Security mechanisms in OAuth and OpenID Connect

Luci André Knudsen, Identity Day Norway, 2026-04-16

Luci André Knudsen

they/them

- IAM wizard/Sr. Security Engineer @ Aidn
- Lecturer @ University of Agder
- Dog @ the Internet

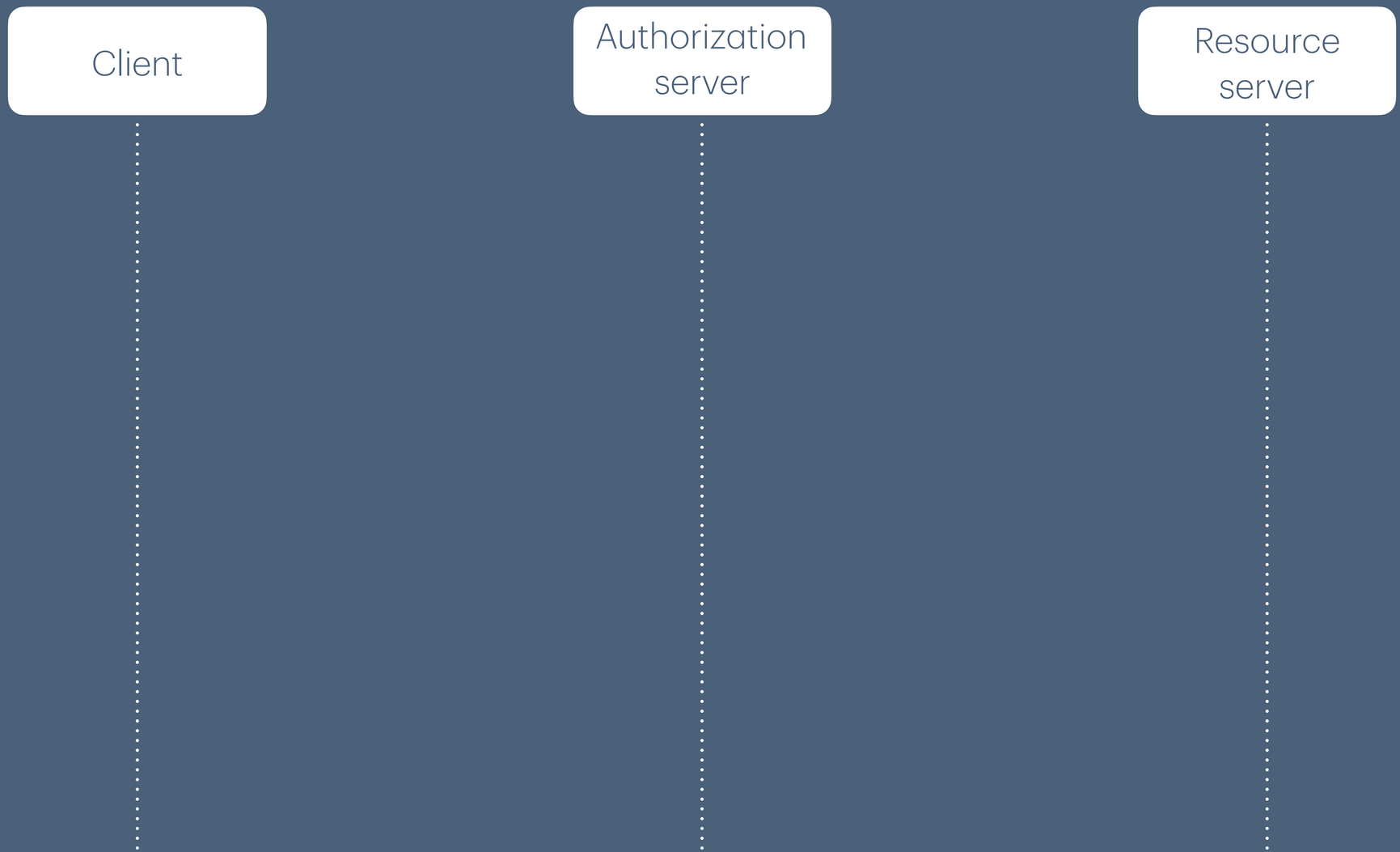
aidn

Agenda

- Recap of OAuth and OpenID Connect
- OAuth and OIDC security
- Attack vectors and mitigation

