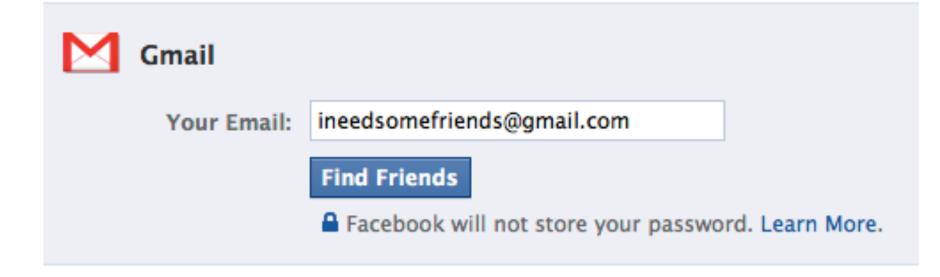
Standards in Production Today

OAuth 2.0 and OIDC in Action

OAuth 2.0 in a nutshell



Replacement for password-sharing pattern

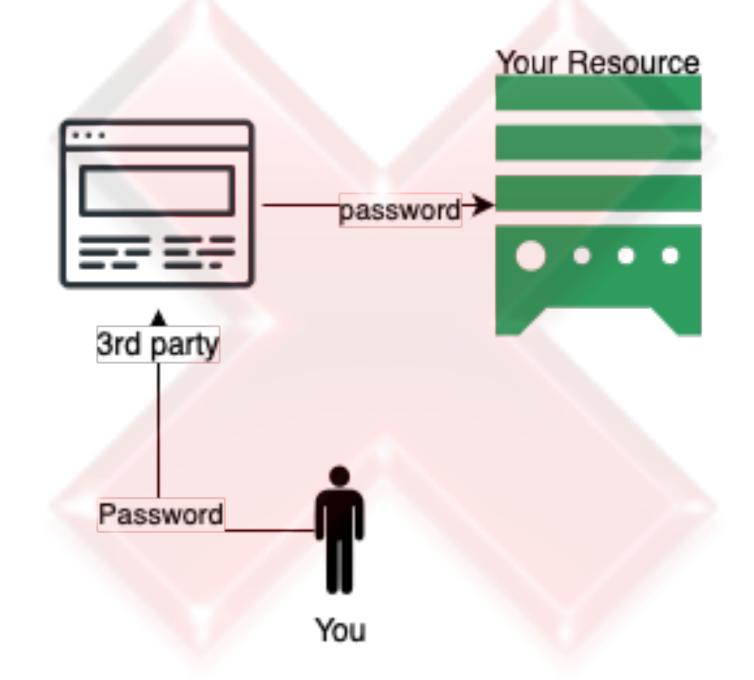


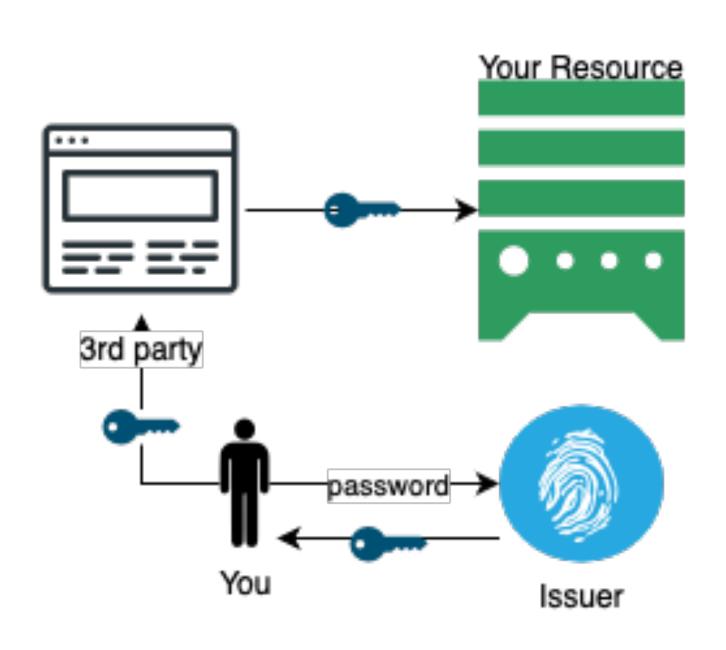
 Around 2007 engineers from Twitter, Google gathered for the need of an API access delegation protocol

