SECURE APPS WITH OAUTH & OIDC

ANDREAS TELL

CADEC 2020.01.23 & 2020.01.29 | CALLISTAENTERPRISE.SE

CALLISTA

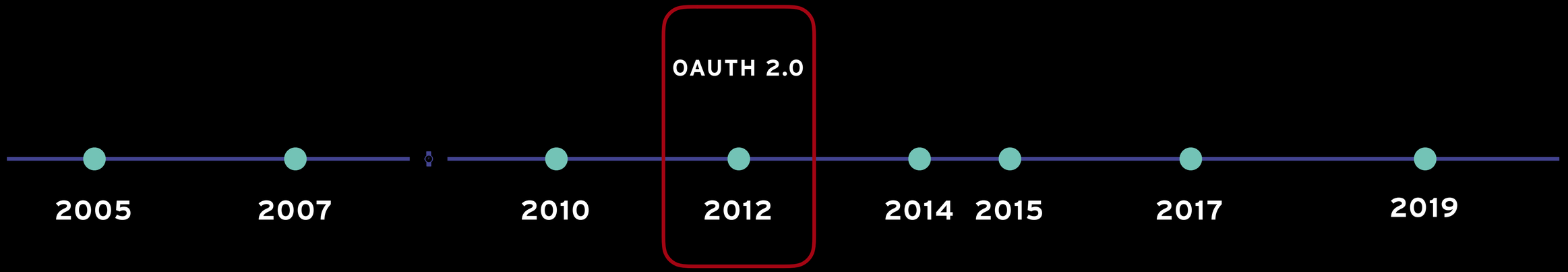
AGENDA

- Intro
  - Background
- OAuth/OIDC - Technology overview
  - Demo
- Best Current Practice anno 2020 for securing Apps* with OAuth/OIDC
- Wrap Up

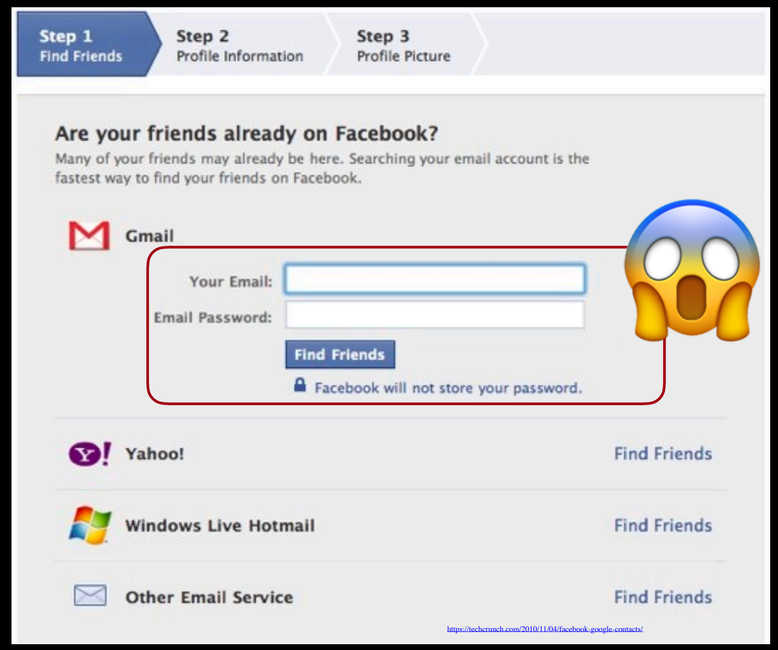
* JavaScript Single Page Apps (SPA) and Mobile Apps

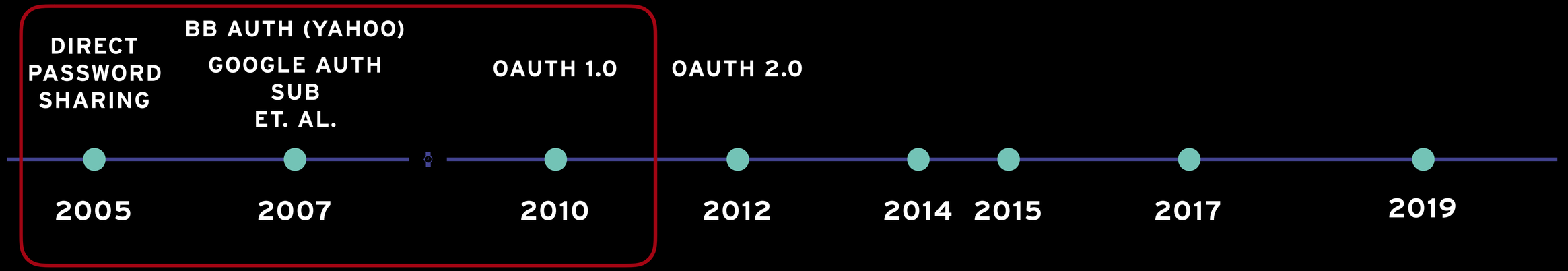
INTRO : WHAT IS OAUTH (2.0)?

- RFC 6749 - Internet Engineering Task Force (IETF)
- An extensible framework for *Access Delegation*
  - » "the one that controls a resource grants access to a software application to do something with that resource on their behalf"
  - Is **not** for authentication
  - Does **not define authorization per se** (as in access control) - specifies the *delegation* of authorization information



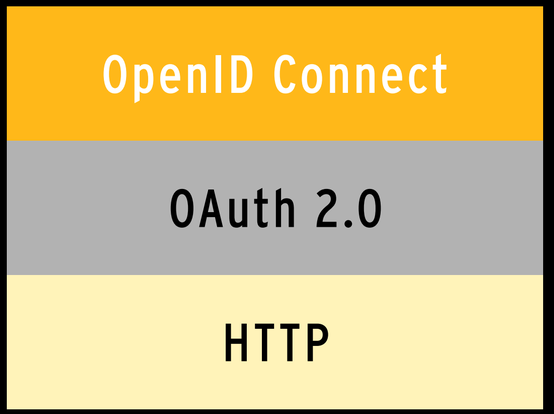
INTRO: BACKGROUND - PRIMARY USE CASE

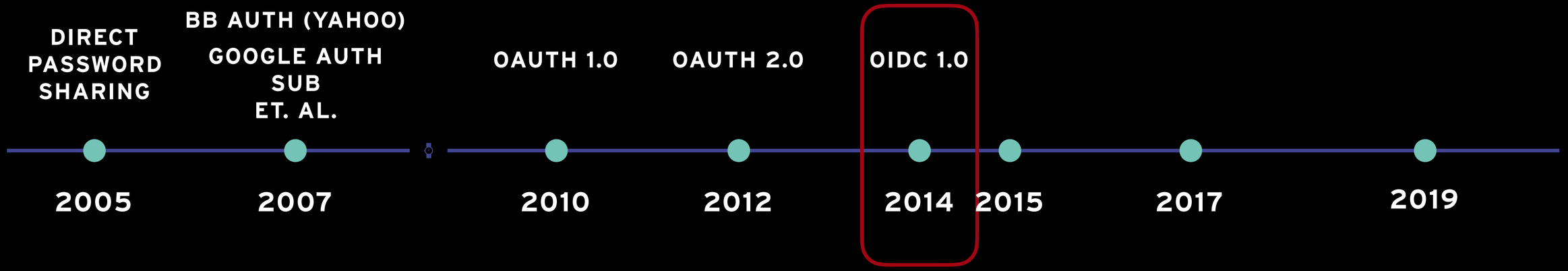




INTRO

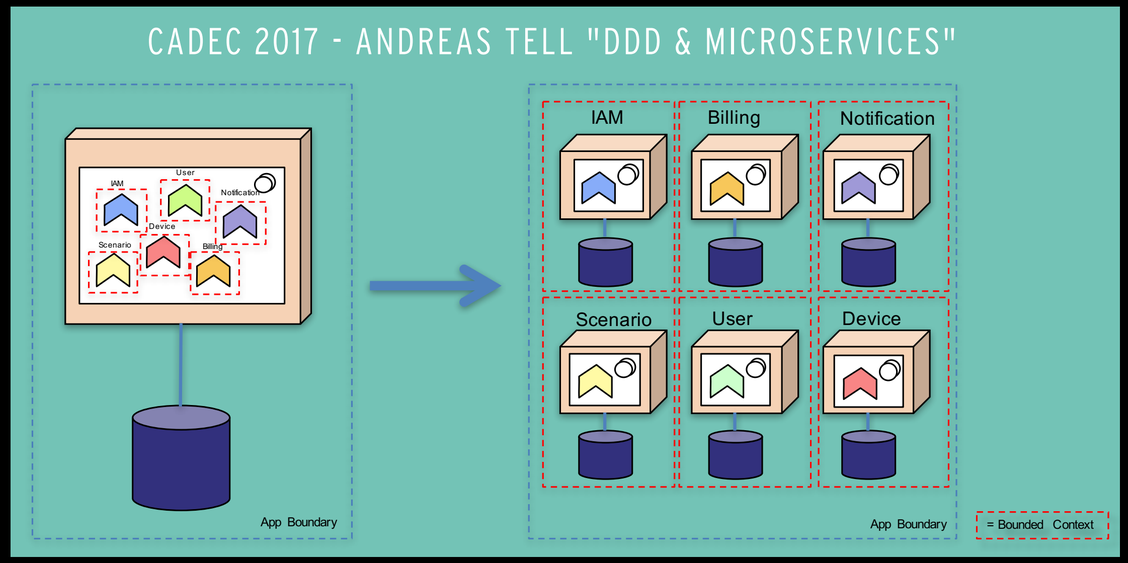
- OpenID Connect - OIDC
  - A simple identity layer on top of OAuth
  - Built on the process flows of OAuth with some extensions added for authentication
  - Adds standard vocabulary (i.e. parameters) and endpoints
  - User identity is encoded in an "ID Token"