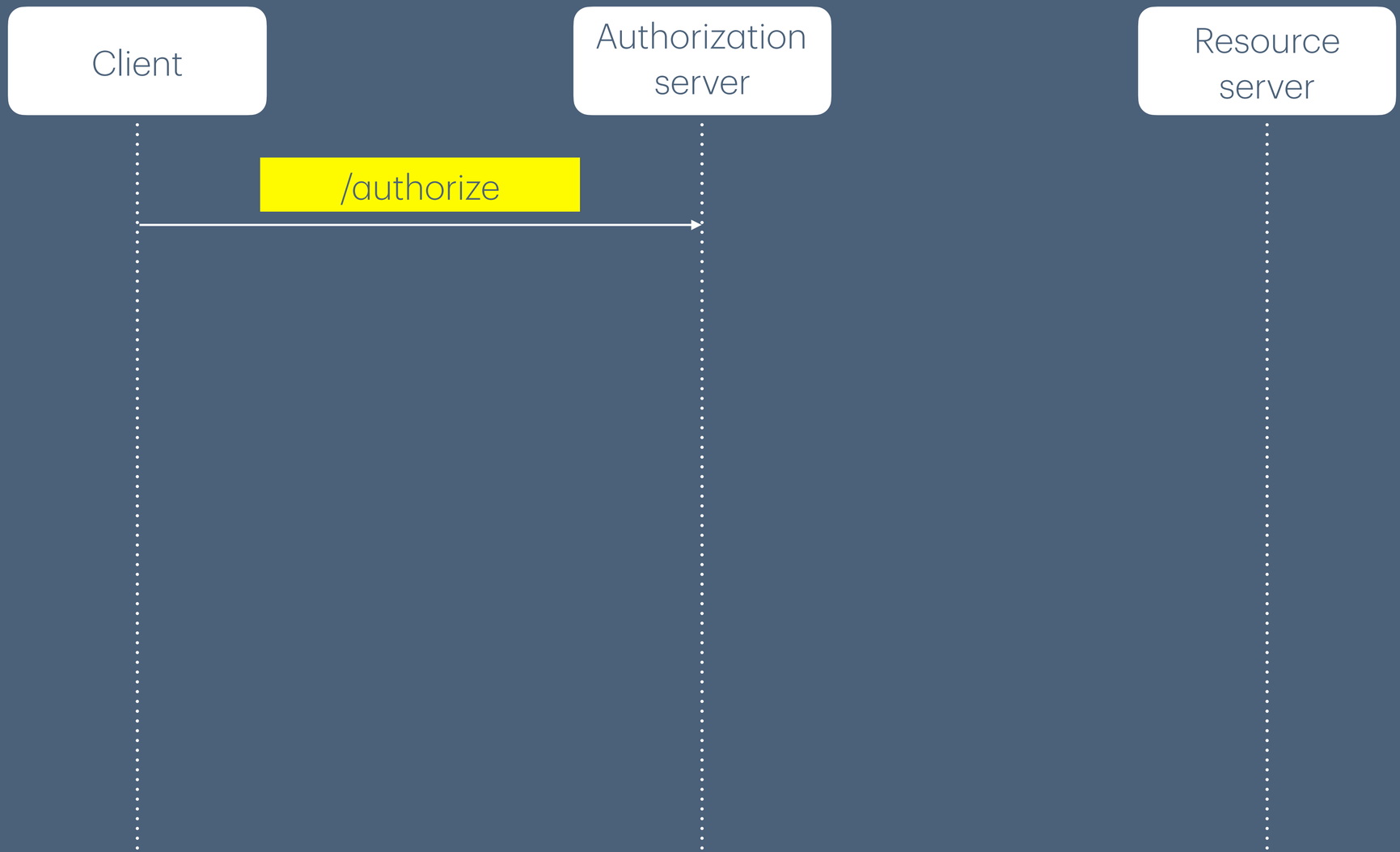
Recap of OAuth

Authorization code grant



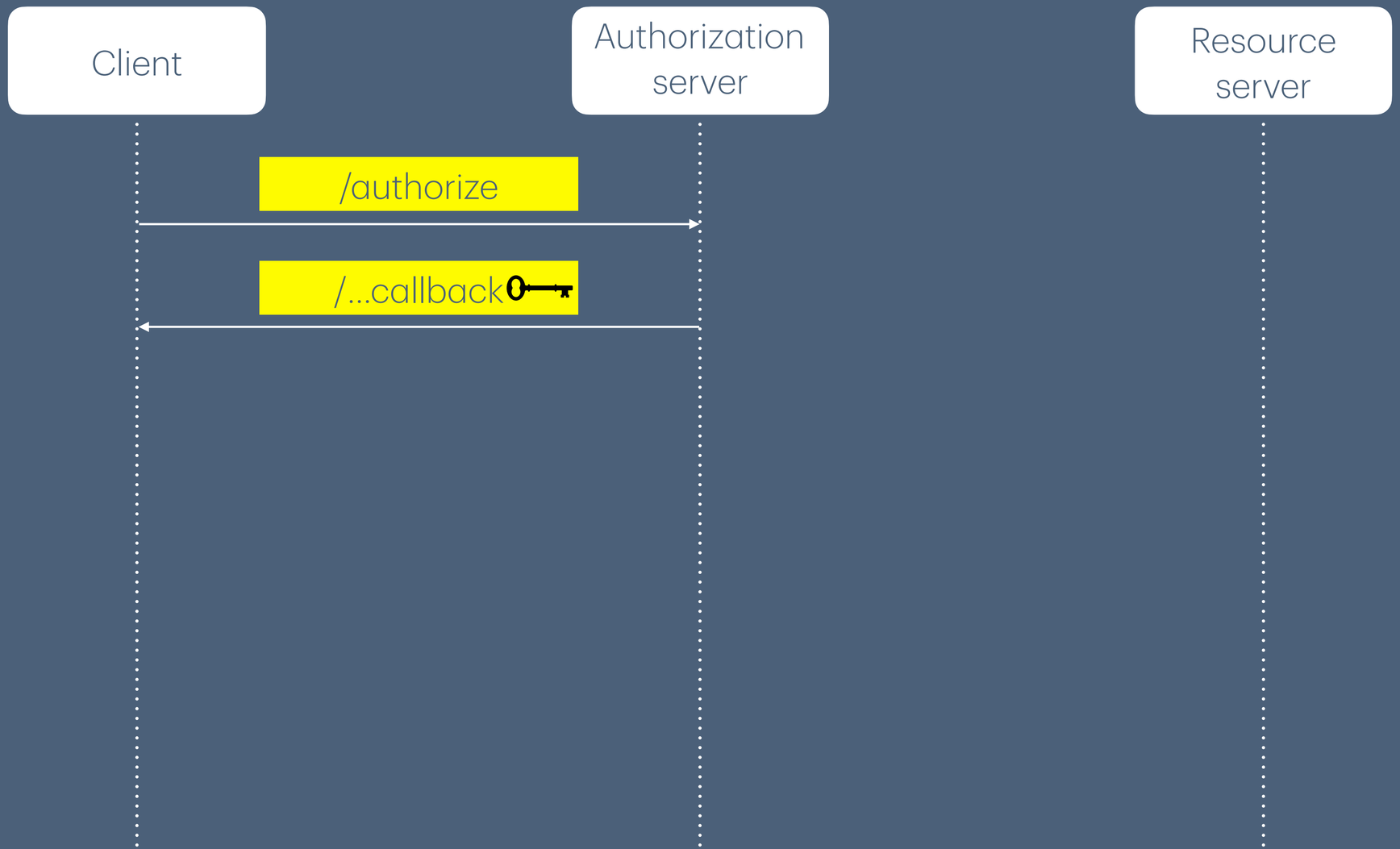
Recap of OAuth

Authorization code grant



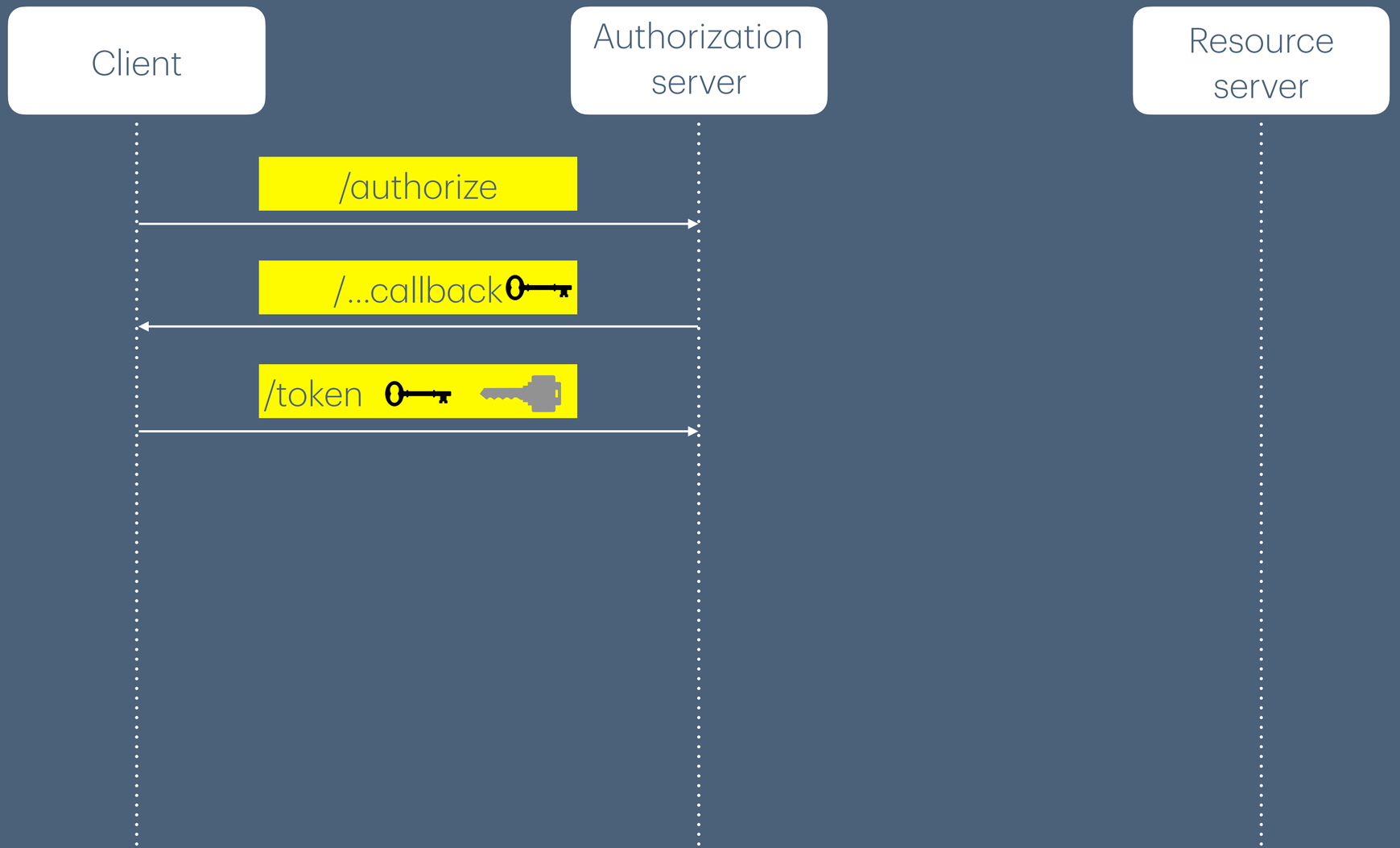
Recap of OAuth

Authorization code grant



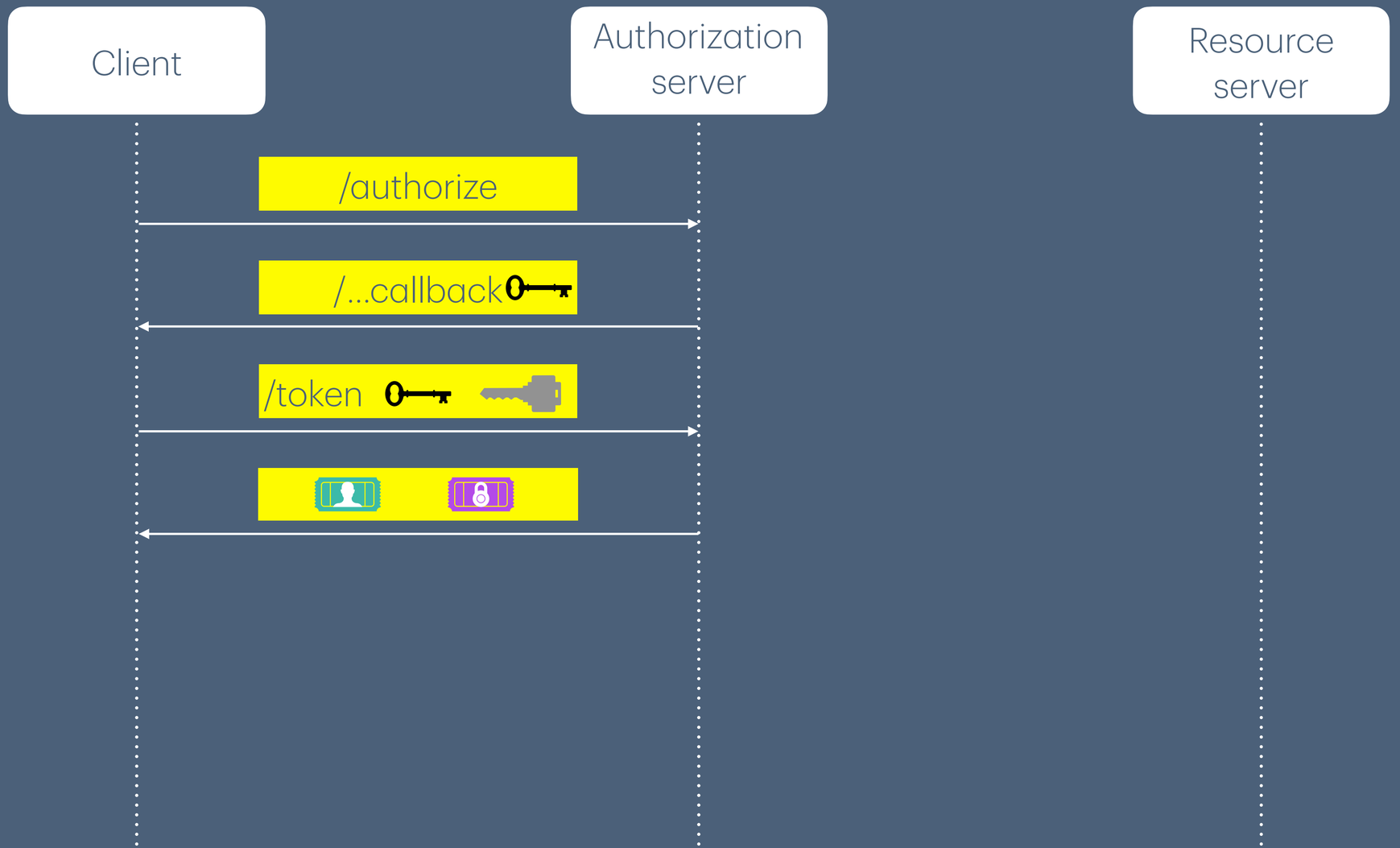
Recap of OAuth

Authorization code grant



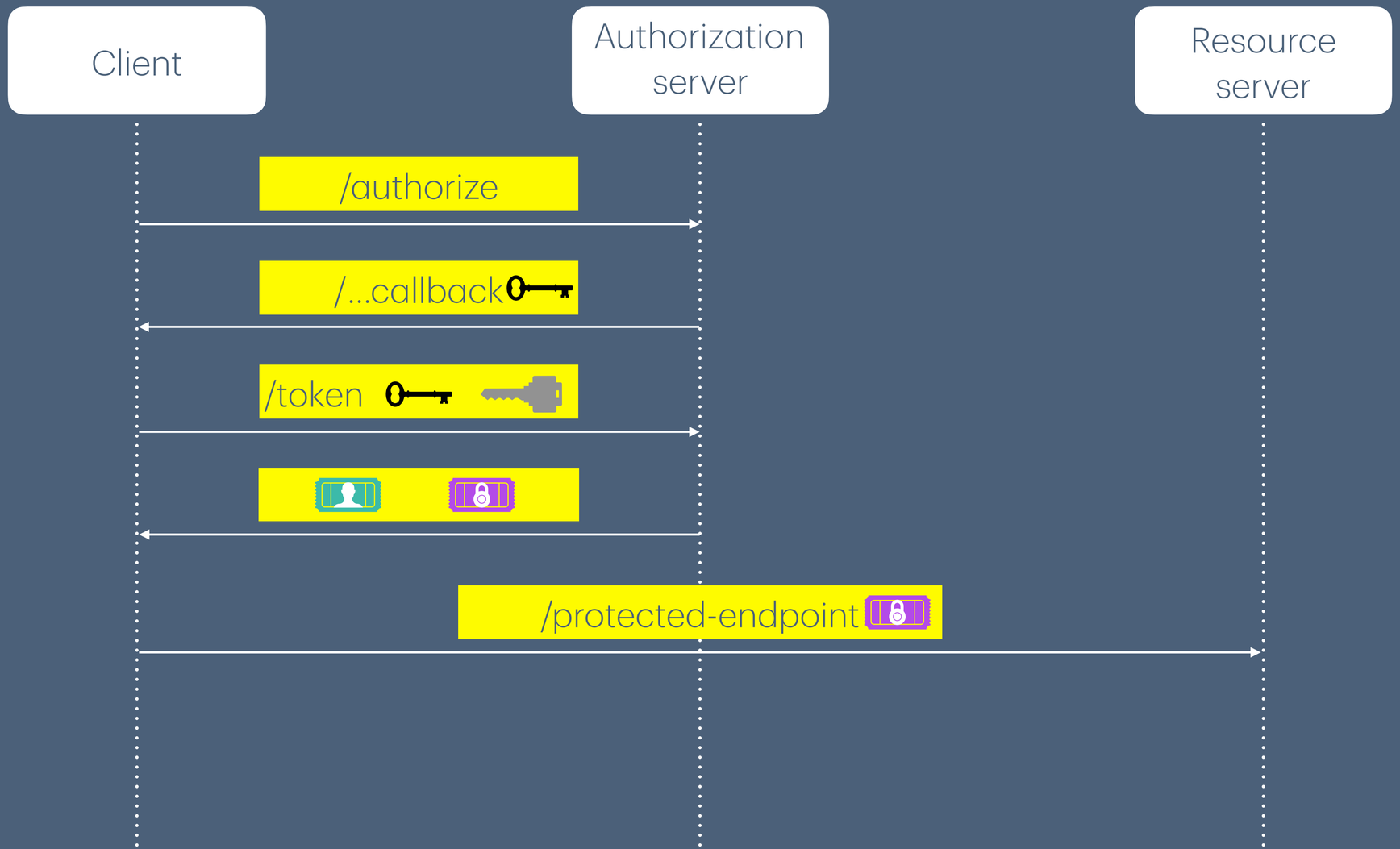
Recap of OAuth

Authorization code grant



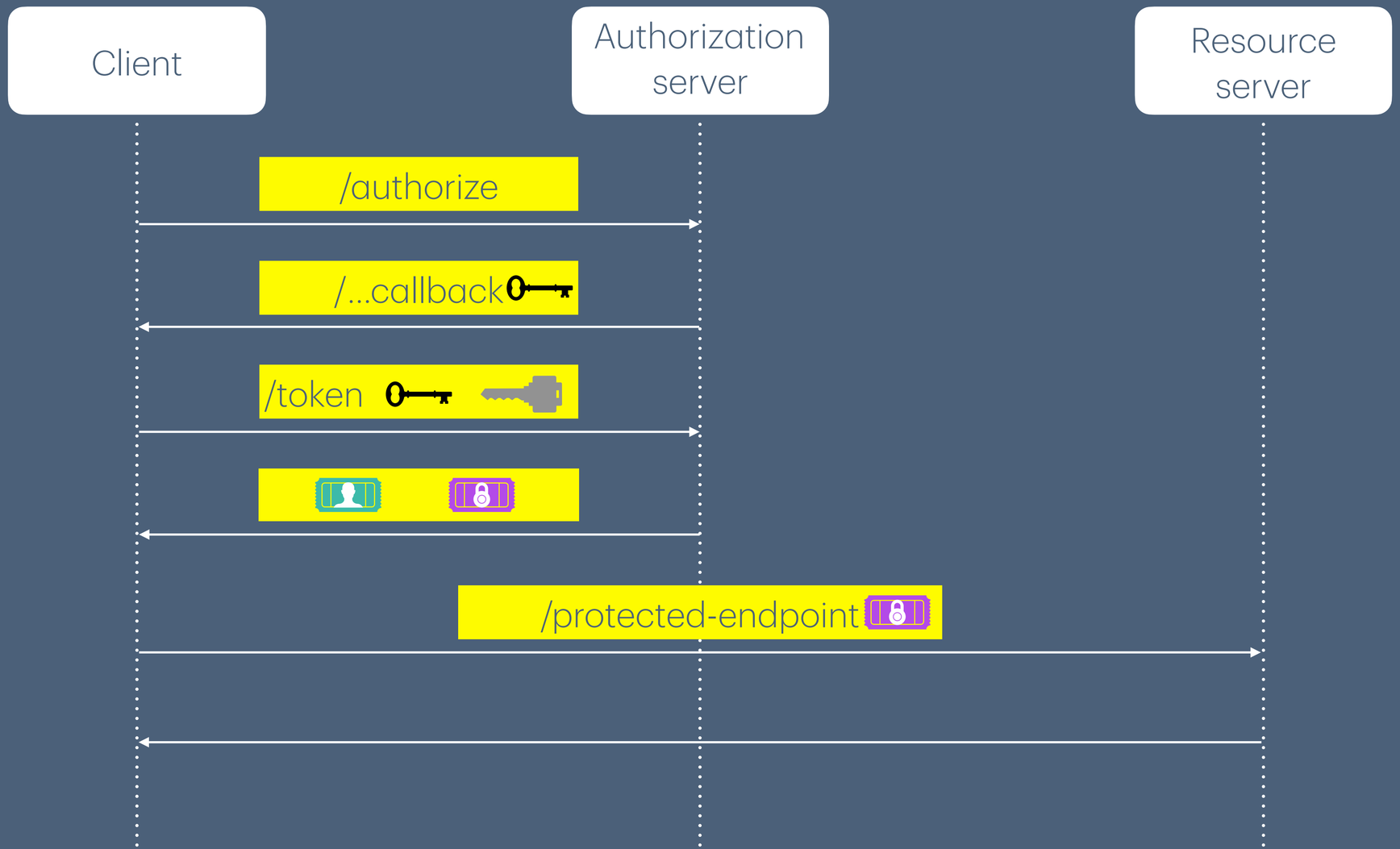
Recap of OAuth

Authorization code grant



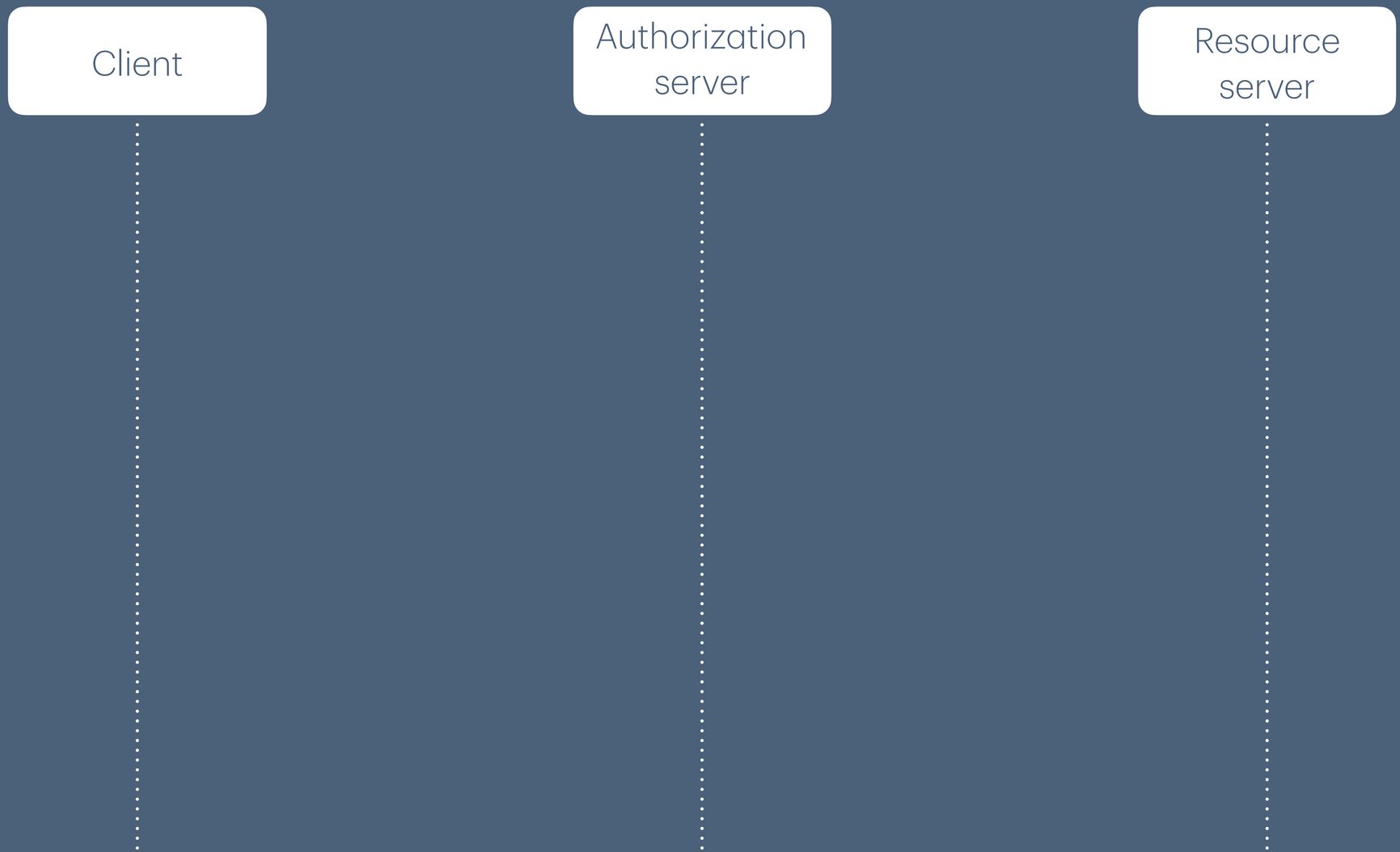
Recap of OAuth

Authorization code grant



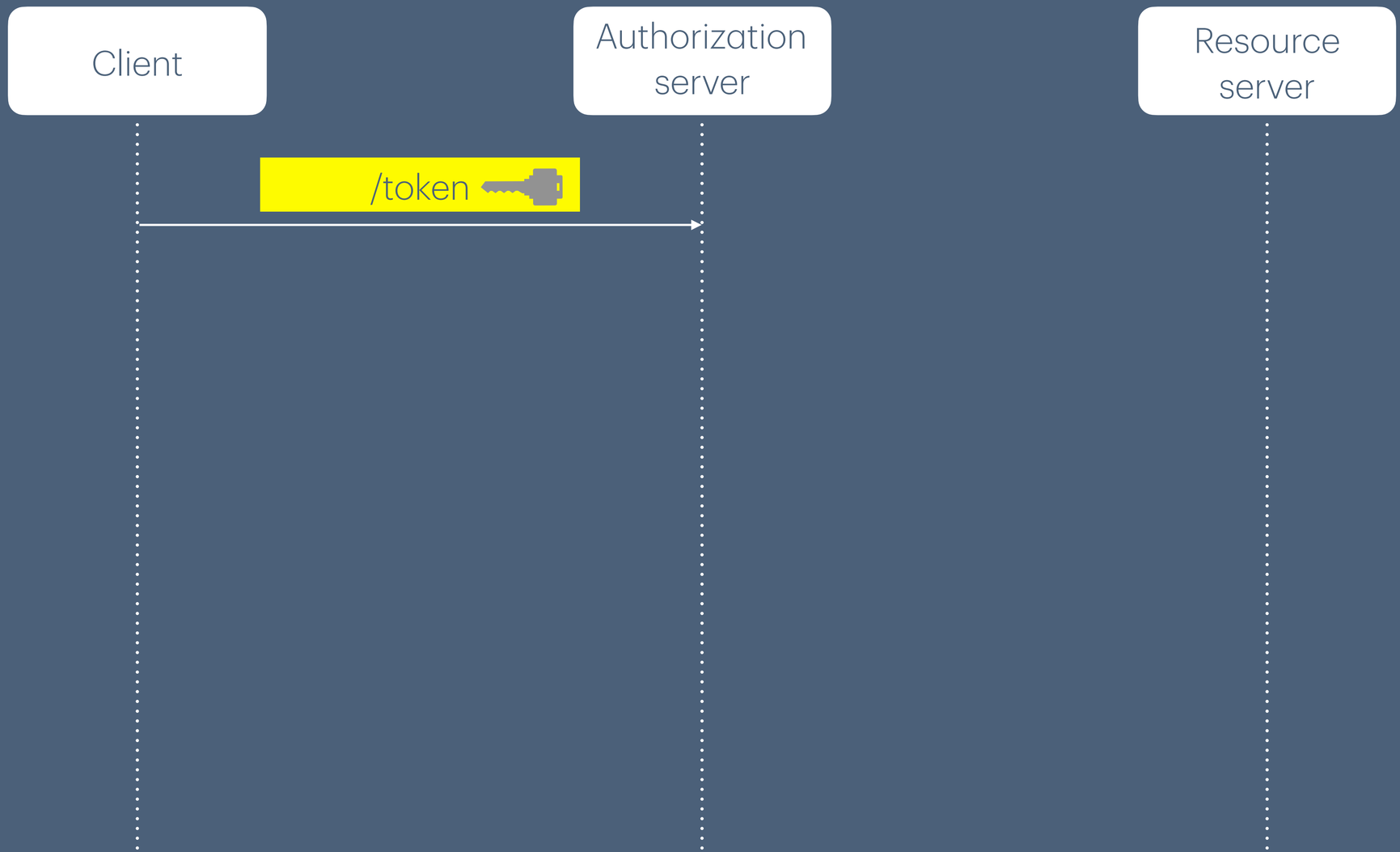
Recap of OAuth

Client credentials grant



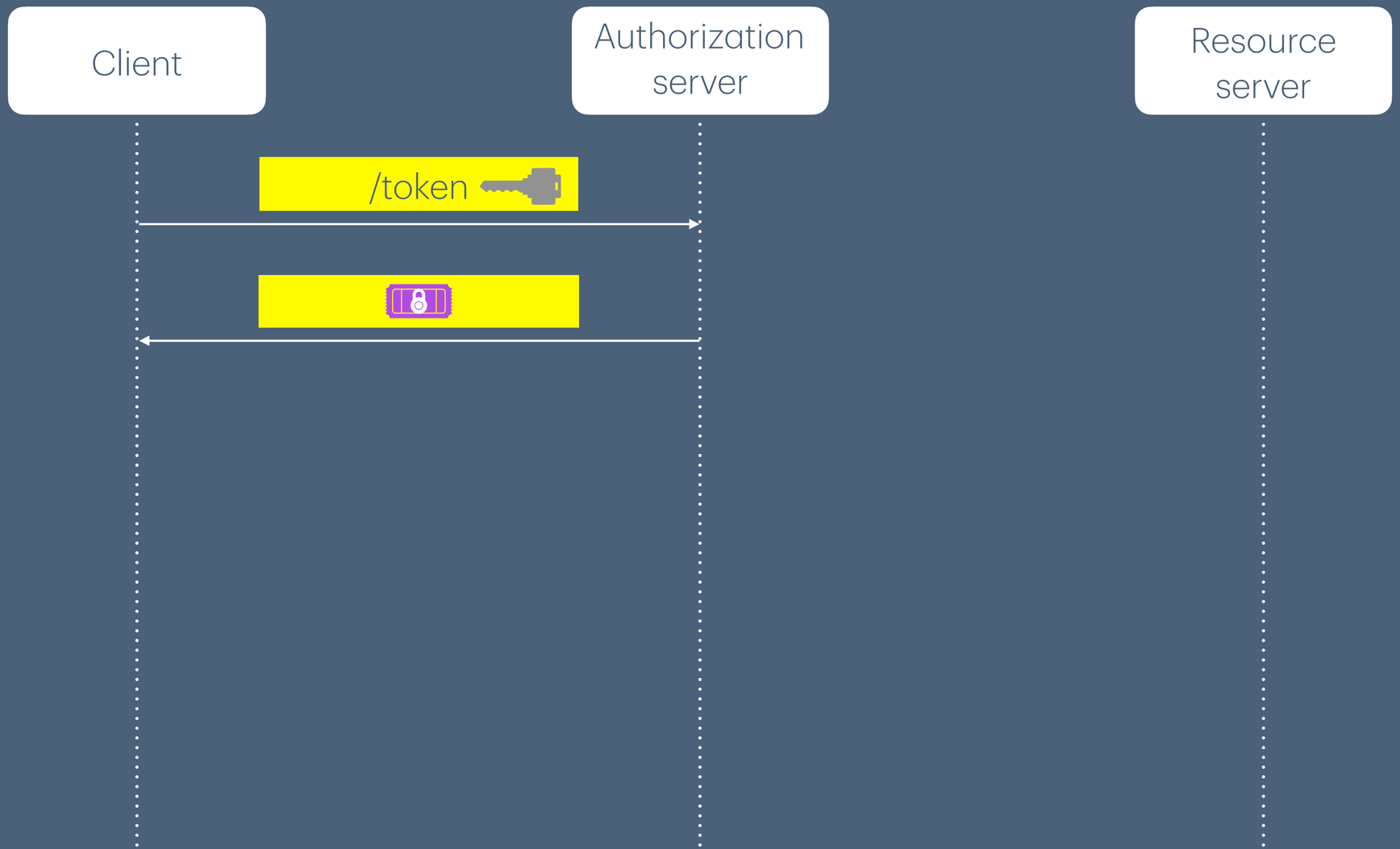
Recap of OAuth

Client credentials grant



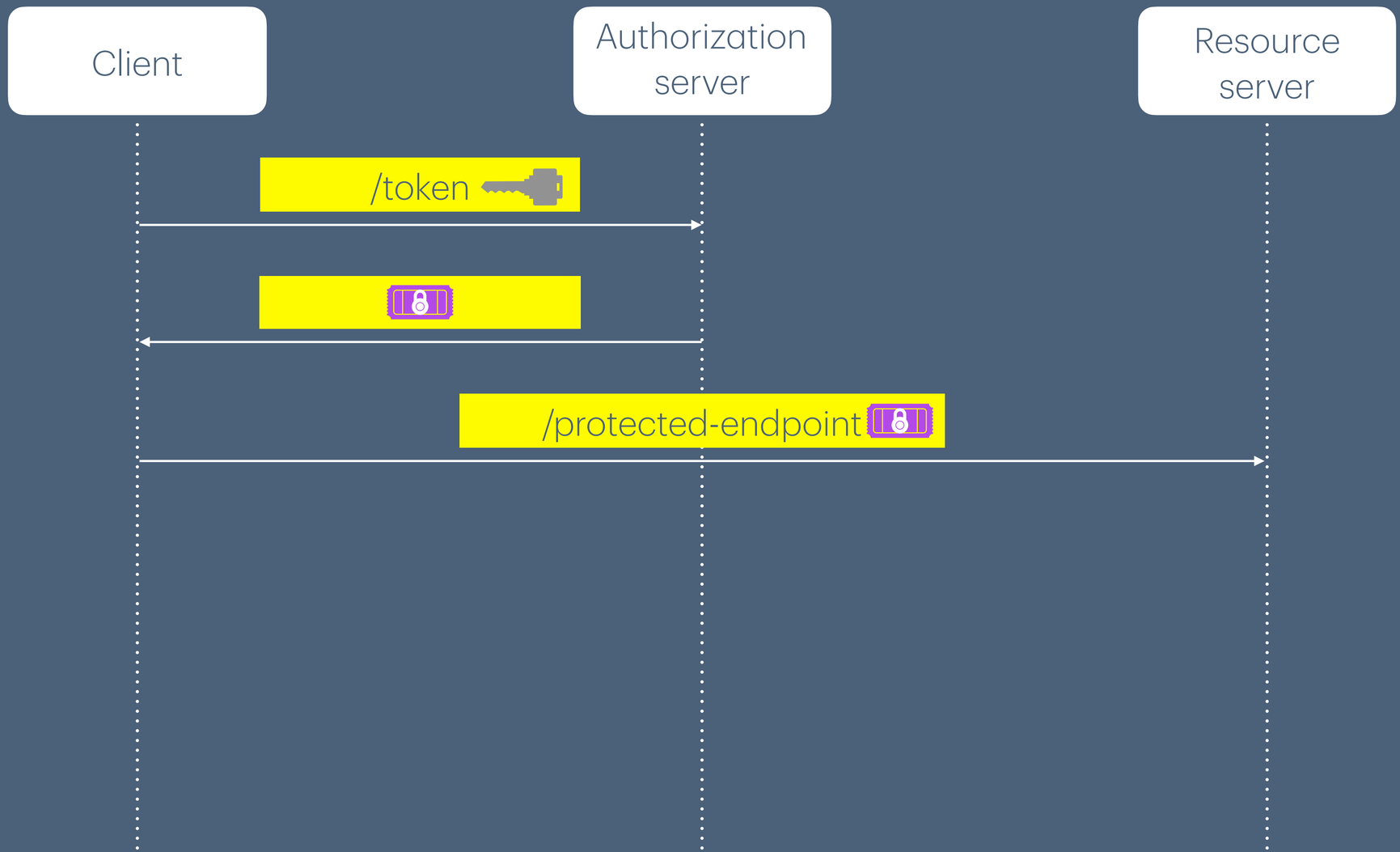
Recap of OAuth

Client credentials grant



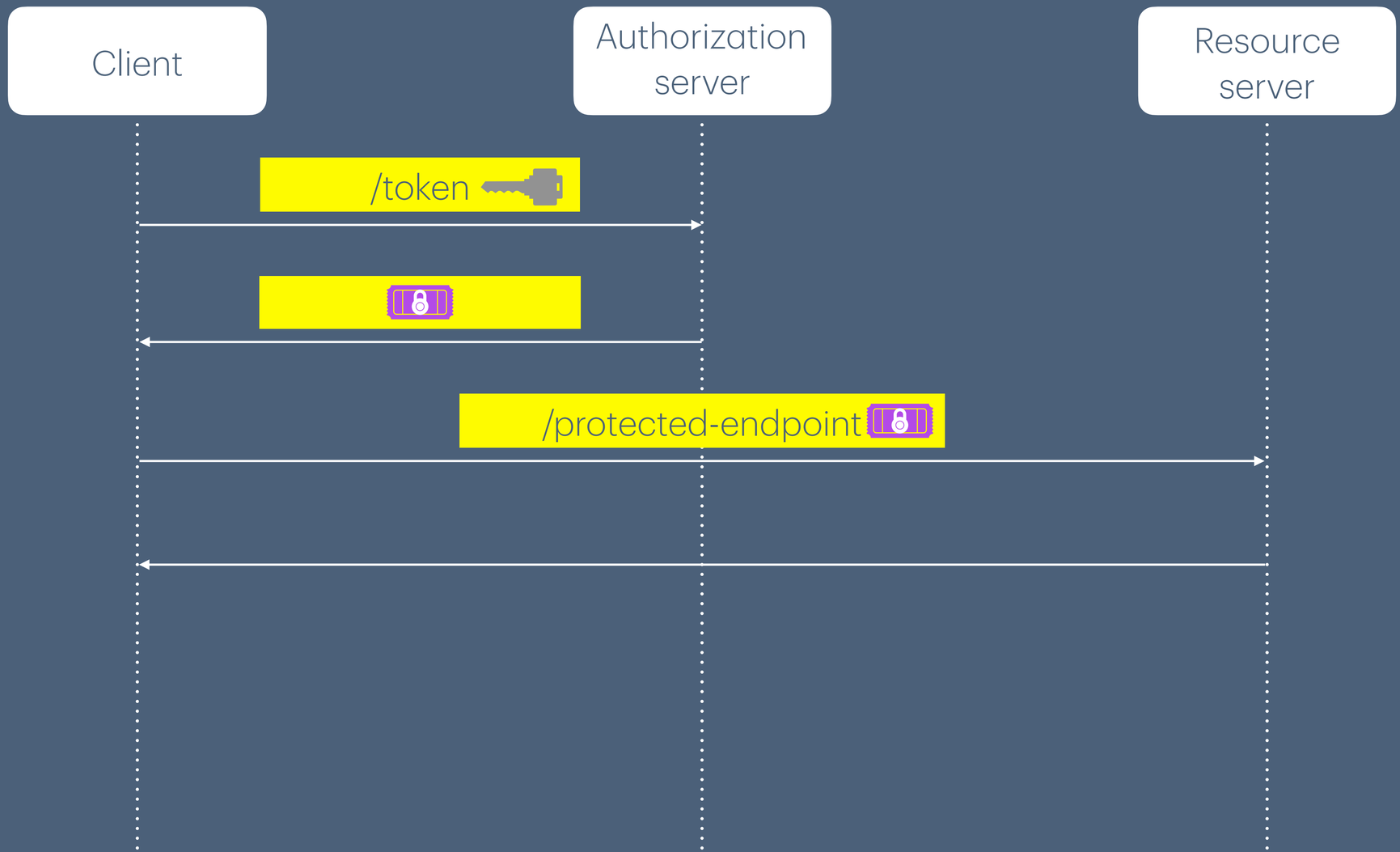
Recap of OAuth

Client credentials grant



Recap of OAuth

Client credentials grant



OAuth and OIDC Security

- OAuth and OIDC is HUGE!
- Status per February 2025.

Quantifying the Explosion
(final and near-final specs)

- 31 OAuth specs
- 13 JOSE & JWT specs
- 13 OpenID Connect specs
- 2 other related IETF specs used by OpenID Connect
- 5 FAPI specs
- 1 MODRNA spec
- 3 eKYC-IDA specs
- 4 SecEvent specs
- 4 Shared Signals specs
- 9 Wallet specs
- 11 COSE specs
- 2 Passwordless Login specs
- 3 Zero-Knowledge Proof specs

- **101 OAuth & OpenID-related specs!**

- And more on the way...
  - 12 active OAuth WG specs, etc.

