OAuth 2.0 meets verifiable credentials and blockchain-based tokens

Nikos Fotiou

https://mm.aueb.gr    https://www.sofie-iot.eu/
About this presentation

• Partially based on:


• On going work in the context of H2020-SOFIE
About SOFIE*

• SOFIE enables interoperability between existing IoT platforms
  • Utilizes distributed ledger technologies
  • 3-year EU Horizon 2020 project, will end in December 2020

• SOFIE functionality will be provided through its framework**
  • “Privacy and Data Sovereignty,” and “Identity, Authentication, and Authorization” are two key components of the SOFIE framework

* Secure Open Federation for Internet Everywhere https://www.sofie-iot.eu/
** https://github.com/SOFIE-project/Framework
About SOFIE

• SOFIE enables interoperability between existing IoT platforms
  • Utilizes distributed ledger technologies
  • 3-year EU Horizon 2020 project, will end in December 2020

• SOFIE functionality will be provided through its framework
  • “Privacy and Data Sovereignty,” and “Identity, Authentication, and Authorization” are two key components of the SOFIE framework

OAuth 2.0

JWT
OAuth 2.0-based authorization

Client

Authorization request

Authorization grant

Authorization server

Access token

Resource request, token

Resource server

Resource owner

Authorization grant

Resource owner

Authorization grant

Authorization server

Access token
OAuth 2.0-based authorization

Client ➔ Authorization request ➔ Authorization grant ➔ Resource owner

Resource owner ➔ Authorization grant ➔ Authorization server ➔ Access token ➔ Authorization server

Authorization grant ➔ Access token ➔ Authorization server ➔ Resource request, token ➔ Resource server

Verifiable Credentials ➔ Authorization grant ➔ Authorization server

Blockchain-backed JWTs ➔ Authorization server ➔ Resource request, token ➔ Resource server


SOFIE Clients

- Resource owners do not interact with clients
- Clients may not even have UI (e.g., IoT device)
  → Client credentials are the “recommended” authorization grant for this case
- But we want to avoid long, hard to manage ACLs in the authorization server

<table>
<thead>
<tr>
<th>Client Identifier</th>
<th>Resource Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
<td>[Resource 1, Resource 2, Resource 3]</td>
</tr>
<tr>
<td>Client 2</td>
<td>[Resource K, Resource L]</td>
</tr>
</tbody>
</table>

→ Verifiable Credentials can solve this problem
VC in a nutshell

• A standard* way to express credentials on the Web

* W3C, Verifiable Credentials Data Model 1.0, https://www.w3.org/TR/vc-data-model/
VC in a nutshell

• A standard* way to express credentials on the Web

* W3C, Verifiable Credentials Data Model 1.0, https://www.w3.org/TR/vc-data-model/
# VC Structure

<table>
<thead>
<tr>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issuer Id</strong></td>
</tr>
</tbody>
</table>

**Credential**
- Type
- Subject Id
- Claims

**Proof**
An example credential

```json
sofie_credential = {
    "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://w3id.org/credentials/v1",
    ],
    "id": "https://www.sofie-iot.eu/credentials/examples/1",
    "type": ["VerifiableCredential"],
    "issuer": "did:naci:E390CF3B5B93E921C45ED978737D89F61B8CAFF9DE768FA5F630A20386BCCA3B",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
        "id": "did:naci:A490CF3B5B93E921C45ED978737D89F61B8CAFF9DE768FA5F630A20386BCCA62",
        "type": ["AllowedURLs"],
        "acl": [
            {
                "url": "http://sofie-iot.eu/device1",
                "methods": ["GET","POST"]
            },
            {
                "url": "http://sofie-iot.eu/device2",
                "methods": ["GET"]
            }
        ]
    }
}
```
An example credential

sofi_credential = {
    "@context" : [
        "https://www.w3.org/2018/credentials/v1",
        "https://mm.aueb.gr/contexts/access_control/v1"
    ],
    "id" : "https://www.sofie-iot.eu/credentials/examples/1",
    "type" : ["VerifiableCredential"],
    "issuer" : "did:mac:1:E390CF335B93E921C45ED978737D89F61B8CAFF9DE768FA5F63DA20386BCCA3B",
    "issuanceDate" : "2010-01-01T19:23:24Z",
    "credentialSubject" : {
        "id" : "did:mac:1:A490CF38593E921C45ED978737D89F61B8CAFF9DE768FA5F63DA20386BCCA62",
        "type" : ["AllowedURLs"],
        "ecl" : [ { "url" : "http://sofie-iot.eu/device1", "methods" : ["GET","POST"] } ],
        { "url" : "http://sofie-iot.eu/device2", "methods" : ["GET"] } ]
}
}
Interacting with SOFIE’s PDS component

HTTP POST http://sofie-iot.eu/token
“grant_type=vc&vc=...”

HTTP 401
'WWW-Authenticate: VC challenge=[...]

HTTP POST http://sofie-iot.eu/token
“grant_type=vc&vc=...proof=...”

....

HTTP 200
{"access_token":.....}
PDS configuration

filters = [
   ["$.@context[*]", "https://mm.aueb.gr/contexts/access_control/v1"],
   ["$.issuer", ["did:nacl:....", "another issuer", "or this issuer"]],
   ["$.credentialSubject.acl[?@.url='http://sofie-iot.eu/device1'].methods[*]","GET"]
]
The use of JWT in SOFIE

• JWT is a standard mean for transferring claims, used in many authorization systems
• Usually they are used as Bearer tokens
• We leverage blockchain to provide
  • Proof-of-possession
  • Revocation
  • Delegation
The Ethereum blockchain

- Decentralized “smart contract” executed by untrusted nodes
- Smart contract code and state are public
- Smart contract execution is deterministic
- State modification are permanently recorded in the blockchain
- Users identified by a public key. The hash of the public key is used as the “address” of the user. The private key is used for signing “transactions”
ERC-721

ERC-721 tokens
• Token Id
• Owner Id
• Metadata
ERC-721

ERC-721 tokens
- Token Id
- Owner Id
- Metadata

ERC-721 token management contract
- ownerOf()
- transferFrom()
- approve()
- getApproved()
- tokenURI()
JWT

Client

{  
  "iss": Authorization Server
  "aud": ...  
  "sub": client identifier
  "exp": ...
  "jti": token identifier
}

Authorization server

Access token
JWT + ERC-721

Client

{  
  "iss": Authorization Server
  "aud": ...
  "sub": ...
  "exp": ...
  "jti" : token identifier
}

Authorization server

Access token

ERC-721 token

Token Id : jti
Owner Id : Client identifier
Metadata: JWT
Accessing legacy resource servers

• It facilitates logging and auditing services
Accessing resource servers with BC read access

Client → Resource server
- Resource request, token
- Verify Client key ownership
- Resource

Resource server →
- ownerOf(), tokenURI()
Revocation

- Revocation is asynchronous
- Authorization server does not have to be online
Delegation

- Delegation is not transitive
- Revocation is not affected
Fair exchange

Client ➔ Access token ➔ Authorization server ➔ ERC-721 token ➔ Payment ➔ transferFrom()

Token identifier
Owner : Authorization server
Metadata: JWT
A note about blockchains

• (Public) blockchains have privacy issues, introduce delays (~13sec per transaction) and monetary costs (~$0.10 to create a token, $0.02 to revoke or delegate)
  • In no payments are involved then private, or testing chains can be used.
Thank you

fotiou@aueb.gr
https://www.sofie-iot.eu/
https://github.com/SOFIE-project/Framework