APPLICABILITY - WHEN USE IT?

- HTTP and user-agent centric
- Developed for apps and APIs
  - Also for first-party apps
- A very god fit for distributed apps (Mini/Micro Services)
  - Centralized Identity and Access Mgmt (IAM)

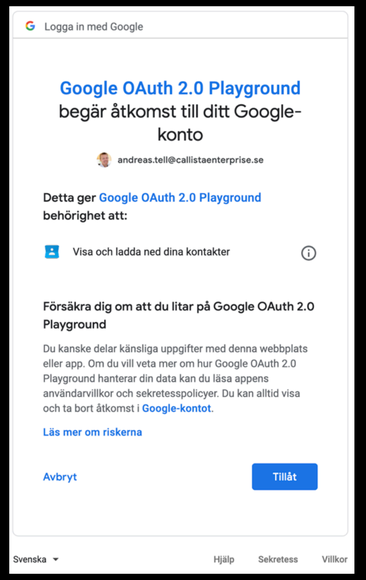


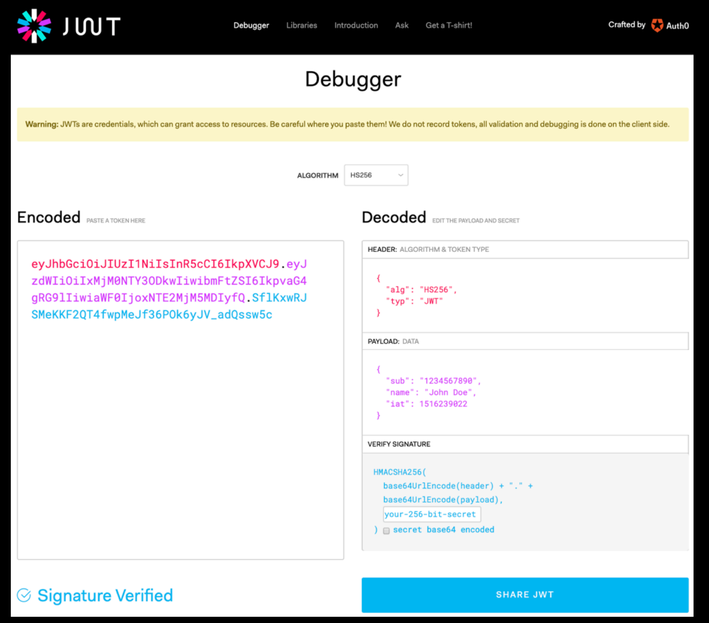
TECHNOLOGY OVERVIEW

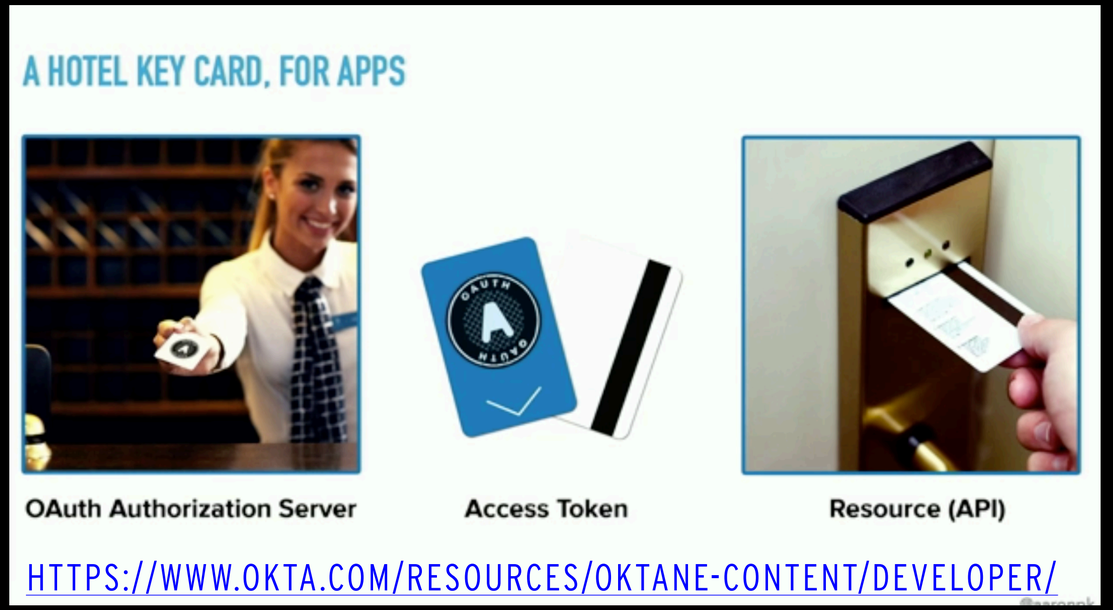
- OAuth
- OIDC
- Direct Password Sharing
- ID Token
- Authorization
- Authentication
- Delegated Authorization
- Access Token
- Refresh Token
- Resource Owner
- Resource Server
- Auth. Server
- OpenID Provider (OP)
- Client
- Relying Party (RP)
- Public vs Confidential Client
- Client Id
- Client Registration
- Grant Types
- Authorization Code Flow
- Authorization Code
- Implicit Flow
- Front channel
- Backchannel
- Redirect
- JWT
- Signature Validation
- Claims
- Opaque Token
- Reference Token
- First-party authorization
- Third-party authorization
- Client Impersonation
- Token Interception
- Auth Cookie
- CORS
- PKCE
- SHA256
- Code Challenge
- State parameter
- Nonce
- Social Login
- MFA
- Token Replay
- Client Credentials Flow
- Custom URI Scheme
- User Store
- User-Info Endpoint
- Subject Identifier
- Same Site Cookies
- HTTP Only Cookies
- Sender Constrained Tokens
- CSRF
- Poached easter eggs
- XSS
- Replay Attack
- TLS
- mTLS
- Token Leakage
- ALG claim
- Signature
- Well known URI endpoint
- SSO
- Device Flow
- Resource Owner Password Flow
- Refresh Token Flow