Michael B. Jones, *"The Cambrian Explosion of OAuth and OpenID Specifications"*, 2025

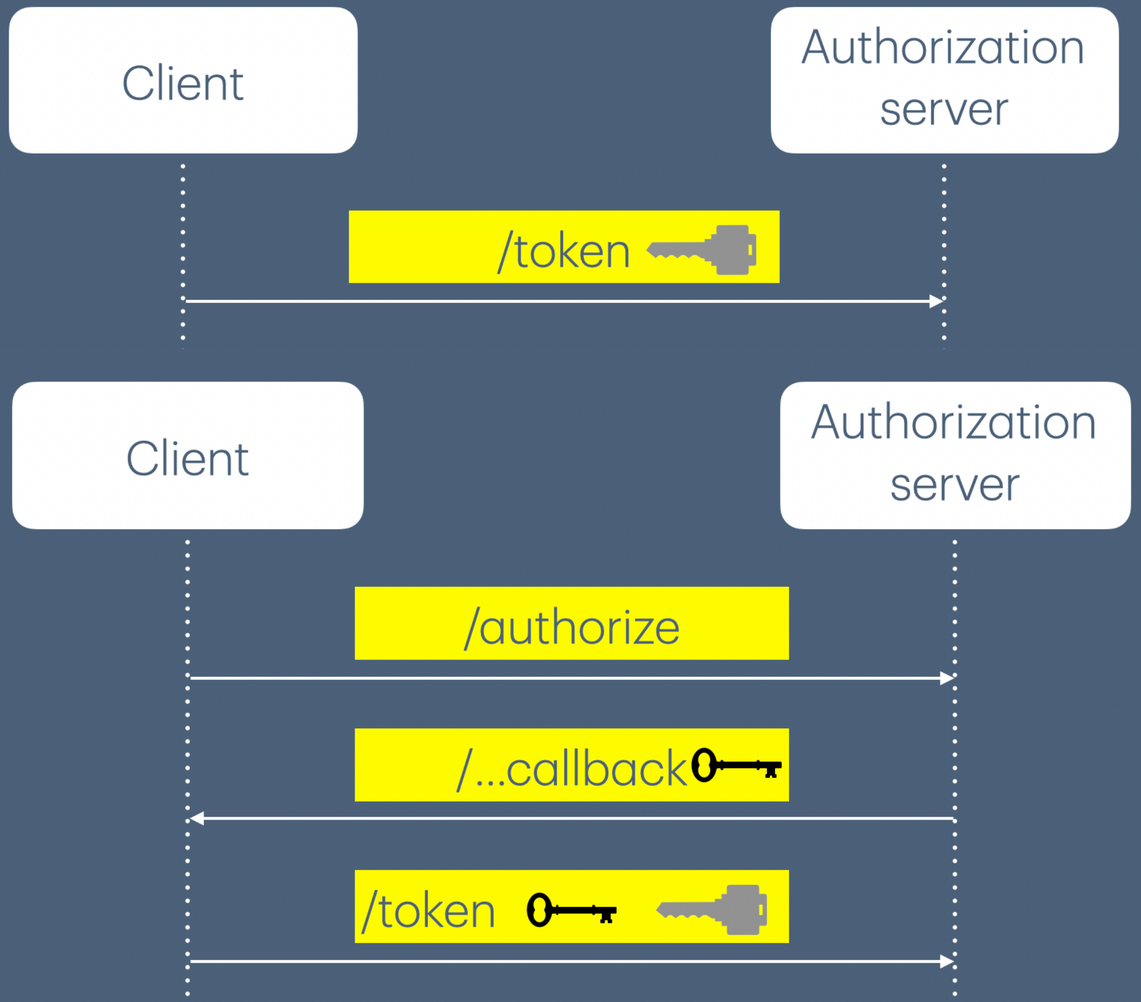
OAuth and OIDC Security

- RFC 9700: Best Current Practice for OAuth 2.0 Security
- FAPI 2.0 Security Profile

Client authentication

private_key_jwt

- In the base framework client authentication is done with a client password.
- Using *private_key_jwt* the authorization server only needs the public key of a client.
- **No more shared secrets!**

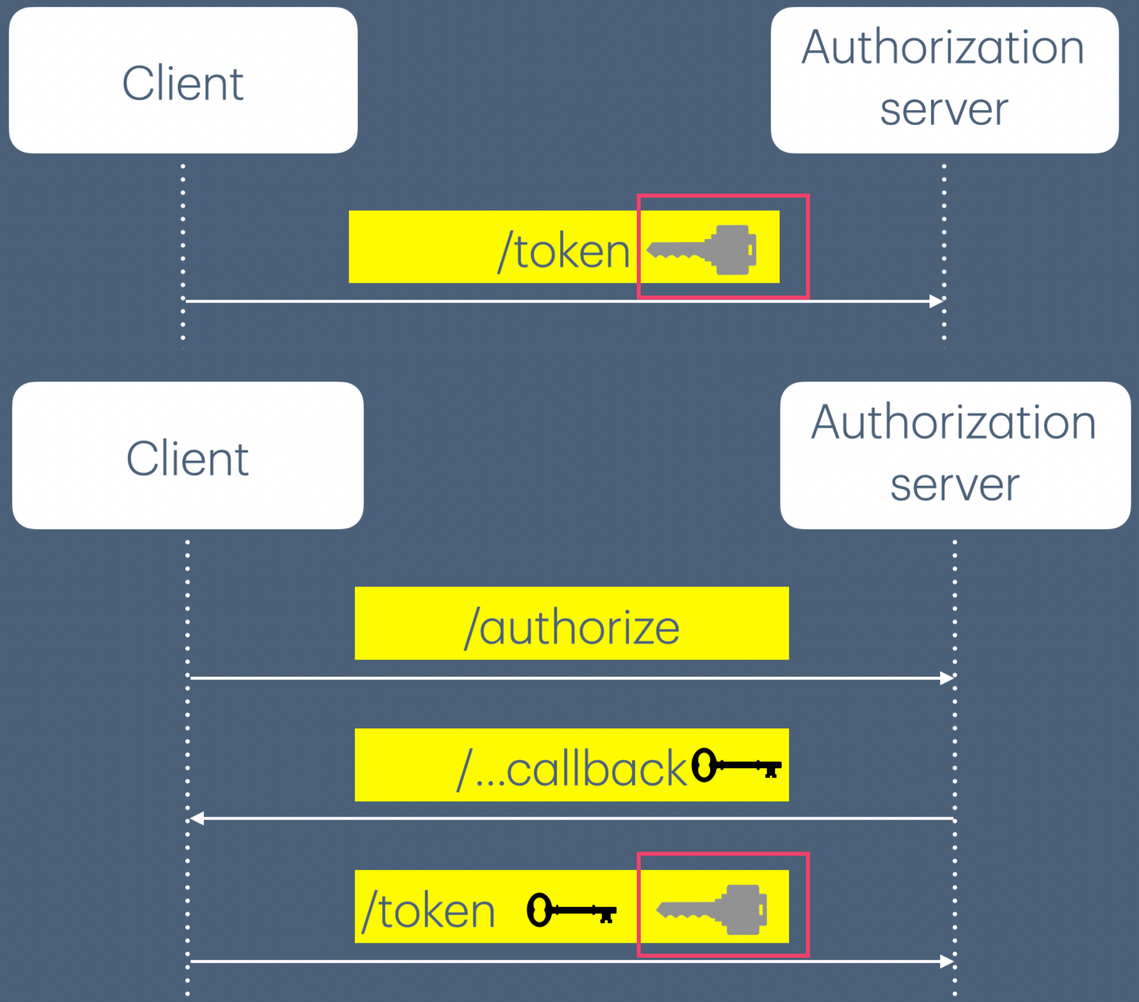


RFC 7521, RFC 7523

Client authentication

private_key_jwt

- In the base framework client authentication is done with a client password.
- Using *private_key_jwt* the authorization server only needs the public key of a client.
- **No more shared secrets!**



RFC 7521, RFC 7523

Client authentication

private_key_jwt

```
grant_type=client_credentials&client_id=myClient&client_secret=secretValue
```

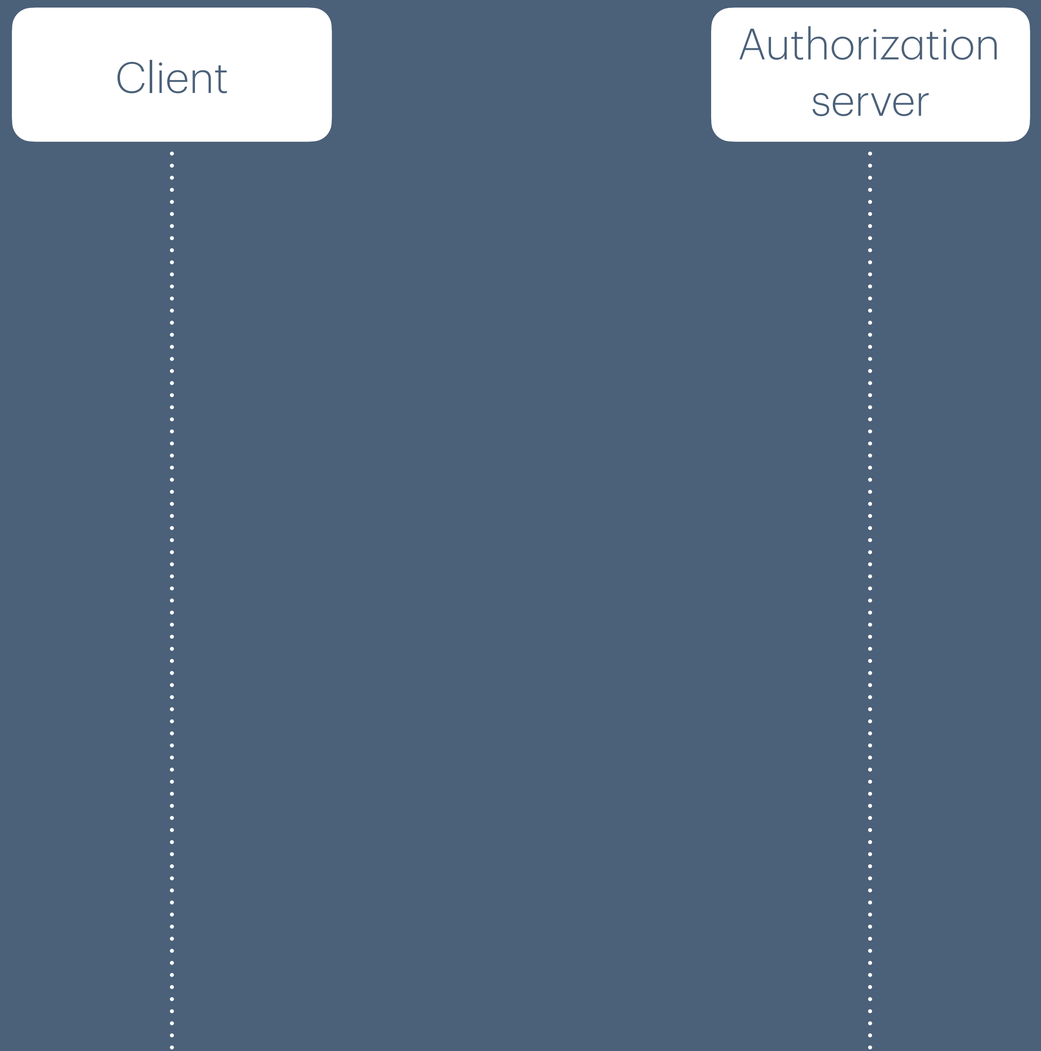
```
{
  "alg": "RS256",
  "kid": "5DC654C985D1037A16D82FCA3B9B8843",
  "typ": "client-authentication+jwt"
}.{
  "sub": "myClient",
  "iat": 1741157397,
  "jti": "9fef2cb095b54f4aa29ee43fefba7778",
  "nbf": 1741157397,
  "exp": 1741157403,
  "iss": "myClient",
  "aud": "https://server.luci.no"
}.[Signature]
```

```
grant_type=client_credentials&
client_assertion_type=urn:ietf:params:oauth:
client-assertion-type:jwt-bearer&
client_assertion=eyJhbGciOiJSUzI[...].
eyJzdWIi[...].
b78MIzFC[...]
```

