client_secret`



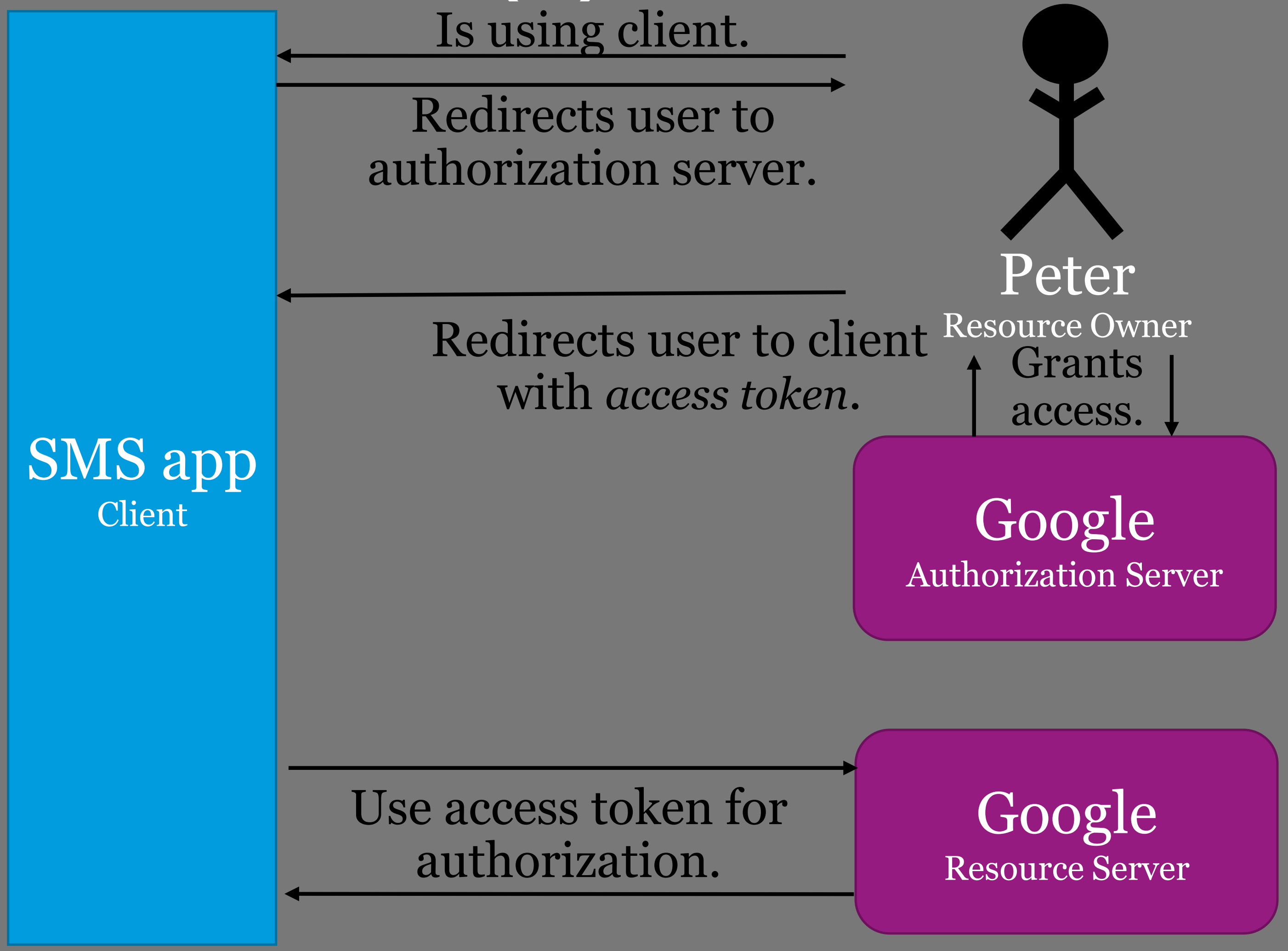
OAUTH 2.0 - BASIC FLOW



OBTAINING THE TOKEN (1)

