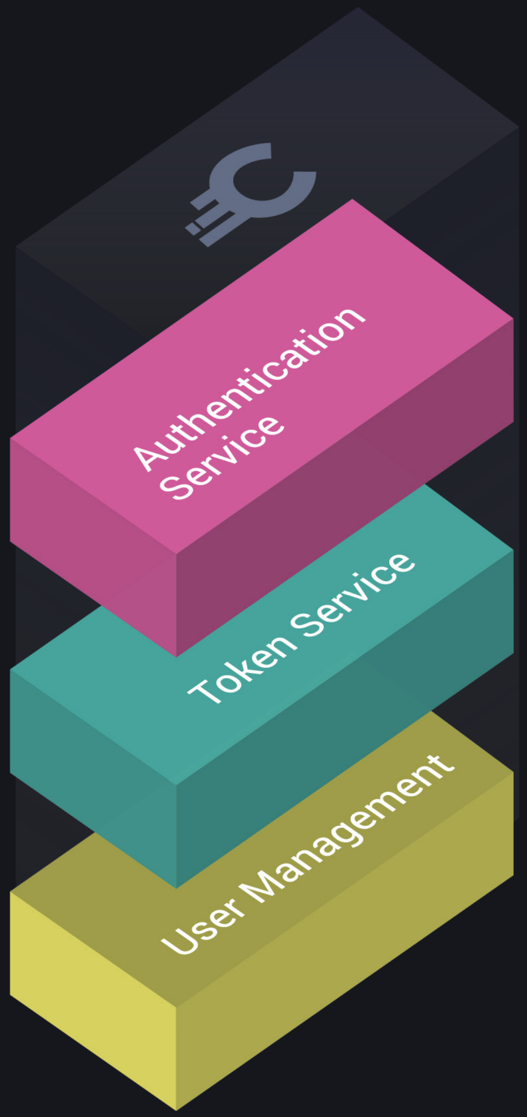
Financial-Grade Security, Not Just for Banks