RFC 7521, RFC 7523

Authorization code injection

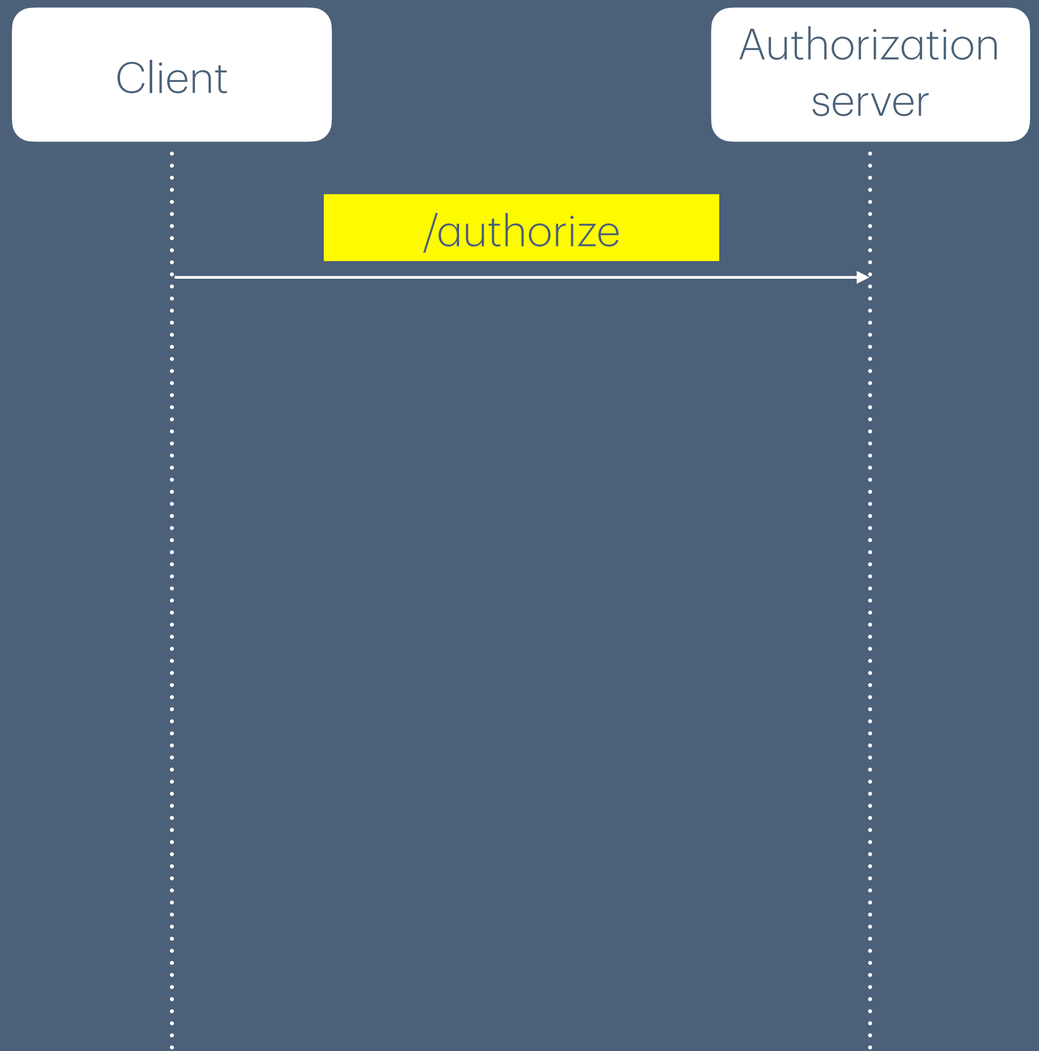
PKCE



RFC 7636

Authorization code injection

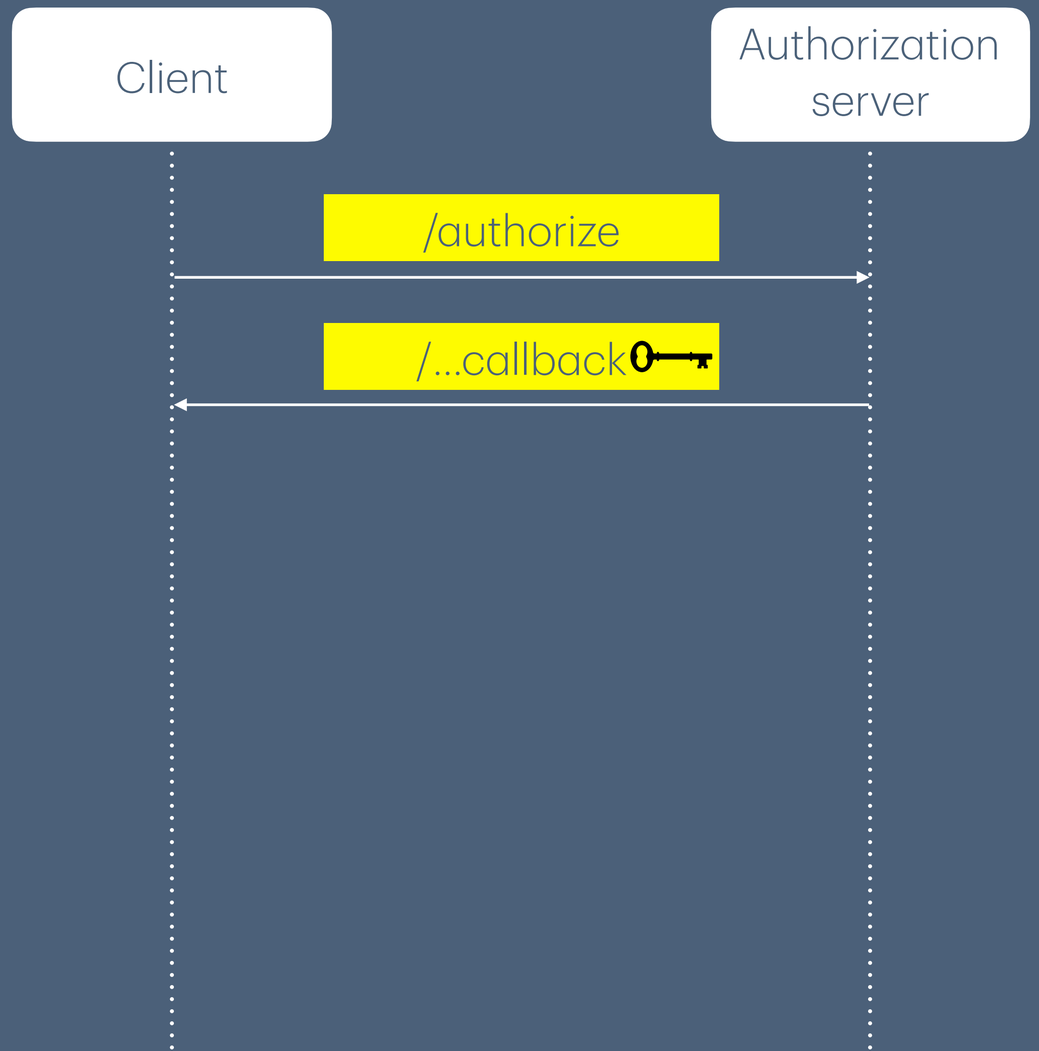
PKCE



RFC 7636

Authorization code injection

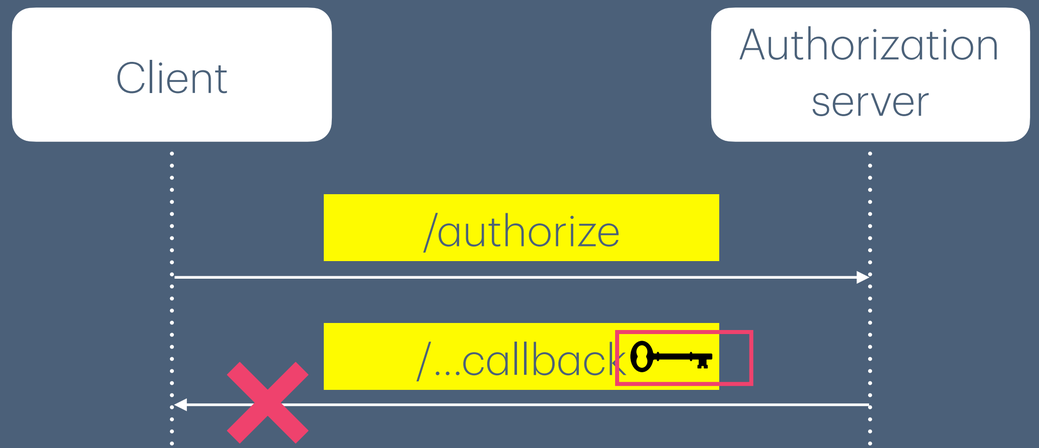
PKCE



RFC 7636

Authorization code injection

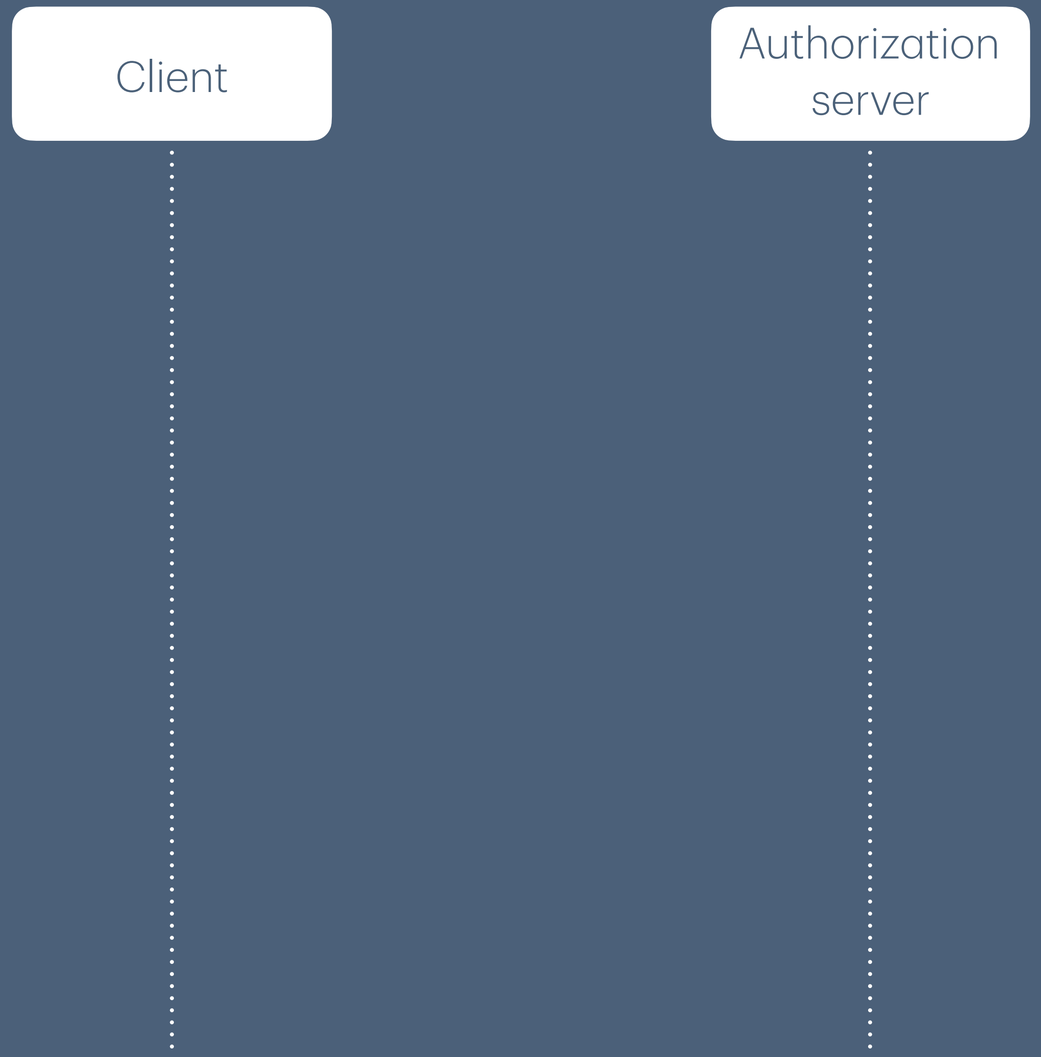
PKCE



RFC 7636

Authorization code injection

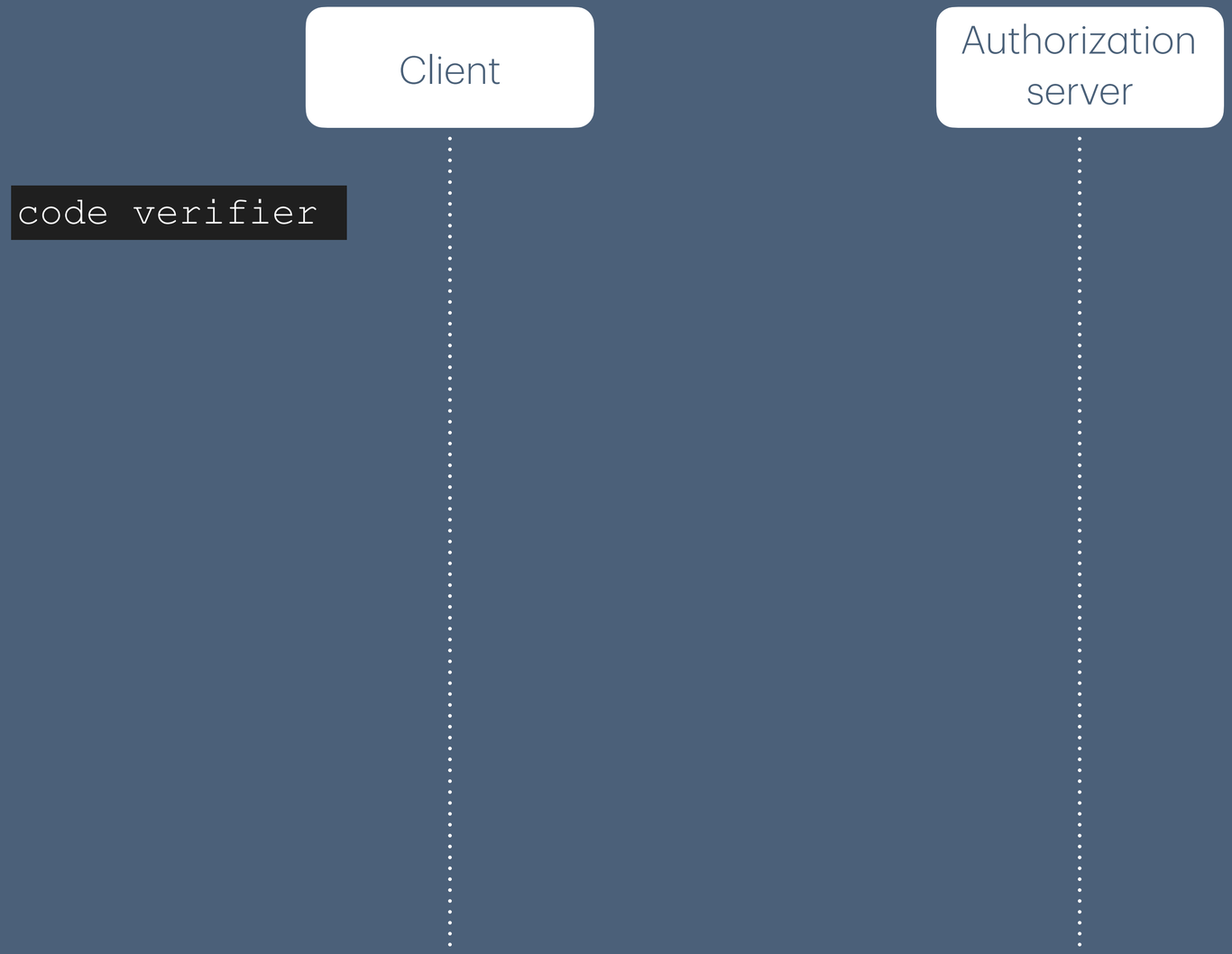
PKCE



RFC 7636

Authorization code injection

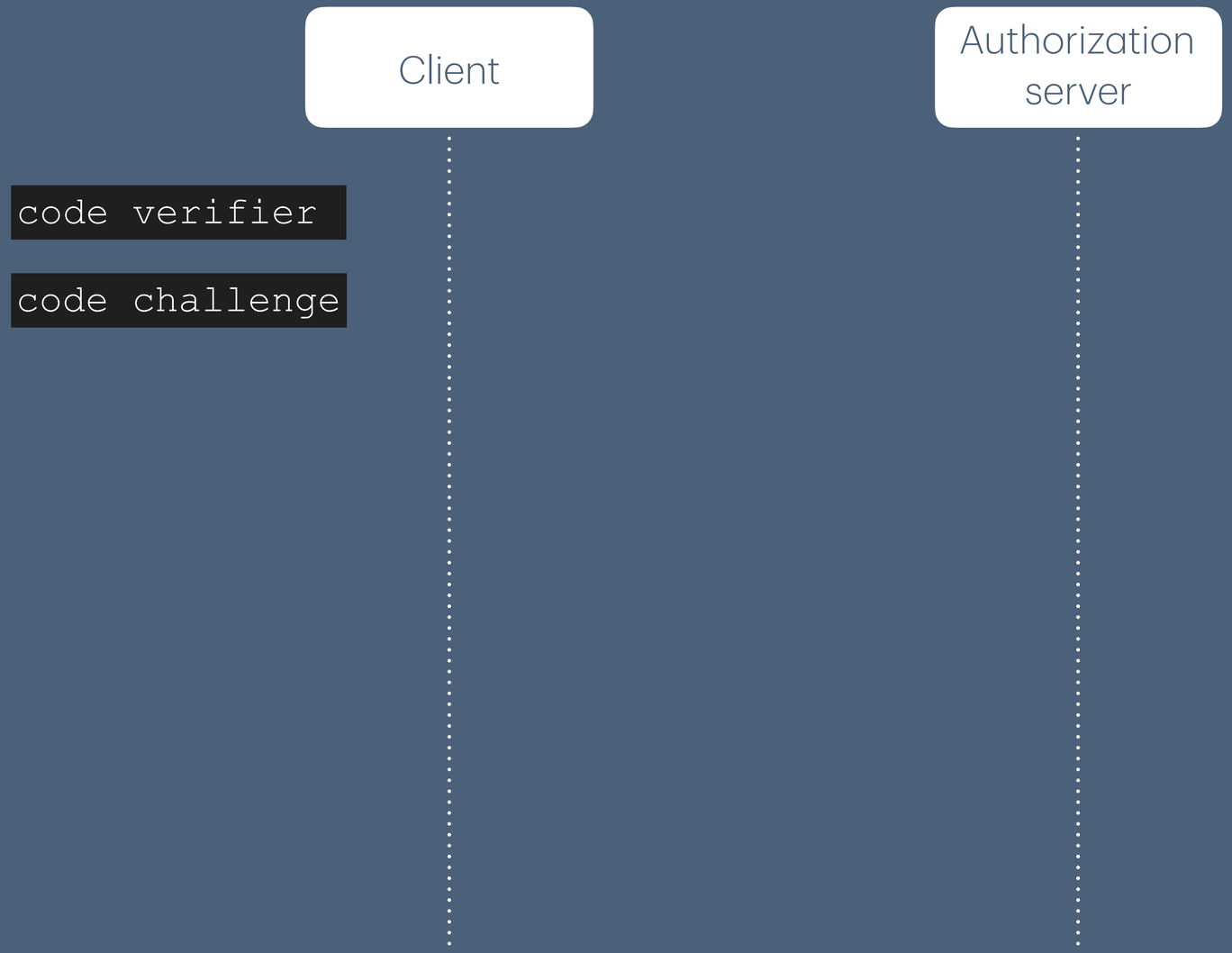
PKCE



RFC 7636

Authorization code injection

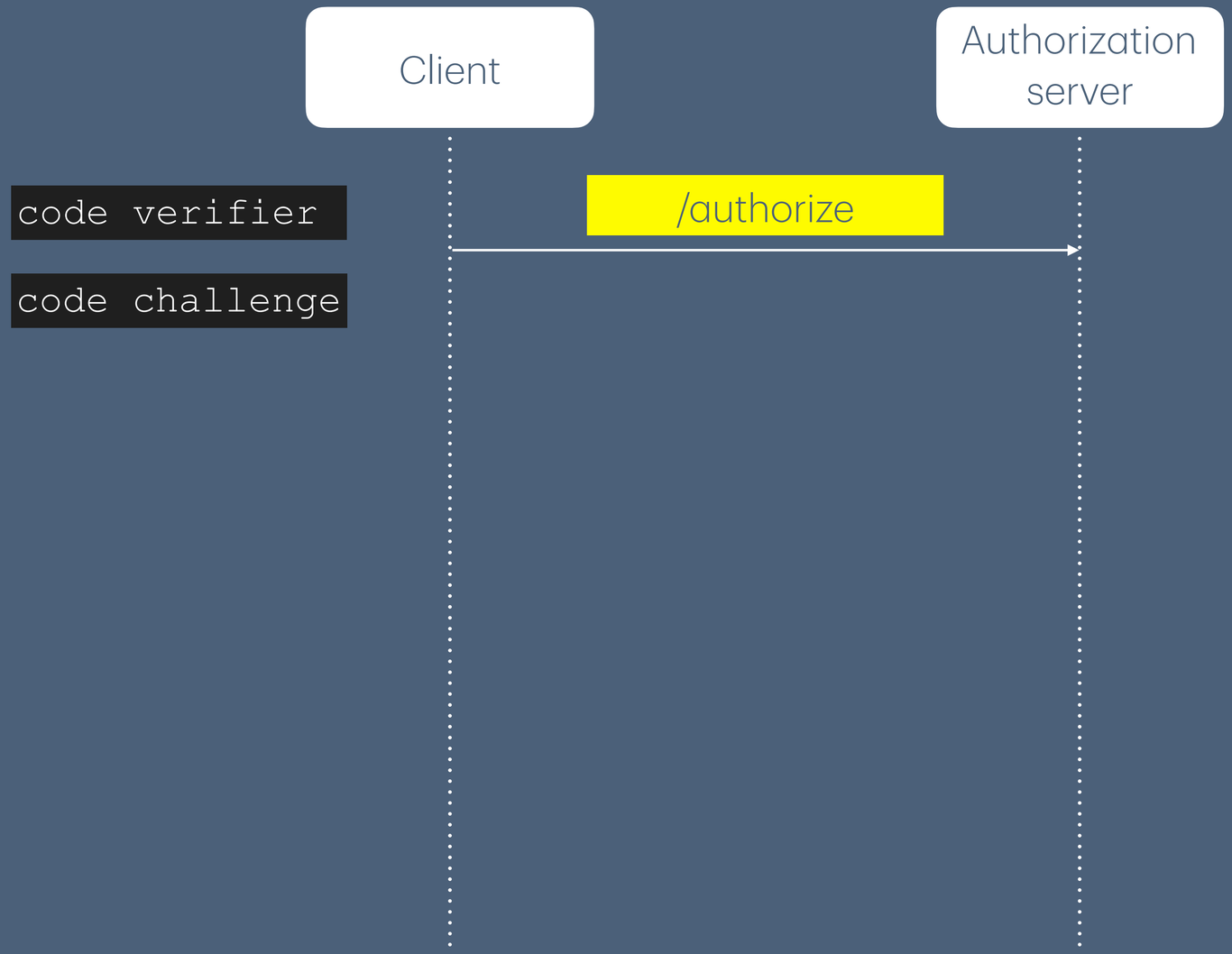
PKCE



RFC 7636

Authorization code injection

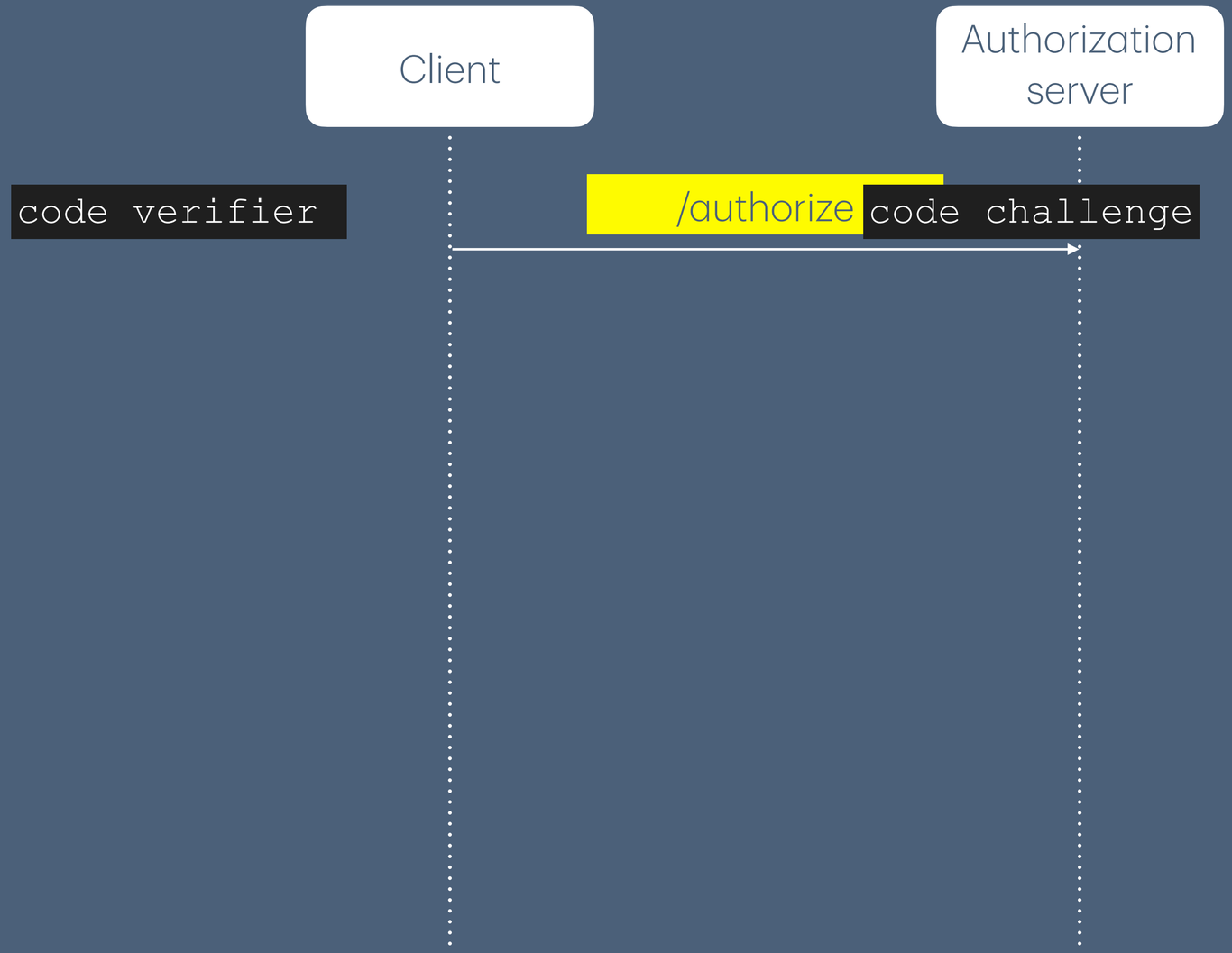
PKCE



RFC 7636

Authorization code injection

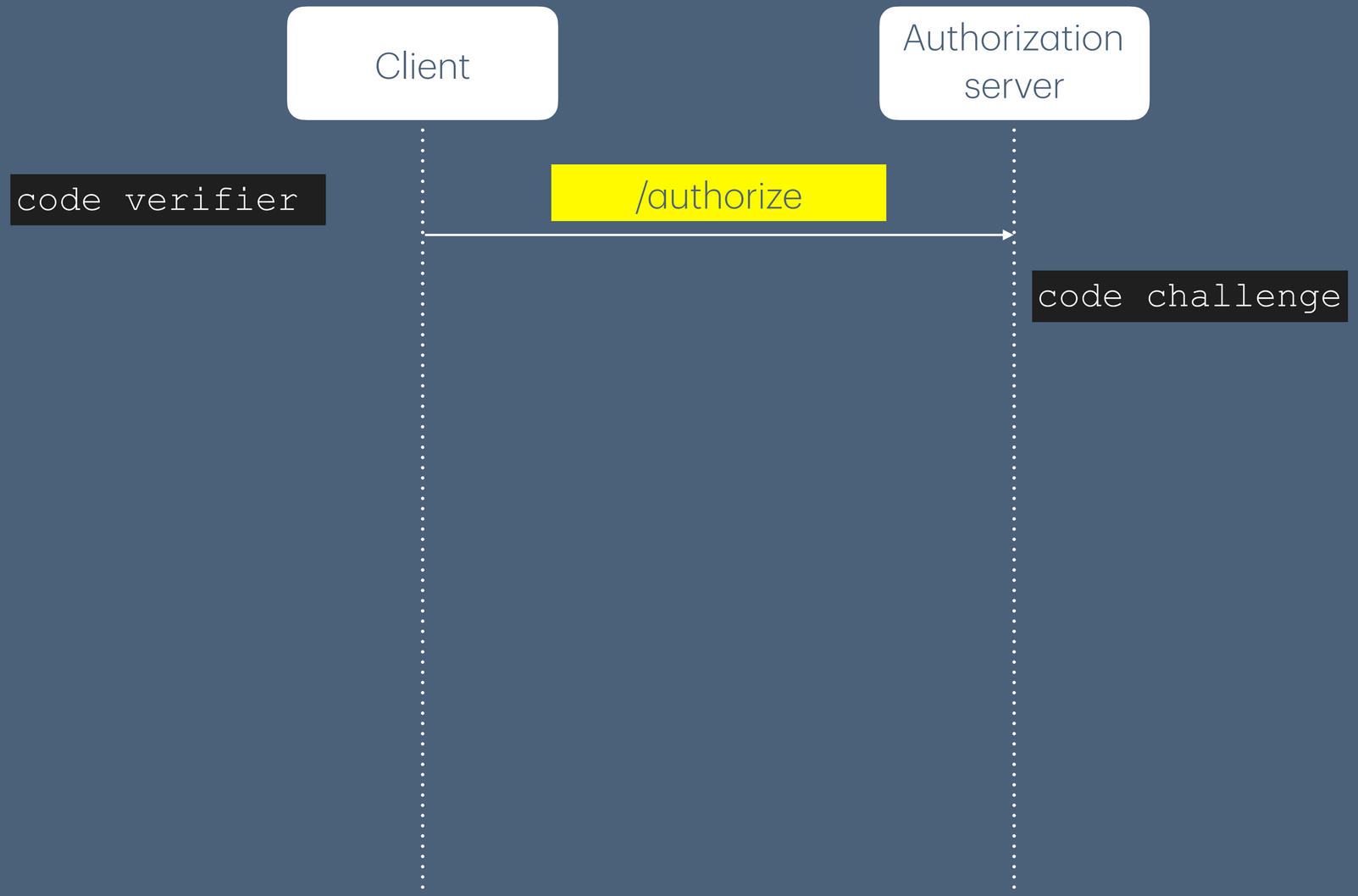
PKCE



RFC 7636

Authorization code injection

