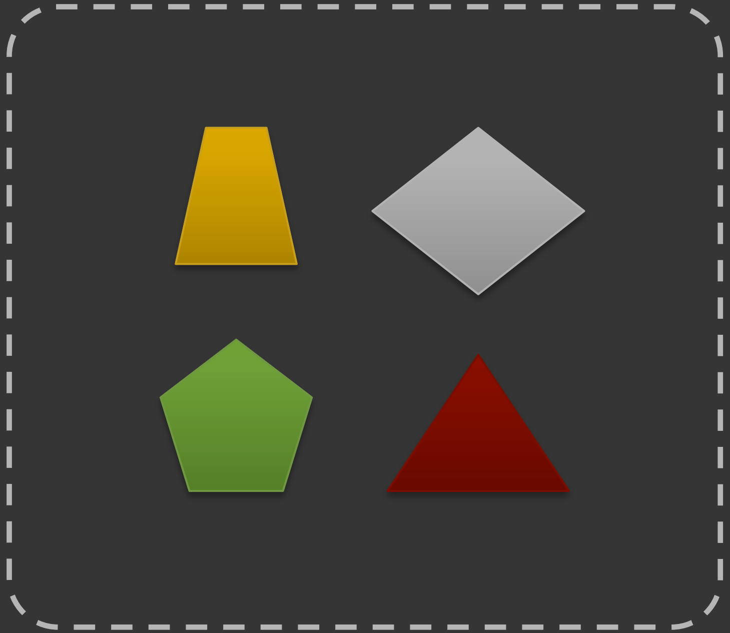
twobotechnologies

OAuth and OpenID Connect for Microservices

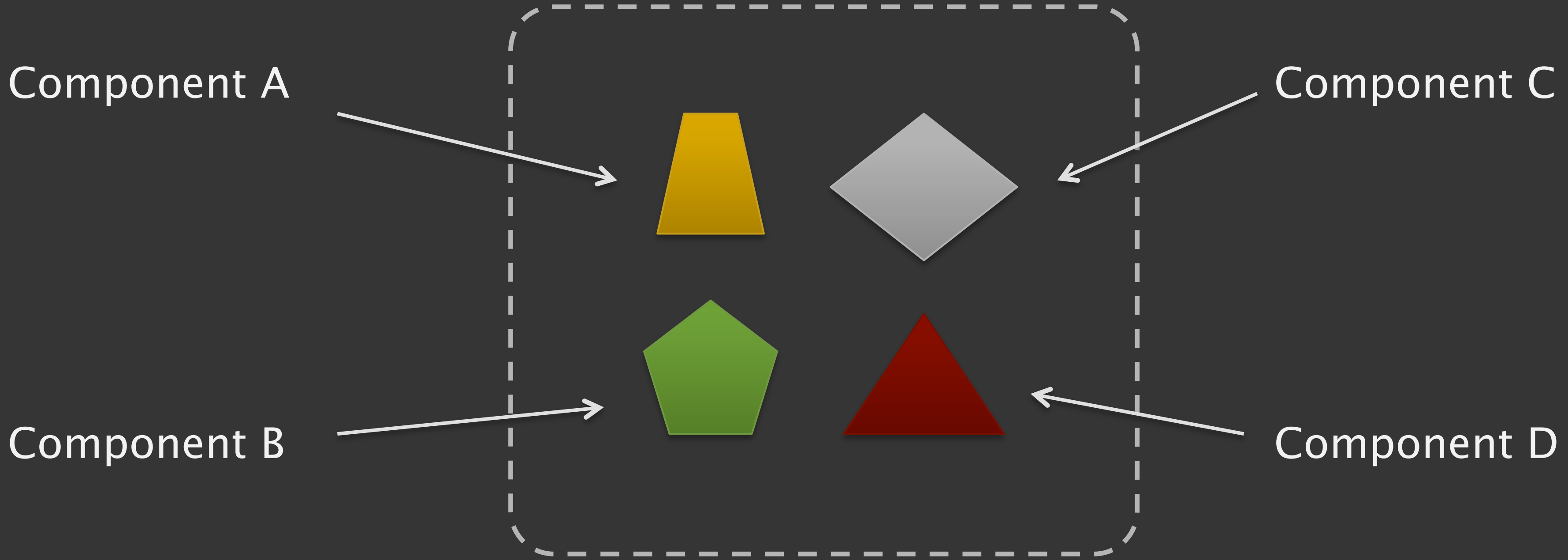
A homogenous solution for a heterogeneous problem

Jacob Ideskog – Identity Specialist at Twobo Technologies

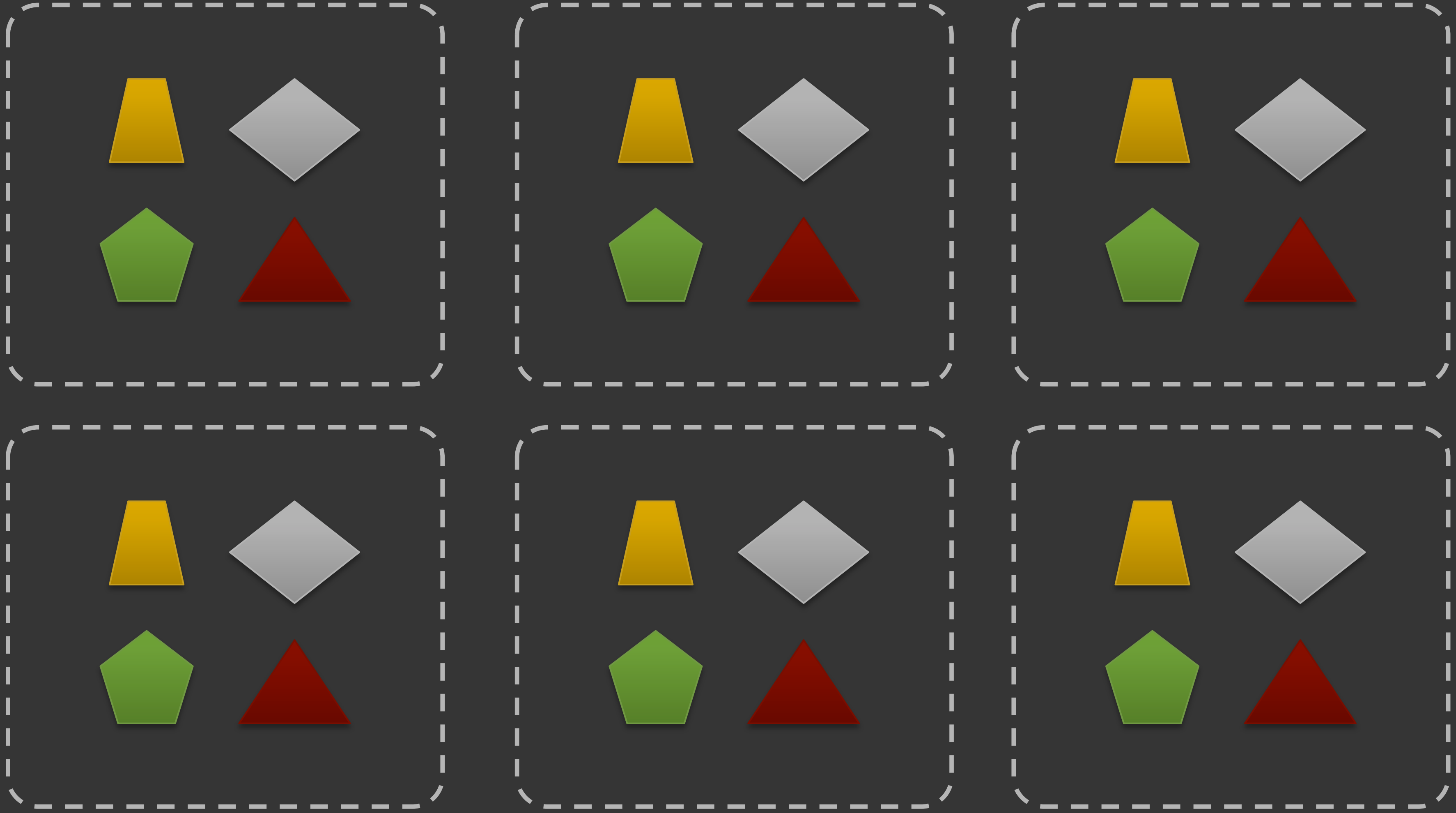
A Traditional Service



With Traditional Subsystems



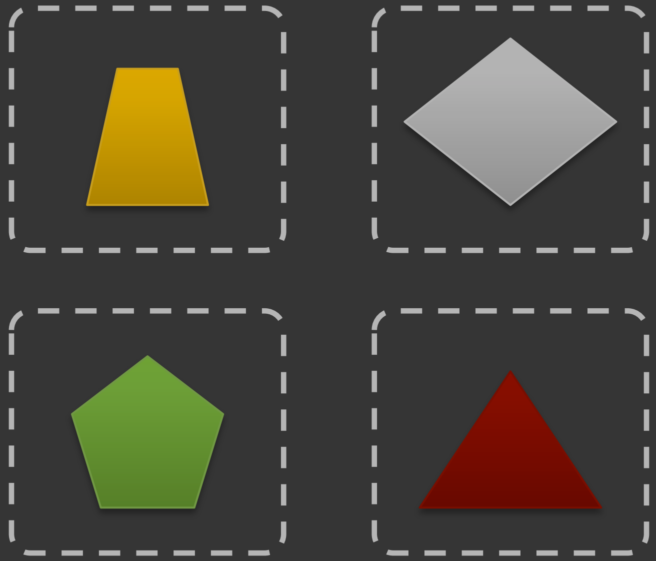
… and traditional scalability



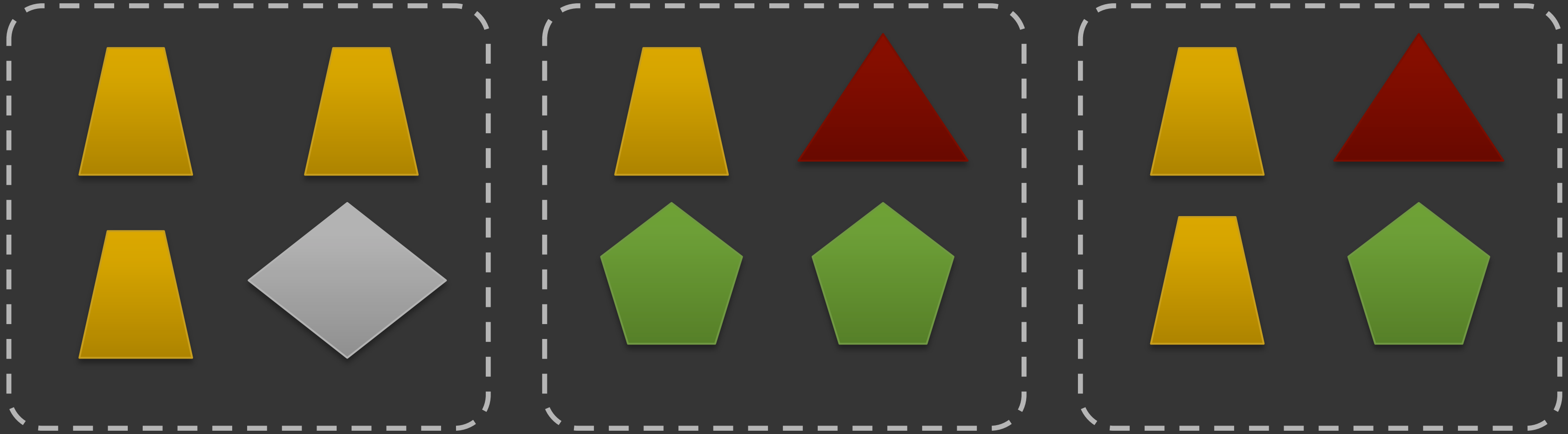
But this is not always how we build systems

