

# Universita degli Studi di Parma



DIPARTIMENTO DI INGEGNERIA ED ARCHITETTURA

Corso di Laurea di ingegneria Informatica, Elettronica e delle  
Telecomunicazioni

---

## OAuth2

**Relatore:**

Prof. Ing. Poggi Agostino

**Tesi di laurea di:**

Corradi Alessandro

A.A. 2019/2020

# Contents

---

|                                  |          |
|----------------------------------|----------|
| <b>Introduction</b>              | <b>1</b> |
| The problem . . . . .            | 1        |
| <b>OAuth2 Protocol</b>           | <b>2</b> |
| . . . . .                        | 2        |
| Obtain an access token . . . . . | 2        |
| Revoke an access token . . . . . | 2        |
| <b>SSO with OpenID</b>           | <b>3</b> |
| JWT . . . . .                    | 3        |
| JWS e JWE . . . . .              | 3        |
| Validazione JWS . . . . .        | 3        |

# Introduction

---

## The problem

Necessity of communication between client and server. Different ways to achieve the goal, here are a few of them:

1. Directly send the user id to the server

```
$ curl -v -X POST http://localhost:8080/db_auth '{"user-id": 1}'
```

2. Http Basic Auth: each API request is signed with a username and a password, encoded in base64

```
$ curl -u Aladdin:OpenSesamus -v http://localhost:8080/basic
```

```
> GET / HTTP/1.1
> Host: google.com
> Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW11cw==
> User-Agent: curl/7.72.0
> Accept: */*
...
```

3. Authentication with PEM keys

4. OAuth protocol

```
$ curl -v -X POST --header 'Authorization: Bearer 1234'
> GET / HTTP/1.1
> Host: www.google.com
> User-Agent: curl/7.72.0
> Accept: */*
> Authorization: Bearer 1234
```

# OAuth2 Protocol

---

## Obtain an access token

1. Redirect to an authorization provider.

`http://google.apis.com?response_type=code&client_id...`

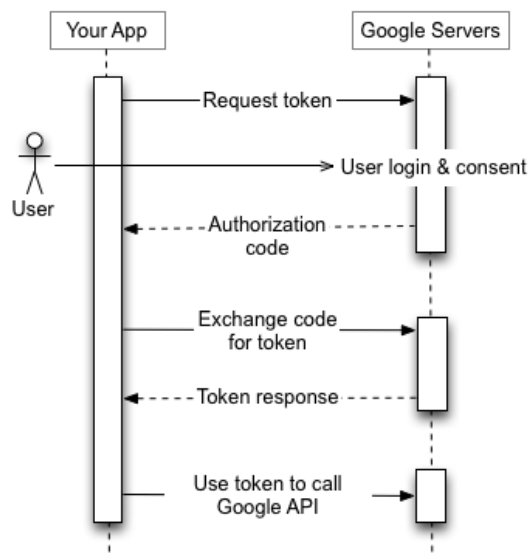
2. If the authorization is successful, the oauth server will redirect to `redirect_uri` passing a `code` as GET parameter. Otherwise `error` contains the reason why the authorization was not successful.

```
POST /oauth/token HTTP/1.1
Host: authorization-server.com

grant_type=authorization_code
&code=xxxxxxxxxxx
&redirect_uri=https://example-app.com/redirect
&client_id=xxxxxxxxxx
&client_secret=xxxxxxxxxx
```

*If an authorization code is used more than once, the authorization server must deny the subsequent request.*

3. The client exchanges the `code`, called also "grant token", with the server to obtain an `access_token`.
4. The server returns the access token with additional informations, such as expire date and JWT.
5. The client uses the access token for APIs requests.



## Revoke an access token

Something something

# SSO with OpenID

---

**JWT**

**JWS e JWE**

**Validazione JWS**

Verify the SHA256 signature

# References

---

- [1] Using OAuth 2.0 to Access Google APIs  
<https://developers.google.com/identity/protocols/oauth2>