CURITY

Jonas Iggbom

Director of Sales Engineering

CURITY



Curity Identity Server

A Unique Combination of IAM and API Management

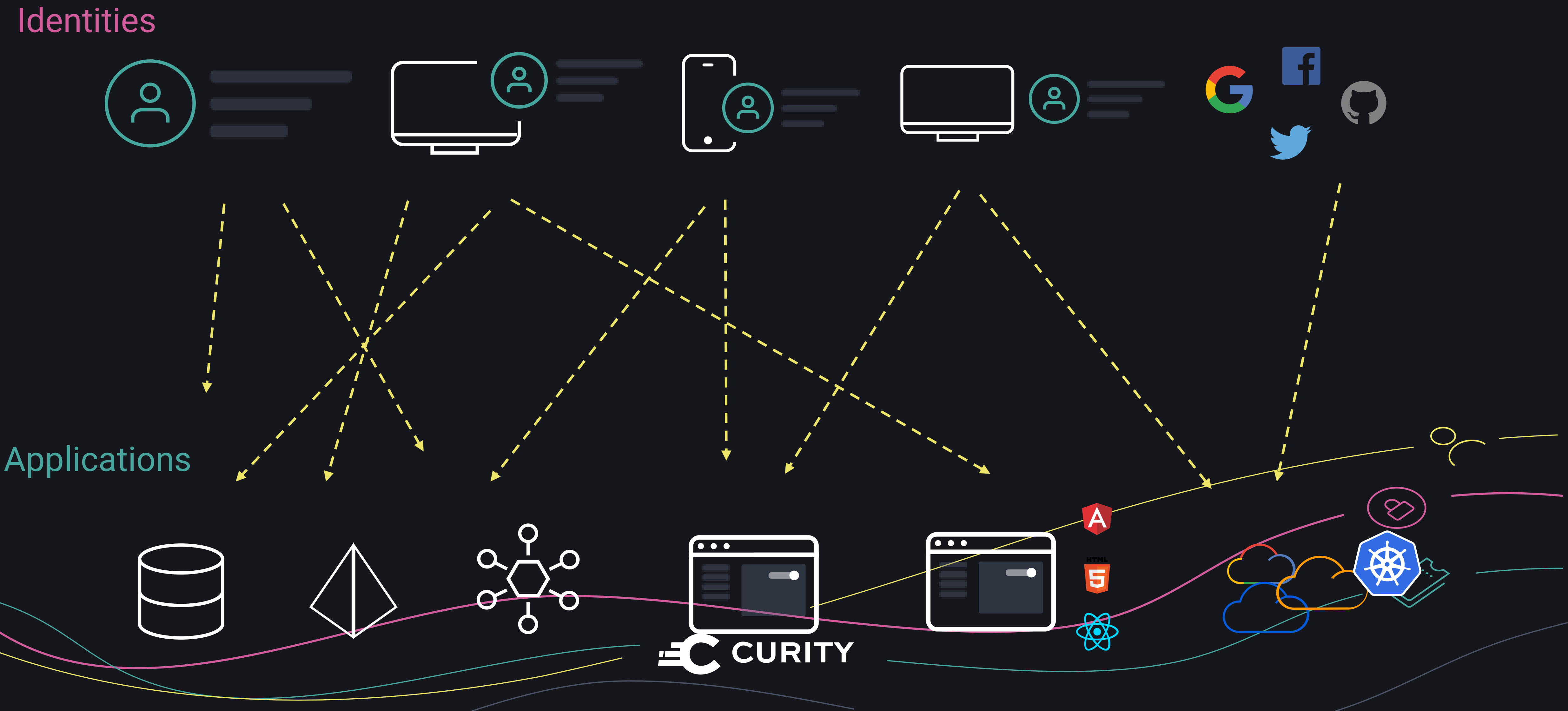
Flexible Authentication

Token-based API Security

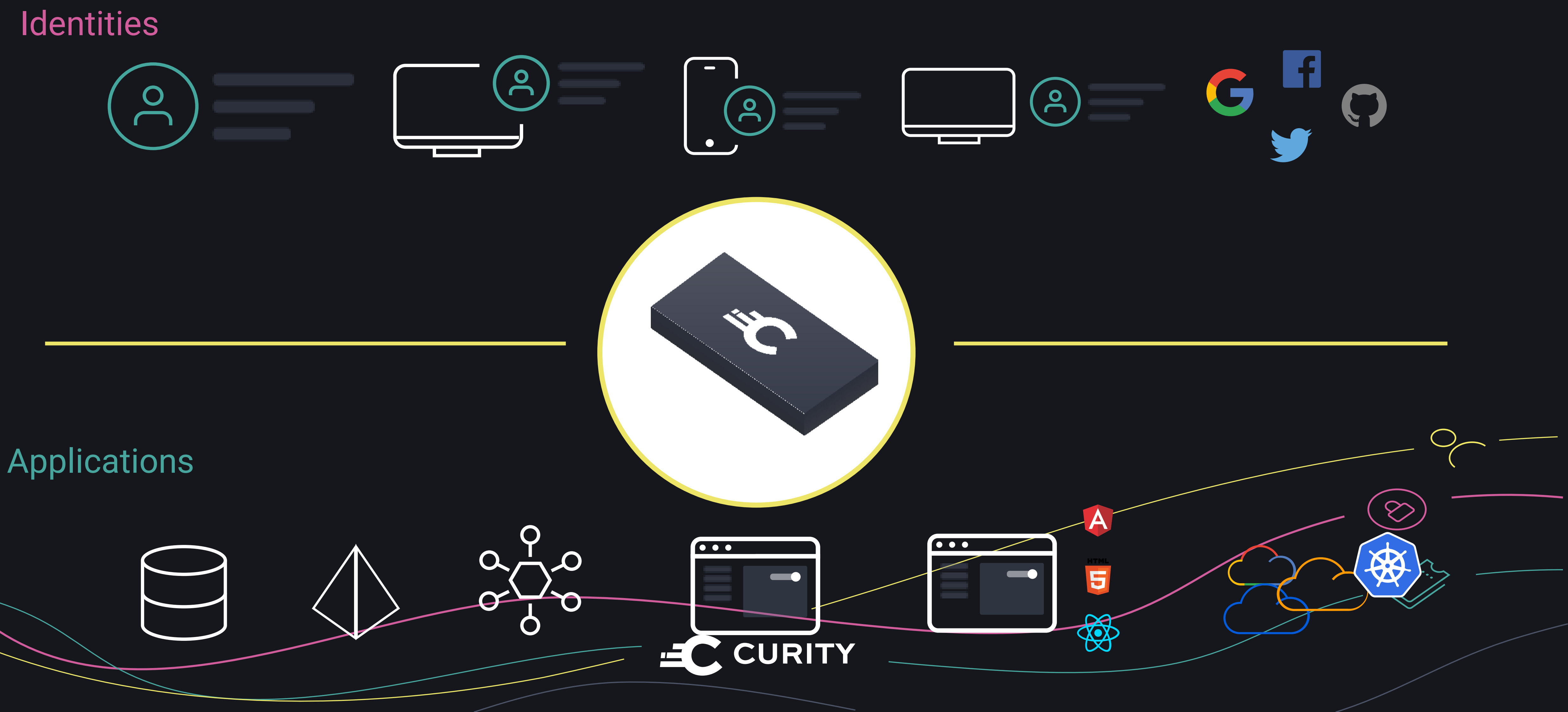
API Driven User Management

CURITY

Complexities of a Digital World



Handle Complexity in the Curity Identity Server



Identities

Applications

Financial Grade API

Finance

Medical

Sensitive data

Financial Grade API

- PSD2
- Europe's General Data Protection Regulation (GDPR)
- UK Open Banking
- Open Banking Brazil
- Australian Privacy Principles (APP)
- South Korea's PIPA
- etc.

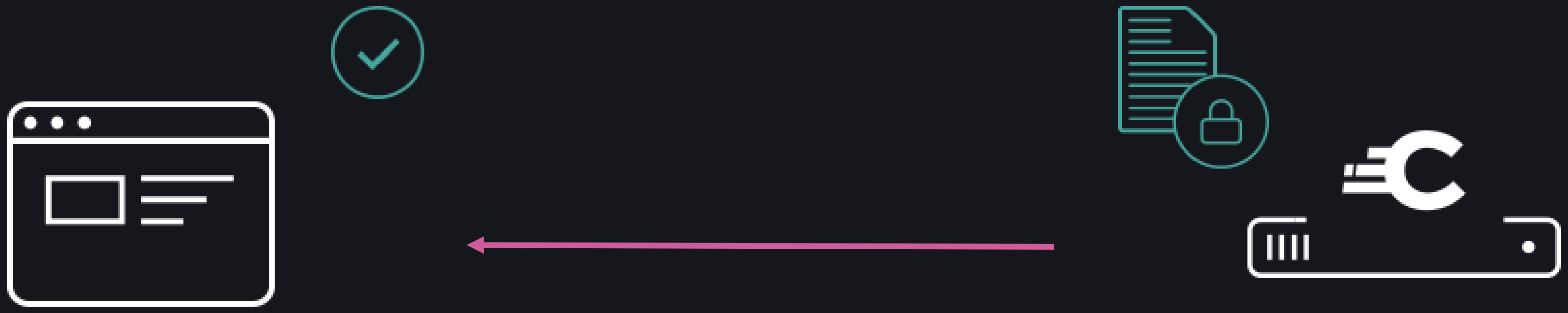
Technologies / Profiles / Patterns

- Mutual Transport Layer Security (mTLS)
  - Sender-constrained access tokens
- Pushed Authorization Request (PAR)
- JWT Secured Authorization Response Mode (JARM)
- Client Initiated Back Channel Authentication (CIBA)
- Phantom / Split token