A microservice

Many microservices

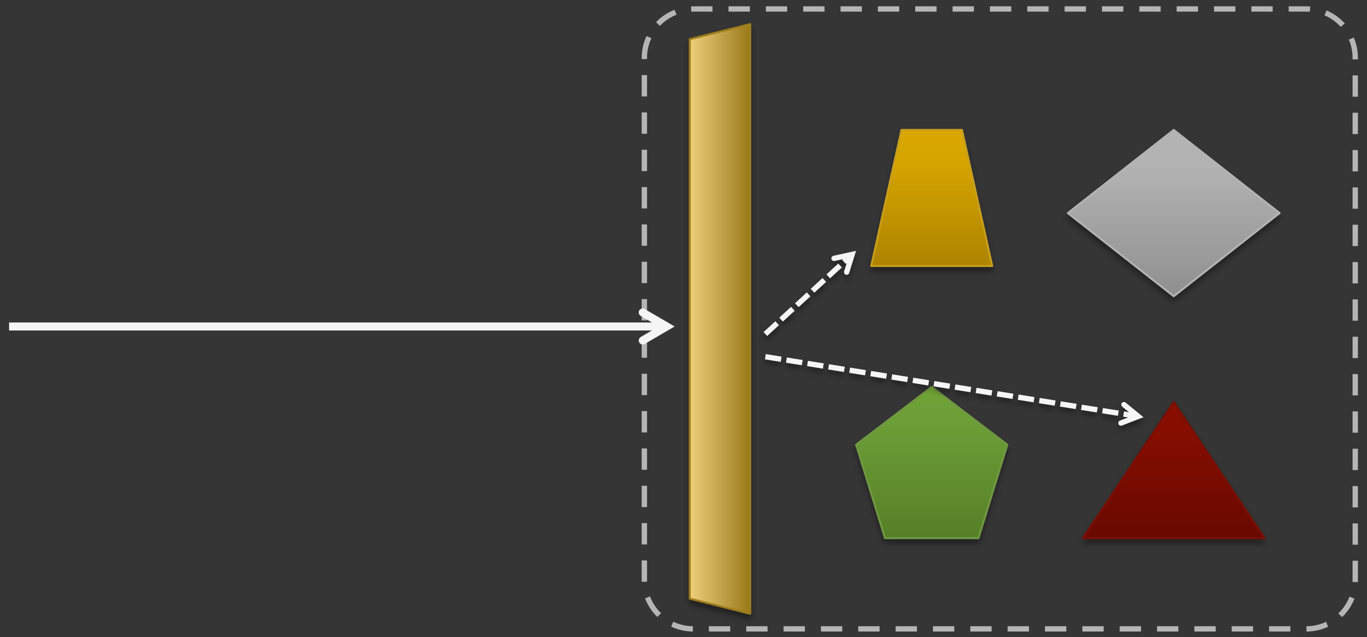


Scaling microservices

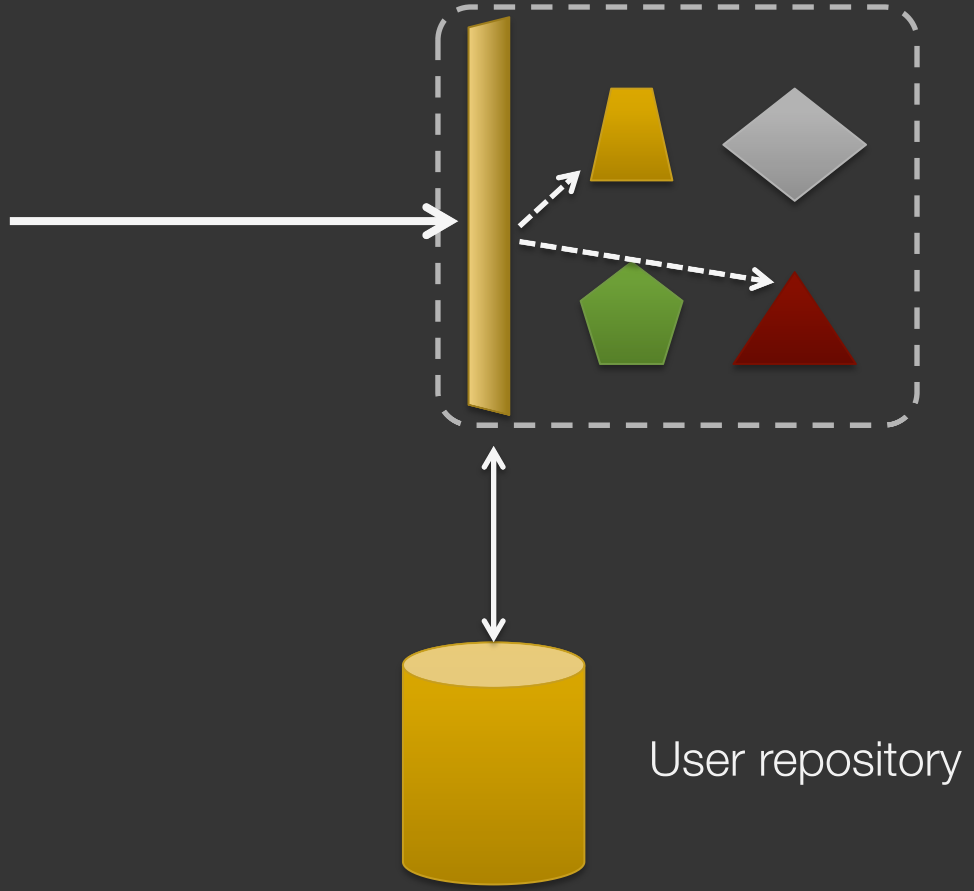


So what’s the problem?

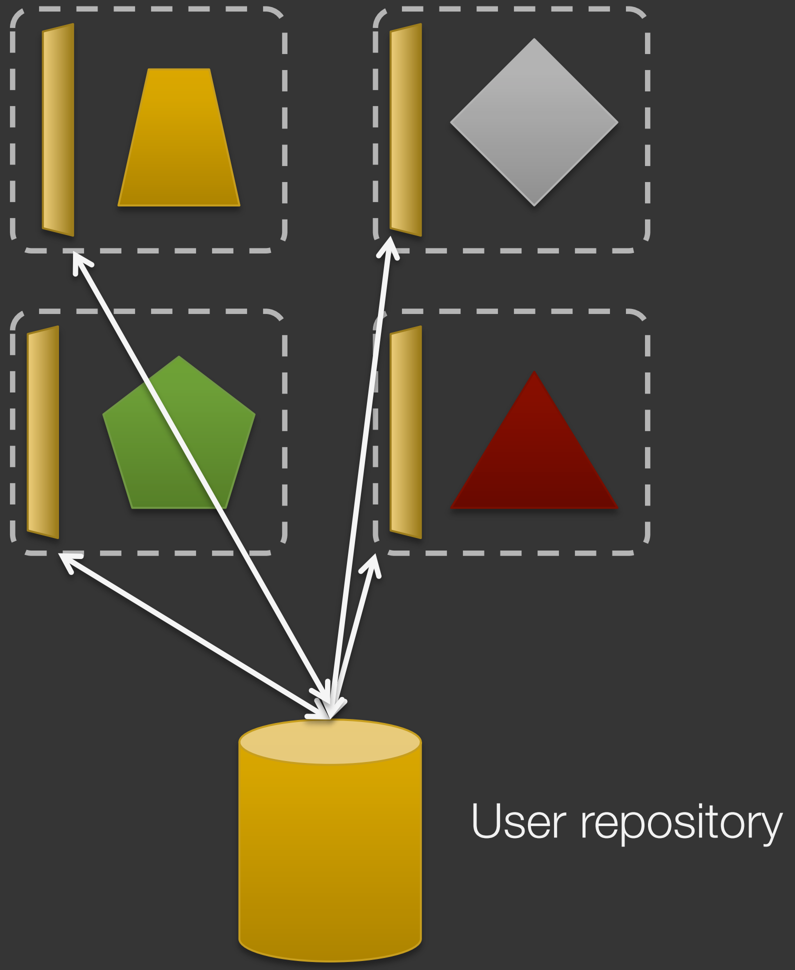
Securing a traditional service



Securing a traditional service



So for microservices that would mean



Not fantastic!

Lets talk about OAuth

It’s not for Authentication
…and not for Authorization

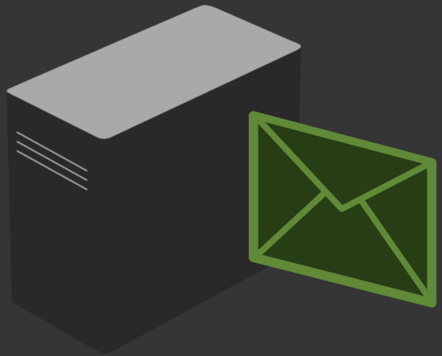
OAuth is a scalable delegation protocol

OAuth has 4 actors

Resource Owner (RO)