TECHNOLOGY OVERVIEW - SCOPE, CONSENTS & TOKENS

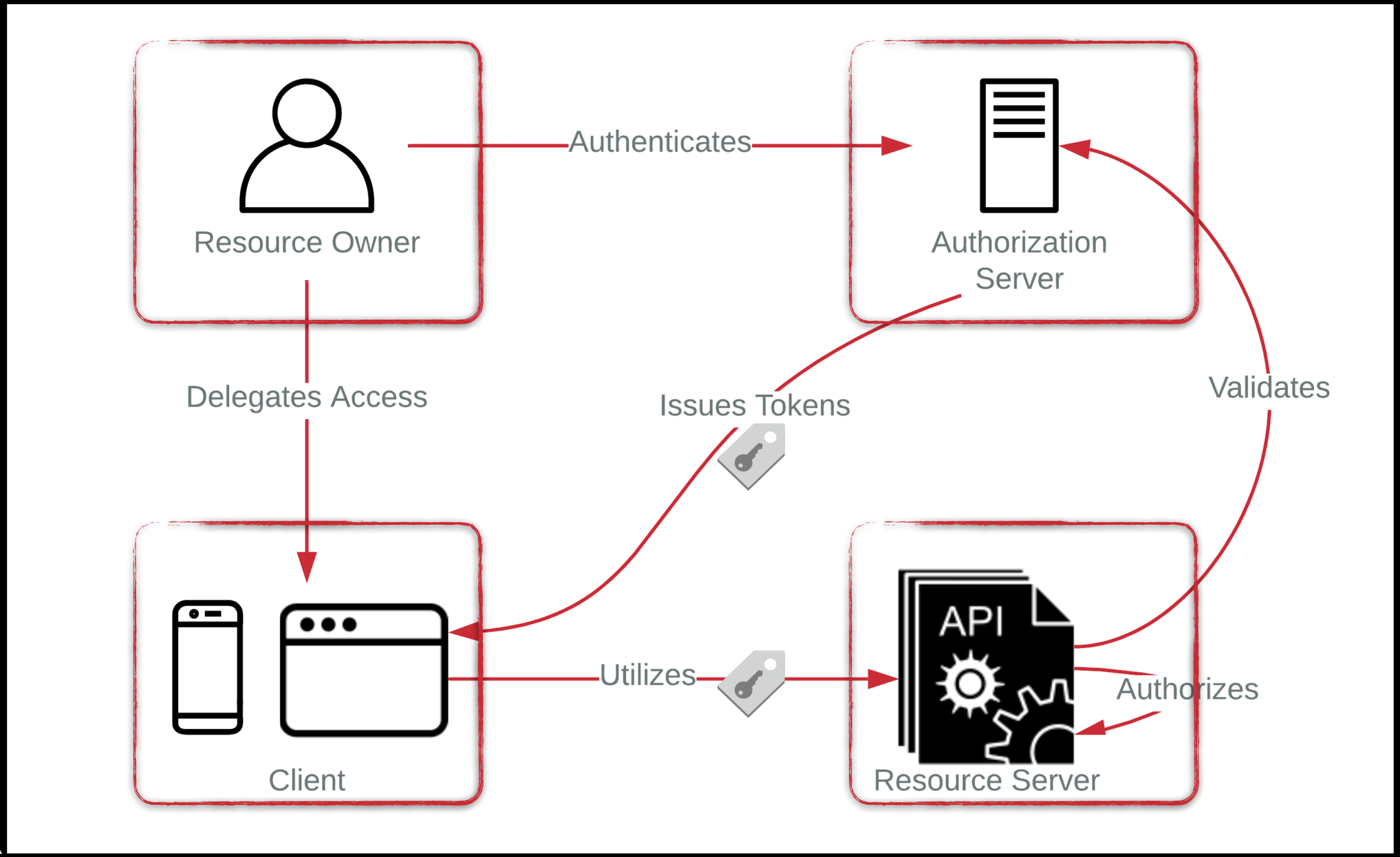
- Scope
- Consent
- Tokens
  - Access Token
  - ID Token (OIDC)
  - Refresh Token
  - Limited lifespan
  - Opaque vs. self-contained







TECHNOLOGY OVERVIEW - OAUTH2 ROLES/ACTORS



TECHNOLOGY OVERVIEW - GRANT TYPES A.K.A "AUTH FLOWS"