There are four ways:

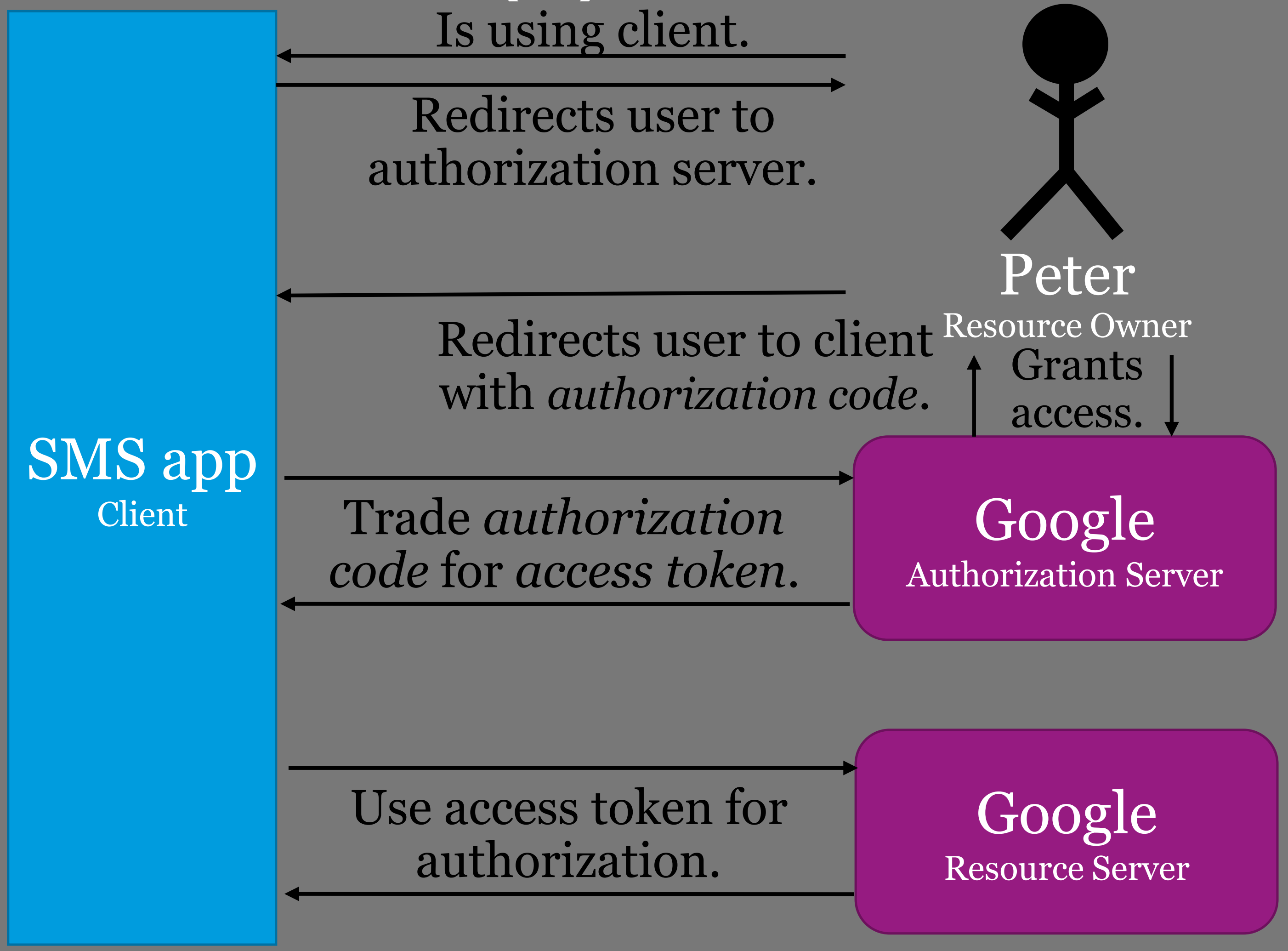
- Implicit
  (client="SPA or smartphone").