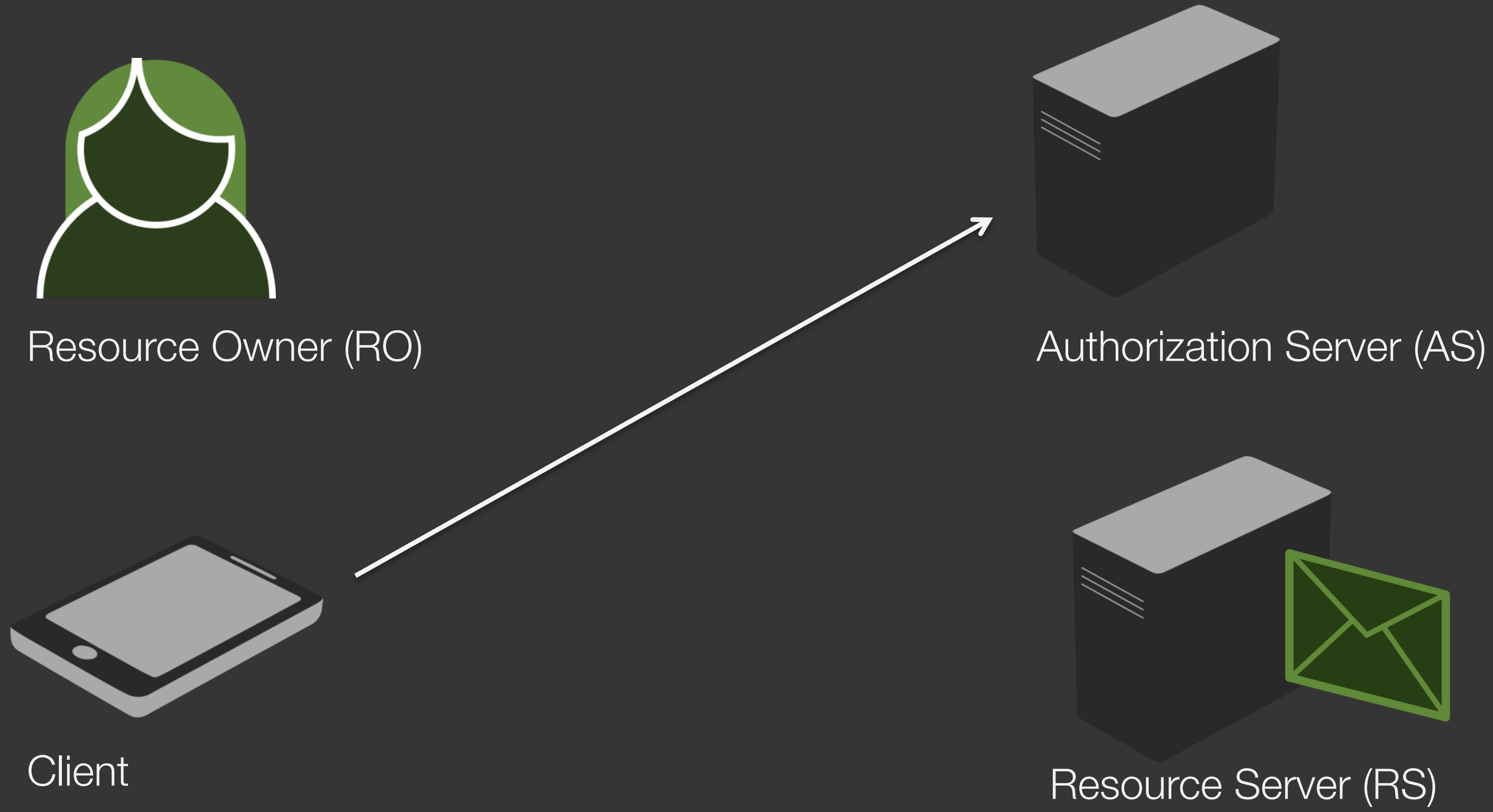
Authorization Server (AS)

Client

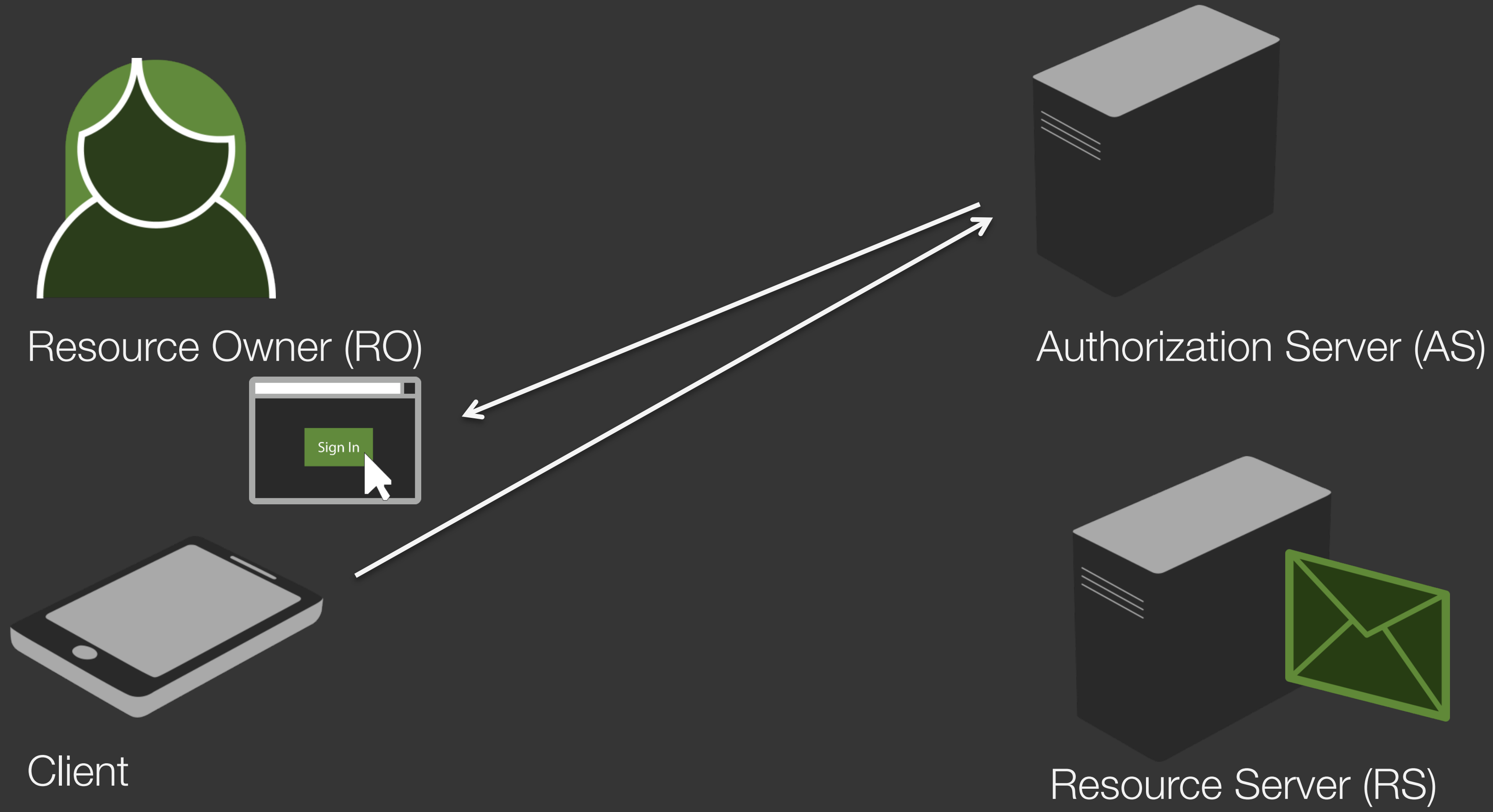


Resource Server (RS)