Implicit Grant

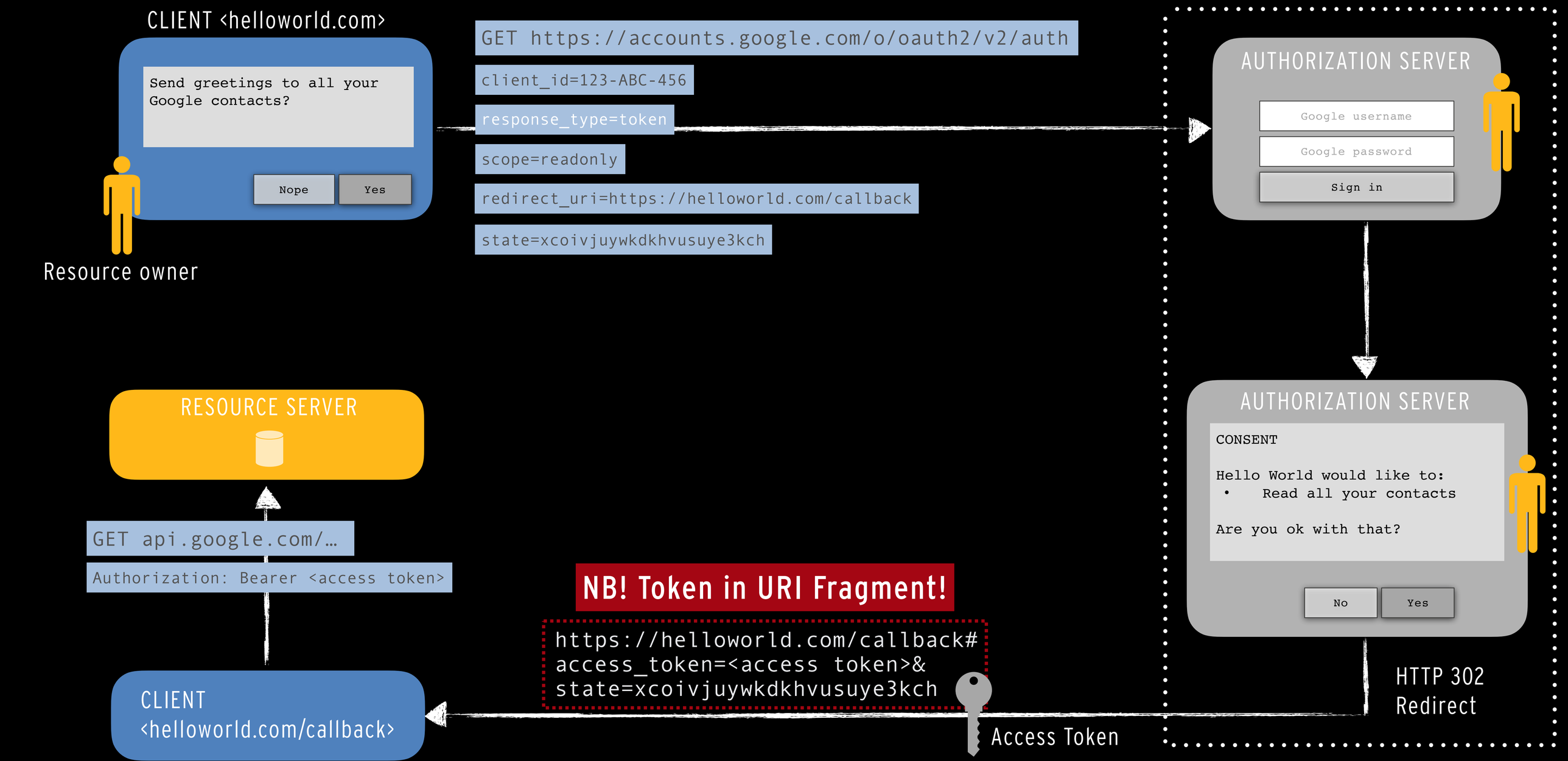
Authorization Code Grant

Client Credentials Grant

Resource Owner Password Credentials Grant

Others: Refresh Flow, Device Flow

TECHNOLOGY OVERVIEW - IMPLICIT GRANT FLOW



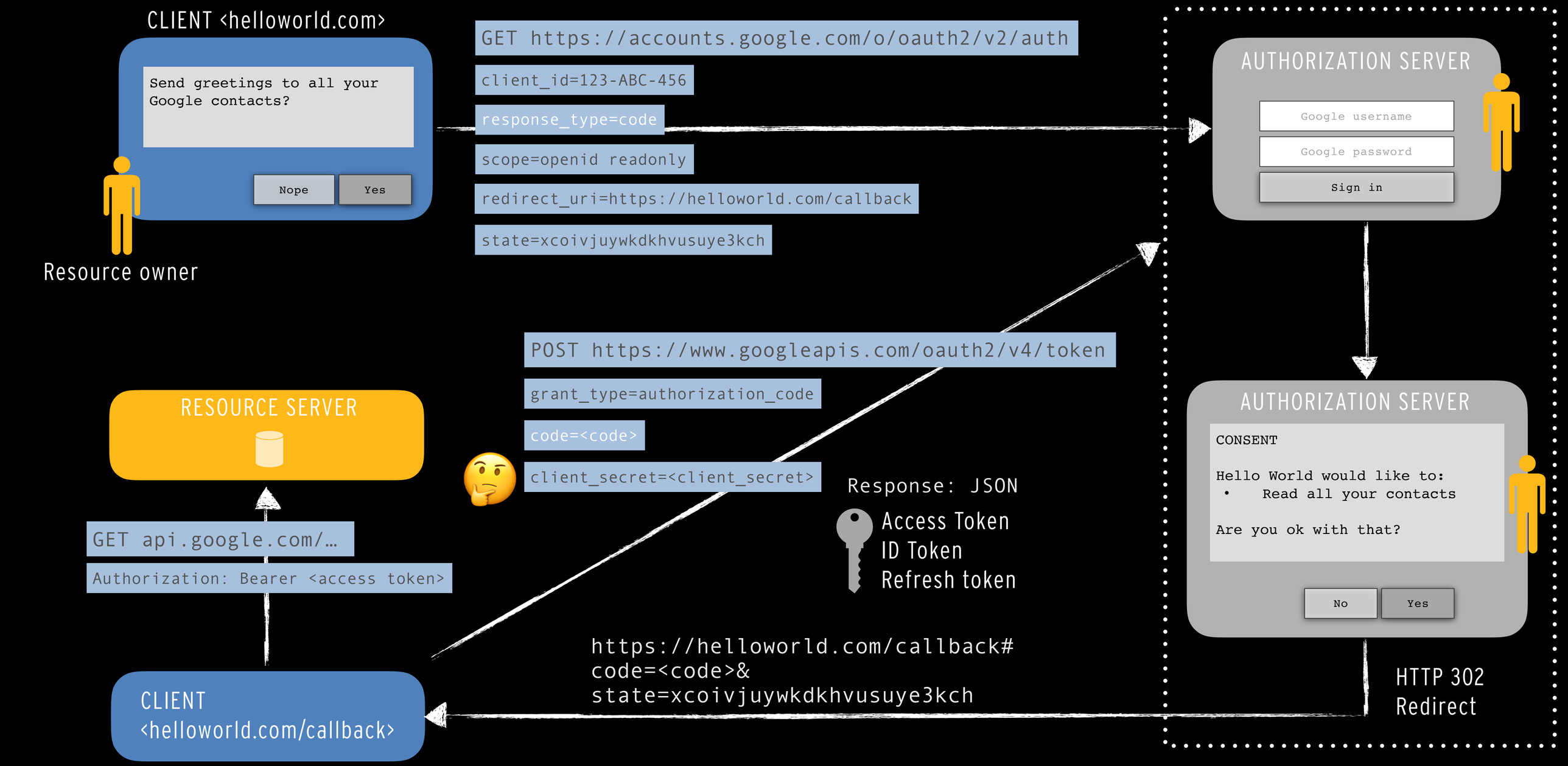
IMPLICIT GRANT WEAKNESSES

- Token in URI prone to "Token Leakage"
  - 3rd party JavaScripts
  - Browser History
  - Referrer Header
  - Browser plugins

```
chrome.history.search({text: '', maxResults: 10}, function(data) {
    data.forEach(function(page) {
        console.log(page.url);
    });
});
```

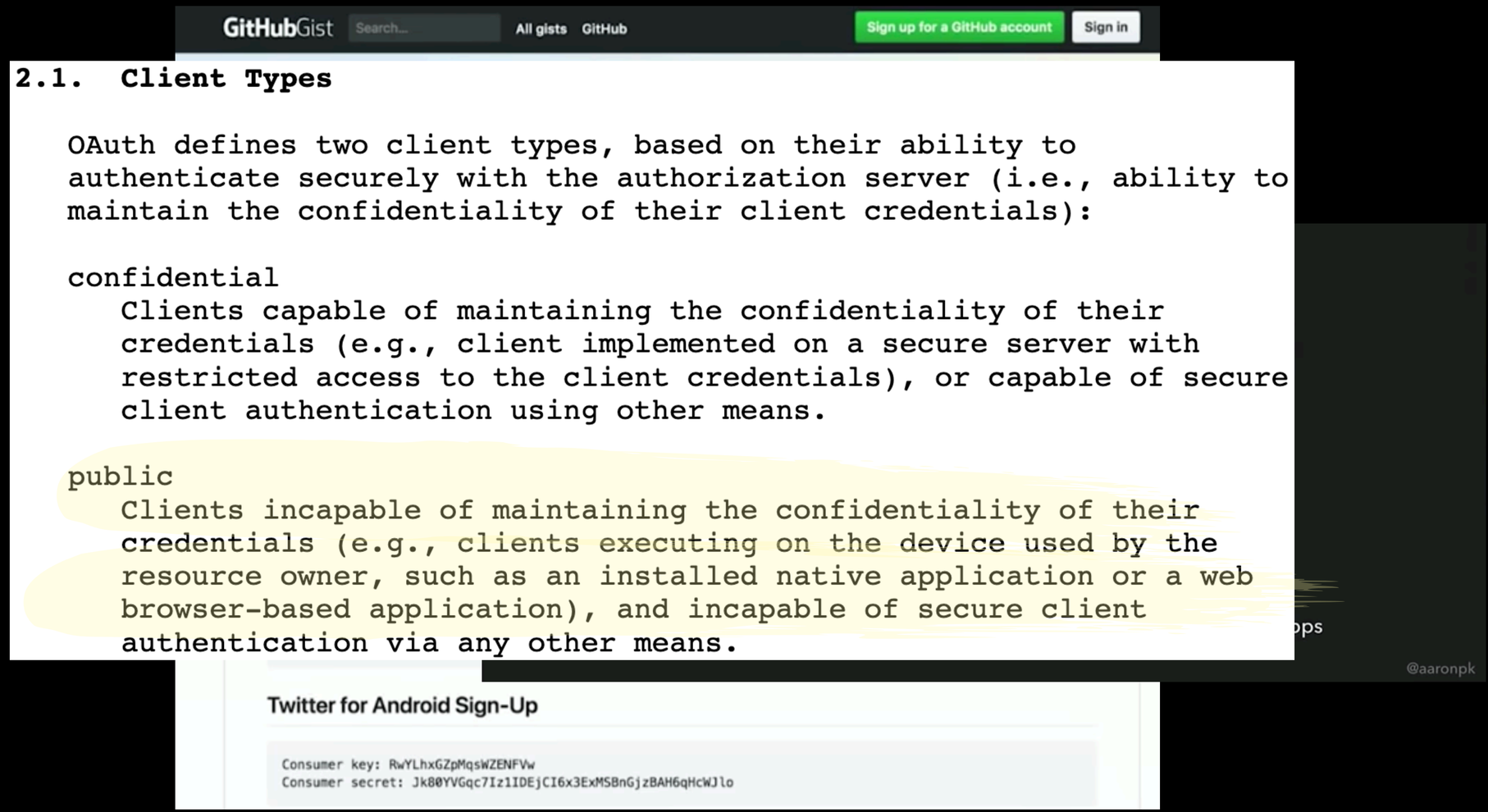
https://developer.chrome.com/extensions/history

TECHNOLOGY OVERVIEW - AUTHORIZATION CODE GRANT FLOW



CONFIDENTIAL VS PUBLIC CLIENTS

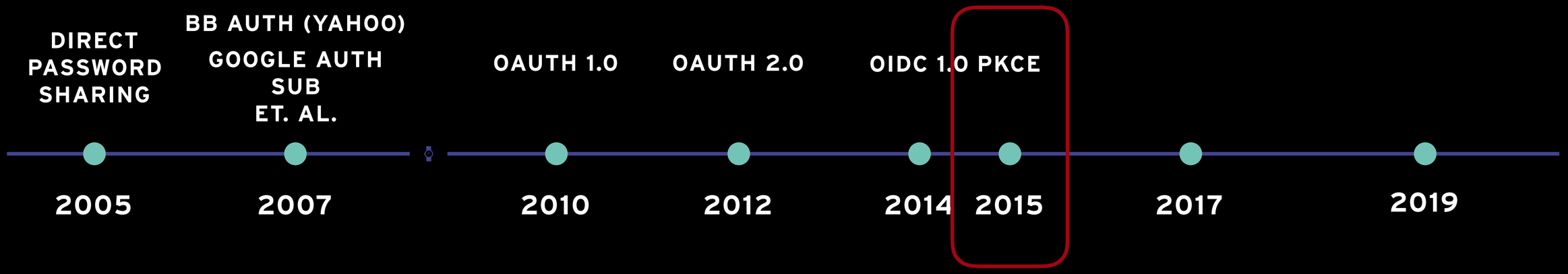
- Core spec (2012) - https://tools.ietf.org/html/rfc6749#section-2.1