OAuth 2.0 in a nutshell



- A complete replacement of OAuth 1.0
- Resource owner, resource server, client, authorization server





Grant Type - What is it



- The way/method to get an access token
- Access token is your "opaque" key card
- Scope, what exactly I can access with it?

More definitions can be found in the RFC -> https://www.rfc-editor.org/rfc/rfc6749

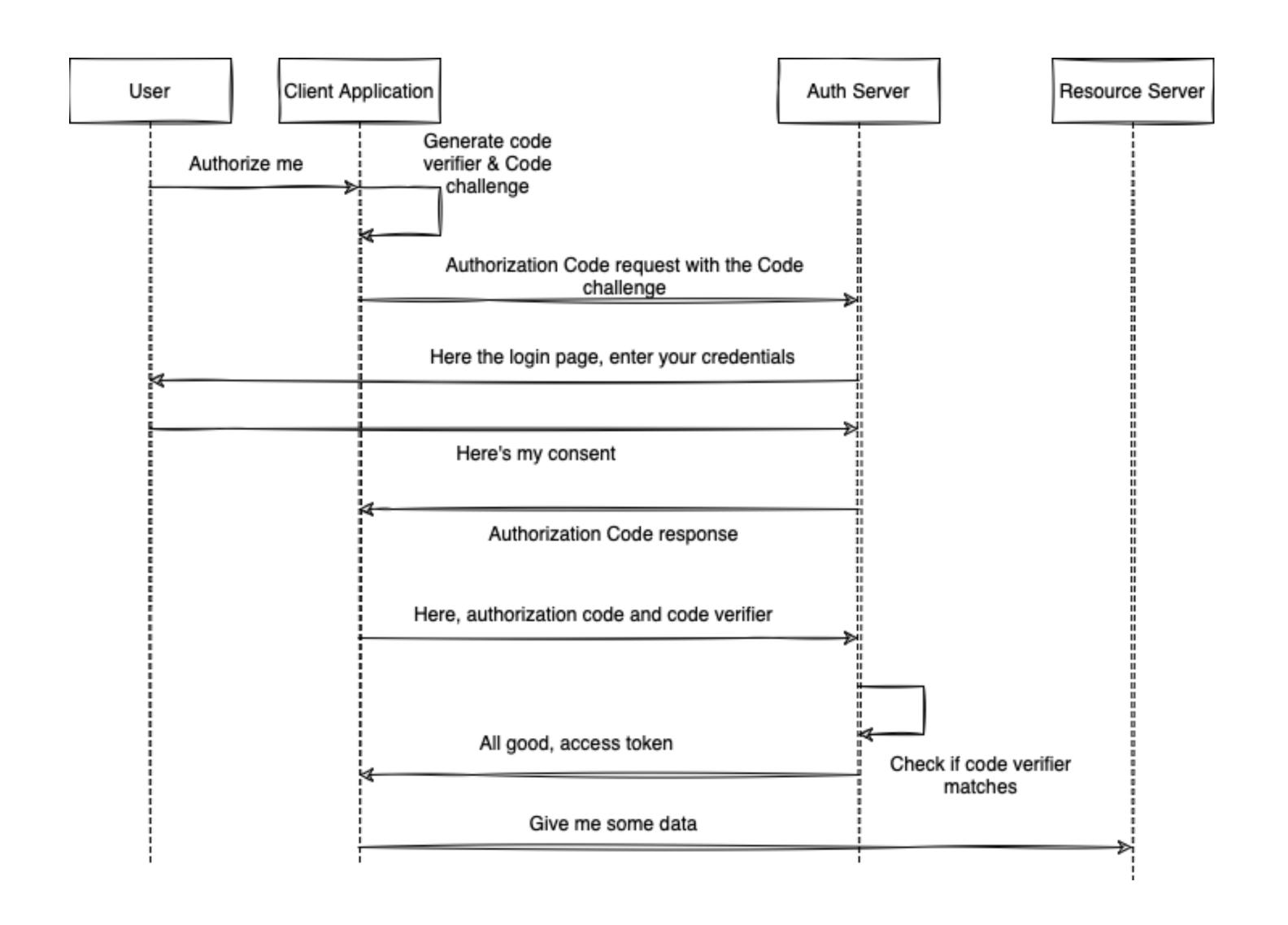
Authorization Code Grant AUTHORIZATION SERVER AUTH DIVISION HEY HERE IS MY LOGIN PAGE, ENTER YOUR CREDENTIALS SO THAT I KNOW ITS YOU ITS MÉ SIR AUTHORIZE ME PLS? HERE AN AUTHORIZATION CODE AUTHORIZATION SERVER AUTH DIVISION SENT ME THIS CODE, CAN I GET A TOKEN? • TOKEN DIVISION CLIENT I WANT ACCESS **APPLICATION** ACCESS TOKEN USER AGENT (BROWSER ETC.) ACCESS TOKEN RESOURCE