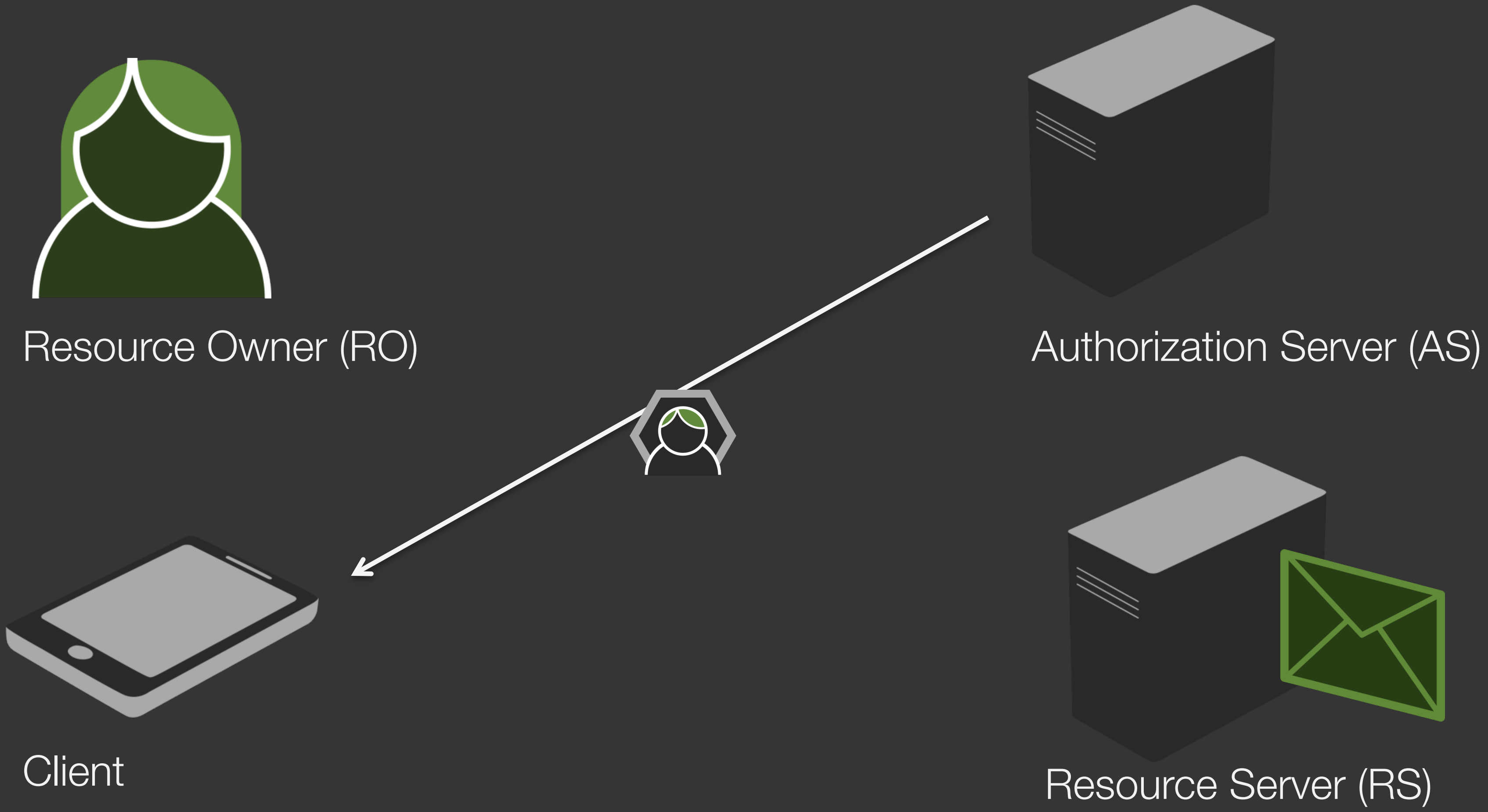
The client requests access



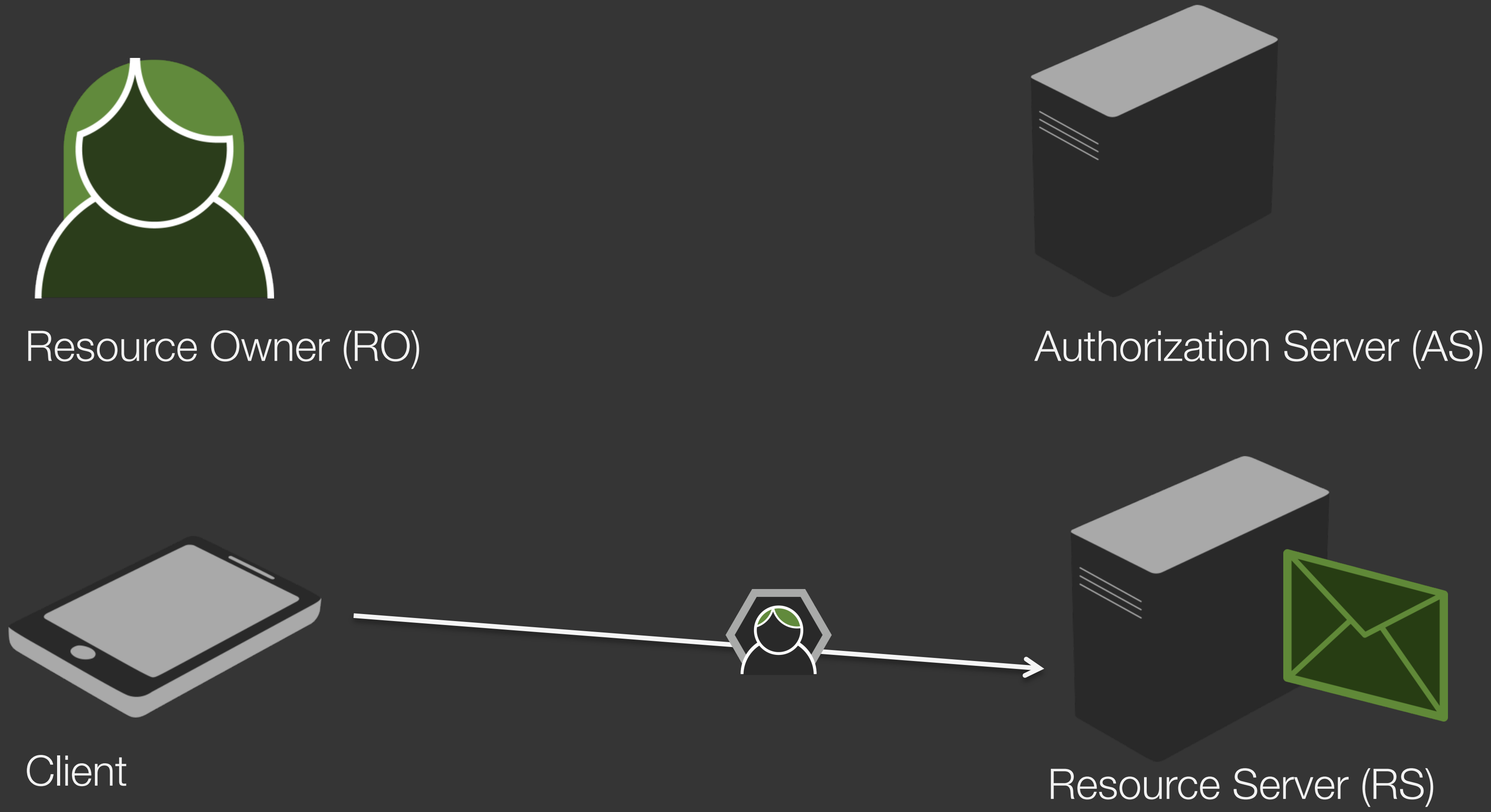
The AS requires the RO to authenticate



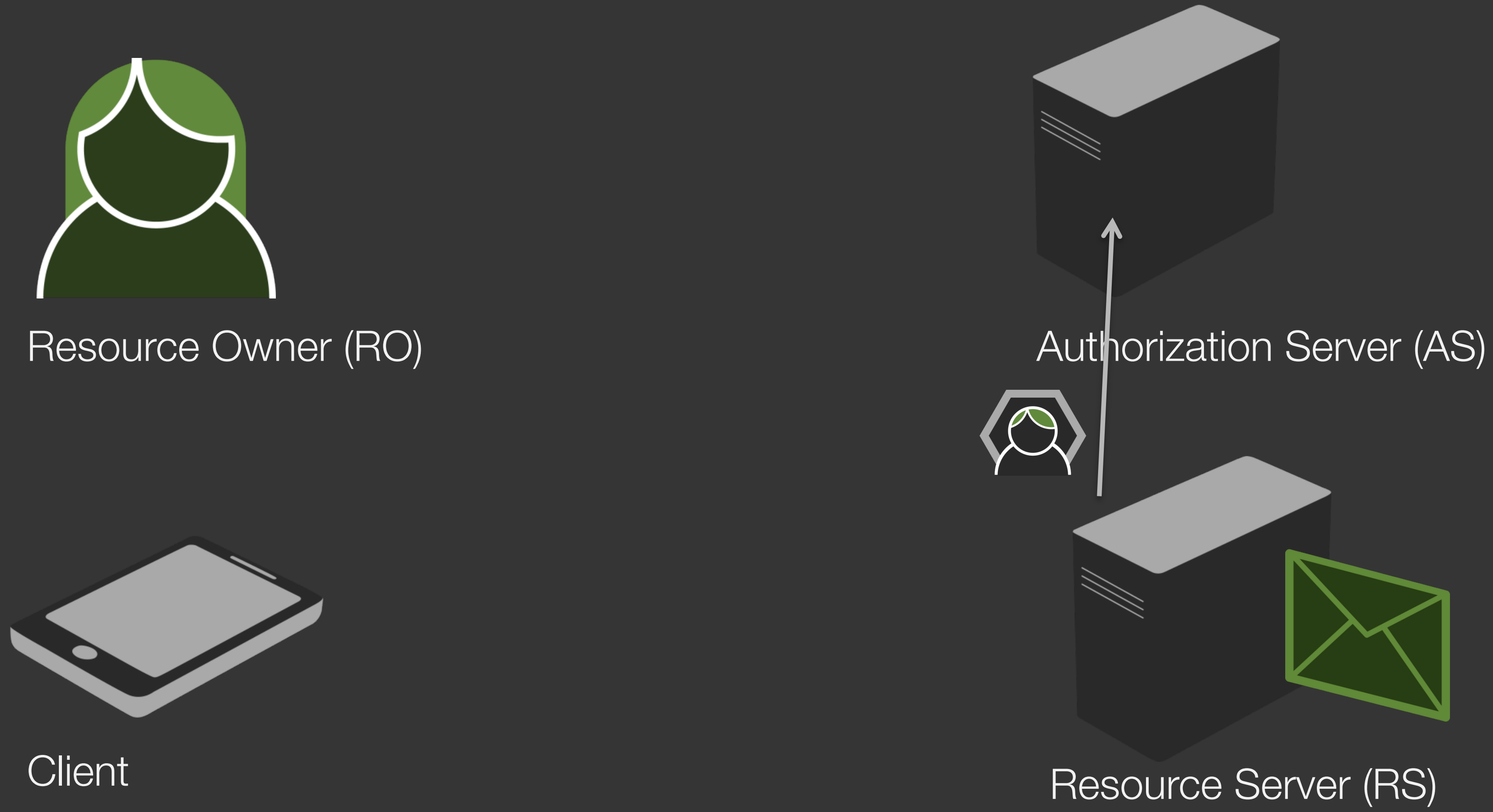
The AS issues the tokens



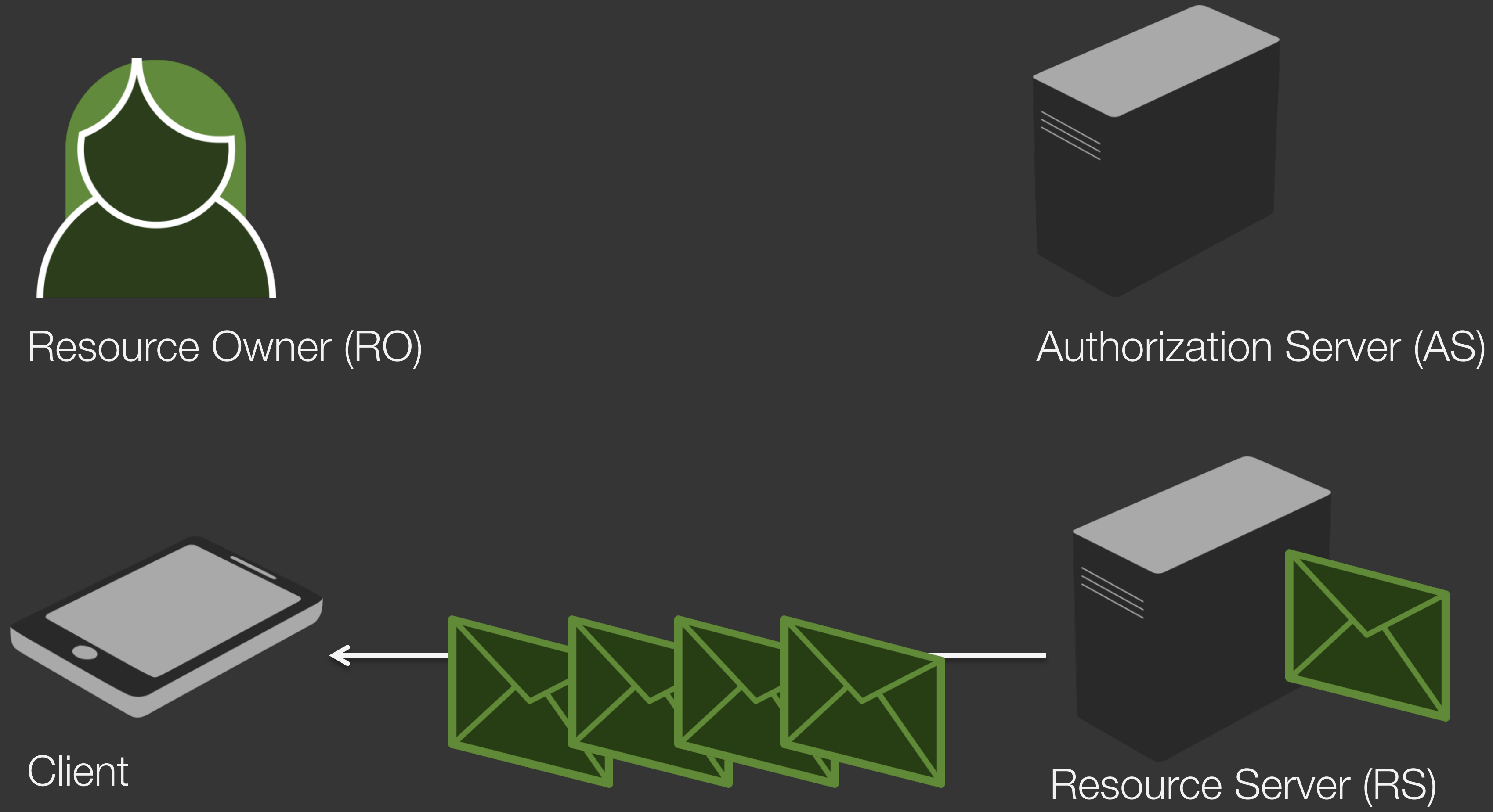
The Client presents the token to the RS



The RS validates the Token



Access!