PKCE

RFC 7636 - Proof of key code exchange

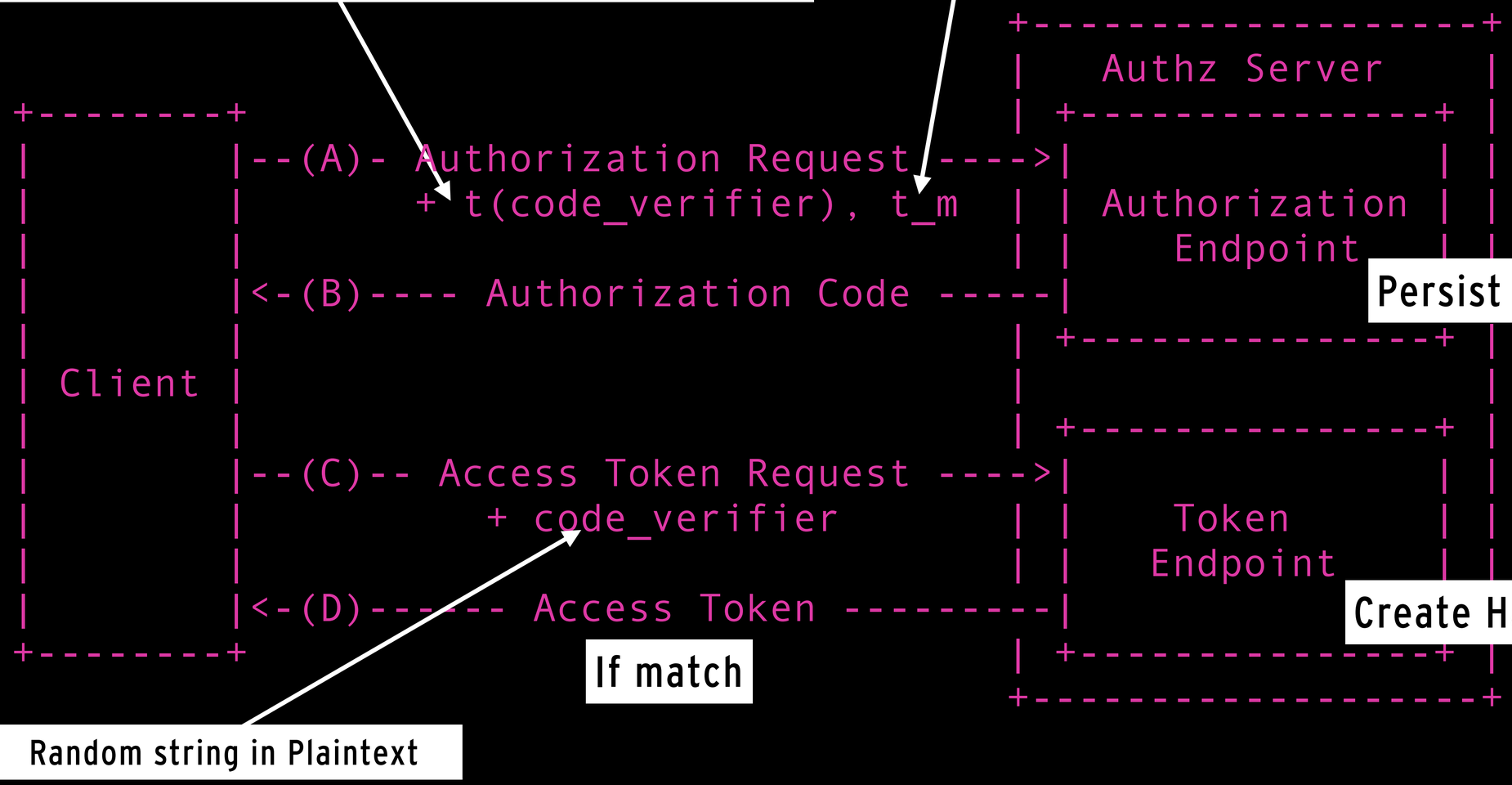
"Pixie"



PKCE

code_challenge
= cryptographic hash of random string (43-128 chars)
(requires WebCrypto)