Is using client.
Redirects user to authorization server.
Peter
Resource Owner
Redirects user to client with *access token*.
Grants access.
SMS app
Client
Google
Authorization Server
Use access token for authorization.
Google
Resource Server

OBTAINING THE TOKEN (2)

There are four ways:

- Implicit.
- Authorization code (client="web app").

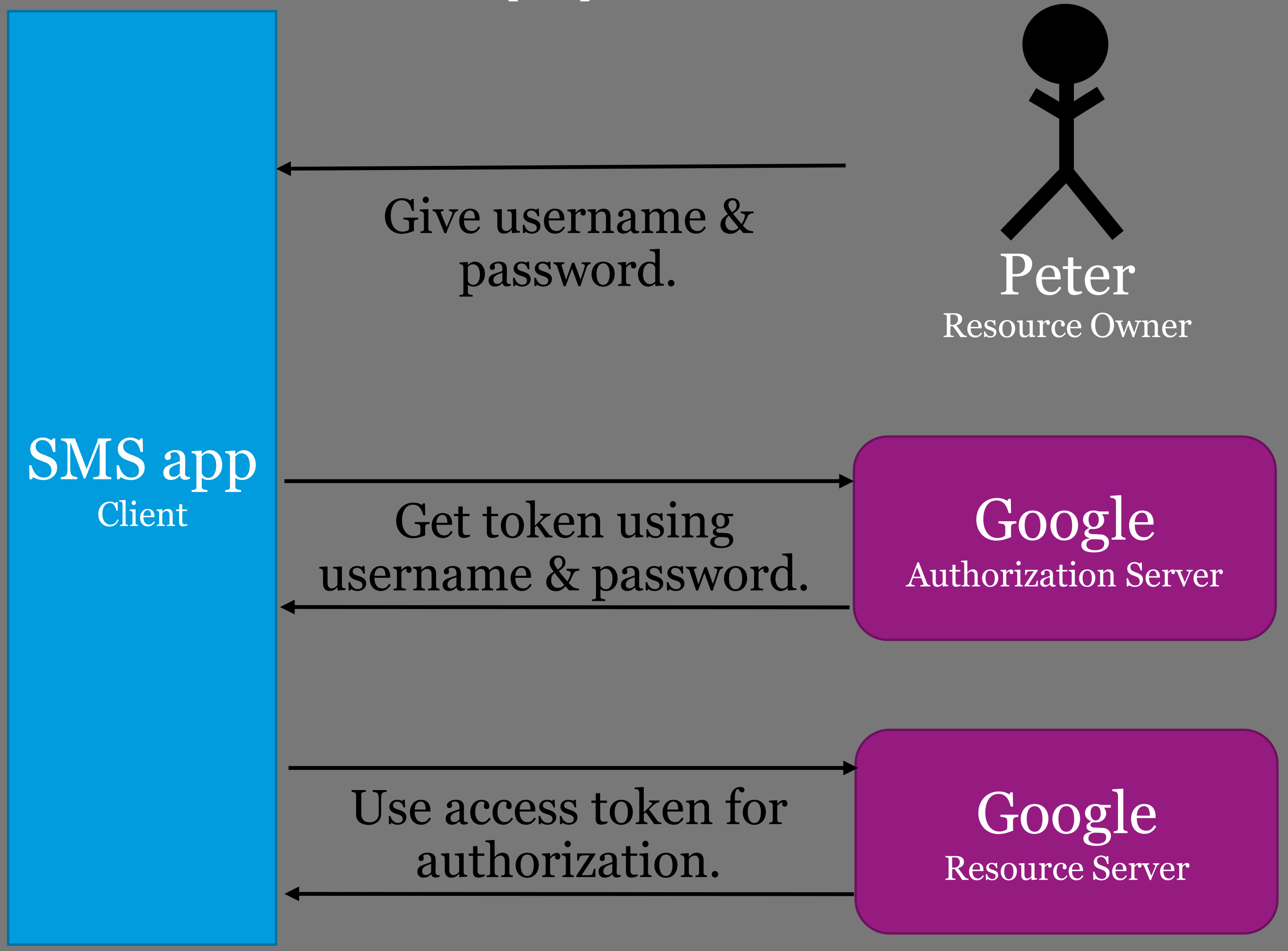


Is using client.
Redirects user to authorization server.