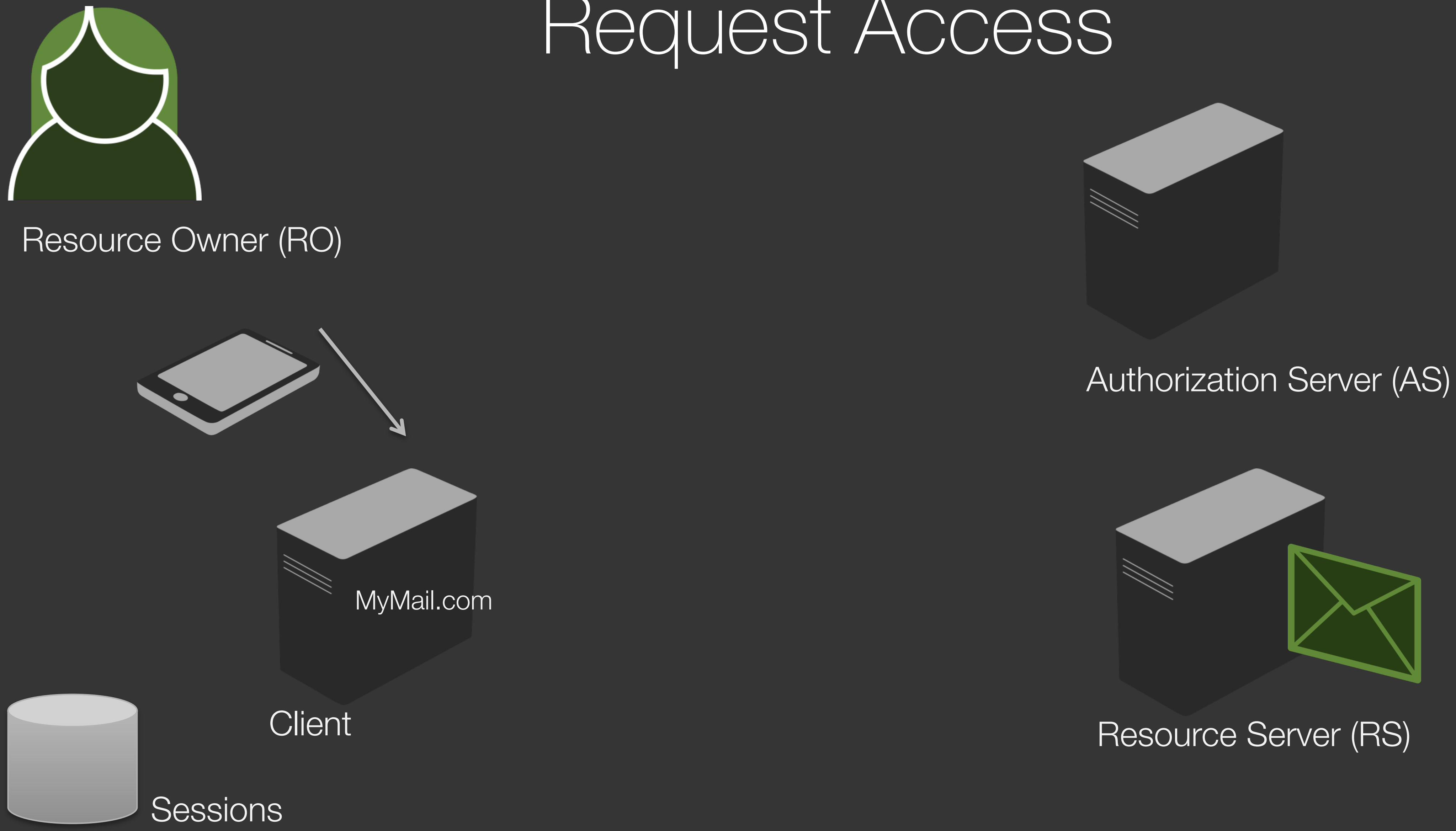

Resource Owner (RO)
Authorization Server (AS)
Client
Resource Server (RS)

One very important thing

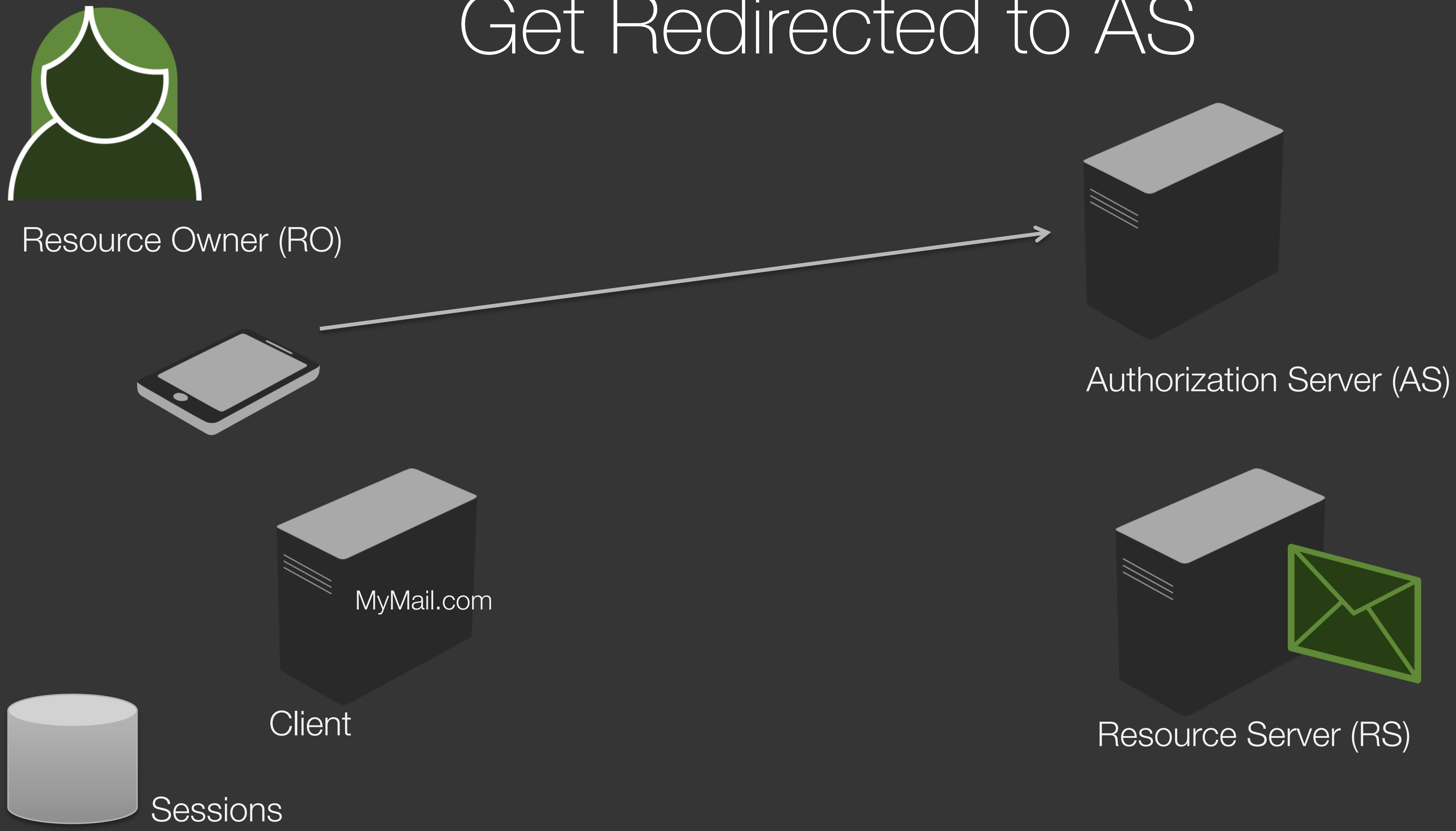
- The Client knows nothing about the user

Open ID Connect
(Simplified)