Hash method
=S256



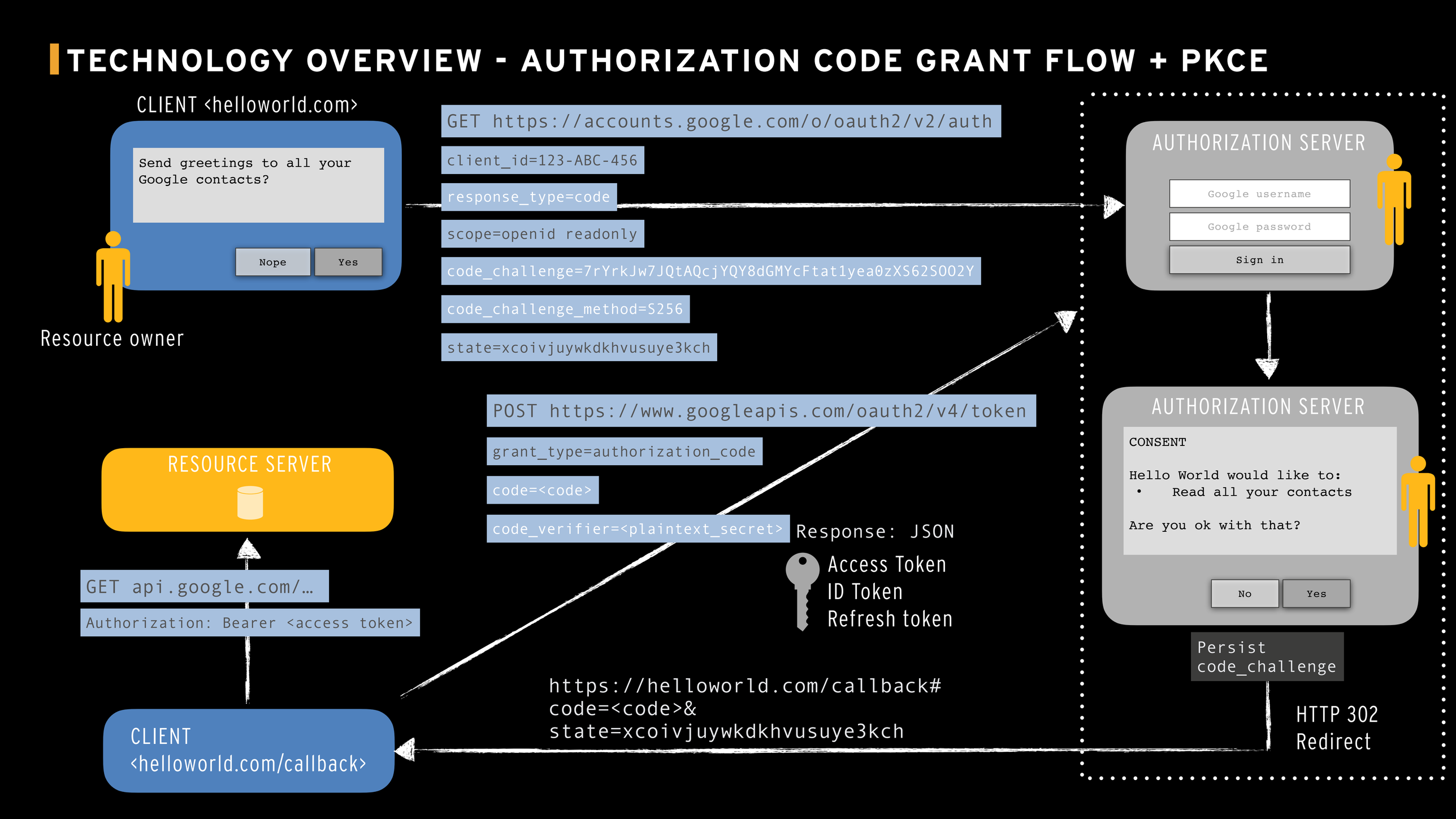
Persist code_challenge

Create Hash and compare

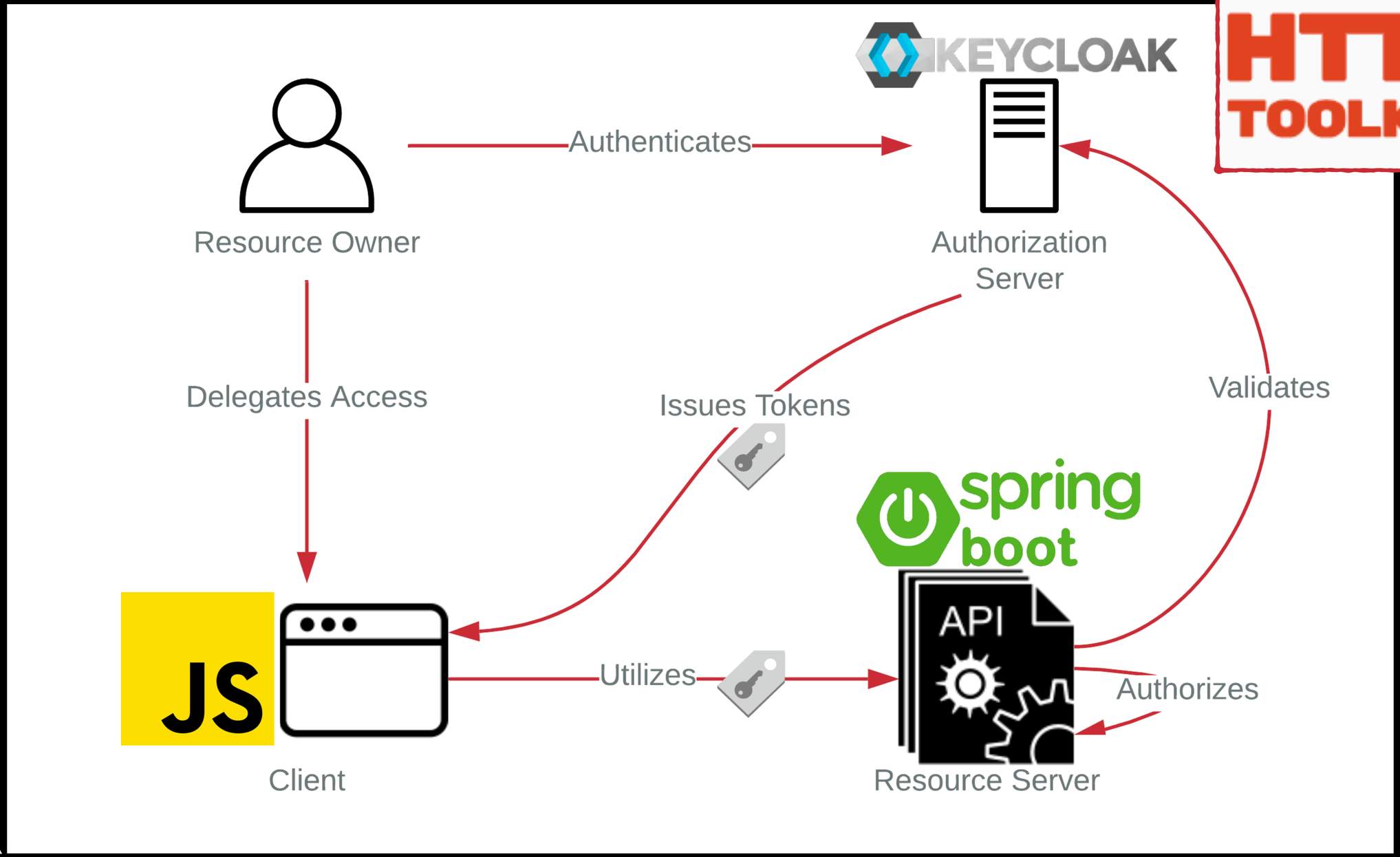
If match

Random string in Plaintext

TECHNOLOGY OVERVIEW - AUTHORIZATION CODE GRANT FLOW + PKCE

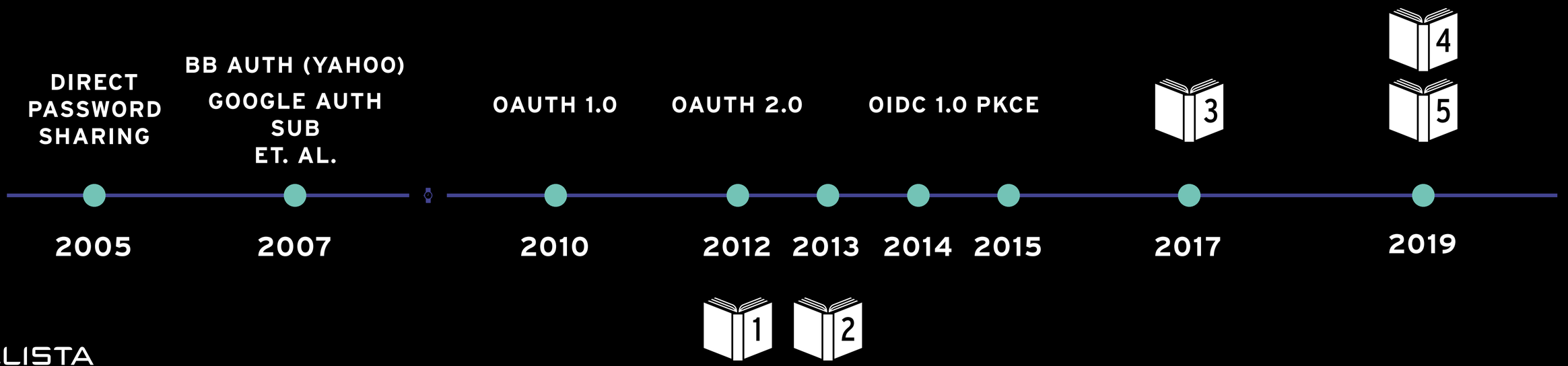


DEMO: AUTH CODE GRANT + PKCE



BEST CURRENT PRACTICE - SECURITY CONSIDERATIONS

1. OAuth 2 main spec: https://tools.ietf.org/html/rfc6749#section-10
2. OAuth 2.0 Threat Model and Security Considerations: https://tools.ietf.org/html/rfc6819
3. OAuth 2.0 for Native Apps: https://tools.ietf.org/html/rfc8252#section-8
4. OAuth 2.0 Security Best Current Practice: https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13
5. OAuth 2.0 for Browser-Based Apps BCP: https://tools.ietf.org/html/draft-ietf-oauth-browser-based-apps-04#section-9



BEST CURRENT PRACTICE (BCP) - GENERAL

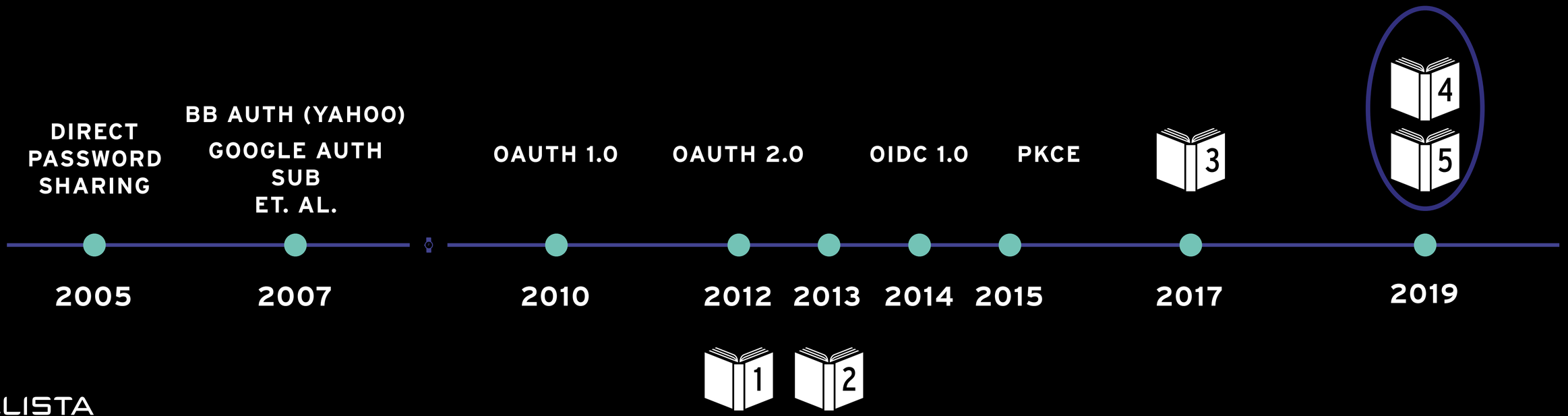
- TLS - end 2 end
- Don't DIY
- Enforce strict URL-redirection rules in Auth Server
- Always use CSRF mitigation techniques
- Verify JWTs
- Manage your tokens!
  - POLP (Principle of Least Privilege)
  - Lifespan/expiration time
  - Rotate Refresh Tokens