Peter
Resource Owner
Redirects user to client with *authorization code*.
Grants access.
SMS app
Client
Trade *authorization code* for *access token*.
Google
Authorization Server
Use access token for authorization.
Google
Resource Server

OBTAINING THE TOKEN (3)

There are four ways:

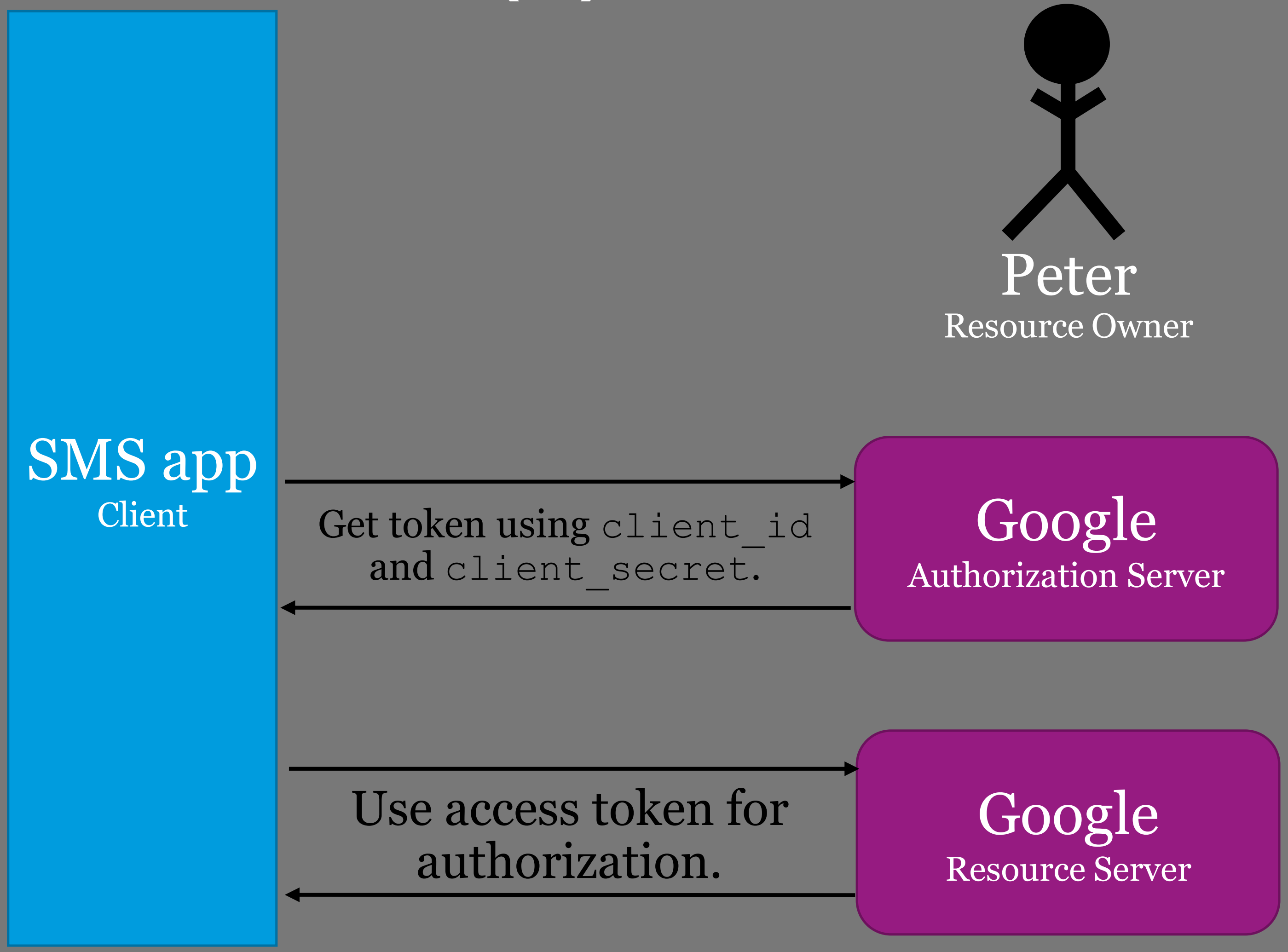
- Implicit.
- Authorization code.
- Resource Owner Password Credentials (for very trustful clients).