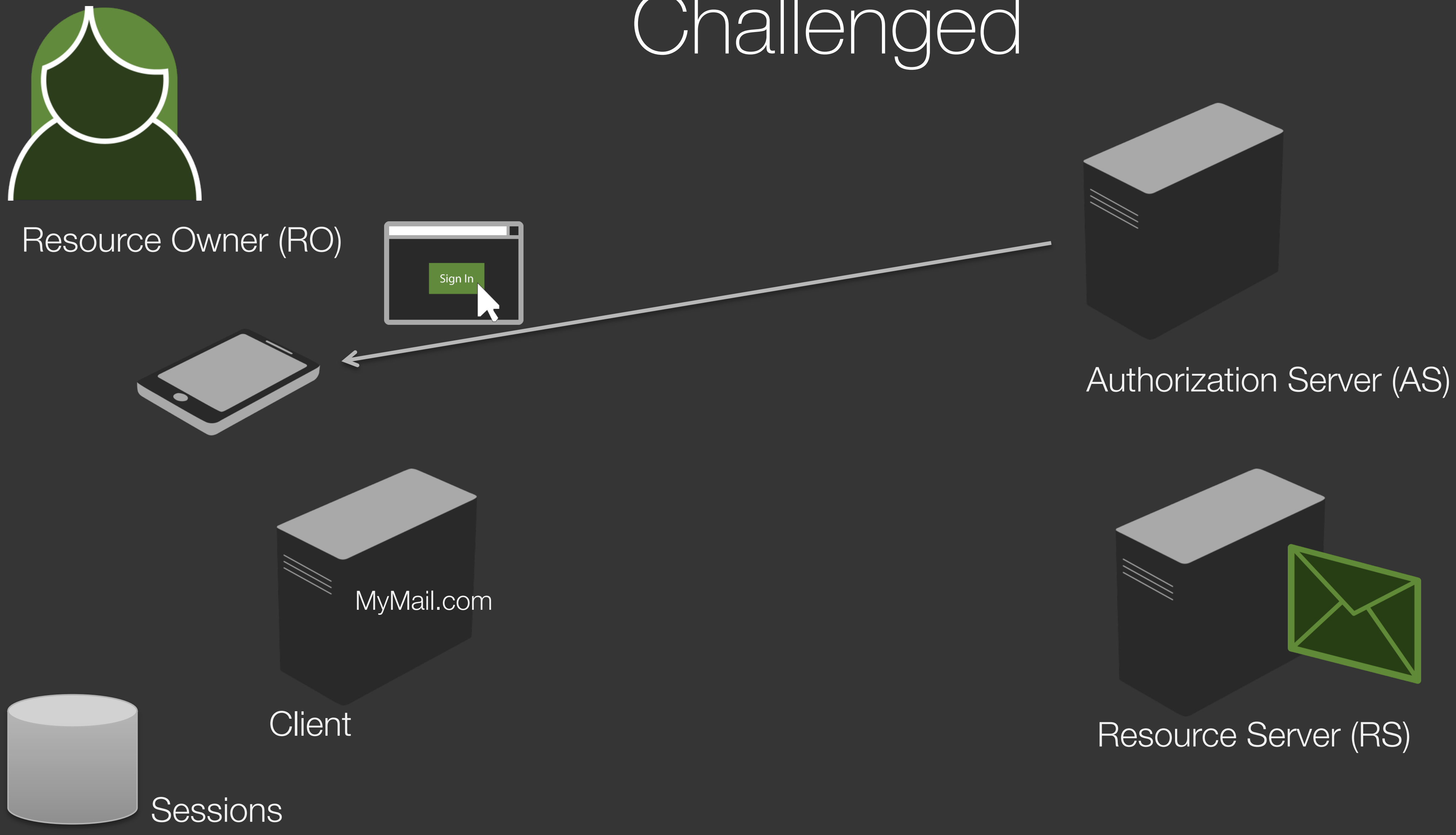
Request Access



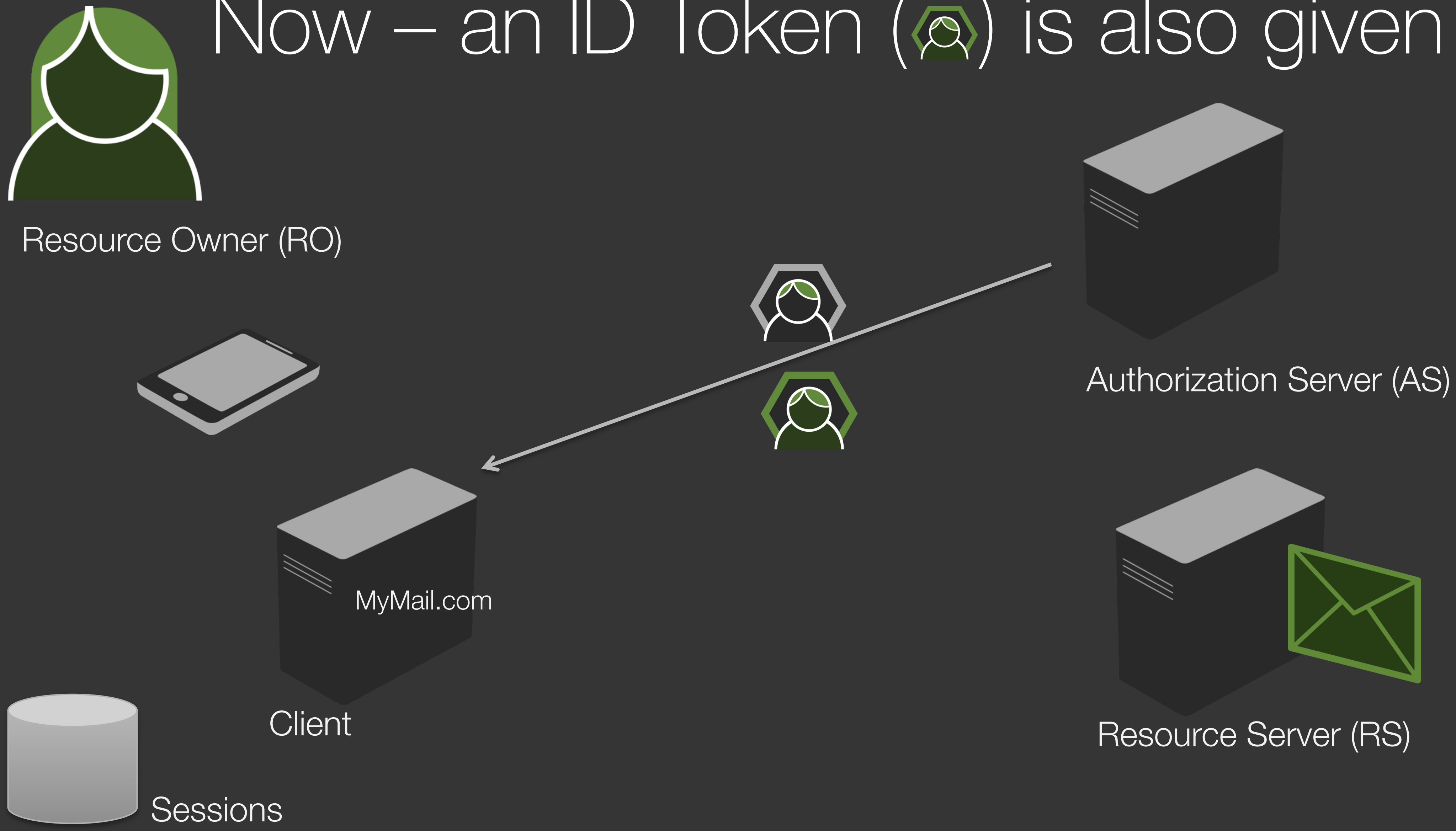
Get Redirected to AS



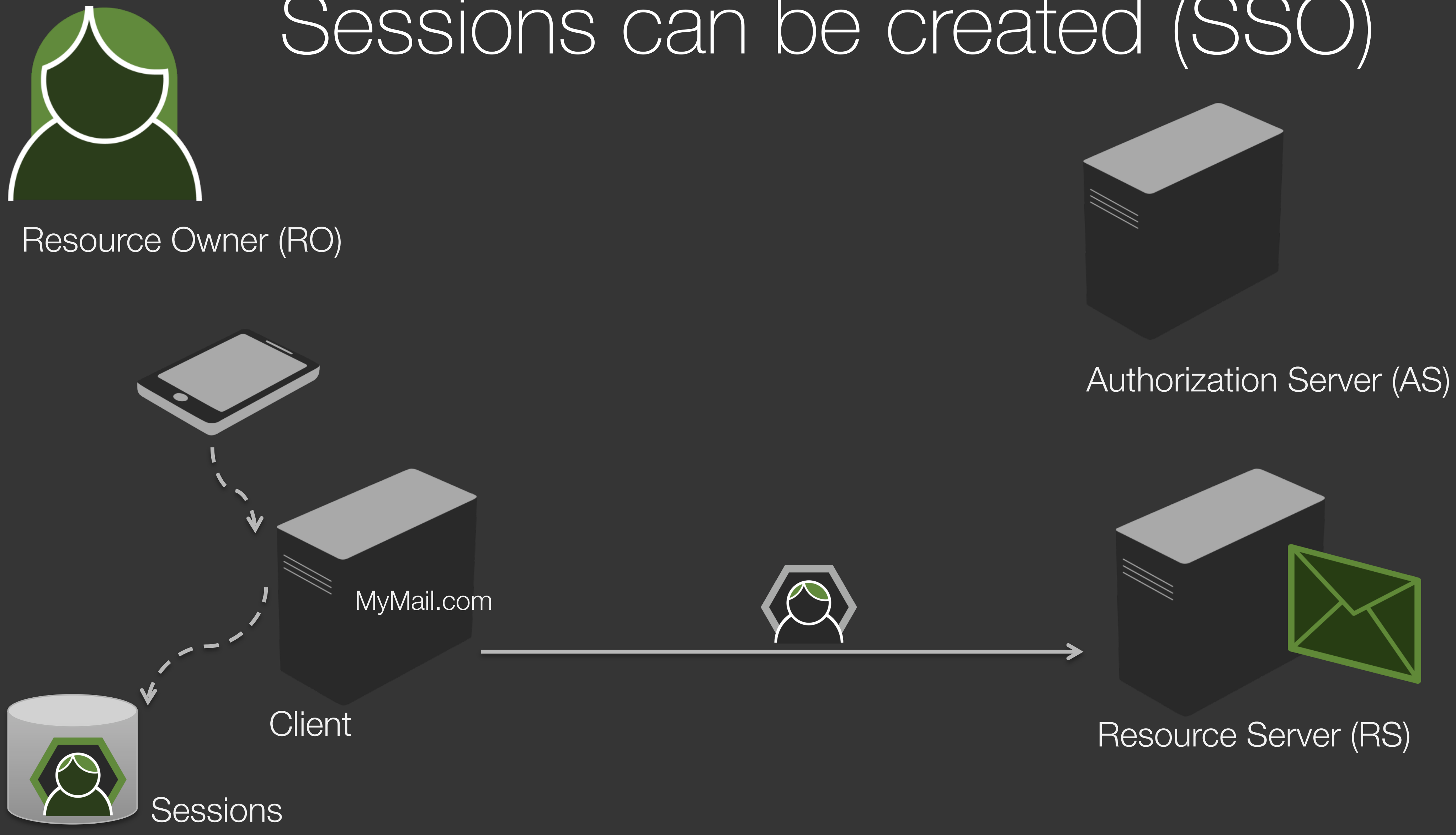
Challenged


Resource Owner (RO)
Sign In
Authorization Server (AS)
MyMail.com
Client
Resource Server (RS)
Sessions