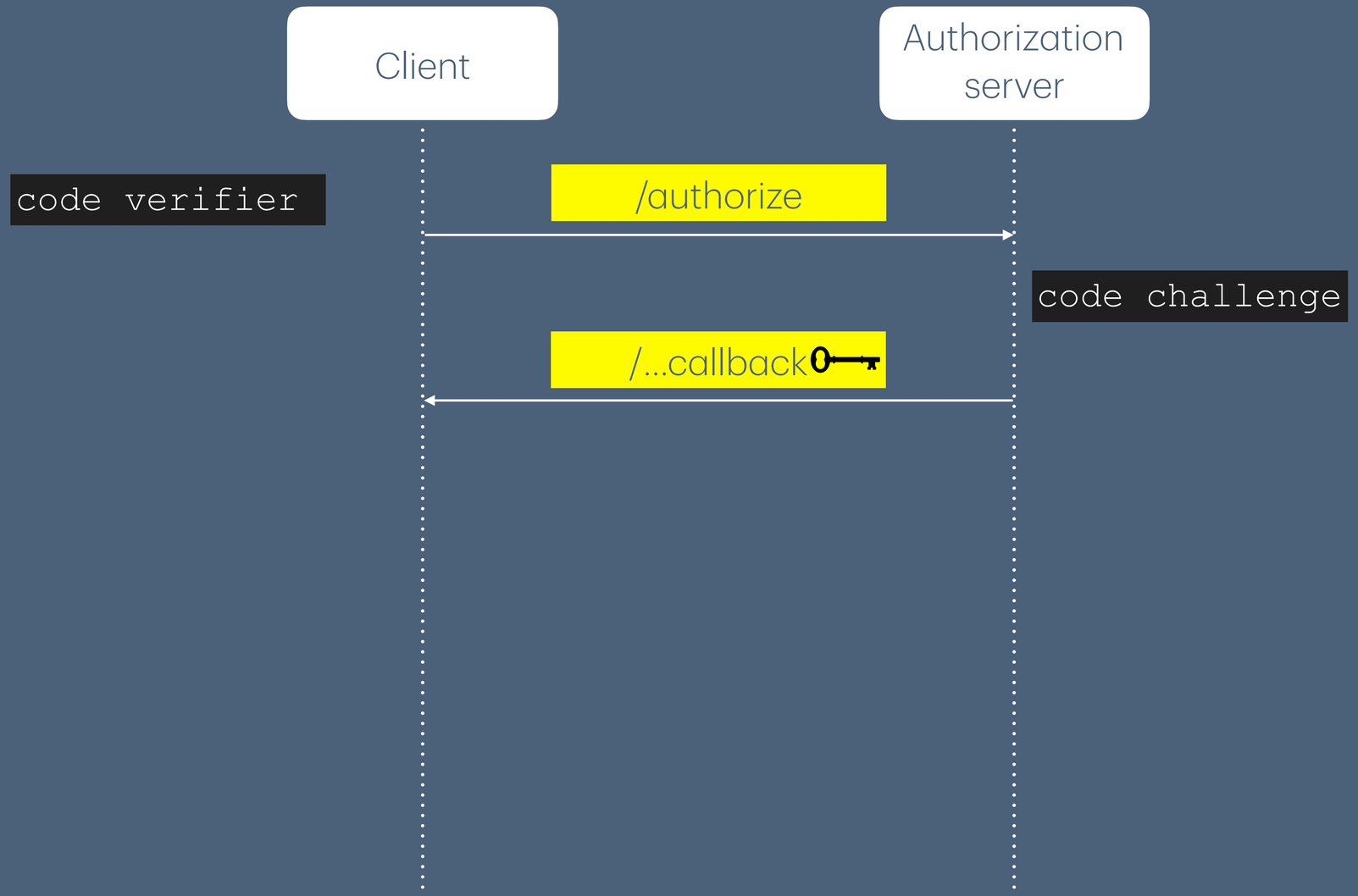
PKCE



RFC 7636

Authorization code injection

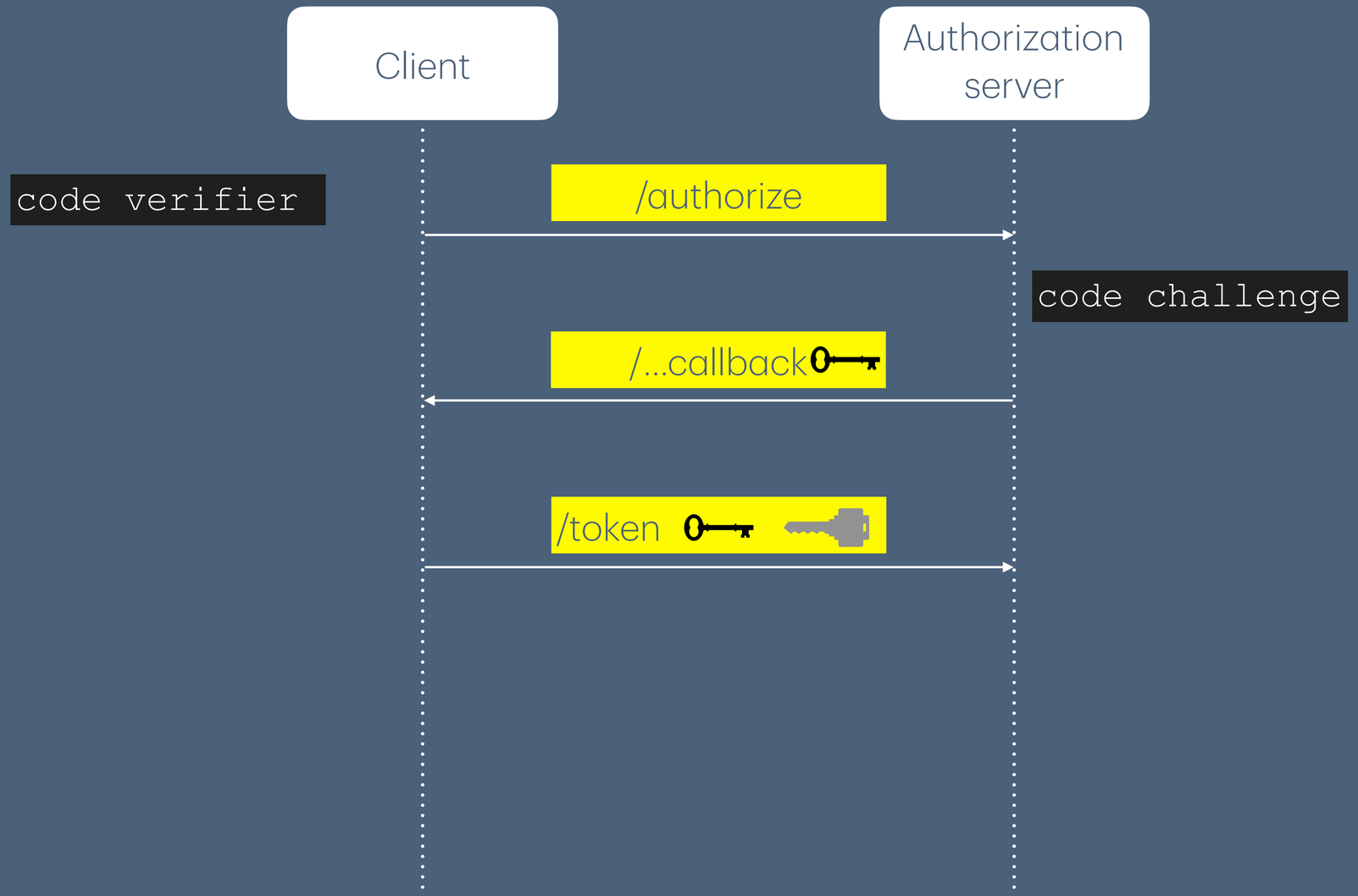
PKCE



RFC 7636

Authorization code injection

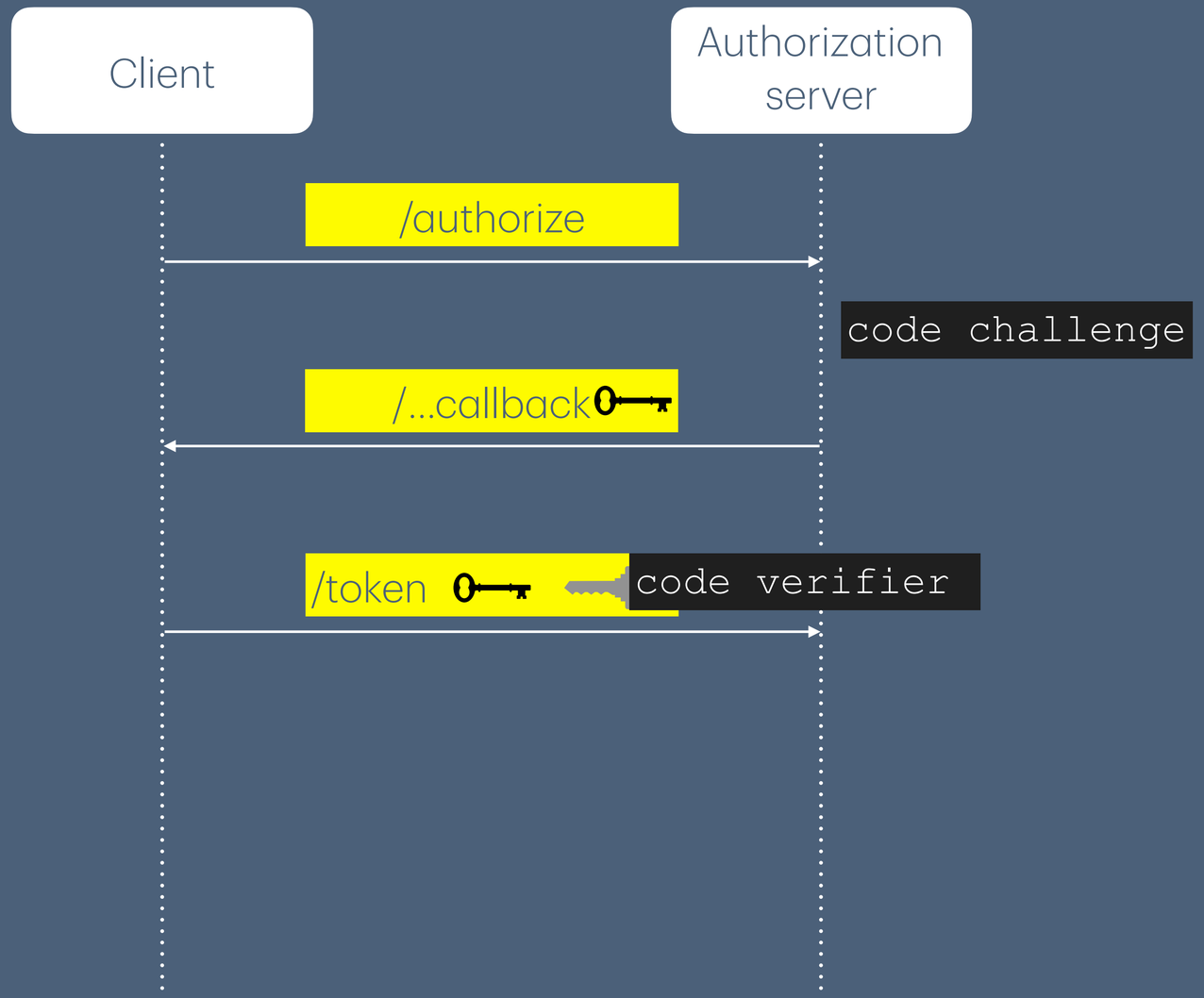
PKCE



RFC 7636

Authorization code injection

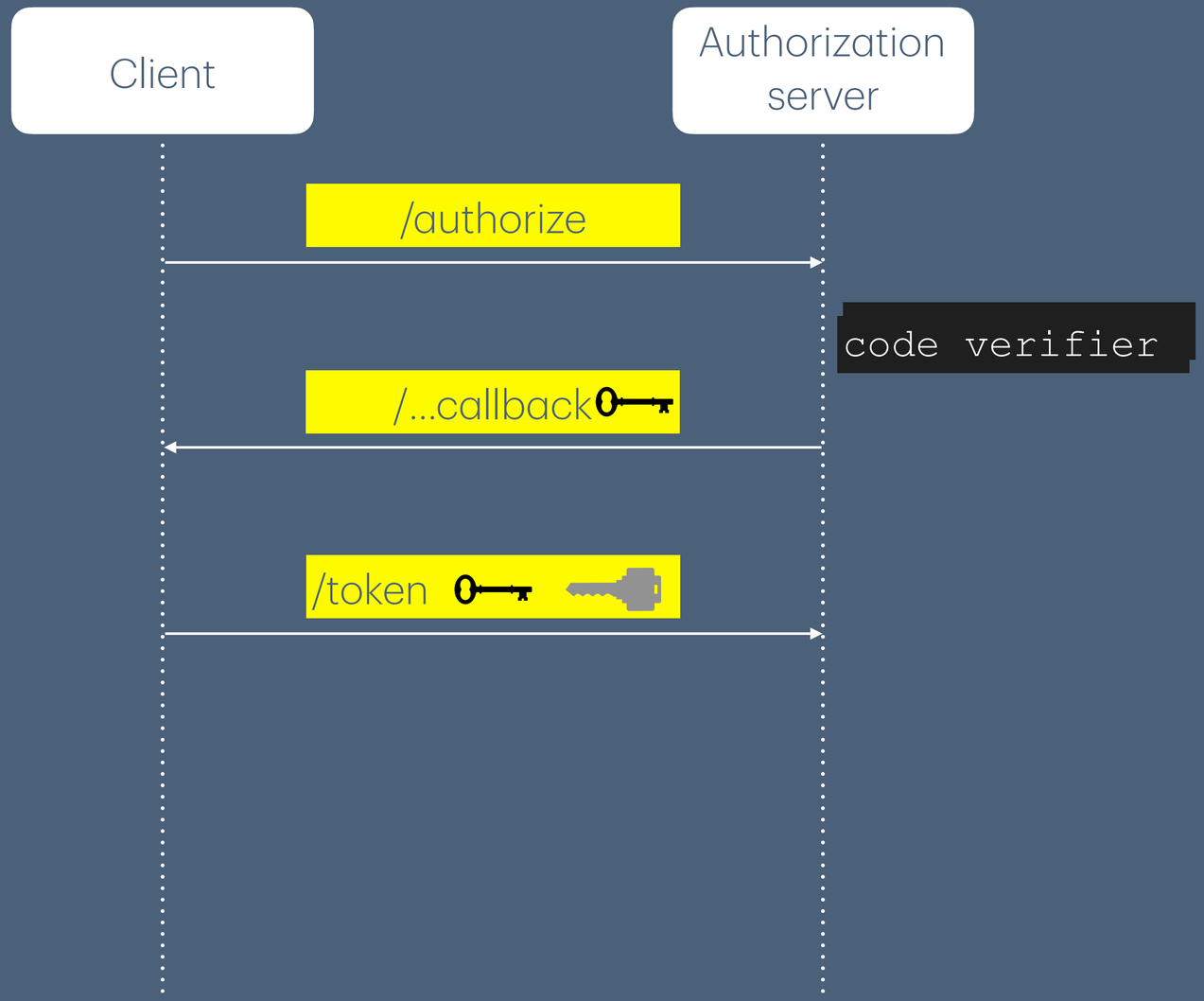
PKCE



RFC 7636

Authorization code injection

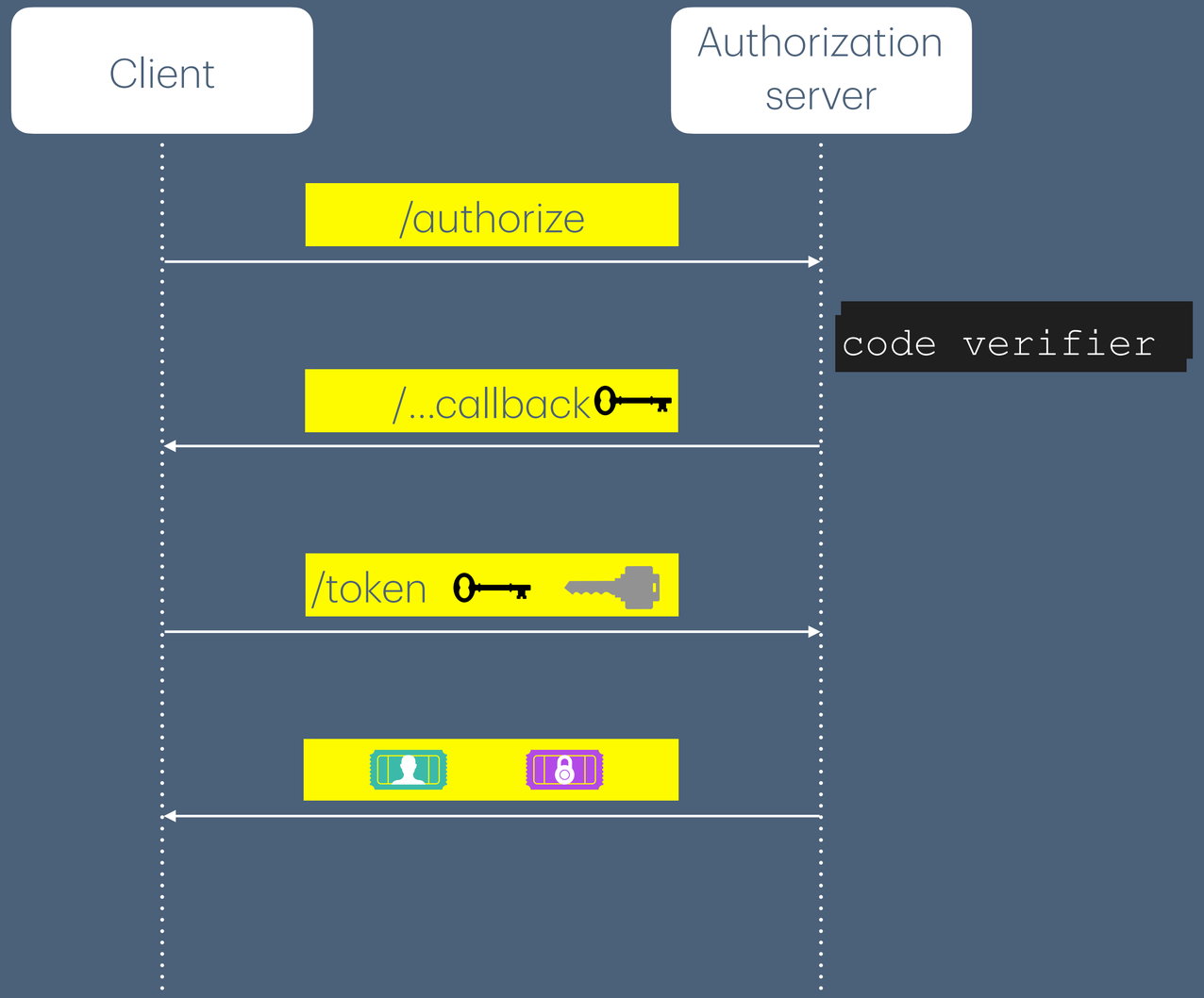
PKCE



RFC 7636

Authorization code injection

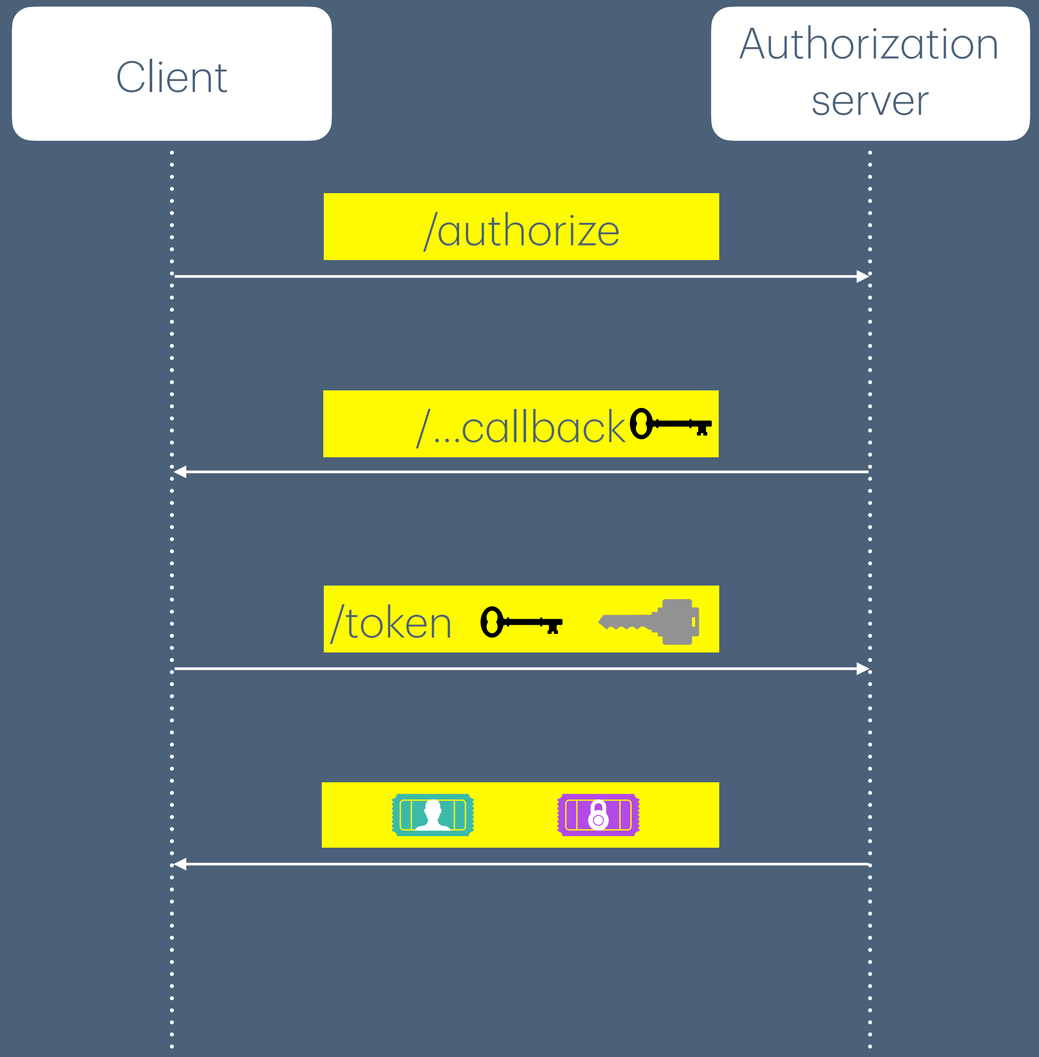
PKCE



RFC 7636

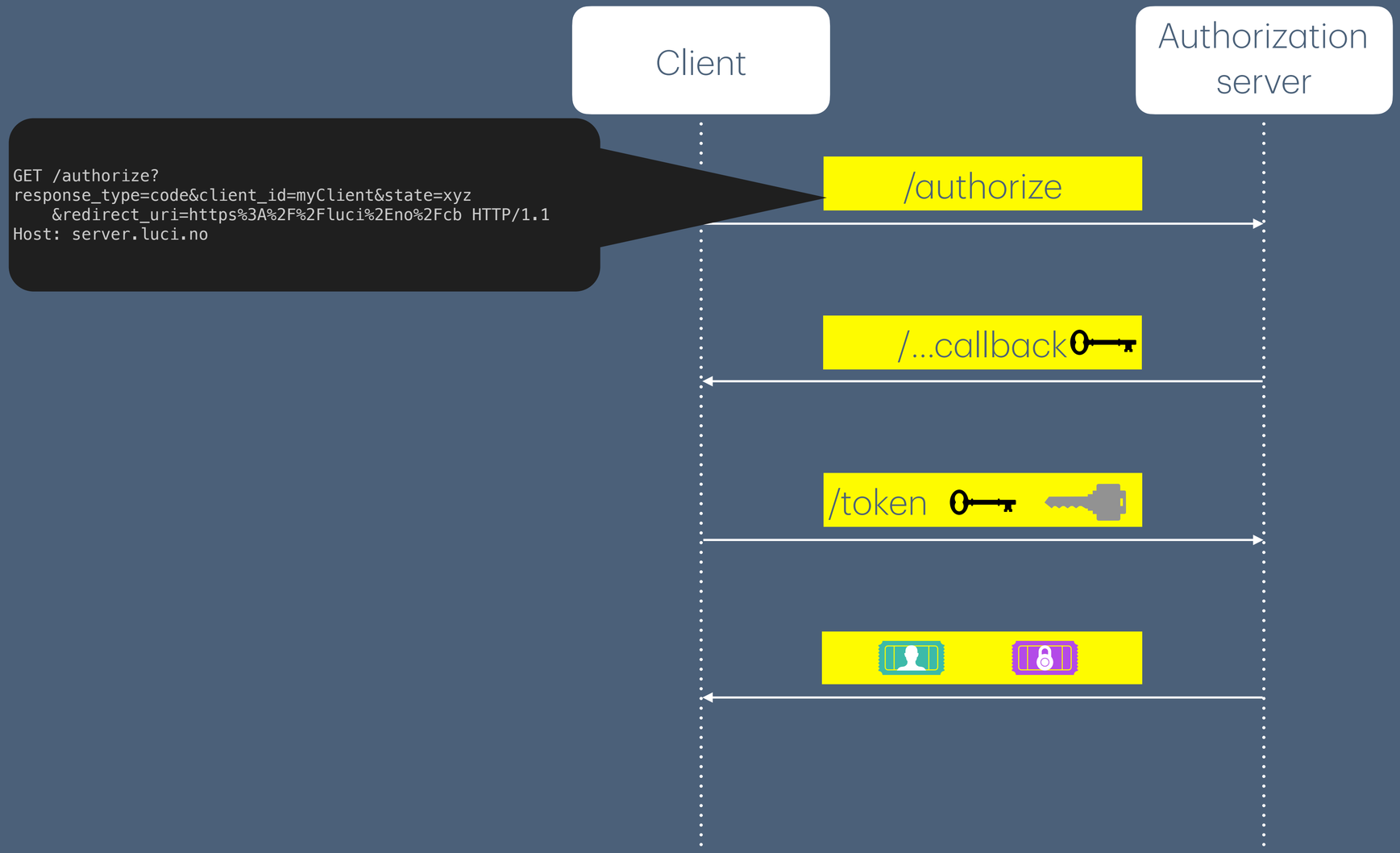
Pushed authorization requests

PAR



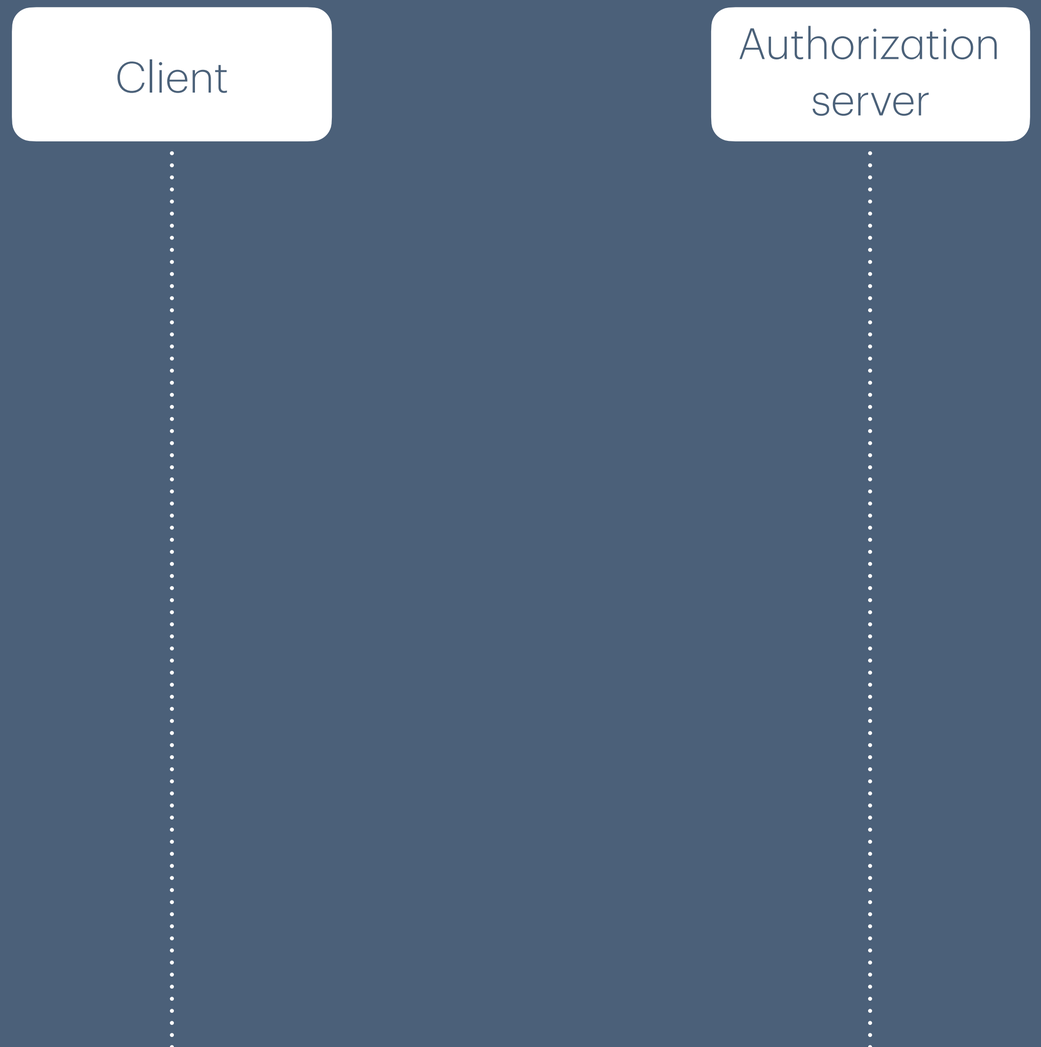
Pushed authorization requests

PAR



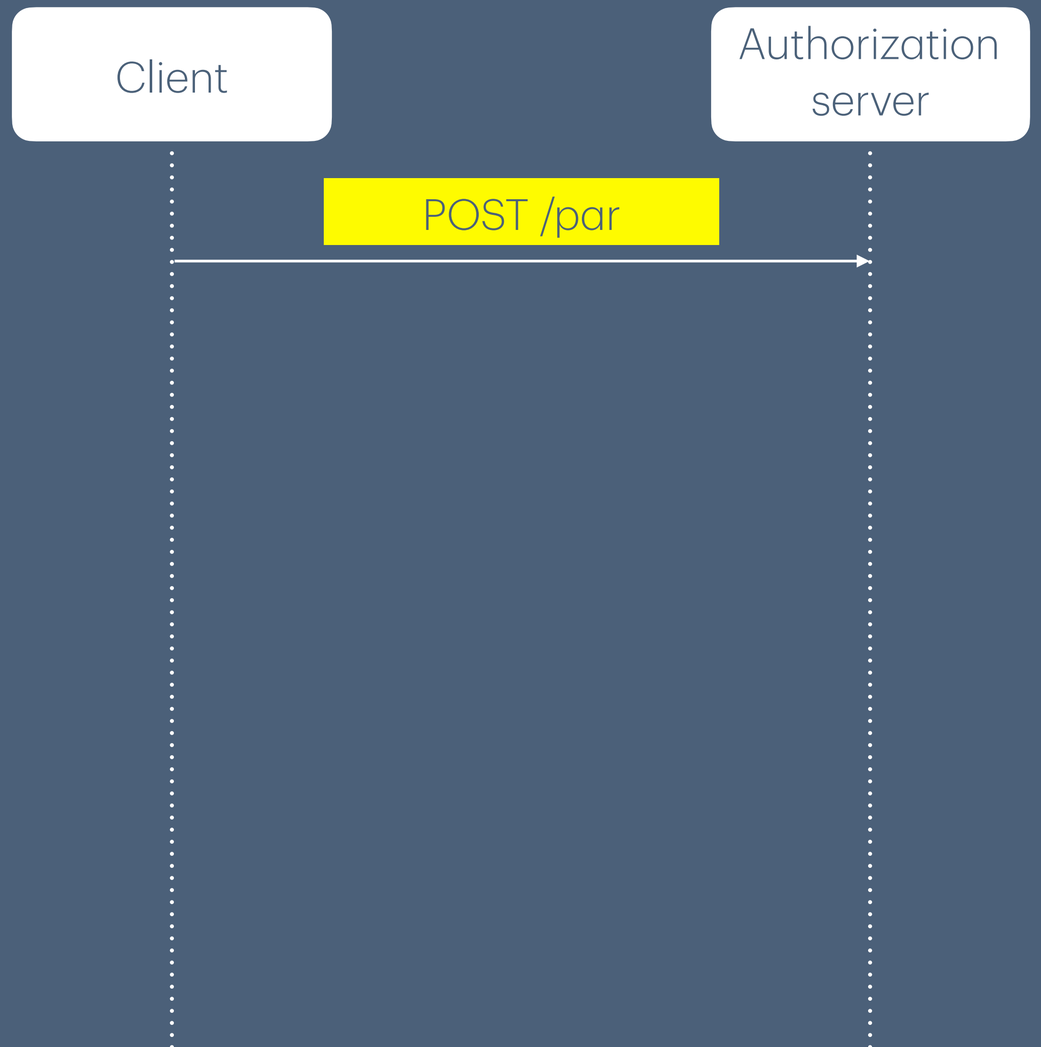
Pushed authorization requests

PAR



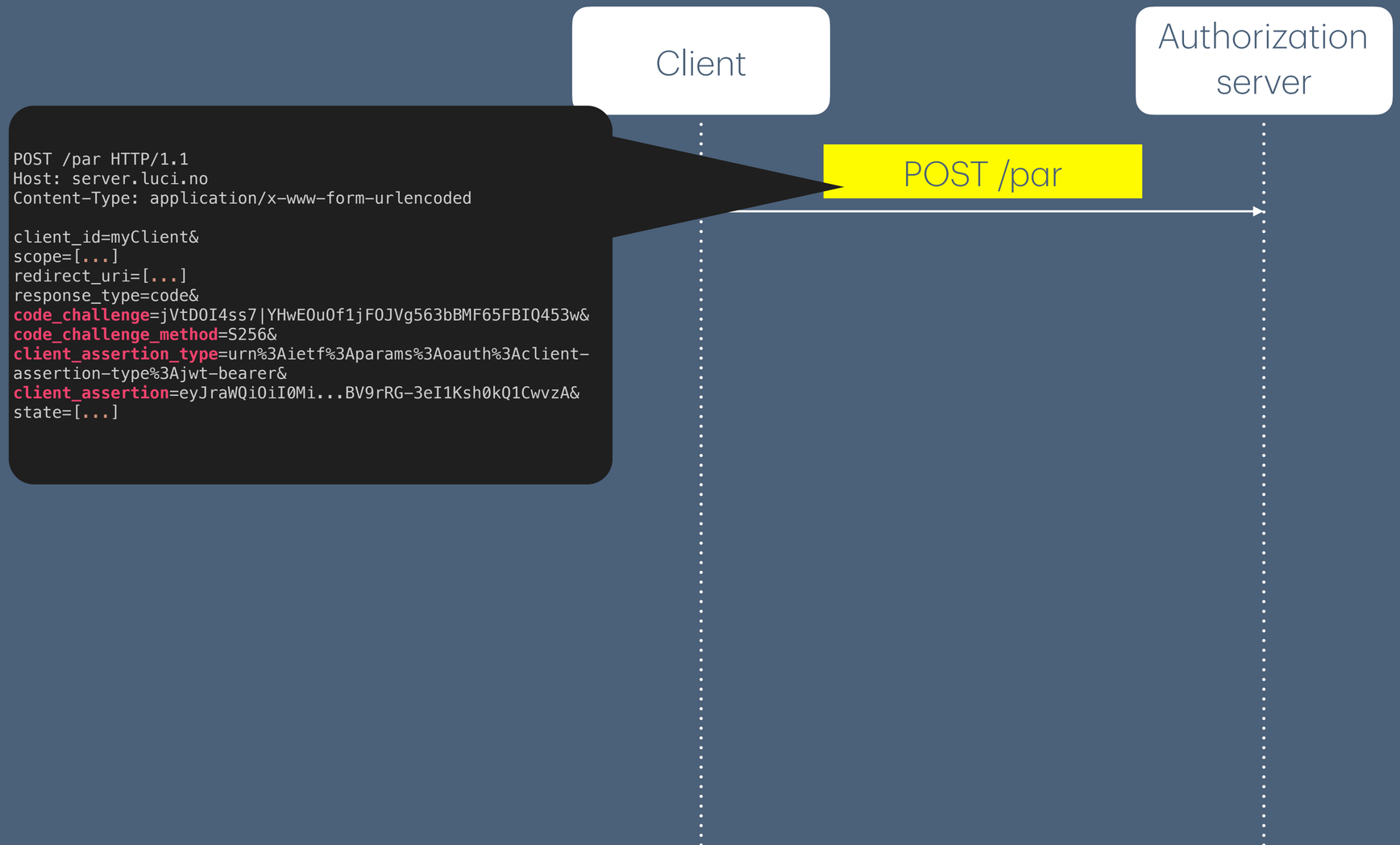
Pushed authorization requests

PAR



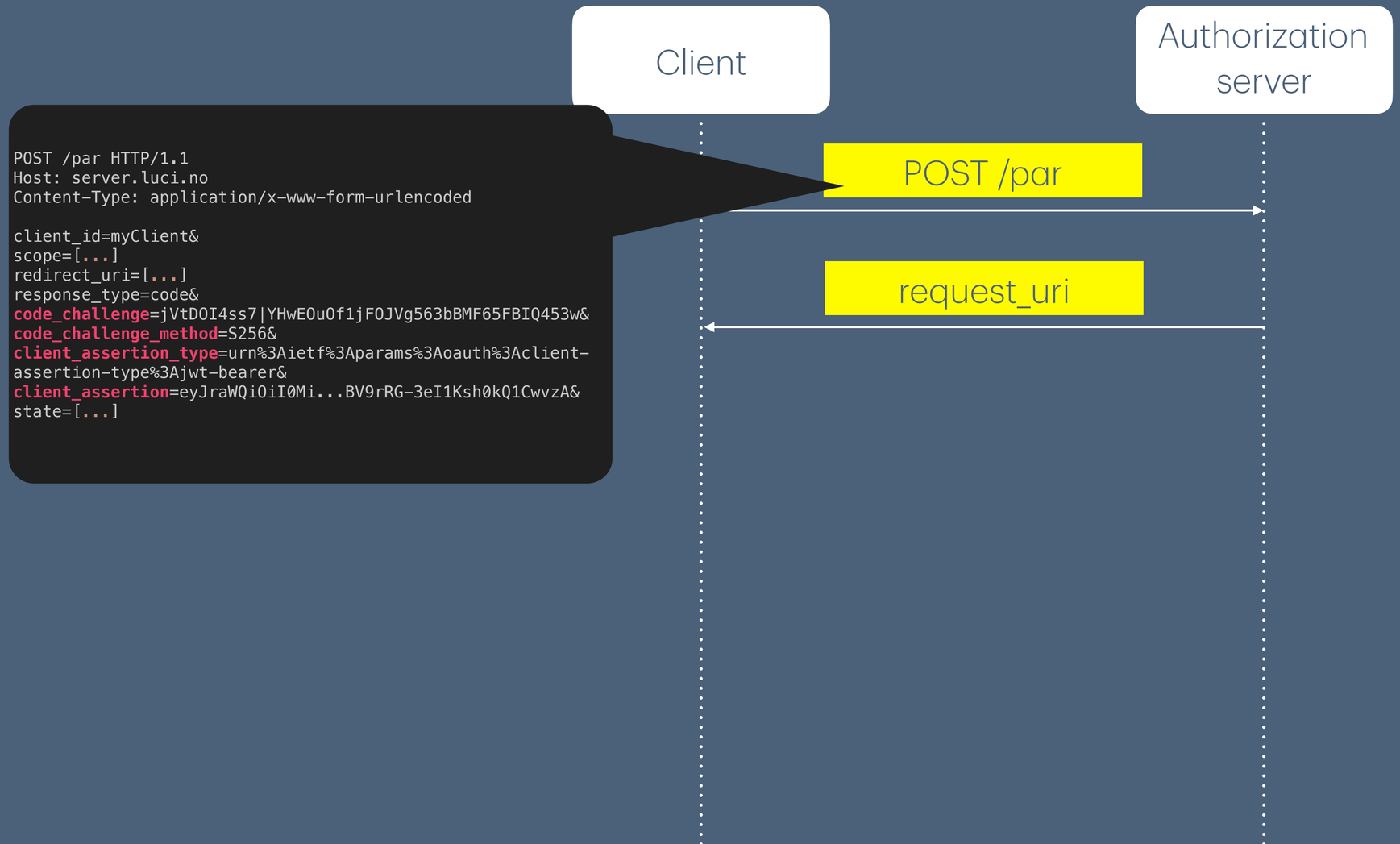