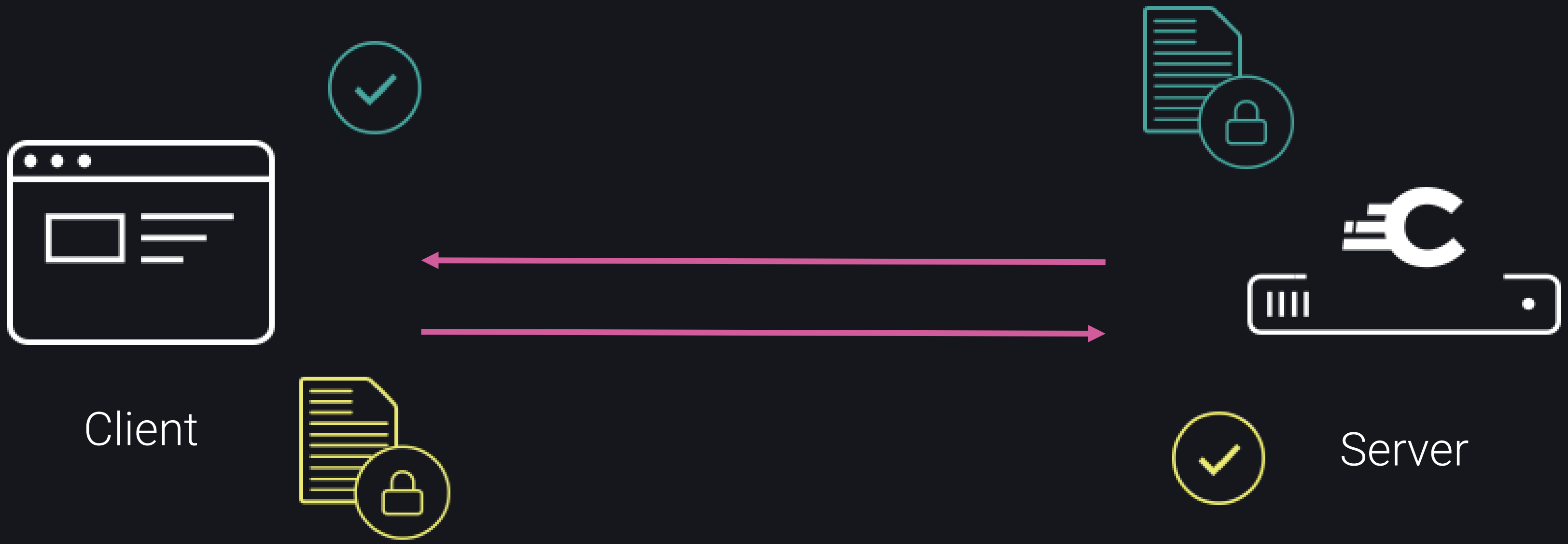
mTLS

CURITY

TLS



mTLS = mutual TLS



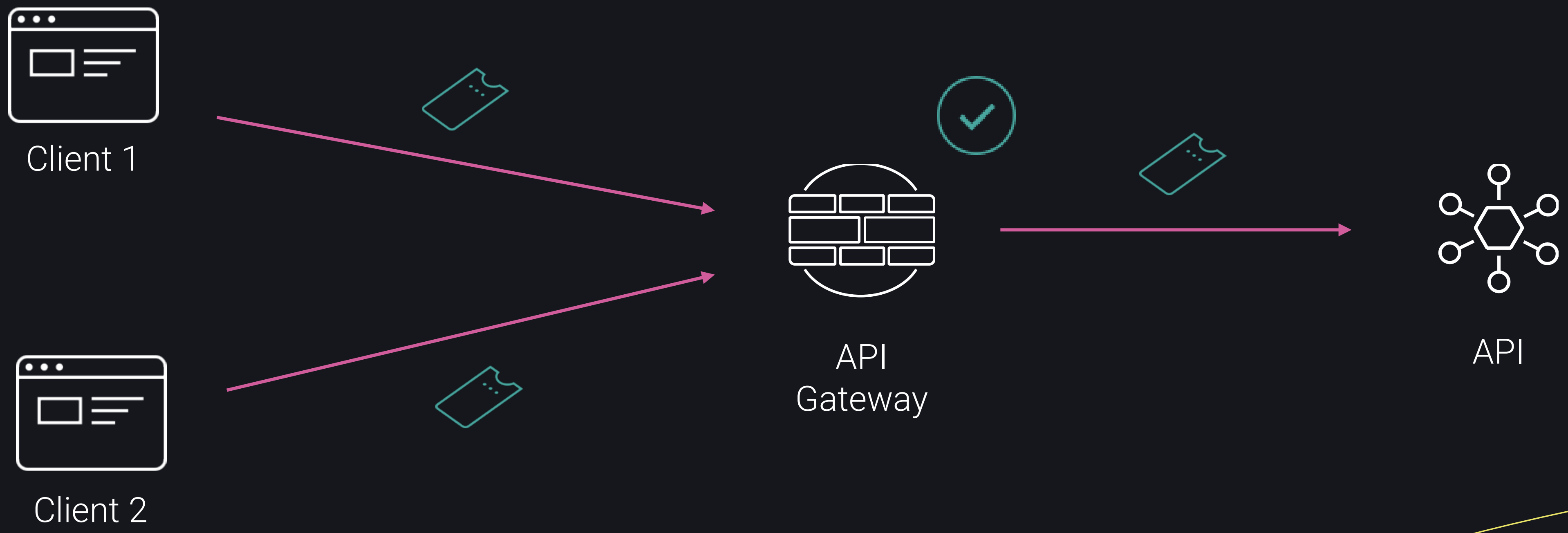
Sender-constrained Tokens

Sender-constrained Tokens

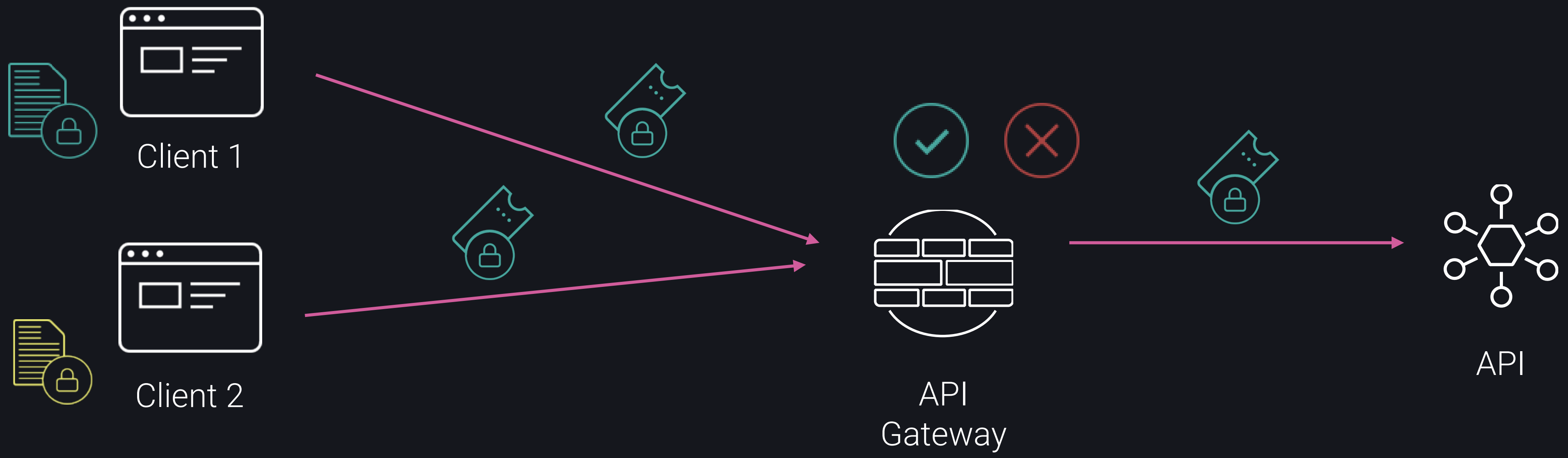
Bearer tokens

Sender-constrained tokens /
Proof-of-Possession tokens

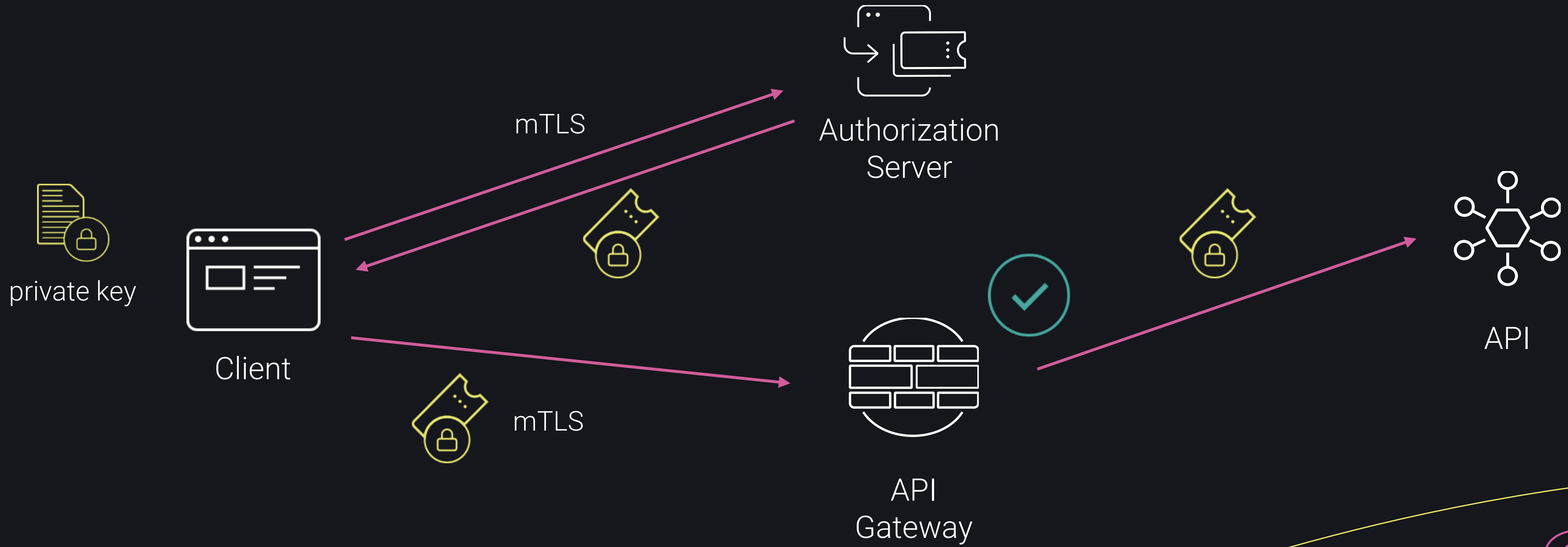
Bearer Tokens



Sender-constrained Tokens



Sender-constrained Tokens



CURITY

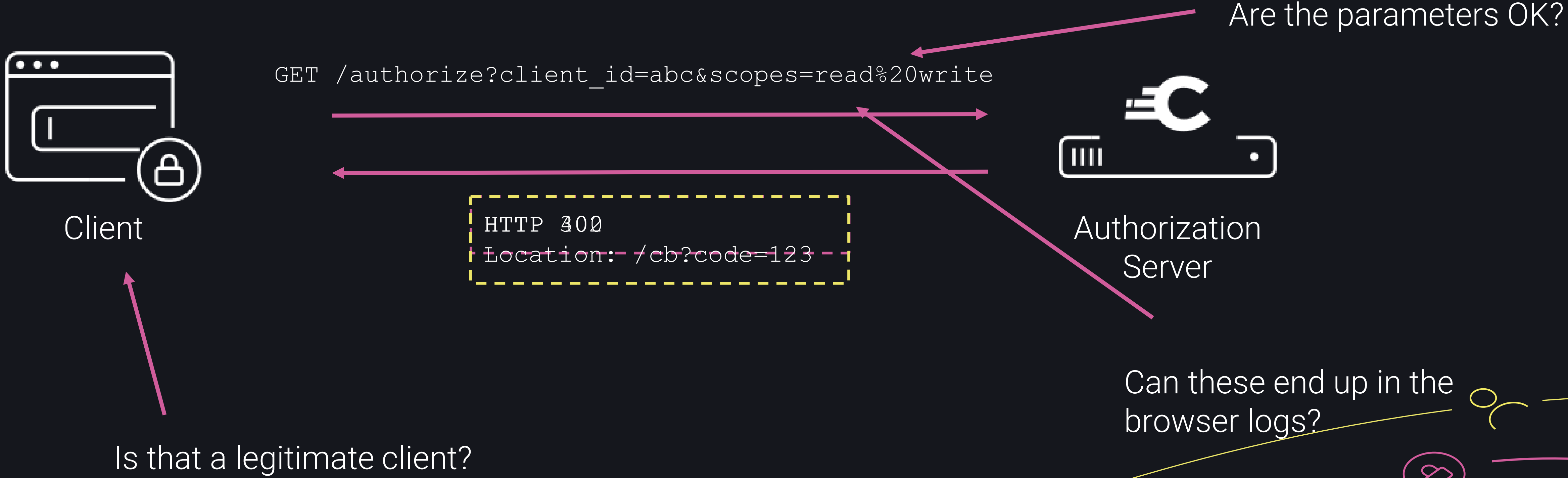
Pushed Authorization Requests (PAR)

CURITY

Pushed Authorization Requests

- Standard defined in RFC 9126.
- Provides means for confidential and integrity-protected authorization requests.