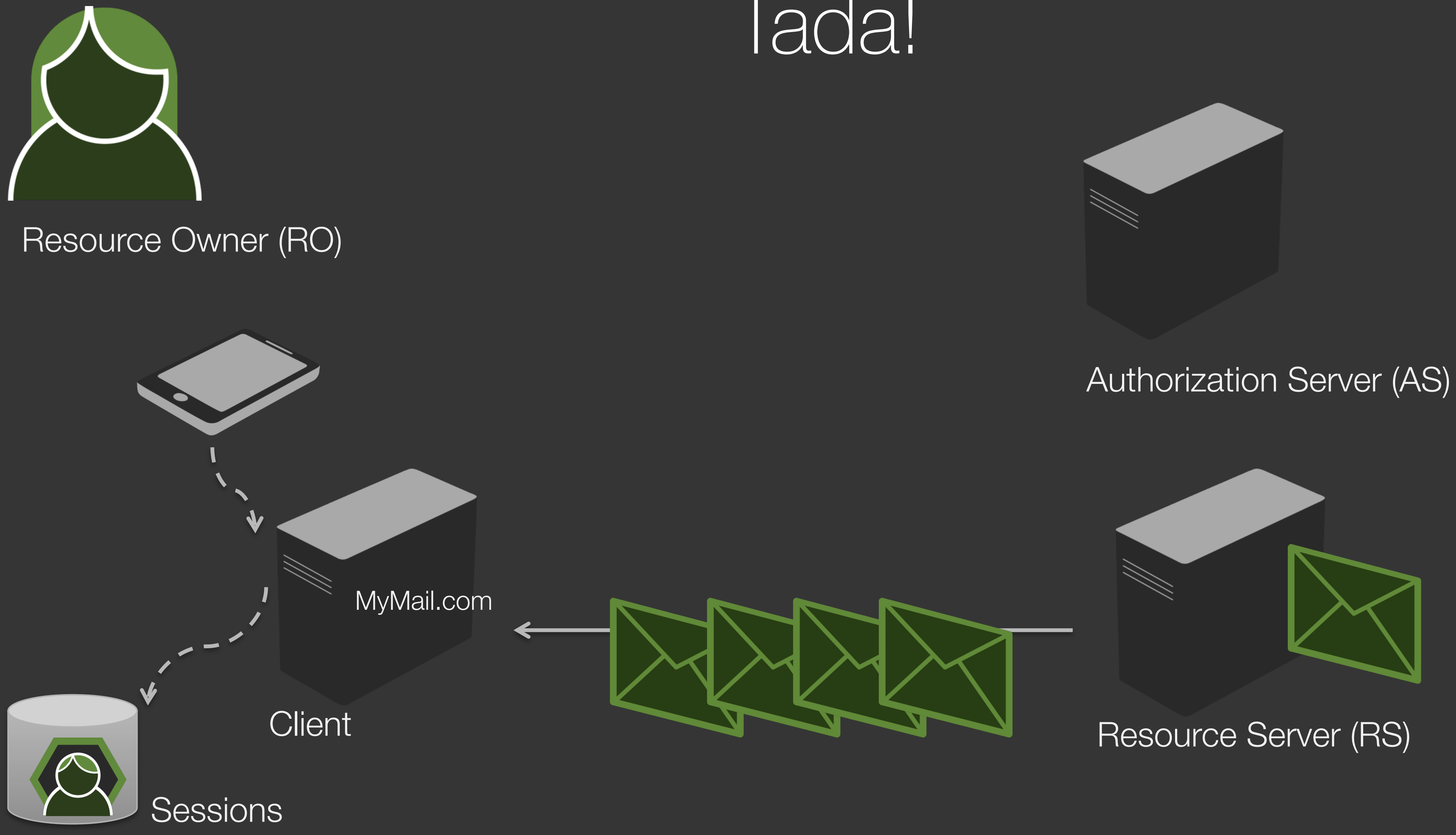
Now – an ID Token ( ) is also given



Sessions can be created (SSO)



Tada!


Resource Owner (RO)
Authorization Server (AS)
MyMail.com
Client
Resource Server (RS)
Sessions

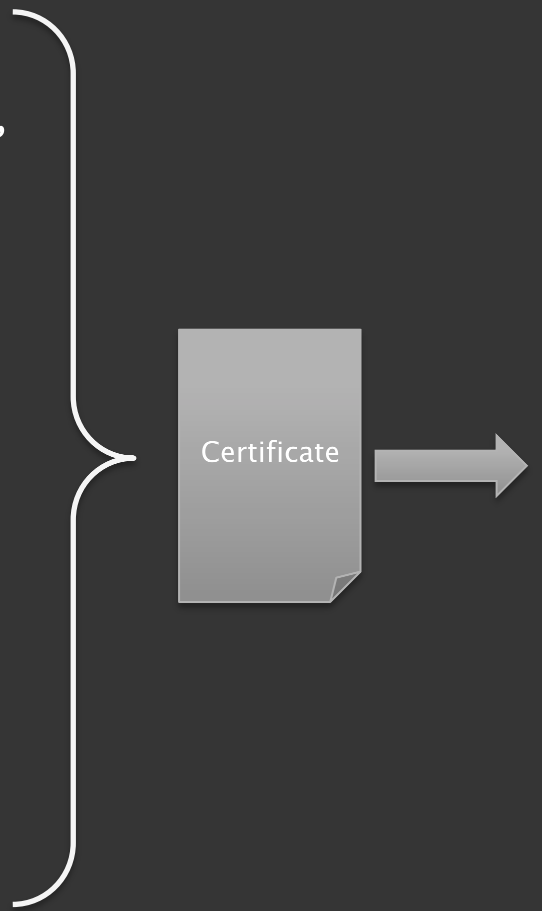
What was interesting there?

TRUST

The ID Token is a JWT (JSON Web Token)

A signed JSON document

```
{
    "iss": "https://fs.oidc.net",
    "x5t": "5F0A1359B4BB9FBB104155908DEC1FDCB5AC8865",
    "typ": "JWT",
    "alg": "RS256”
}
{
    "sub": "janedoe",
    "name" : "Jane Doe",
    "email" : "jane@doe.com",
    "phone_number" "+46 (0) 12345678",
    "aud": "https://mymail.com",
    "iss": "https://fs.oidc.net",
    "nbf": 1409213888783,
    "jti": "622a9973-fc4d-4797-be31-7c2116f549df",
    "exp": 1409213890583,
    "iat": 1409213888783
}
```



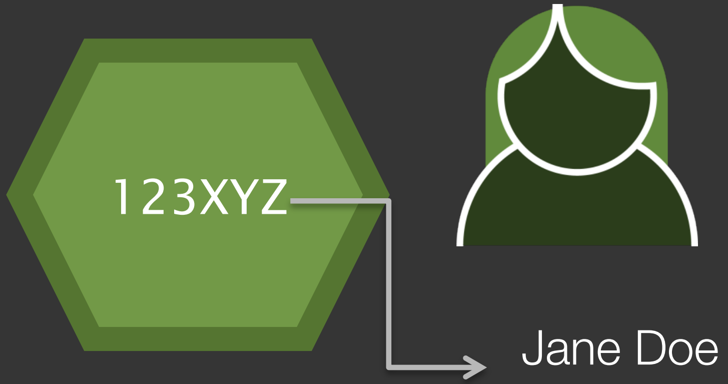
Signature

orQOOKvXN3jbEpBSl0RHAyaQNxcx9DFgtMsJJgMxm9Az6QJMKKy6m0
WvP1UzXZA_nsK16g9etg2yEW9IXbQU0RbSQktUtObRB9SxHtW_AcCk6
93XDAz15Y4aP9DeD62nROzd1MS4FZTmY3Cgzo1-3-
sqW6_4Rgzs94aLO3aLP_zoVtJycCUKtJQhGhPTyjXXYWMsp0E4uTtL8Ri
f7cWu4olme_XNFlAs73pOrfzsQYc1GD2dB70l1M8SDaJZFURr9jAAaavX
7Xqs_FPXY1PZLXLbc3ARXFmRf_-
Z4B6uLCGI2shzl12ni54Yun6dflL9rQwaxXYuNZZodUWchID2cA