Pushed authorization requests

PAR



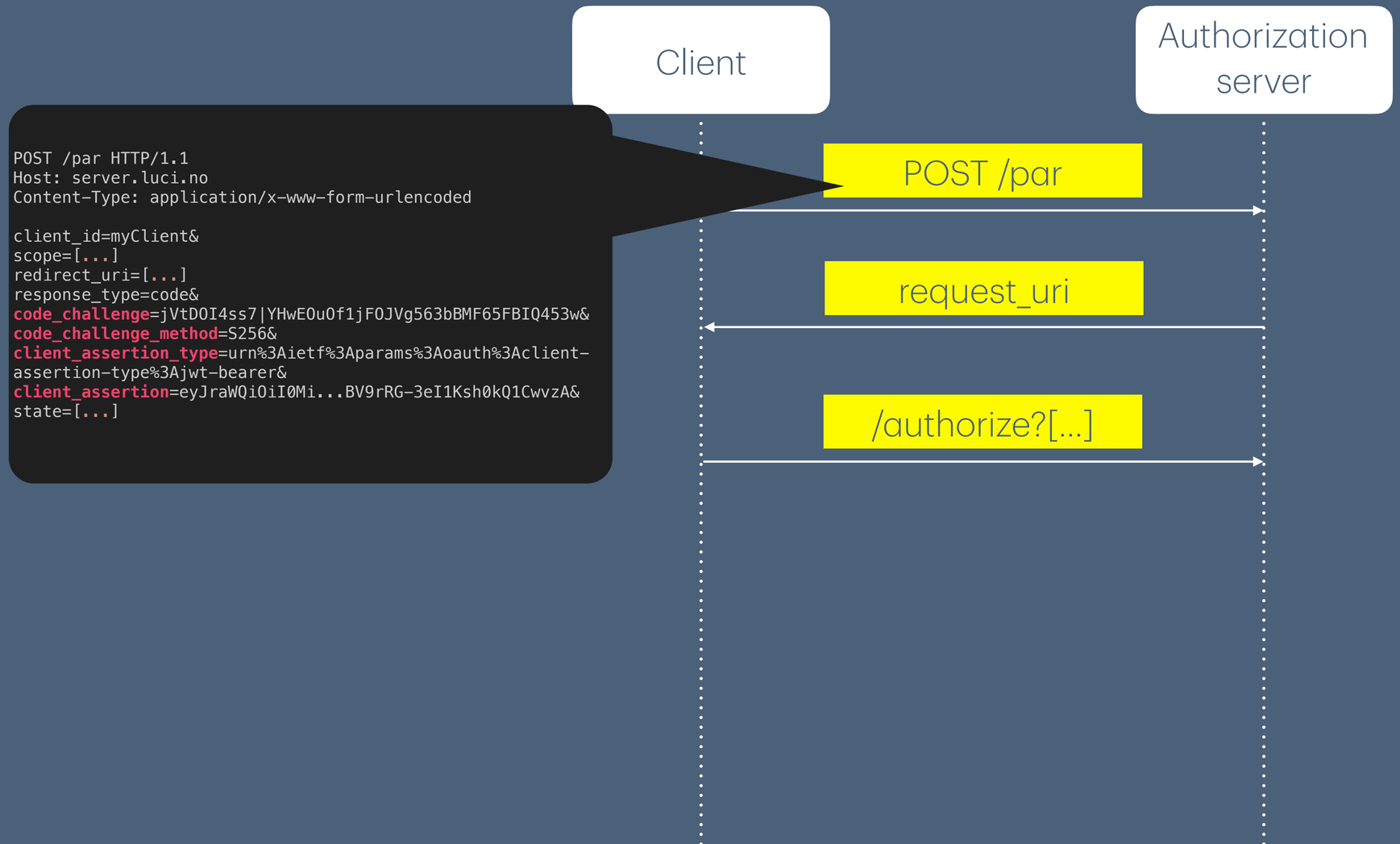
Pushed authorization requests

PAR



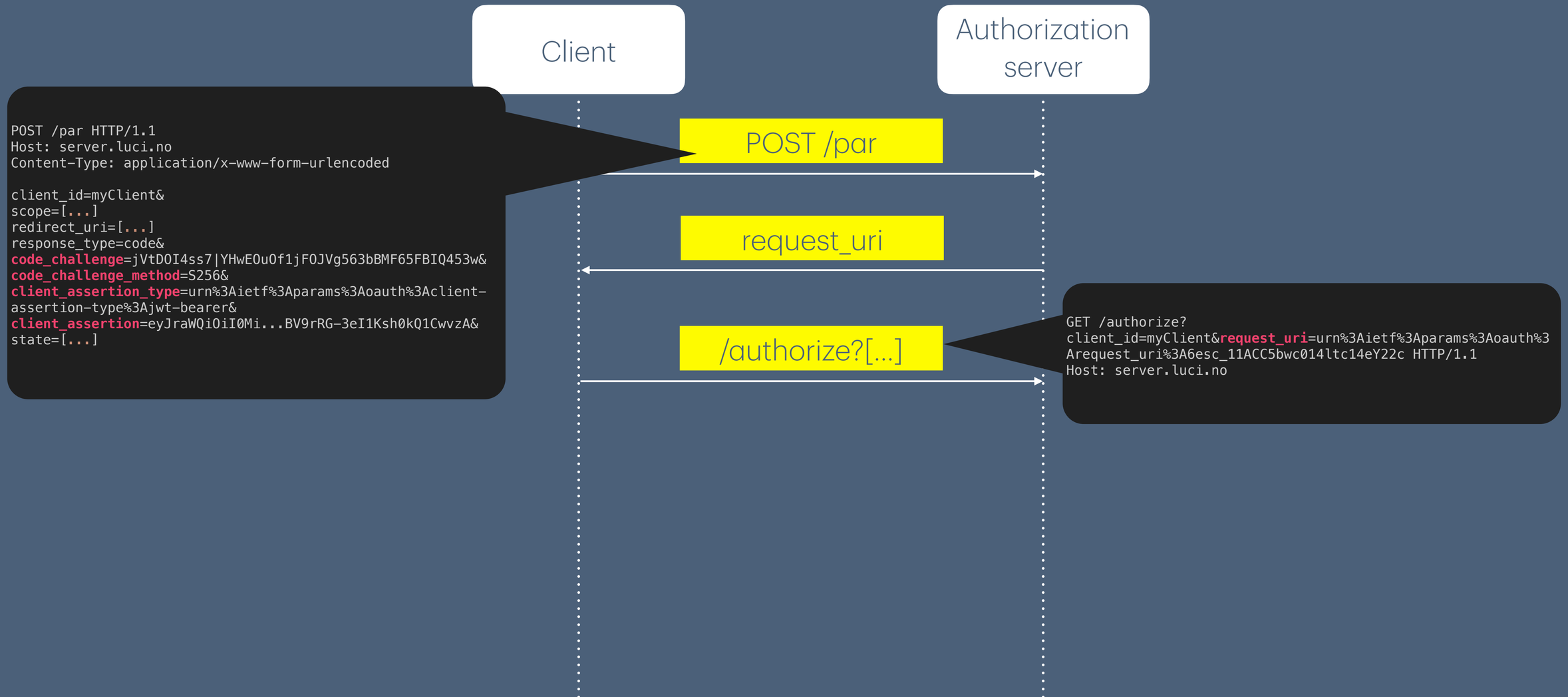
Pushed authorization requests

PAR



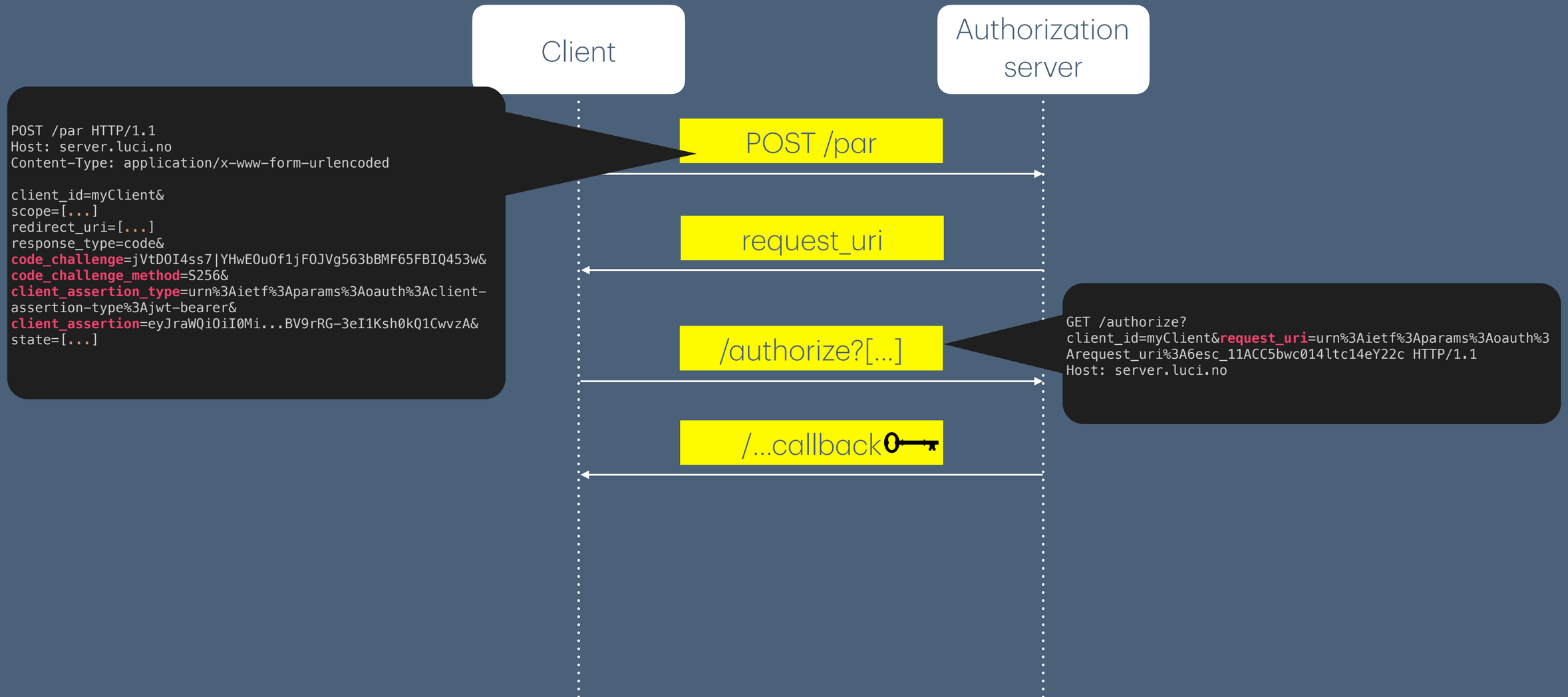
Pushed authorization requests

PAR



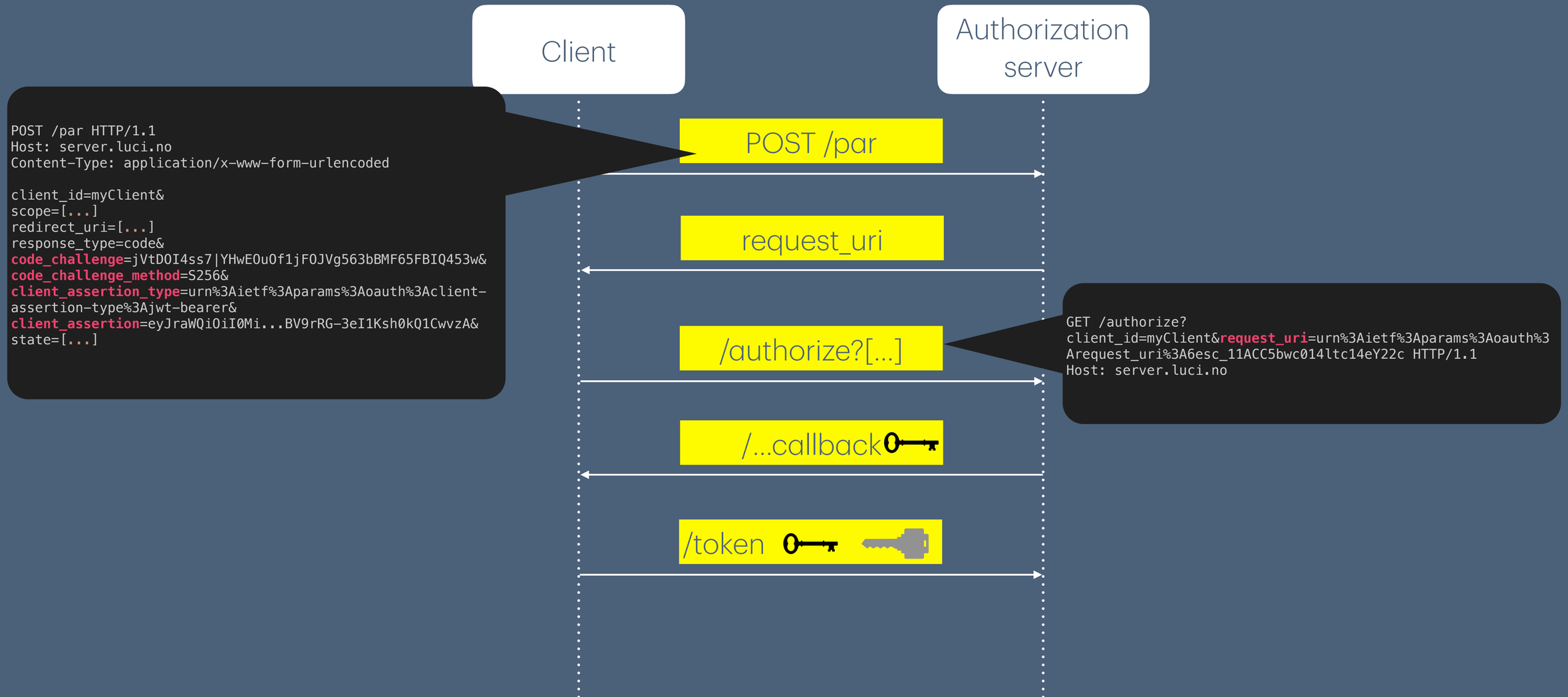
Pushed authorization requests

PAR



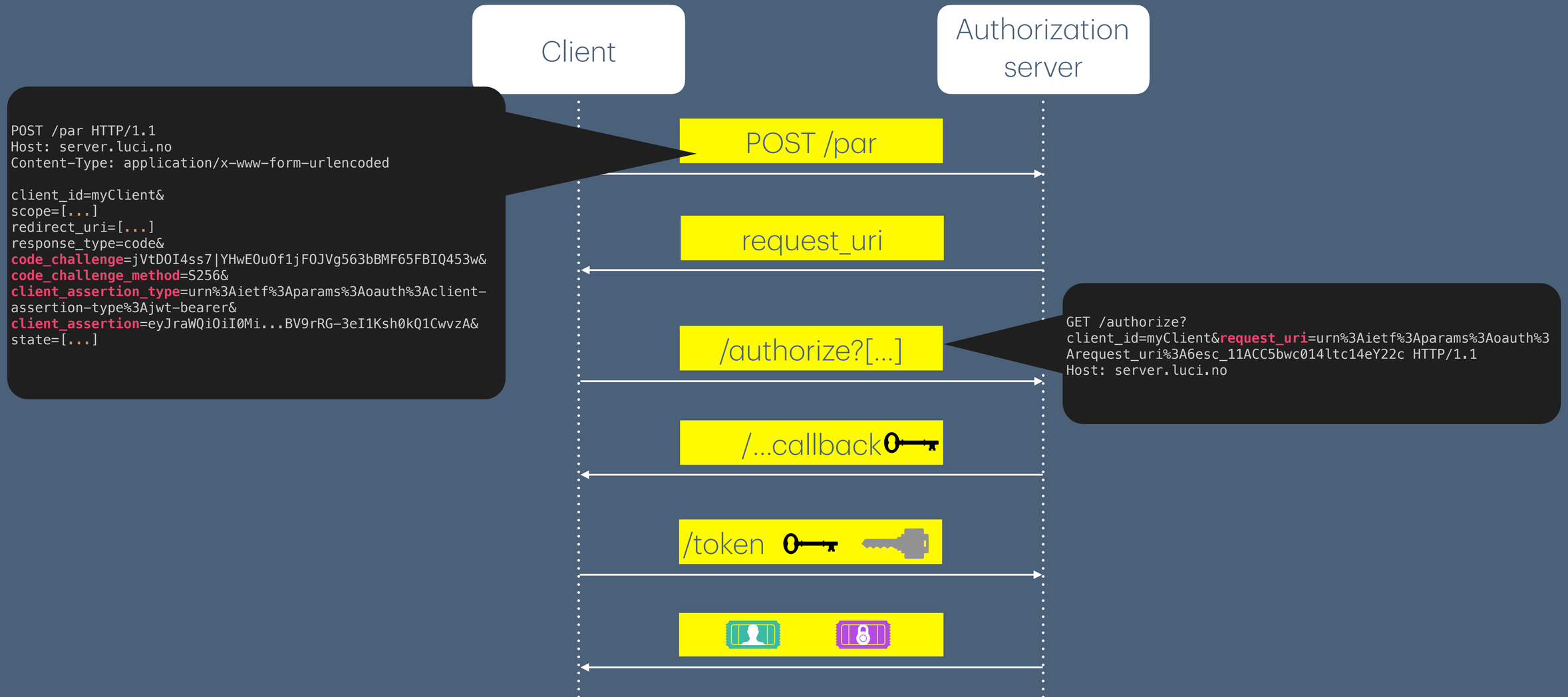
Pushed authorization requests

PAR

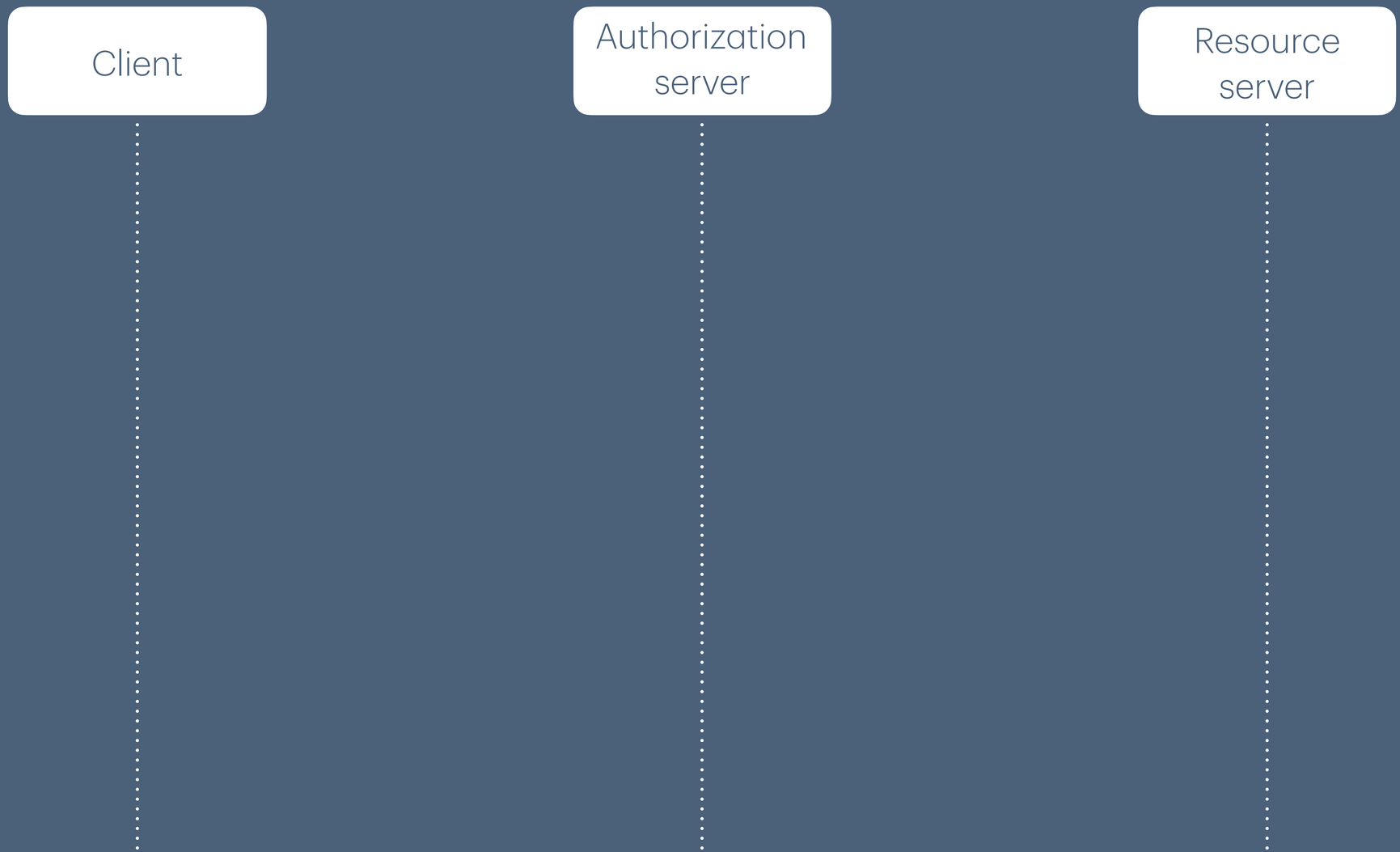


Pushed authorization requests

PAR

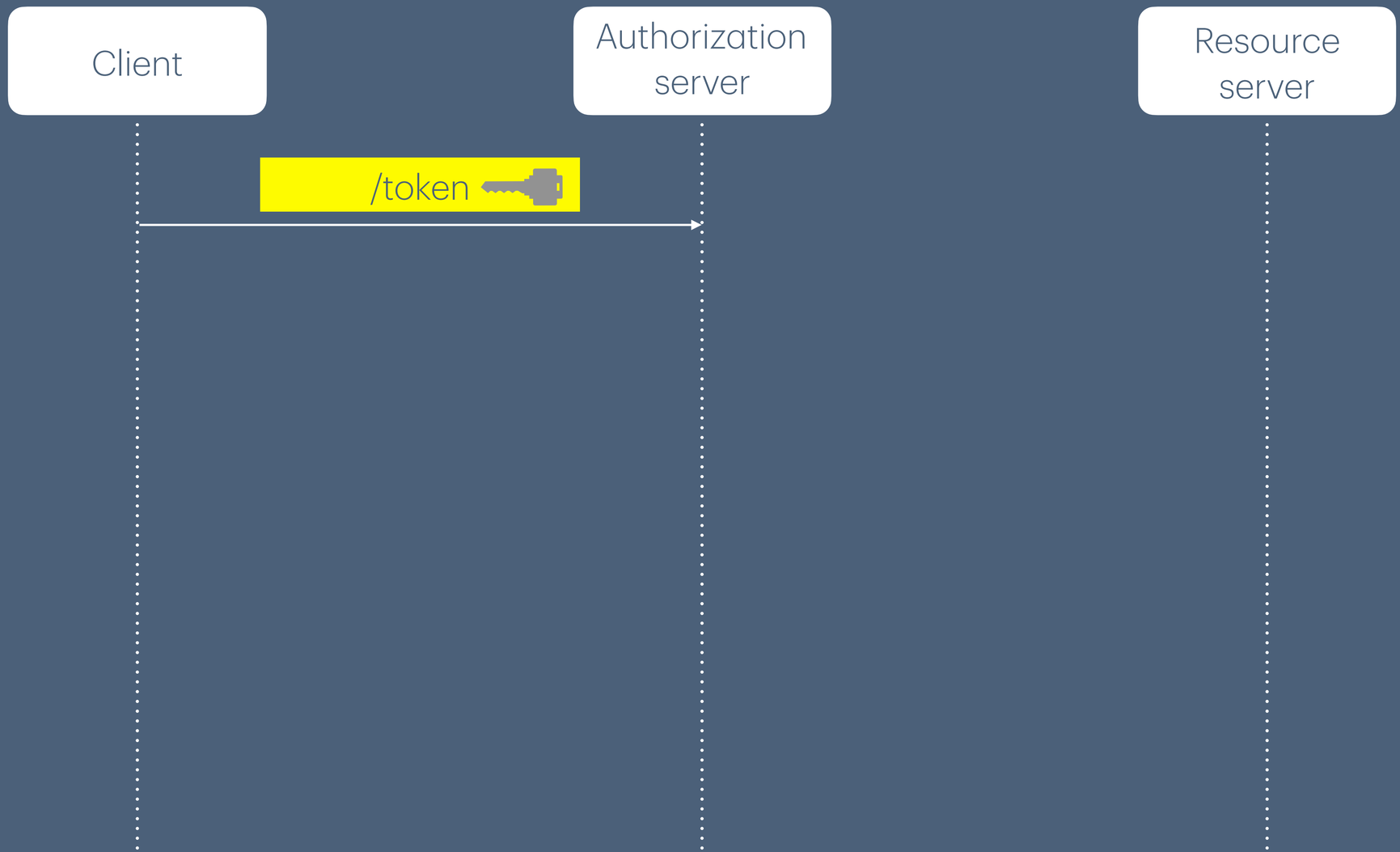


Stolen access tokens



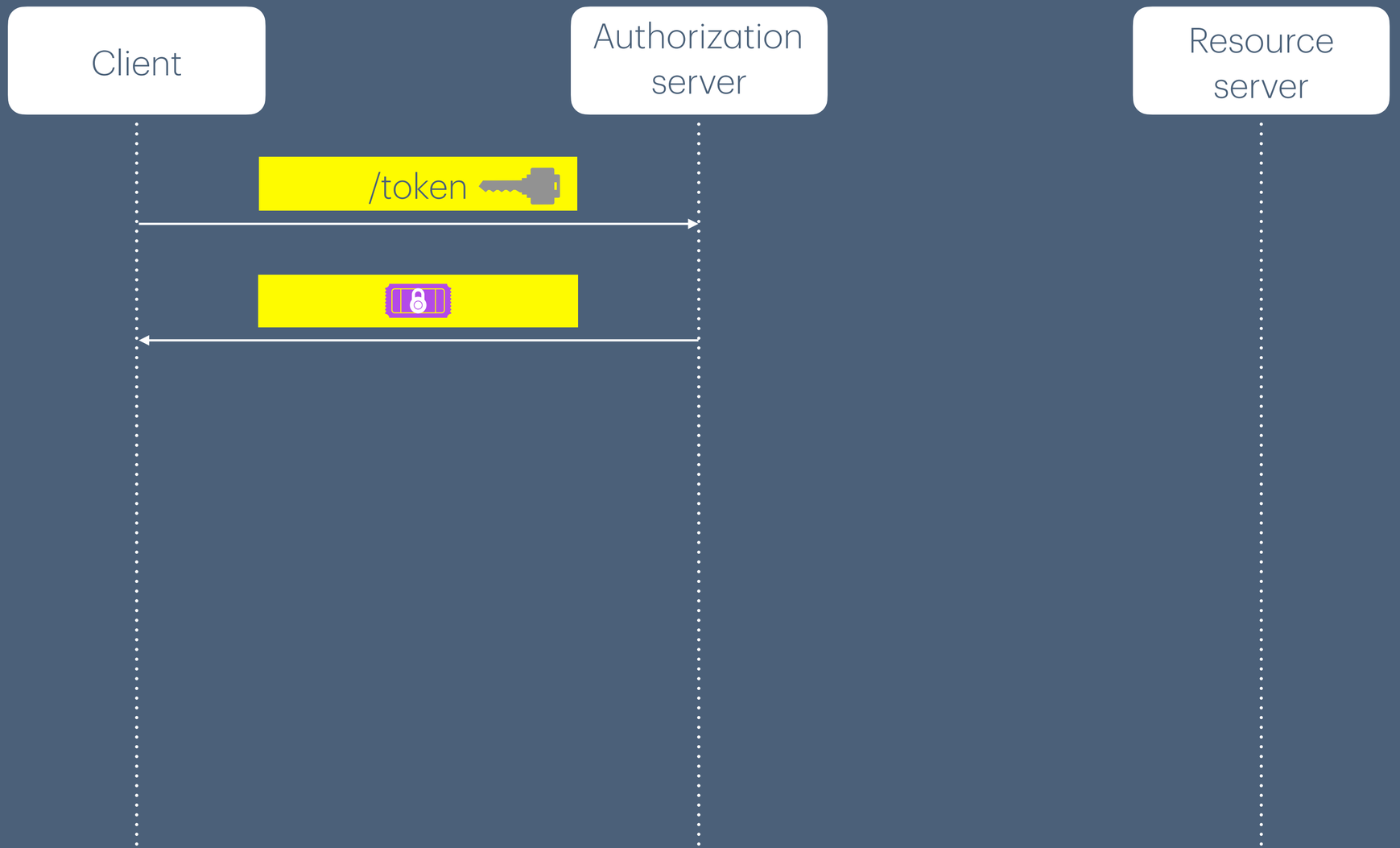
RFC 9126

Stolen access tokens



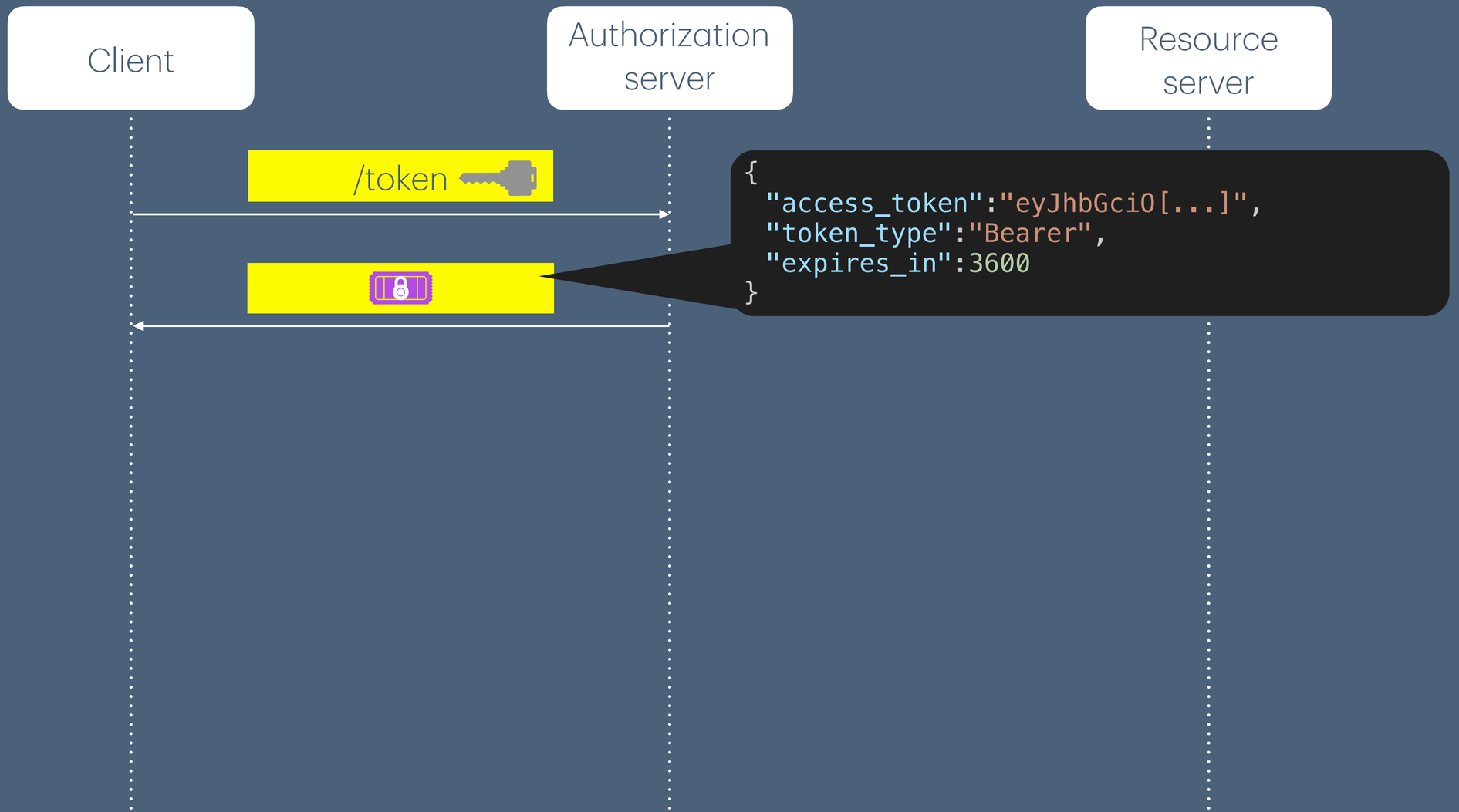
RFC 9126

Stolen access tokens



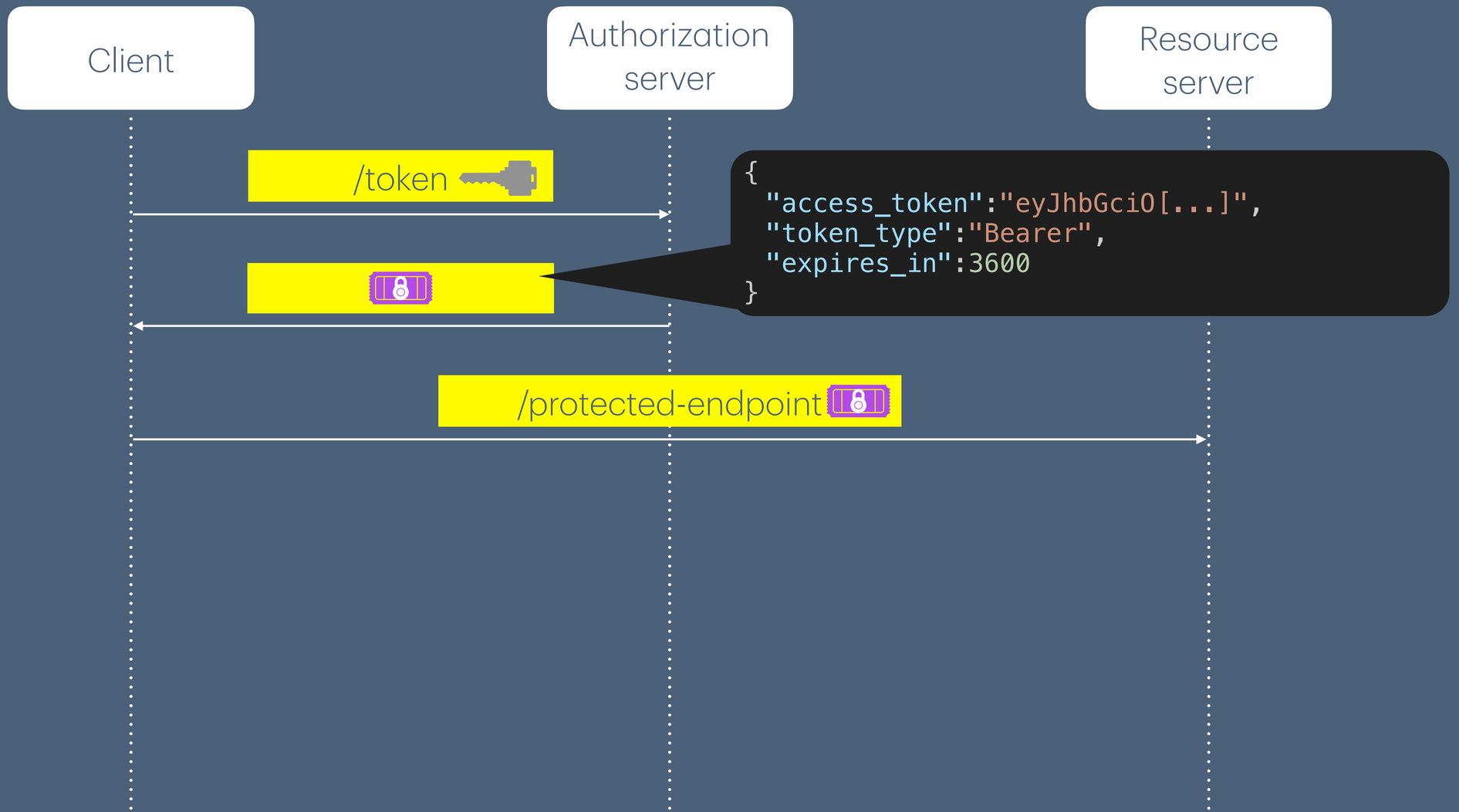
RFC 9126

Stolen access tokens



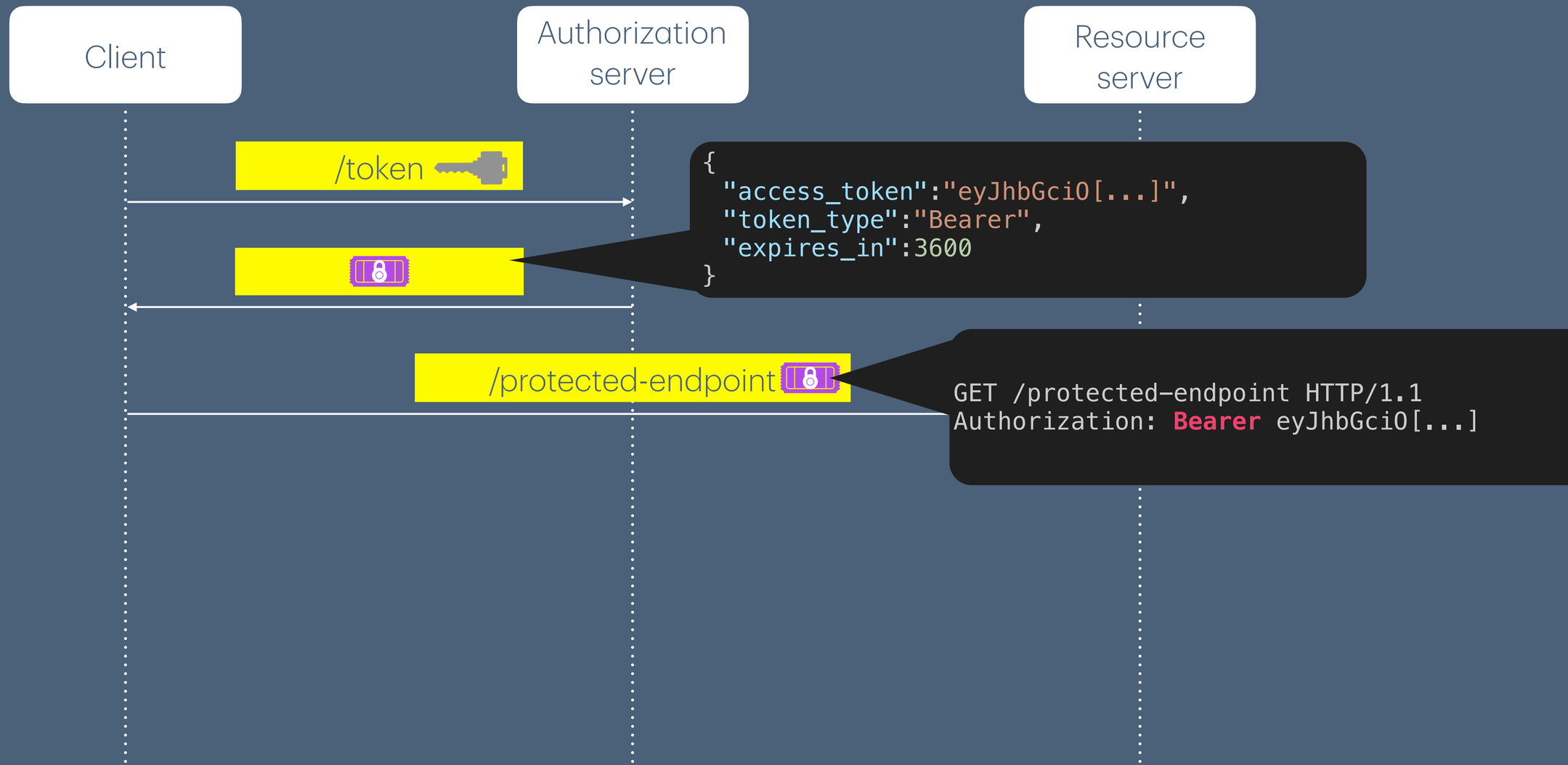
RFC 9126

Stolen access tokens