Standard OAuth Authorization Requests



Are the parameters OK?

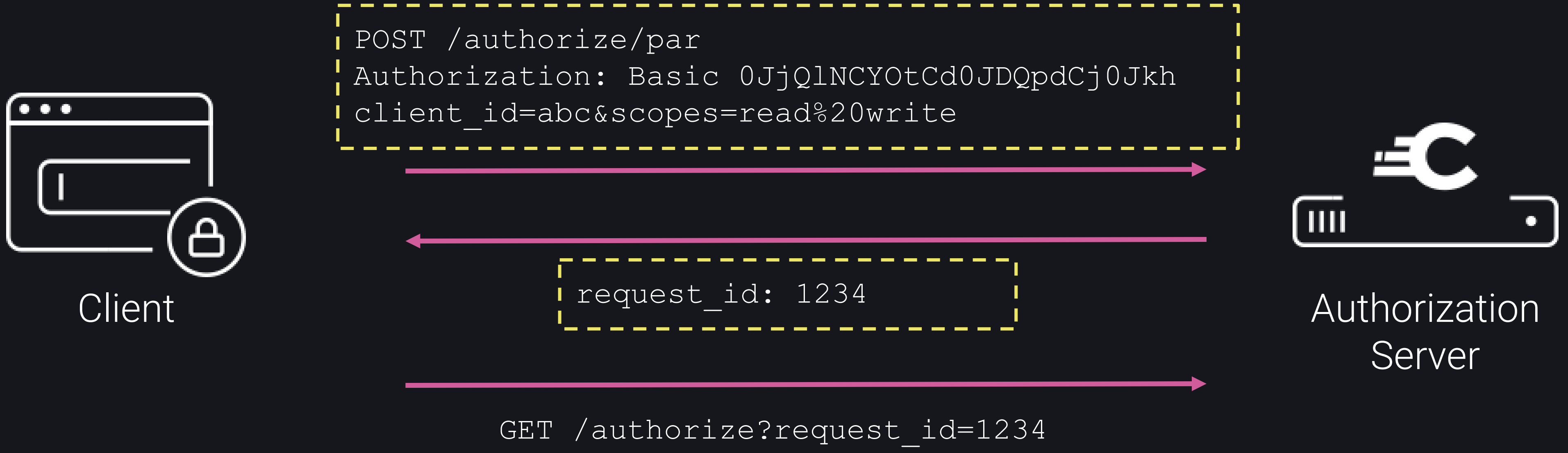
```
GET /authorize?client_id=abc&scopes=read%20write
```

```
HTTP 302
Location: /cb?code=123
```

Can these end up in the browser logs?

Is that a legitimate client?

Pushed Authorization Requests



```
POST /authorize/par
Authorization: Basic 0JjQlNCYOtCd0JDQpdCj0Jkh
client_id=abc&scopes=read%20write
```

```
request_id: 1234
```

```
GET /authorize?request_id=1234
```

Pushed Authorization Requests

- The client is authenticated before the authorization request
- Request parameters do not traverse through unsecure transport and cannot be tampered with
- Ability to ease on redirect URI restrictions

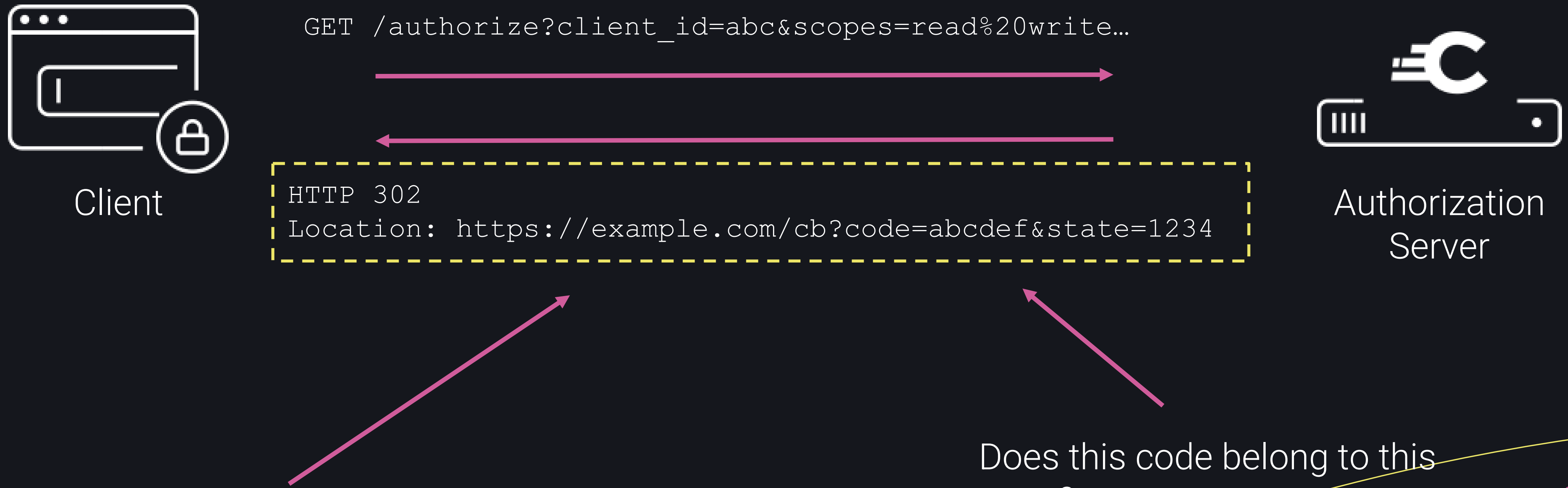
JWT Secured Authorization Response Mode (JARM)

CURITY

JARM

- Draft specification from the OpenID Foundation
- Protects against attacks on the authorization code response

Standard Response



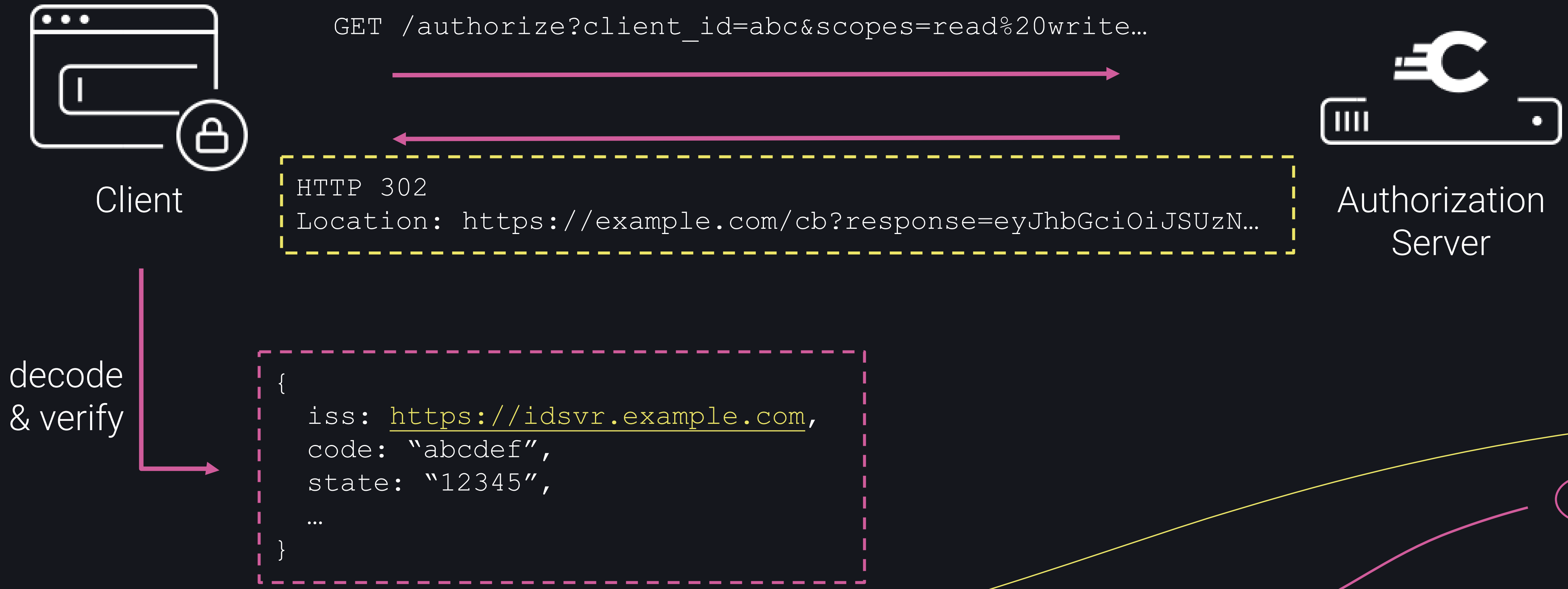
```
GET /authorize?client_id=abc&scopes=read%20write…
```

```
HTTP 302
Location: https://example.com/cb?code=abcdef&state=1234
```

Was it issued by the correct Authorization Server?