HTTPS://SKOLBANKEN.UNIKUM.NET/SKOLBANKEN/PLANERING/3678643542

BEST CURRENT PRACTICE - SPA

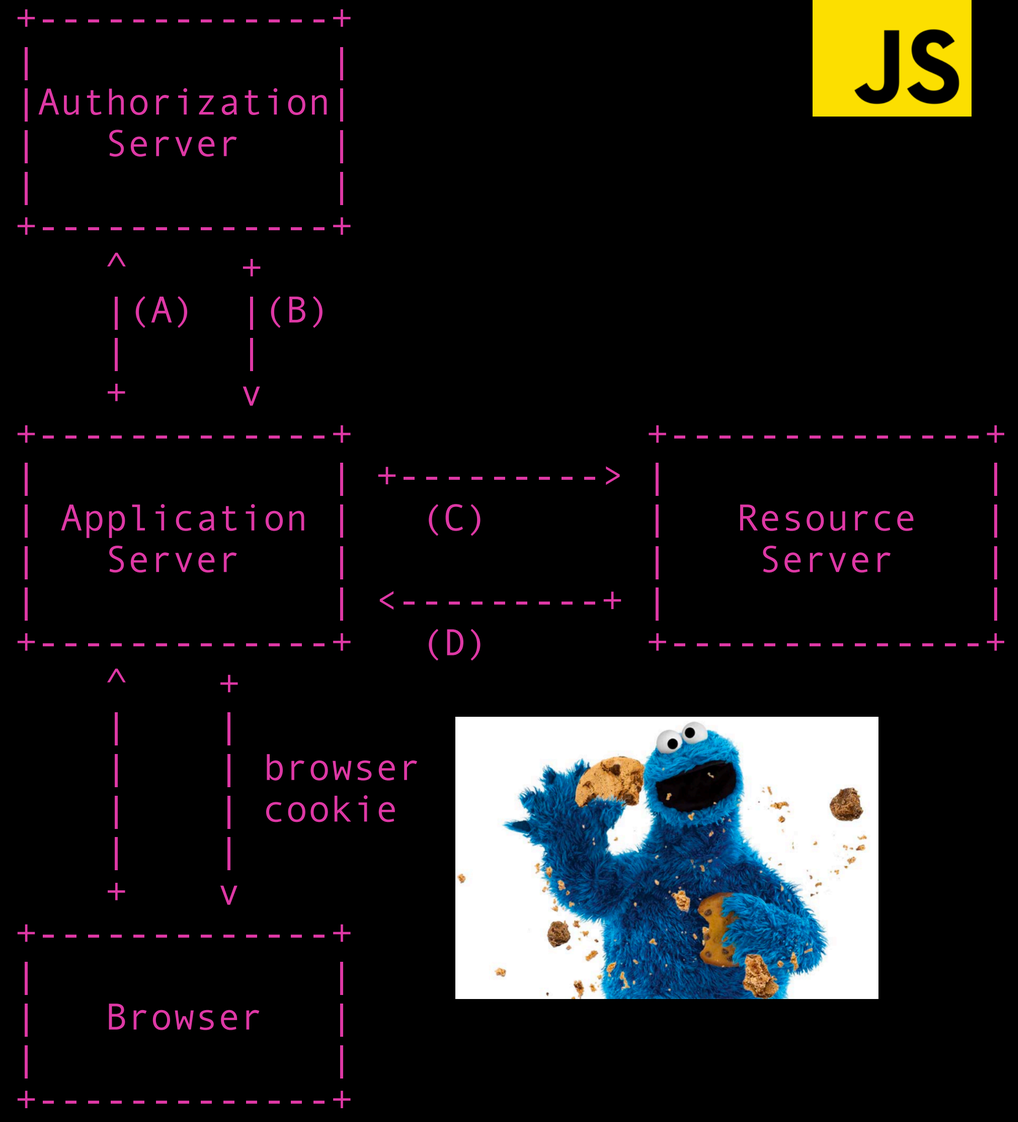
- Use PKCE
- Only use trusted 3rd party JavaScripts
  - Control JavaScript execution via Content Security Policy
- No secure storage in browsers, keep tokens in transient memory

4. OAuth 2.0 Security Best Current Practice
5. OAuth 2.0 for Browser-Based Apps BCP



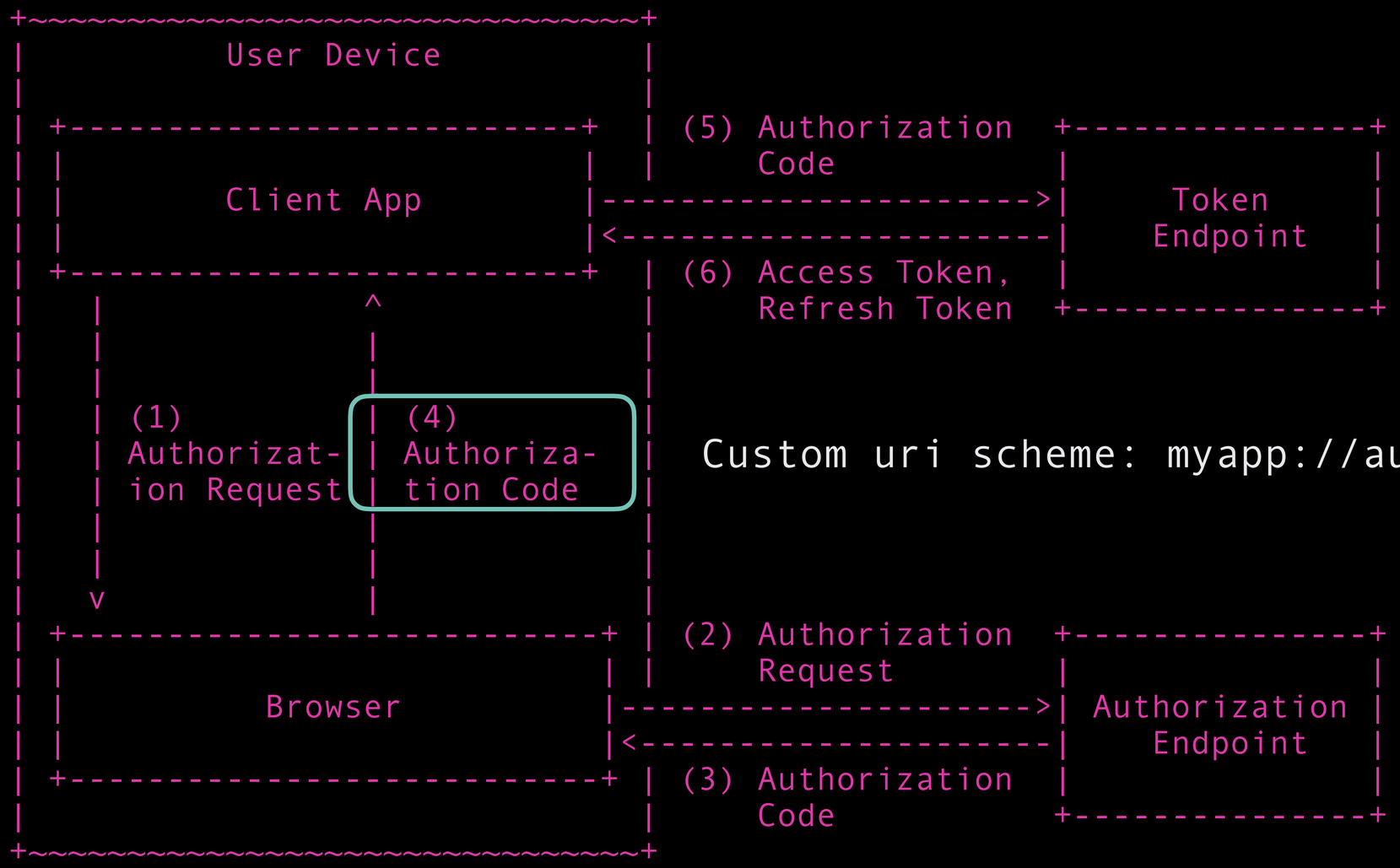
SAME DOMAIN ARCHITECTURE FOR SPA

- Use good old cookie(s) to authenticate
  - » HTTP Only
  - » Secure
- If the SPA has a dedicated backend served from the same origin (domain) as the SPA itself
- Downsides
  - Backend has to proxy calls
  - If the actual tokens should be associated with the cookie session state is introduced