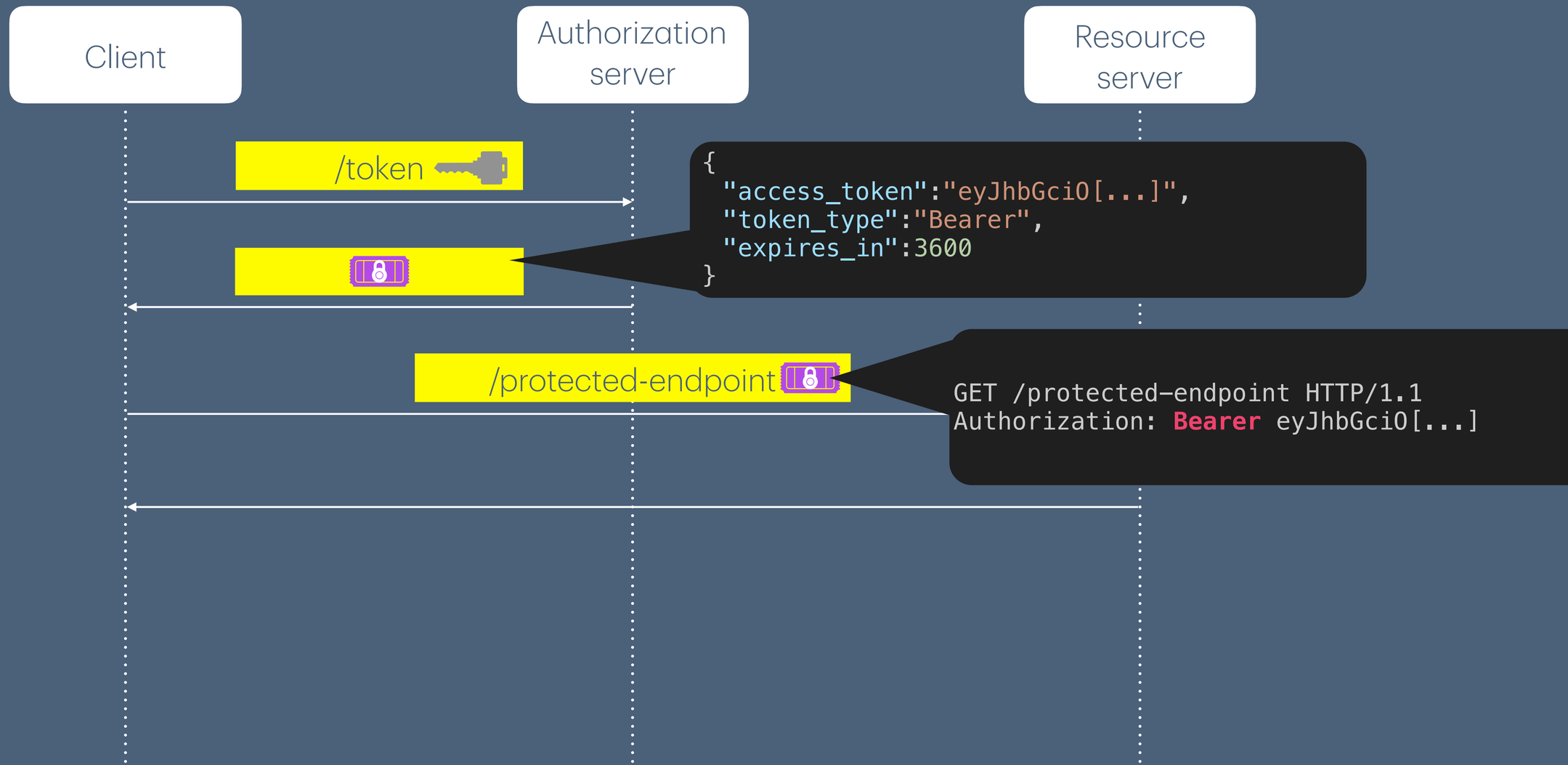
RFC 9126

Stolen access tokens



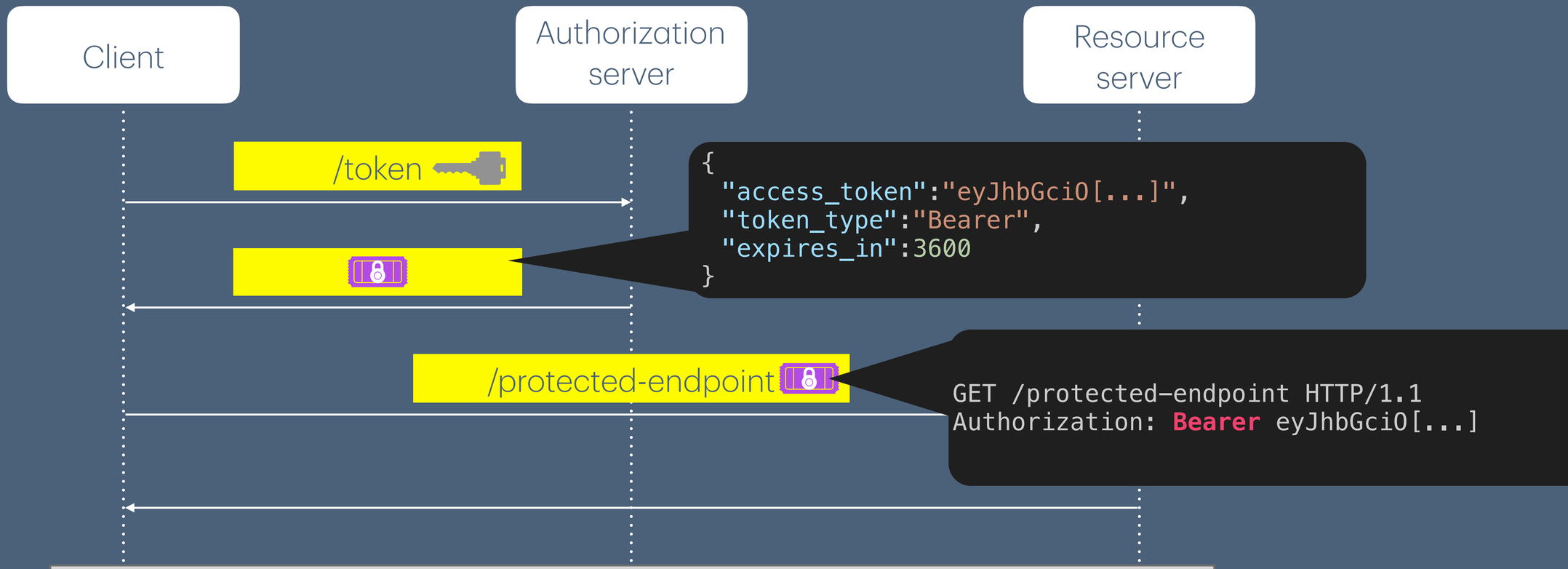
RFC 9126

Stolen access tokens



RFC 9126

Stolen access tokens



Bearer Token:

A security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. (...) (RFC 6750)

RFC 9126

Stolen access tokens

Proof of possession tokens

Stolen access tokens

Proof of possession tokens

A token that is bound to the client that requested it

Stolen access tokens

Proof of possession tokens

A token that is bound to the client that requested it

- Part of OAuth 1.0

Stolen access tokens

Proof of possession tokens

A token that is bound to the client that requested it

- Part of OAuth 1.0
- Removed from OAuth 2.0 -> *Simplified*

Stolen access tokens

Proof of possession tokens

A token that is bound to the client that requested it

- Part of OAuth 1.0