Does this code belong to this state?

JWT Secured Response



```
GET /authorize?client_id=abc&scopes=read%20write…
```

```
HTTP 302
Location: https://example.com/cb?response=eyJhbGciOiJSUzN…
```

```
{
  iss: https://idsvr.example.com,
  code: "abcdef",
  state: "12345",
  …
}
```

JARM

- The code response is integrity-protected.
- Response parameters strongly coupled (mitigates replay attacks).
- Protection from mix-up attacks (ability to verify iss claim).

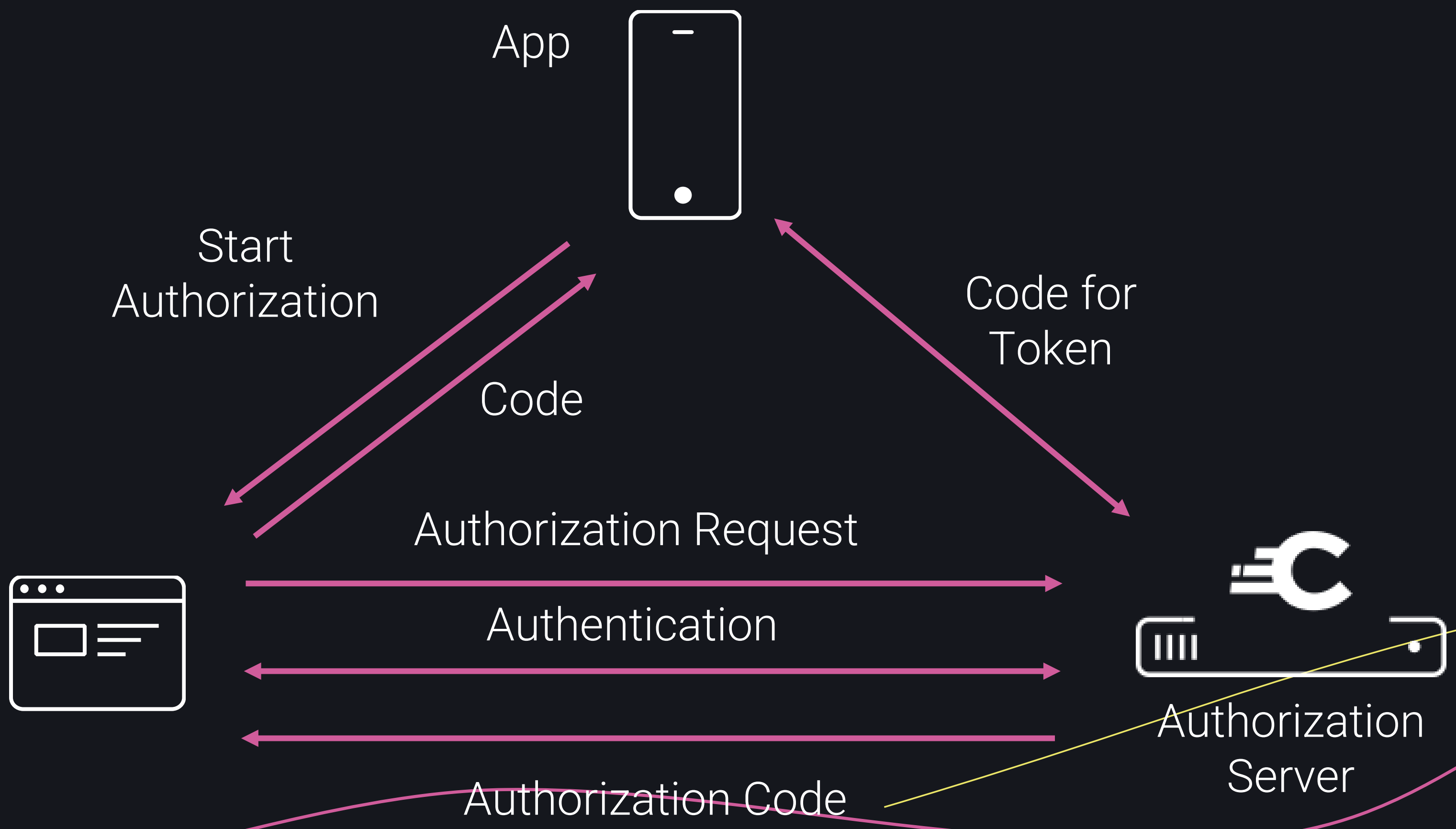
Client Initiated Back Channel Authentication (CIBA)

CURITY

CIBA

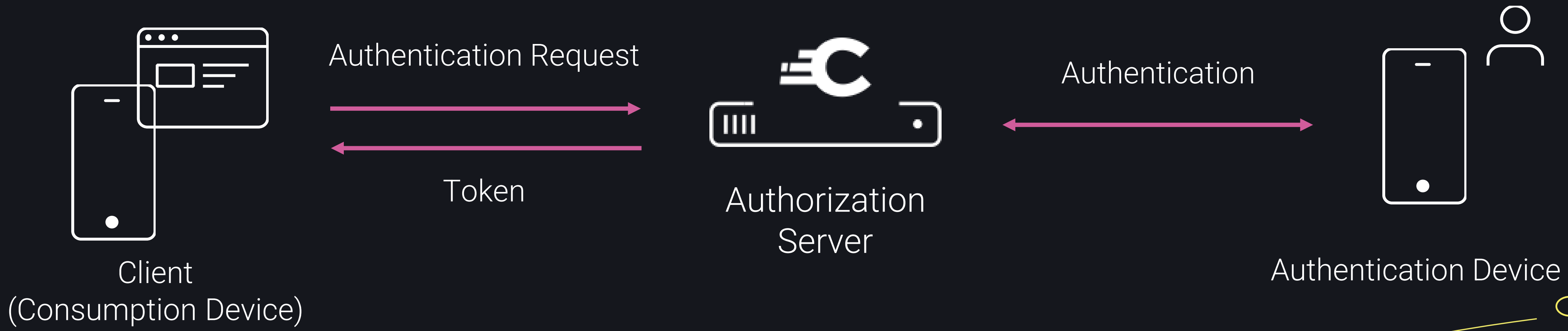
- OpenID Connect Authentication Flow
- Decoupled authentication
- Relying Party initiates authentication

Traditional Front-channel Authentication



CURITY

CIBA


Authentication Request
Token
Authorization Server
Authentication
Client
(Consumption Device)
Authentication Device