Authorization Code with PKCE

Hardened Authorization Code Grant

- Authorization code grant will be deprecated with OAuth 2.1
- Attackers can listen to network to steal the authorization code
- PKCE Proof Key for Code Exchange

Authorization Code with PKCE



Other Grant Types

- Client Credentials
- Implicit grant
- Device code

• Playground for grant types -> https://www.oauth.com/playground/



- Introduced by OpenID foundation
- OpenID Connect is a layer on top of OAuth
- Please welcome a companion token to the access token, ID token
- ID token is for client use only

Sample contents for both tokens

Access Token

```
{
    "sub": "05a725db-be17-4705-b7d0-48b662509850",
    "iss": "my-issuer",
    "client_id": "154ilttrkk6kmk5c87sul2vgve",
    "origin_jti": "71de7070-dbfc-42f9-8da9-9bb0c104b932",
    "event_id": "442a547a-364c-4121-ba1b-3d8684bbef8d",
    "token_use": "access",
    "scope": "get-books",
    "auth_time": 1670261383,
    "exp": 1670261983,
    "iat": 1670261387,
    "jti": "0a0996f7-fec3-4097-98c0-df446605cf93",
    "username": "05a725db-be17-4705-b7d0-48b662509850"
}
```

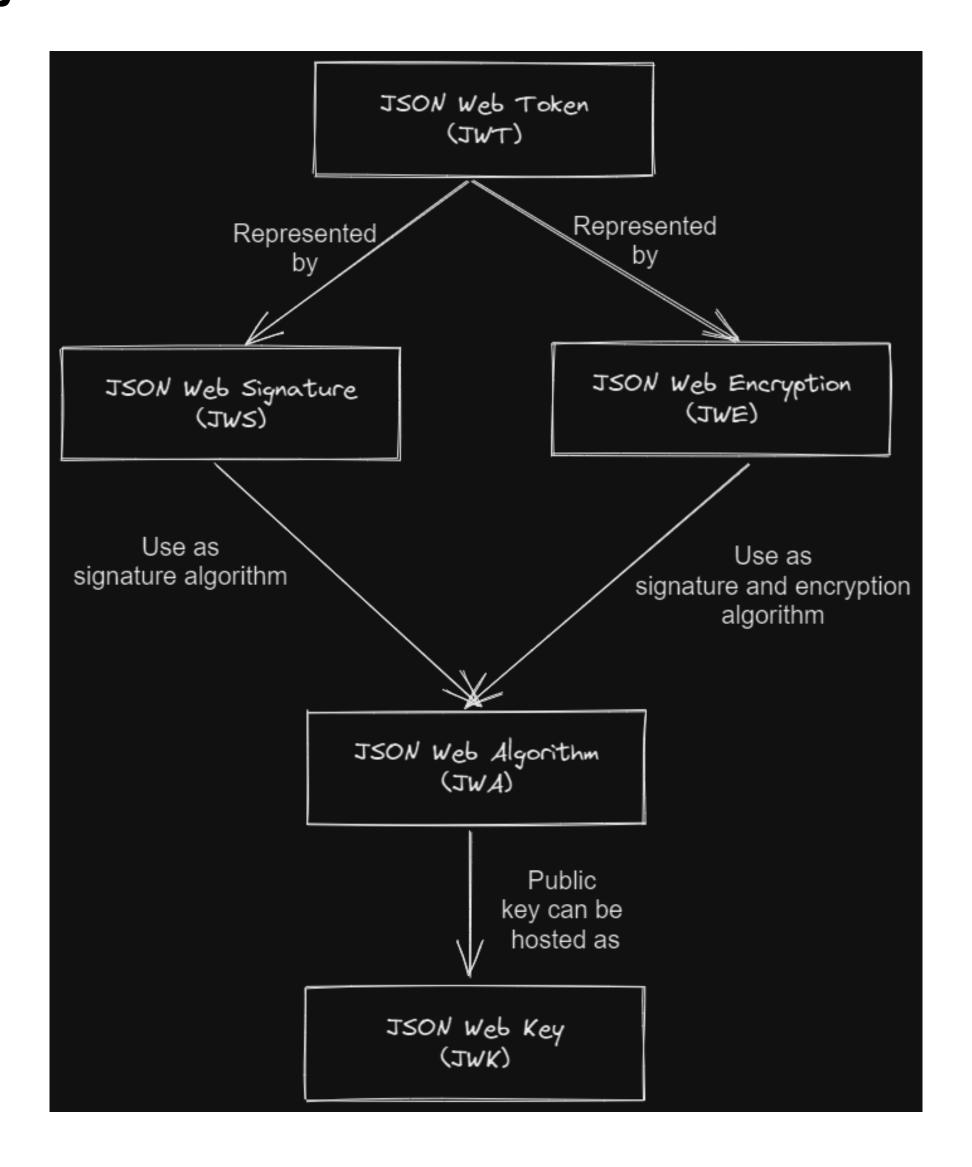
ID Token

```
"sub": "05a725db-be17-4705-b7d0-48b662509850",
"email_verified": true,
"iss": "my-issuer",
"phone_number_verified": true,