SMS app
Client
Give username & password.
Peter
Resource Owner
Get token using username & password.
Google
Authorization Server
Use access token for authorization.
Google
Resource Server

OBTAINING THE TOKEN (4)

There are four ways:

- Implicit.
- Authorization code.
- Resource Owner Password Credentials.
- Client credentials.


SMS app
Client
Peter
Resource Owner
Get token using client_id and client_secret.
Google
Authorization Server
Use access token for authorization.
Google
Resource Server

EXAMPLE

Accessing a user's calendar at Google.

1. Register your application as a client at Google API Console:
    1. Login at: https://console.developers.google.com
    2. Create a new project.
    3. Activate the Google APIs you want to use (Google Calendar).
    4. Obtain `client_id` and `client_secret`.

EXAMPLE

Accessing a user's calendar at Google.

1. Register your application as a client at Google API Console.
2. Ask a user for permission to access her Google calendar:
   1. Redirect user to:
      `https://accounts.google.com/o/oauth2/v2/auth?`
      `client_id=`**`YOUR_CLIENT_ID`**`&`
      `redirect_uri=`**`http://YOUR_SITE.COM/GOOGLE_RESPONSE`**`&`
      `response_type=code&`
      `scope=`**`https://www.googleapis.com/auth/calendar`**
   2. User accepts and is redirected back to:
      `http://YOUR_SITE.COM/GOOGLE_RESPONSE?code=`**`YOUR_CODE`**

EXAMPLE

Accessing a user's calendar at Google.

1. Register your application as a client at Google API Console.
2. Ask a user for permission to access her Google calendar.
3. On the server, exchange authorization code for access token:
    1. Send a POST request to:
       `https://www.googleapis.com/oauth2/v4/token`
       with the following body:
       ```
       code=YOUR_CODE&
       client_id=YOUR_CLIENT_ID&
       client_secret=YOUR_CLIENT_SECRET&
       redirect_uri=http://YOUR_SITE.COM/GOOGLE_RESPONSE&
       grant_type=authorization_code
       ```
    2. Read access token from the body of the response.

EXAMPLE

Accessing a user's calendar at Google.

1. Register your application as a client at Google API Console.
2. Ask a user for permission to access her Google calendar.
3. On the server, exchange access code for access token.
4. Use access token to access the user's calendars:
    1. Send GET request to:
       `https://www.googleapis.com/calendar/v3/users/me/calendarList`
       with the following header:
       `Authorization: Bearer` **`YOUR_TOKEN`**
    2. Read the user's calendars from the body of the response.

EXAMPLE

Useful resources for Google APIs:

- Obtaining token: https://developers.google.com/identity/protocols/OAuth2
  - Specific for web apps: https://developers.google.com/identity/protocols/OAuth2WebServer
- Calendar API scopes: https://developers.google.com/google-apps/calendar/auth
- Calendar API docs: https://developers.google.com/google-apps/calendar/v3/reference/

Try it yourself:

- https://developers.google.com/oauthplayground/