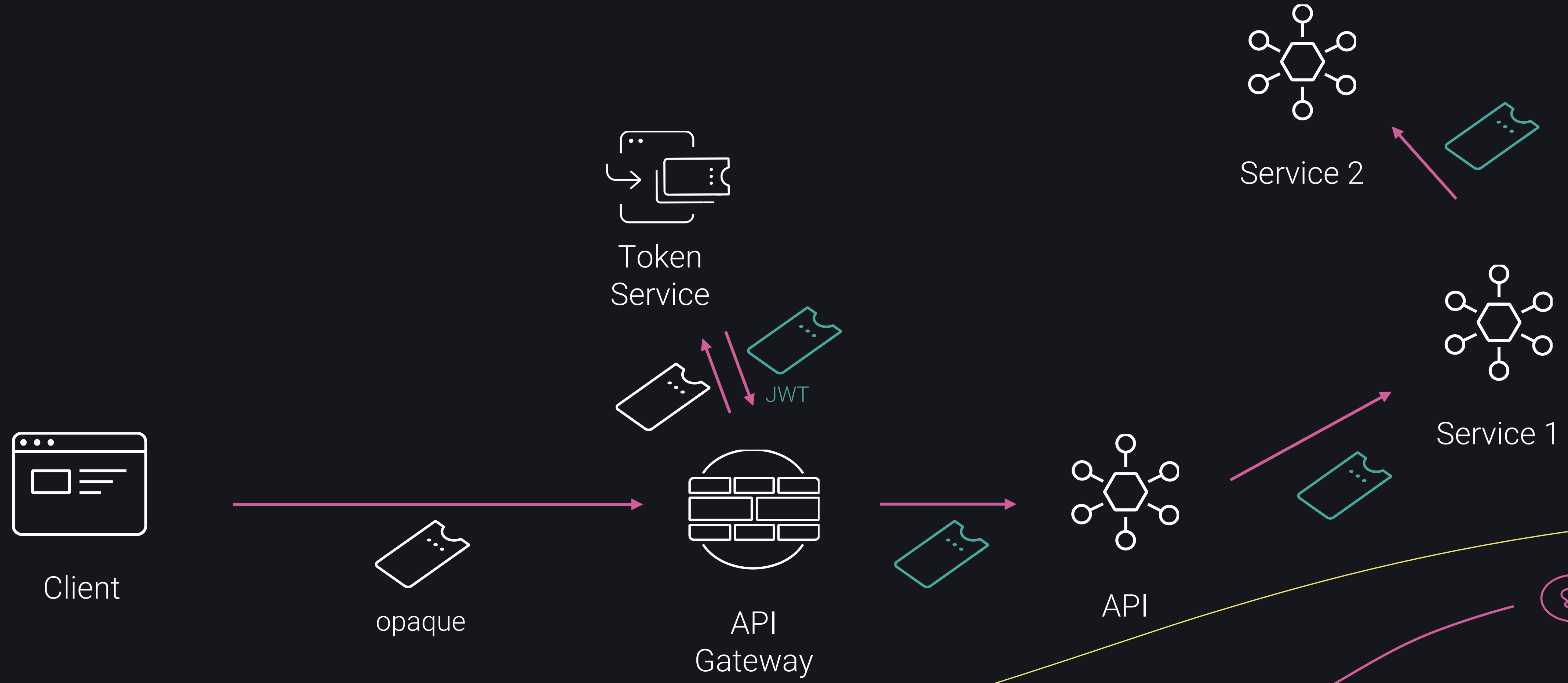
Phantom / Split Token

CURITY

Phantom / Split Token

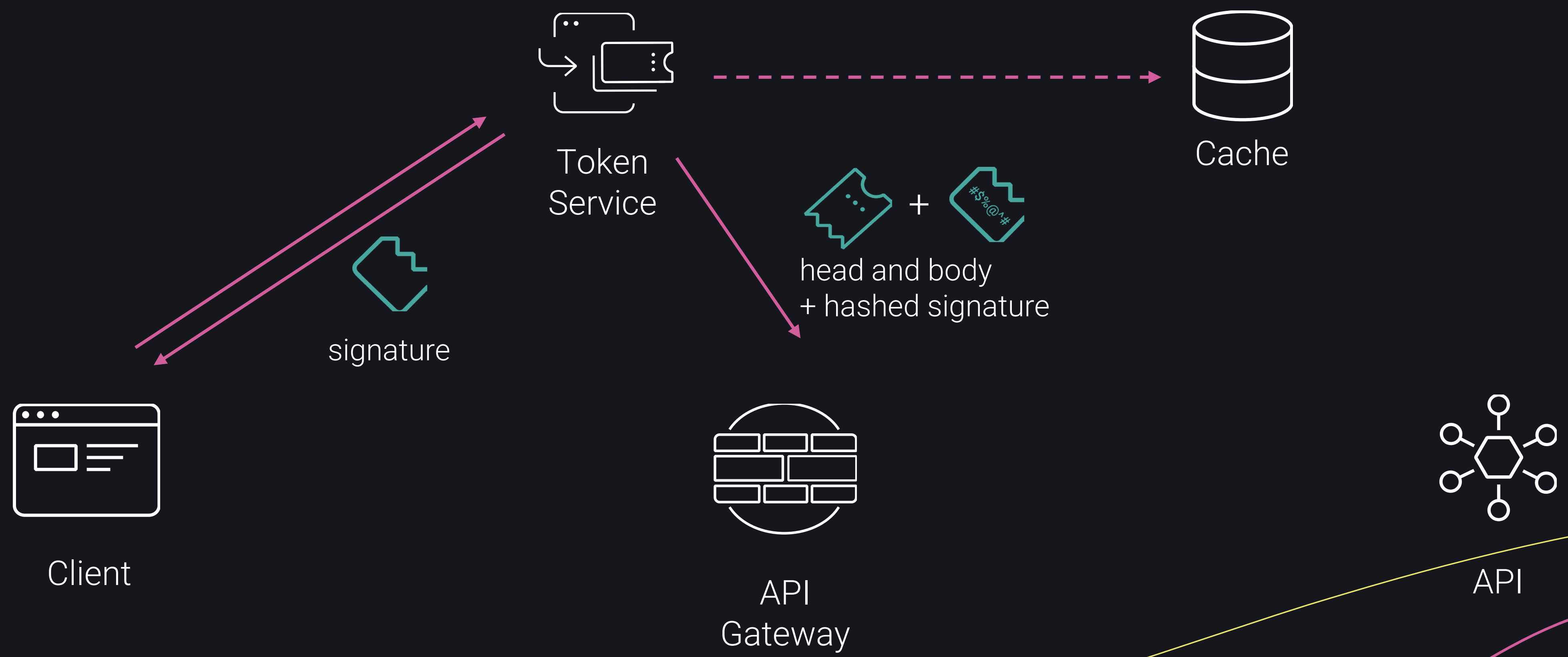
- Reduce data exposure to the client
- PII data
- Token information is for the API, not the client
- Risk of breaking app if relying on specific data in JWT