OAuth Access Tokens can also be JWTs

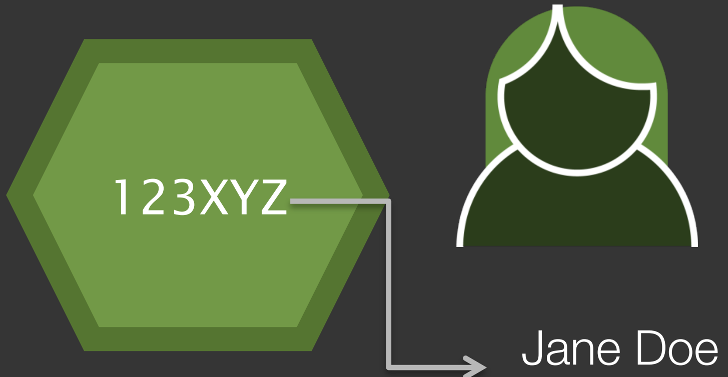
2 types of tokens

By Value



By Reference

By Reference

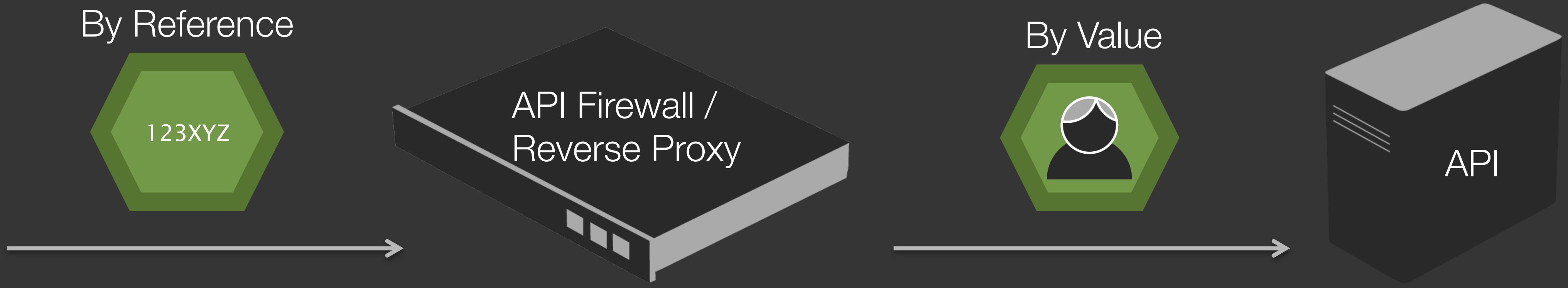


Contains NO information outside the network

By Value

Contains ALL necessary information

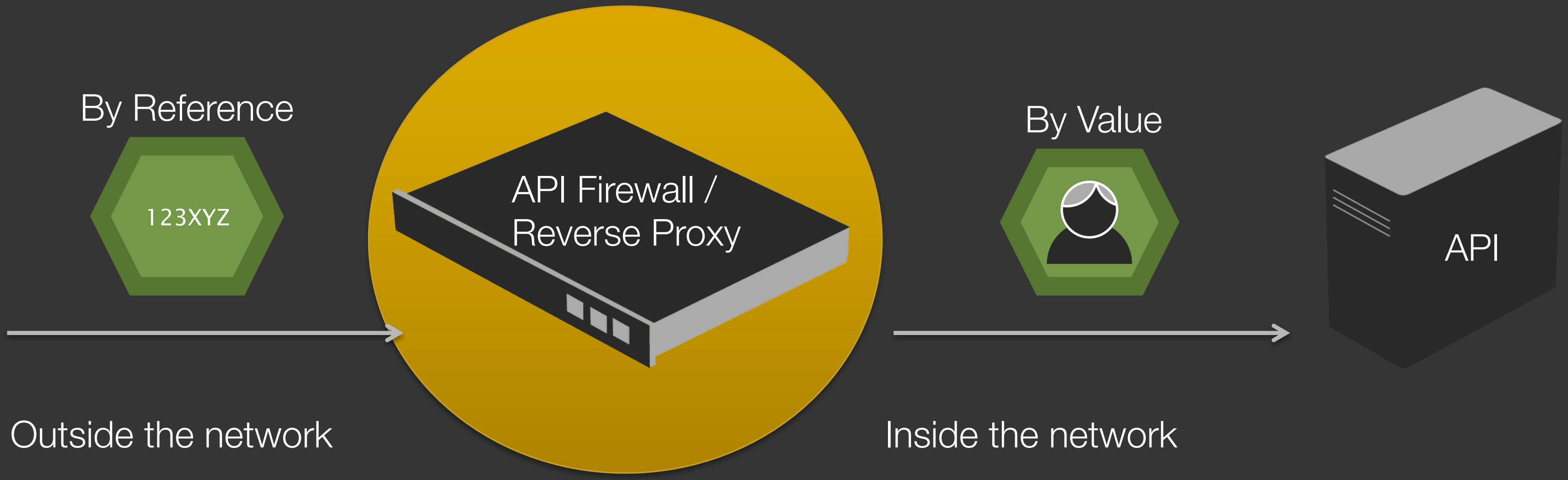
External vs. Internal



Outside the network

Inside the network

Token Translation

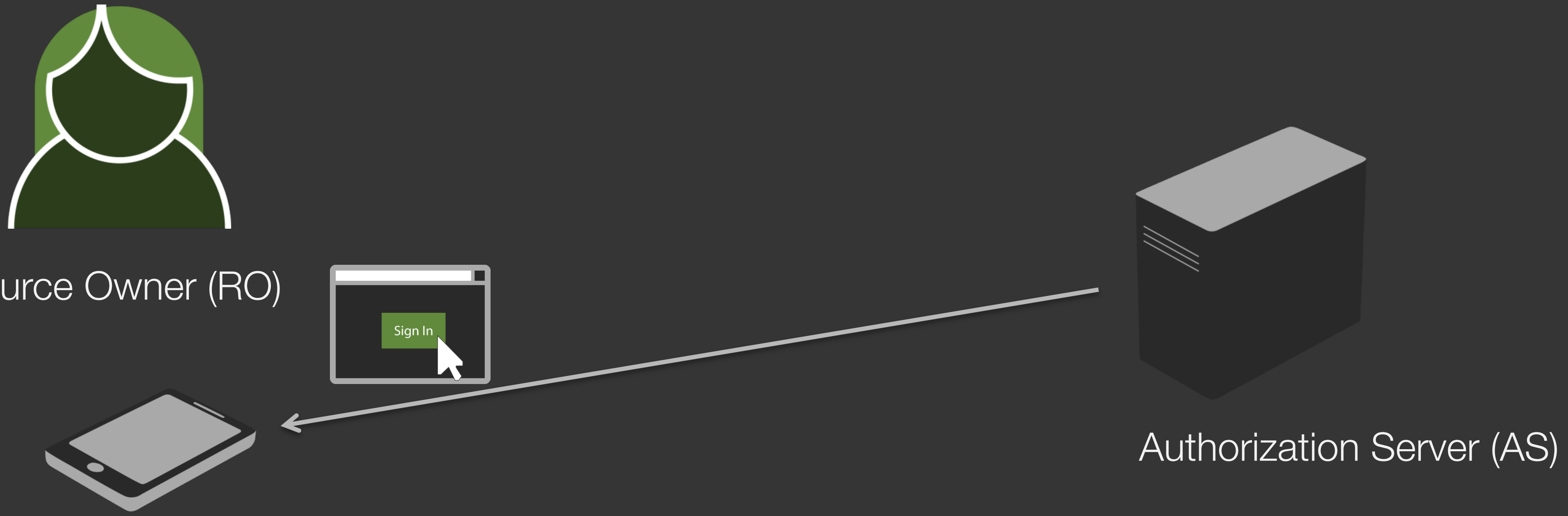


Back to Microservices

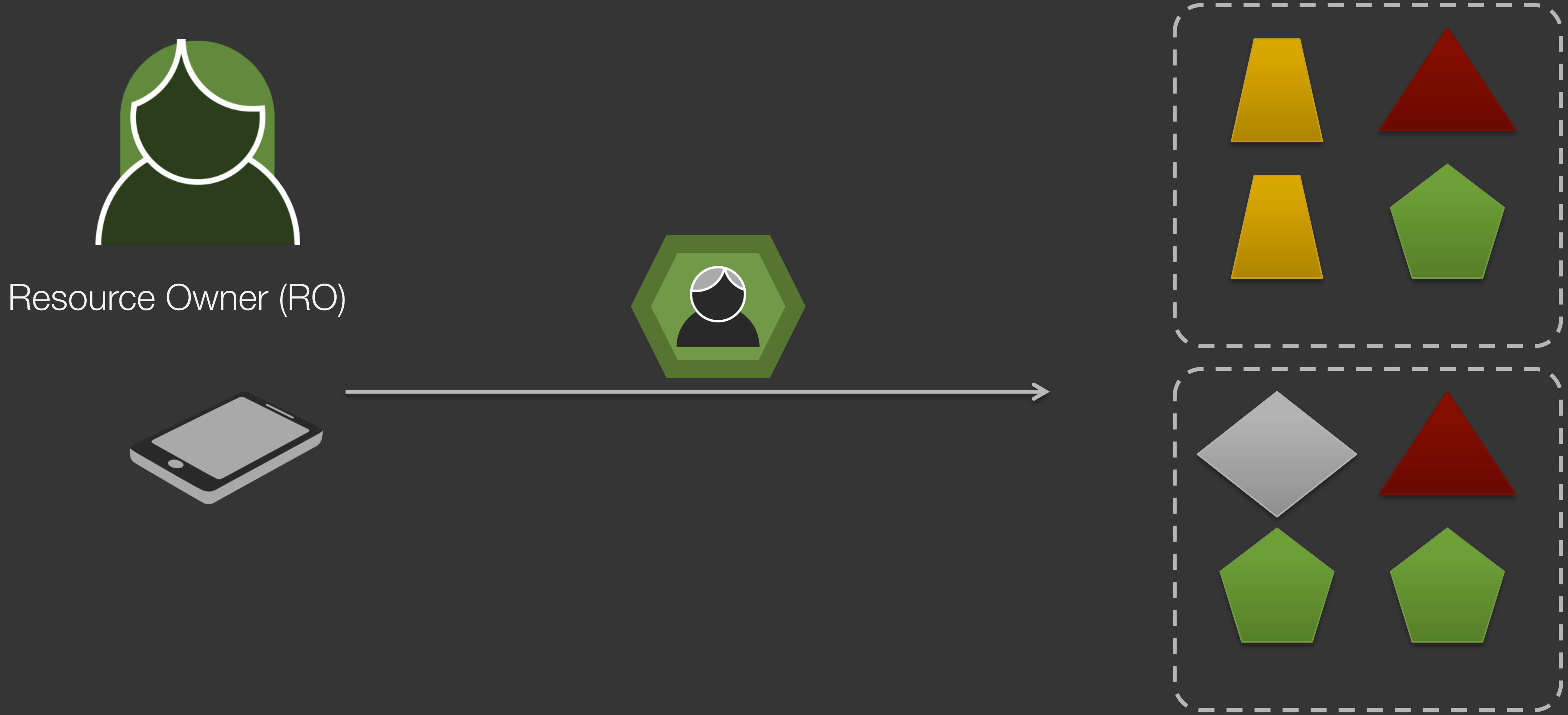
2 Problems

- Identifying the user
- Creating sessions

Leave authentication to the OAuth/OIDC server



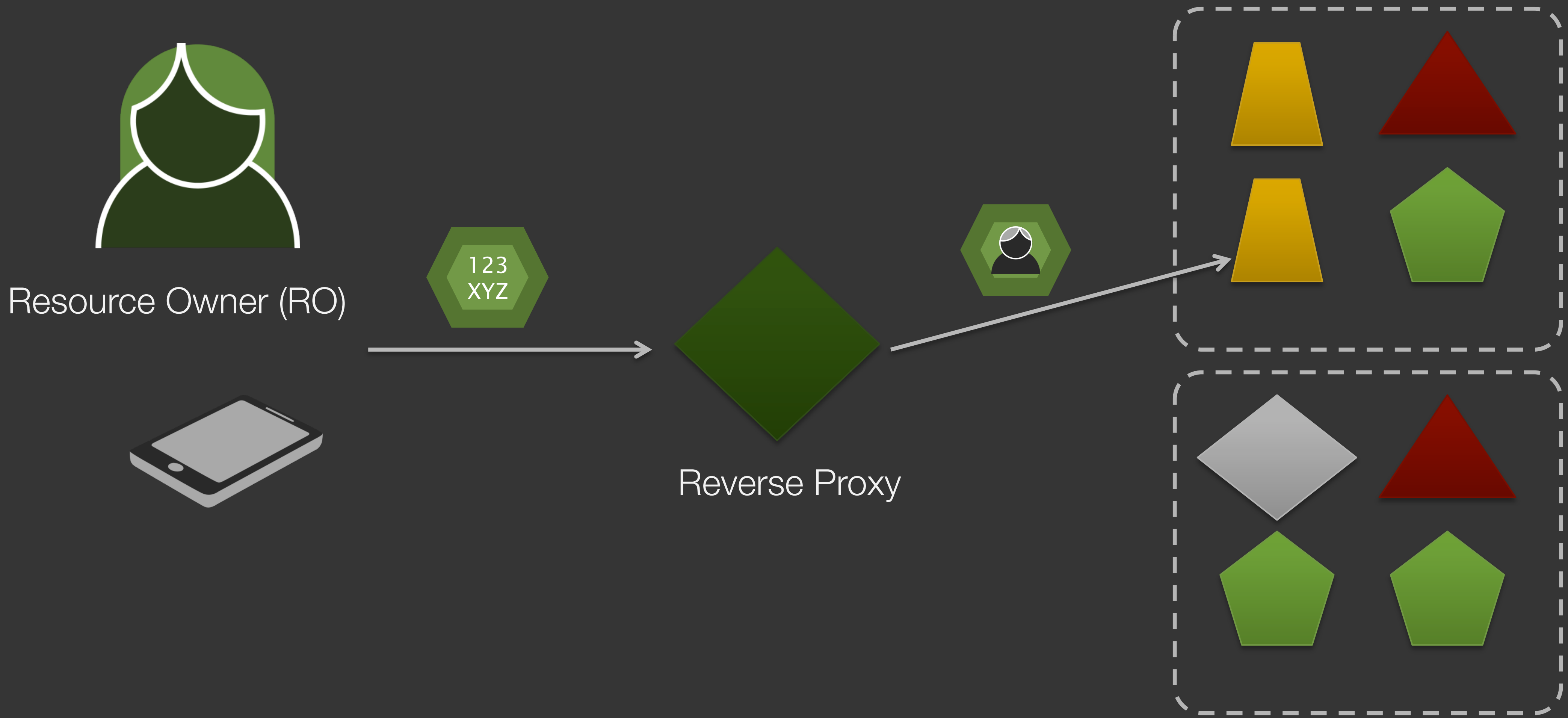
Let all Microservices accept JWTs



BUT…

Translate!

Let all Microservices accept JWTs



Conclusion

- everything is self contained
- standards based
- non-reputable
- scalable

Thank you!