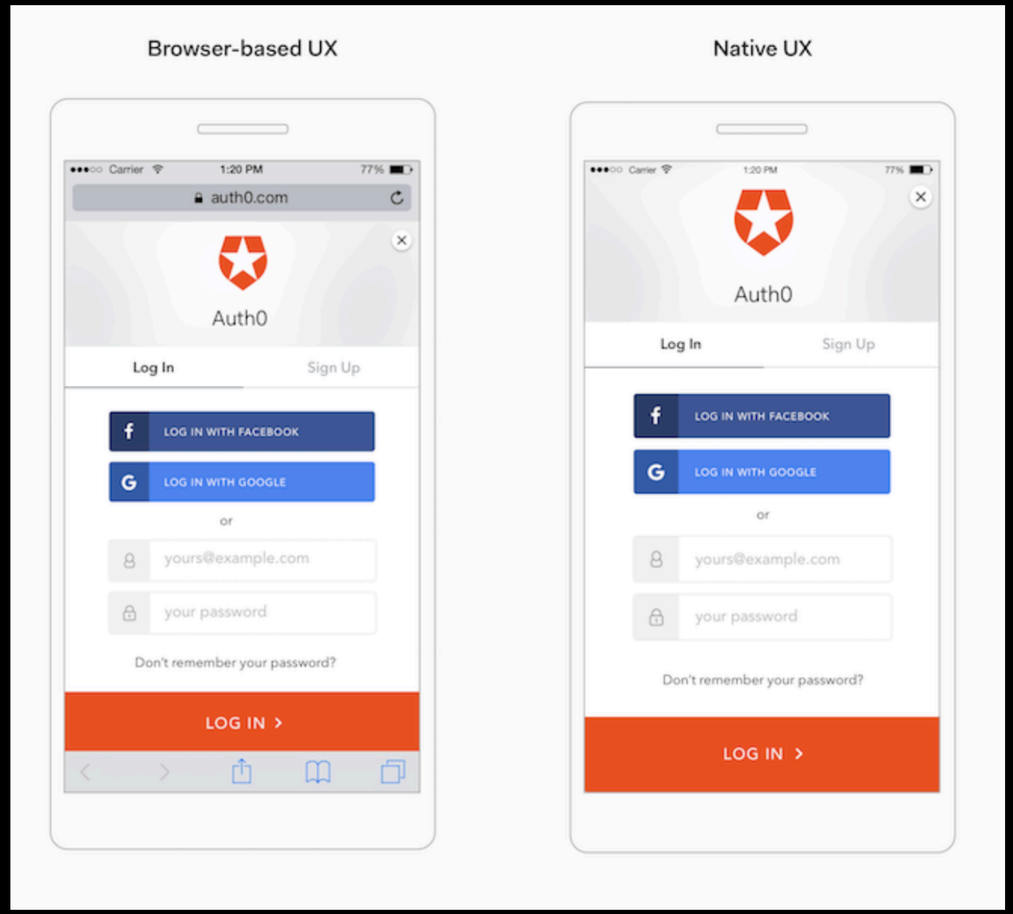
HTTPS://TOOLS.IETF.ORG/HTML/DRAFT-IETF-OAUTH-BROWSER-BASED-APPS-04#SECTION-6.2

NATIVE/MOBILE APPS



BEST CURRENT PRACTICE - MOBILE/NATIVE CLIENTS

- Use PKCE
- Use "app claimed" verified https schemes for redirect if possible
  - iOS Universal Links
  - Android App Links
- Use the real external browser of the OS over "native widget"
  - Easier for user to determine if the request is legit (URL, TLS)
  - Don’t share credentials with the app
  - Can partake in web SSO



HTTPS://AUTH0.COM/DOCS/DESIGN/BROWSER-BASED-VS-NATIVE-EXPERIENCE-ON-MOBILE

WRAP UP

Best current practice for Public Clients
- SPAs and Mobile -
is
Authorization Code Flow with PKCE.

For SPAs with same domain scenarios and dedicated backend;
consider using cookies
to eliminate the need to store tokens in the Browser

Implicit Flow is viable in situations when only obtaining ID Tokens
(i.e response_type=id_token)

WRAP UP: ON THE HORIZON

- "Sender constrained" Tokens:
  - "Level up" Bearer Tokens into "Holder of Key" tokens using cryptography
    - OAuth 2.0 Token binding
    - OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens
  - No wide support, still just a drafts…
- OAuth 2.1 (proposed)
- OAuth 3.0 (new WG TBD)

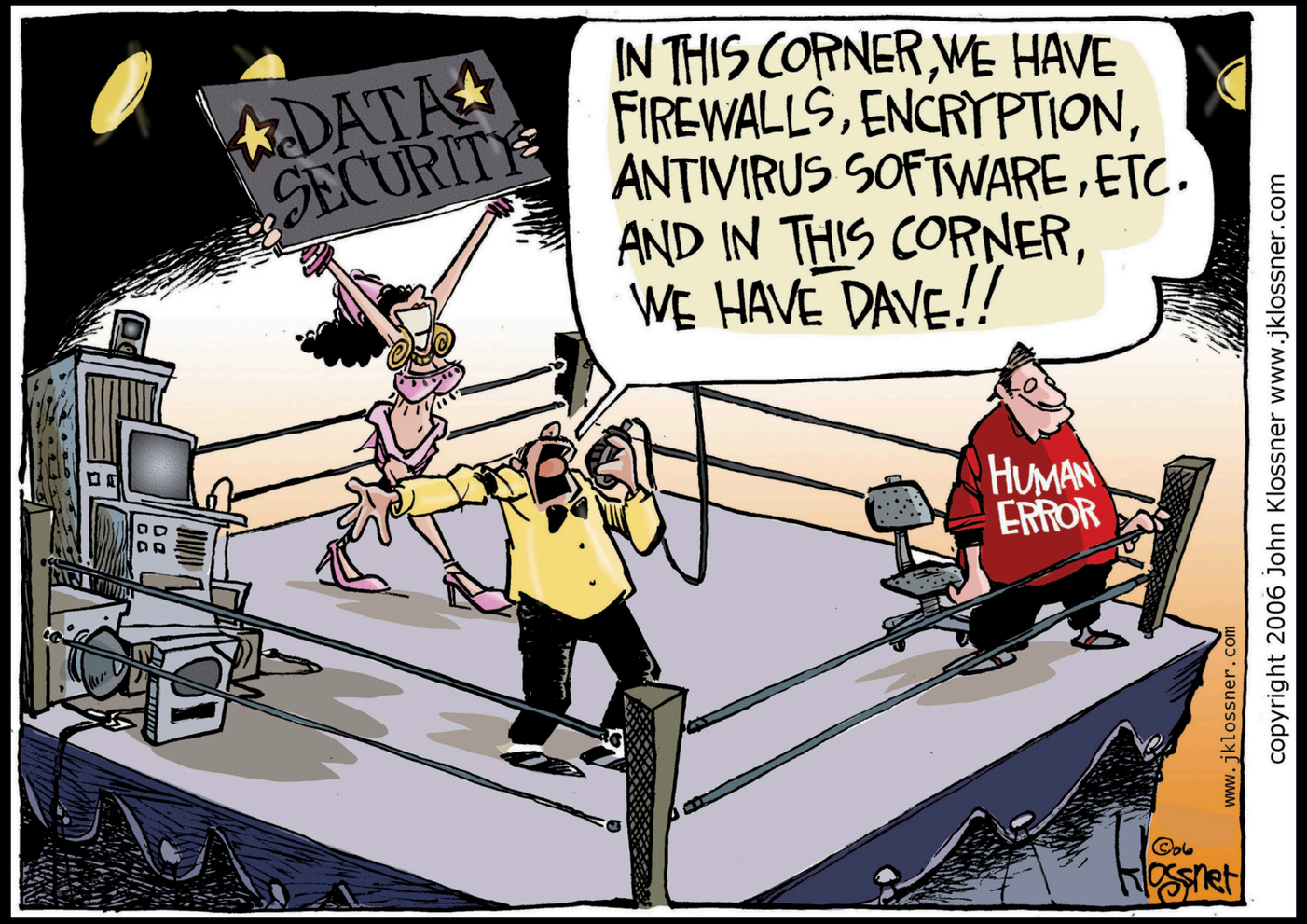
WRAP UP: BALANCED SECURITY MEASURES

https://defence-blog.com/army/oman-to-acquire-k2-black-panther-tanks-from-south-korea.html

https://www.autoevolution.com/news/knight-xv-world-s-most-secure-suv-costs-310000-3286.html#agal_3

https://europe.autonews.com/blogs/renault-twizy-gets-new-life-korea-motorcycle-replacement

Context Is King



THANK YOU!

CALLISTA

QUESTIONS?


andreas.tell@callistaenterprise.se


https://www.linkedin.com/in/andreastell/

REFERENCES

- OAuth 2 main spec: https://tools.ietf.org/html/rfc6749
- OAuth 2.0 Threat Model and Security Considerations: https://tools.ietf.org/html/rfc6819
- OAuth 2.0 for Native Apps: https://tools.ietf.org/html/rfc8252
- OAuth 2.0 Security Best Current Practice: https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13
- OAuth 2.0 for Browser-Based Apps: https://tools.ietf.org/html/draft-ietf-oauth-browser-based-apps-04
- "OAuth2 in Action" - Manning (ISBN 9781617293276)
- OAuth 2.0 and OpenID Connect in plain English: https://www.youtube.com/watch?v=996OiexHze0&feature=youtu.be