The Phantom Token Flow



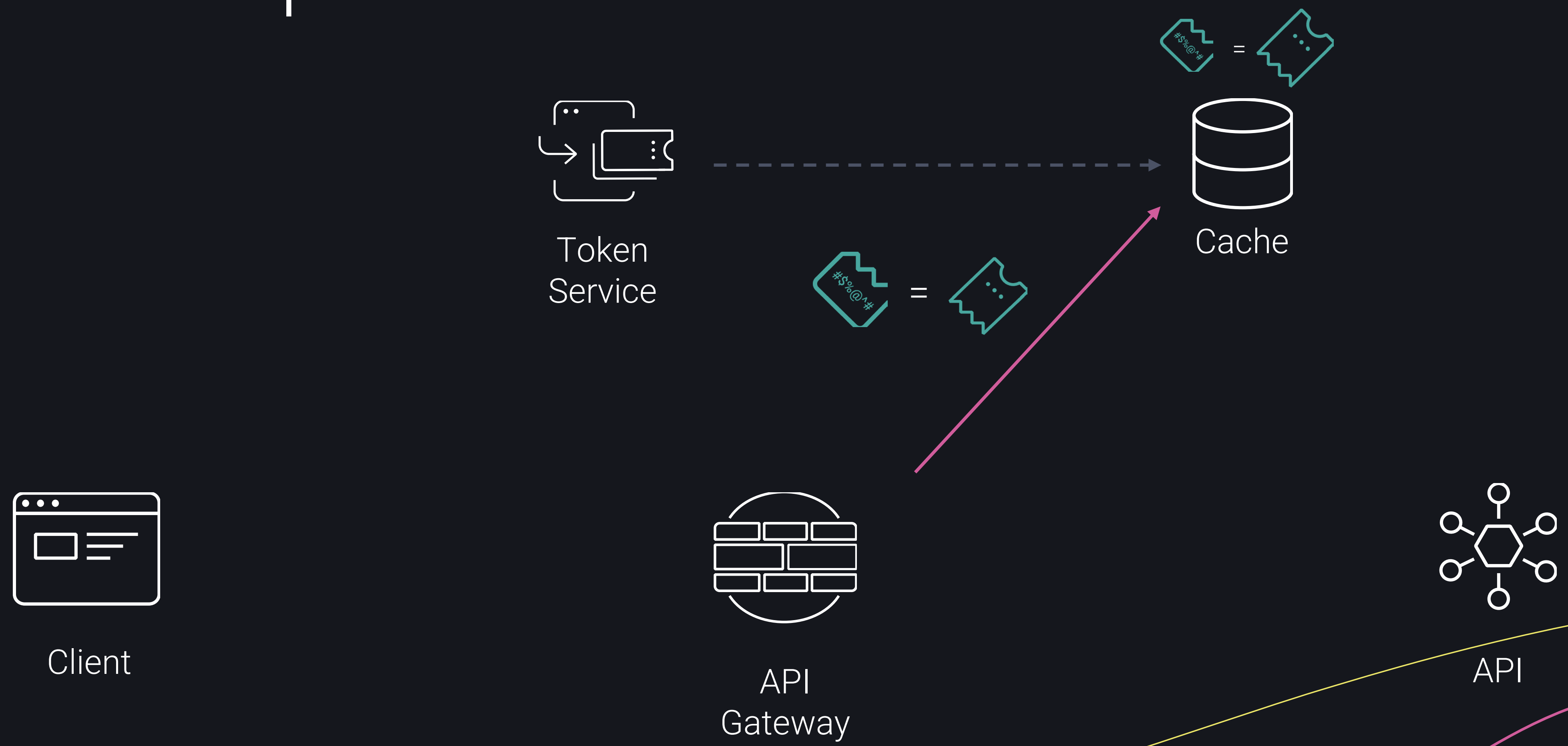
CURITY

The Split Token Flow



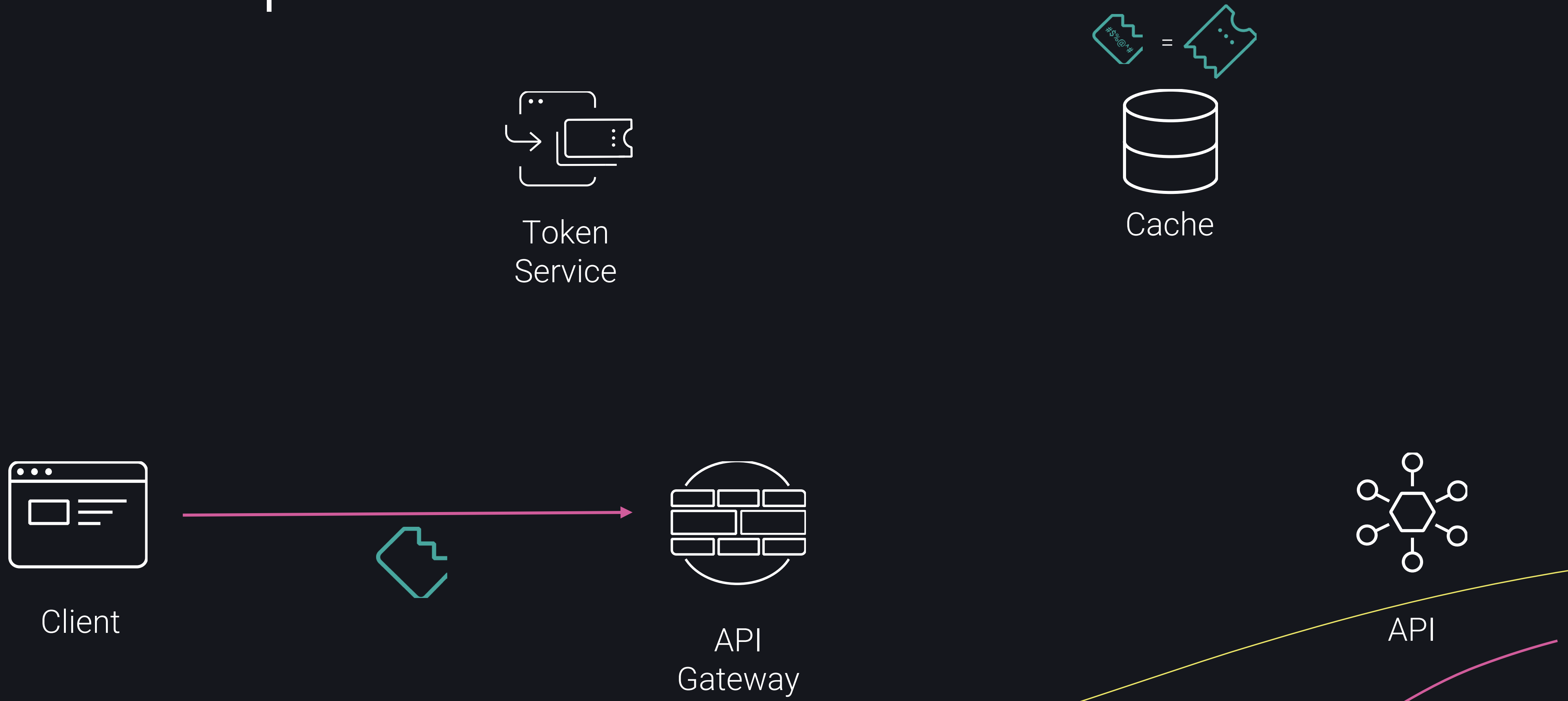
CURITY

The Split Token Flow



CURITY

The Split Token Flow



Token Service

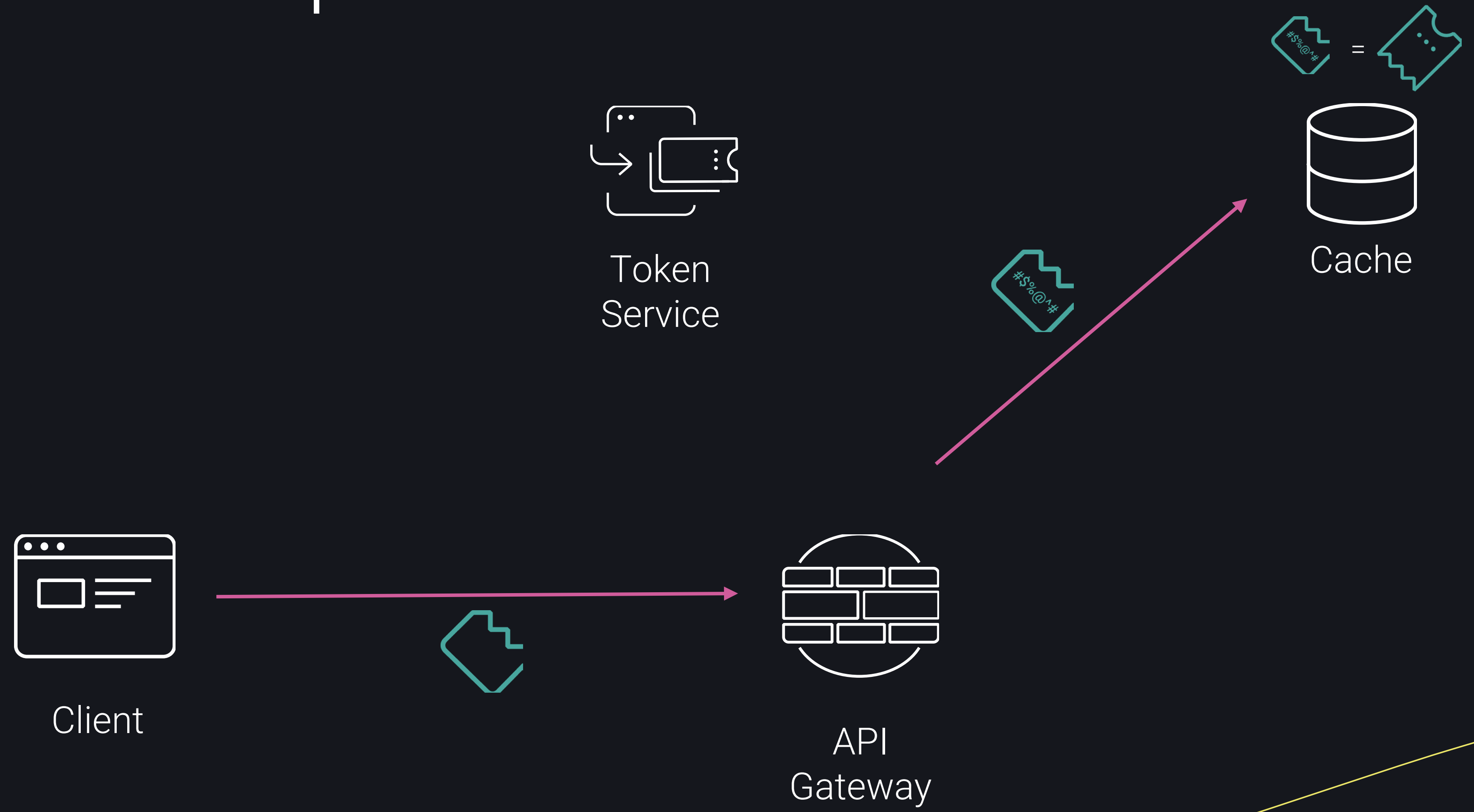
Cache

Client

API Gateway

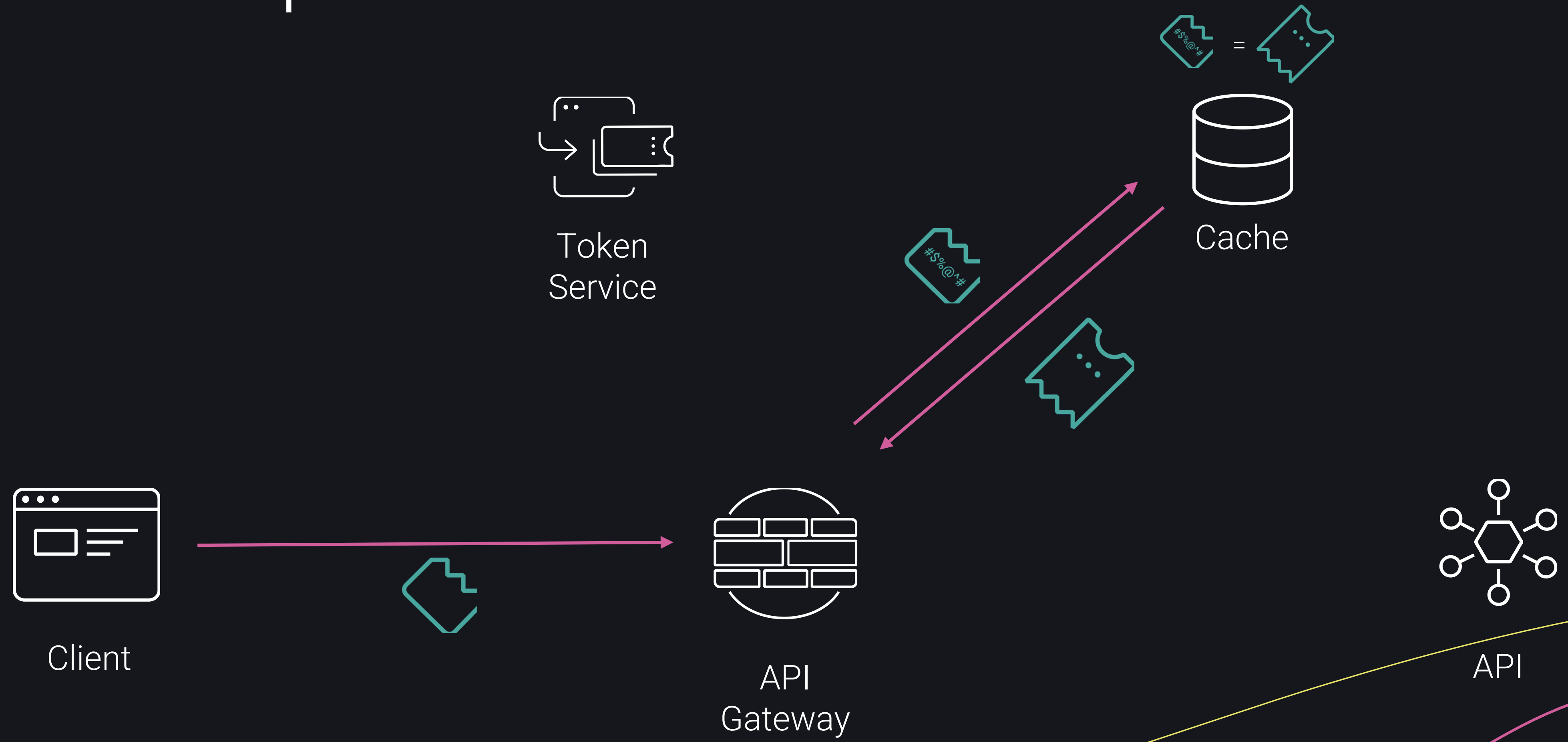
API

The Split Token Flow



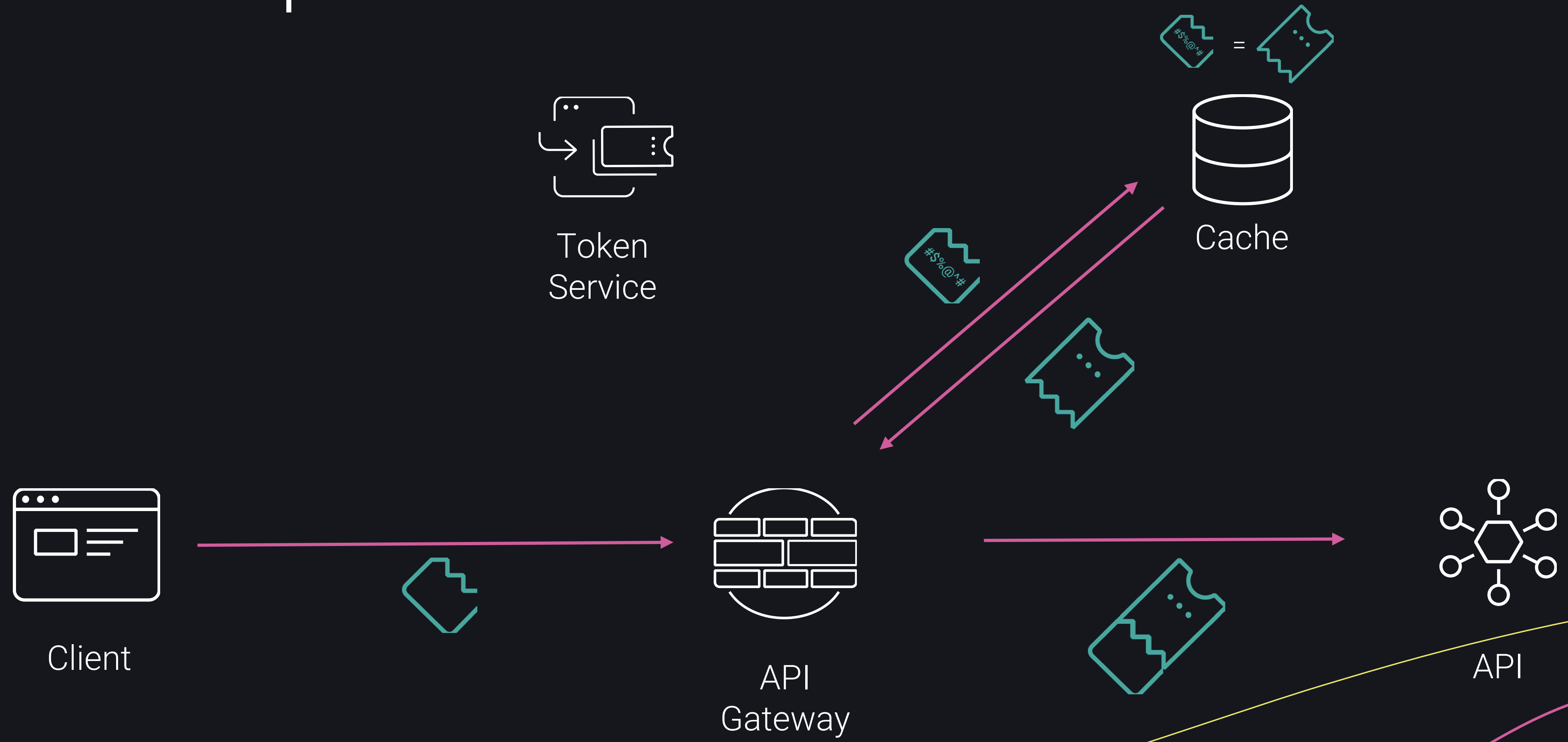
CURITY

The Split Token Flow



CURITY

The Split Token Flow



CURITY

Summary

- FAPI, for higher security environments, not just banks
- Safeguard against usage of stolen/lost tokens
- Protect request and response using PAR & JARM
- Decoupled user authentication flow (CIBA)
- Prevent confidential data from leaking or being misused

Thank You!

curity.io

developer.curity.io

@curityio

info@curity.io

CURITY