"username": "05a725db-be17-4705-b7d0-48b662509850",
"env": "TEST",
"actorUuid": "d68bc7af-db7e-11e8-a4ce-005056b910a4",
"origin_jti": "71de7070-dbfc-42f9-8da9-9bb0c104b932",
"aud": "154ilttrkk6kmk5c87sul2vgve",
"event_id": "442a547a-364c-4121-ba1b-3d8684bbef8d",
"updated_at": 1667998236,
"token_use": "id",
"auth_time": 1670261383,
"name": "d68bc7af-db7e-11e8-a4ce-005056b910a4",
"phone_number": "+1111111111111",
"exp": 1670261983,
"iat": 1670261383,
"jti": "99a70b45-5895-4cf8-937e-0ebd419454c0",
"email": "supporter@example.com"
```

JSON Web Tokens

- The most common structure for Access tokens and ID tokens
- It is securely signed, mostly not encrypted
- It is not the holy way of handling authorization, comes with drawbacks

Json Web Token Terminology



Structure of JWT's

• jwt.io is the central place if you work with JWT's

Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY30DkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1Kx
wRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

Decoded EDIT THE PAYLOAD AND SECRET

```
HEADER: ALGORITHM & TOKEN TYPE
   "alg": "HS256",
   "typ": "JWT"
PAYLOAD: DATA
   "sub": "1234567890",
   "name": "John Doe",
   "iat": 1516239022
VERIFY SIGNATURE
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
```

Thank you:)