- Removed from OAuth 2.0 -> *Simplified*
- RFC 6749: The OAuth 2.0 Authorization Framework

Stolen access tokens

Proof of possession tokens

A token that is bound to the client that requested it

- Part of OAuth 1.0
- Removed from OAuth 2.0 -> *Simplified*
- RFC 6749: The OAuth 2.0 Authorization Framework
- RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage

Stolen access tokens

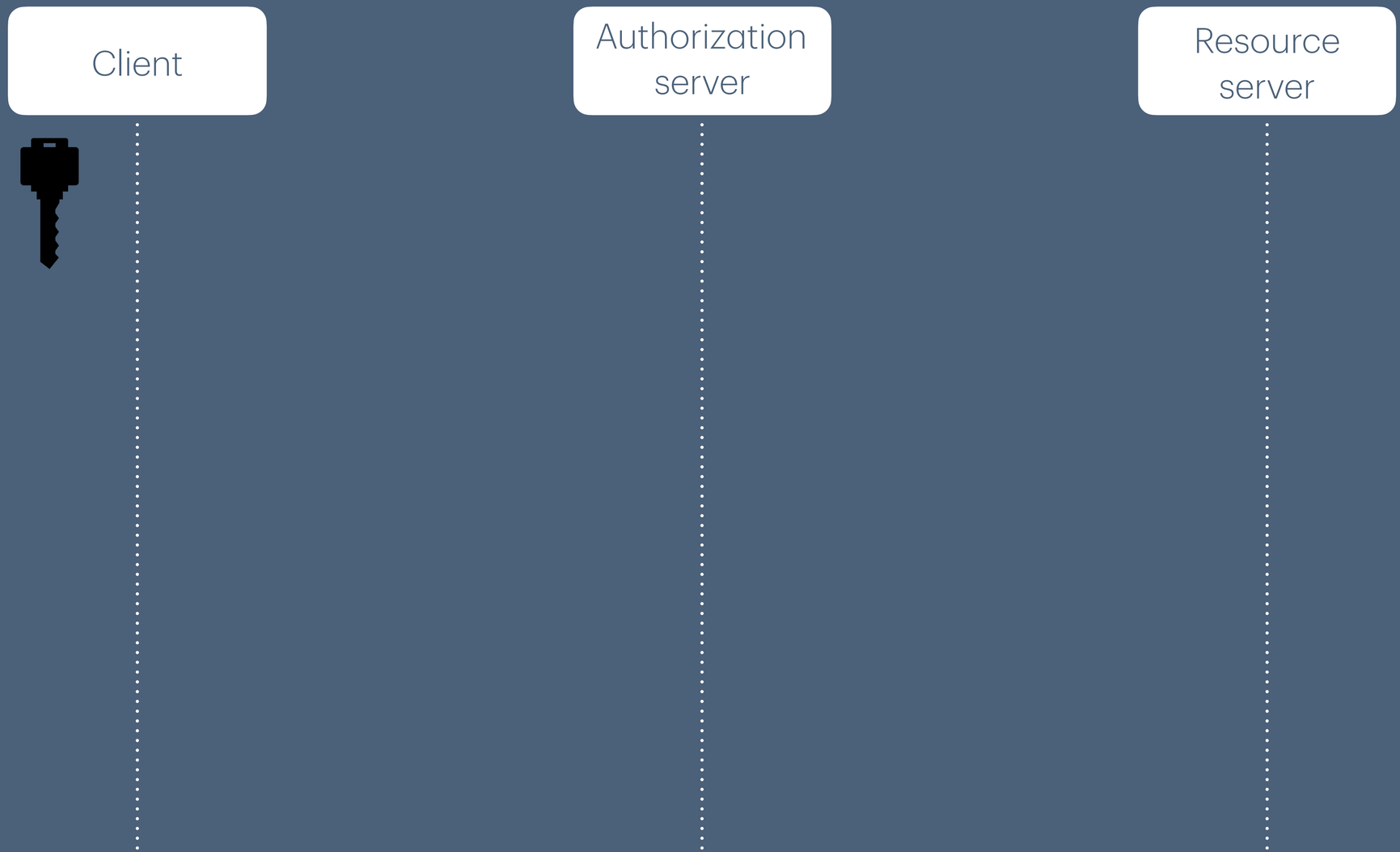
Proof of possession tokens

A token that is bound to the client that requested it

- Part of OAuth 1.0
- Removed from OAuth 2.0 -> *Simplified*
- RFC 6749: The OAuth 2.0 Authorization Framework
- RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage
- FAPI -> *Sender constrained tokens*

Demonstrating proof of possession

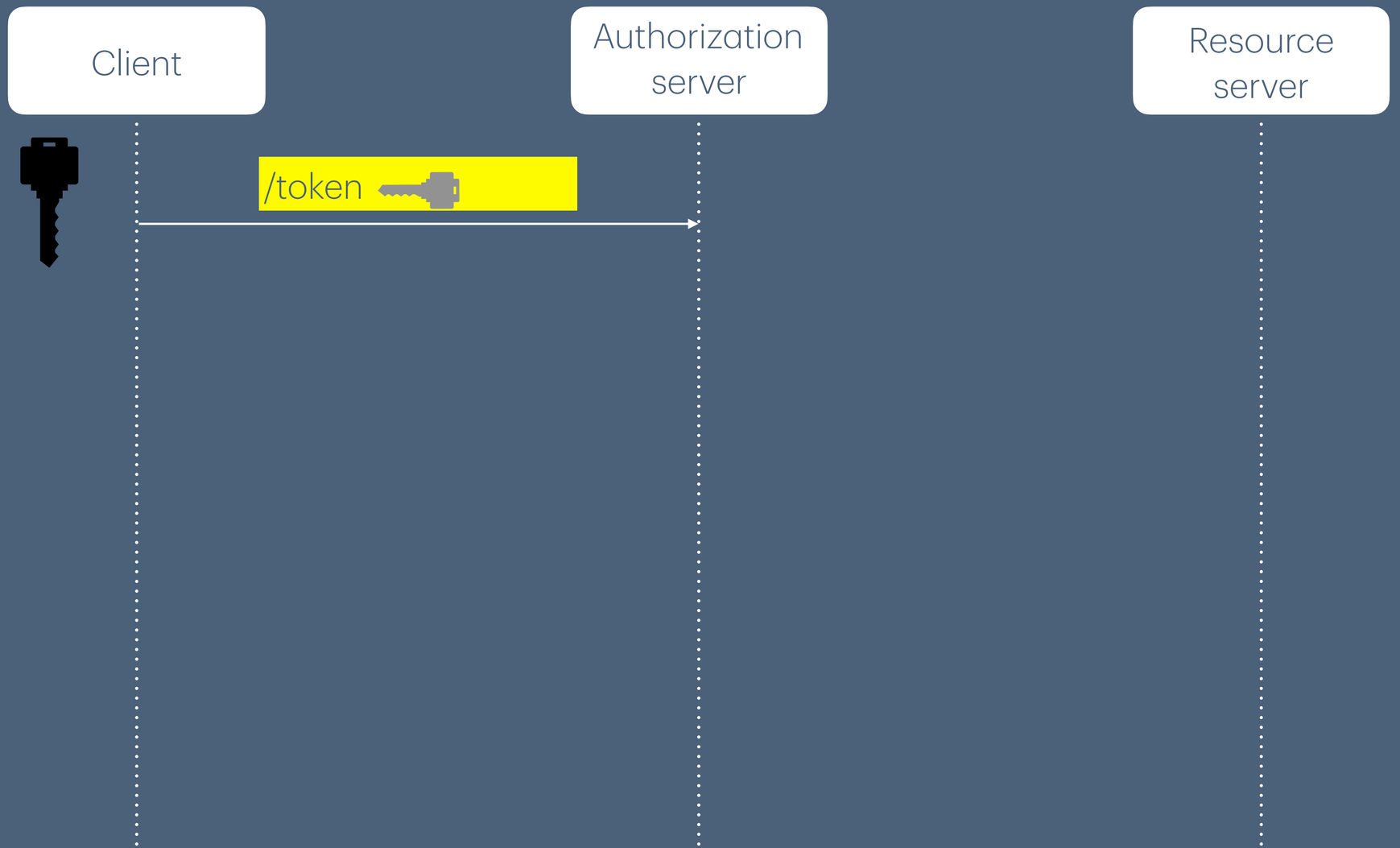
DPOP



RFC 9449

Demonstrating proof of possession

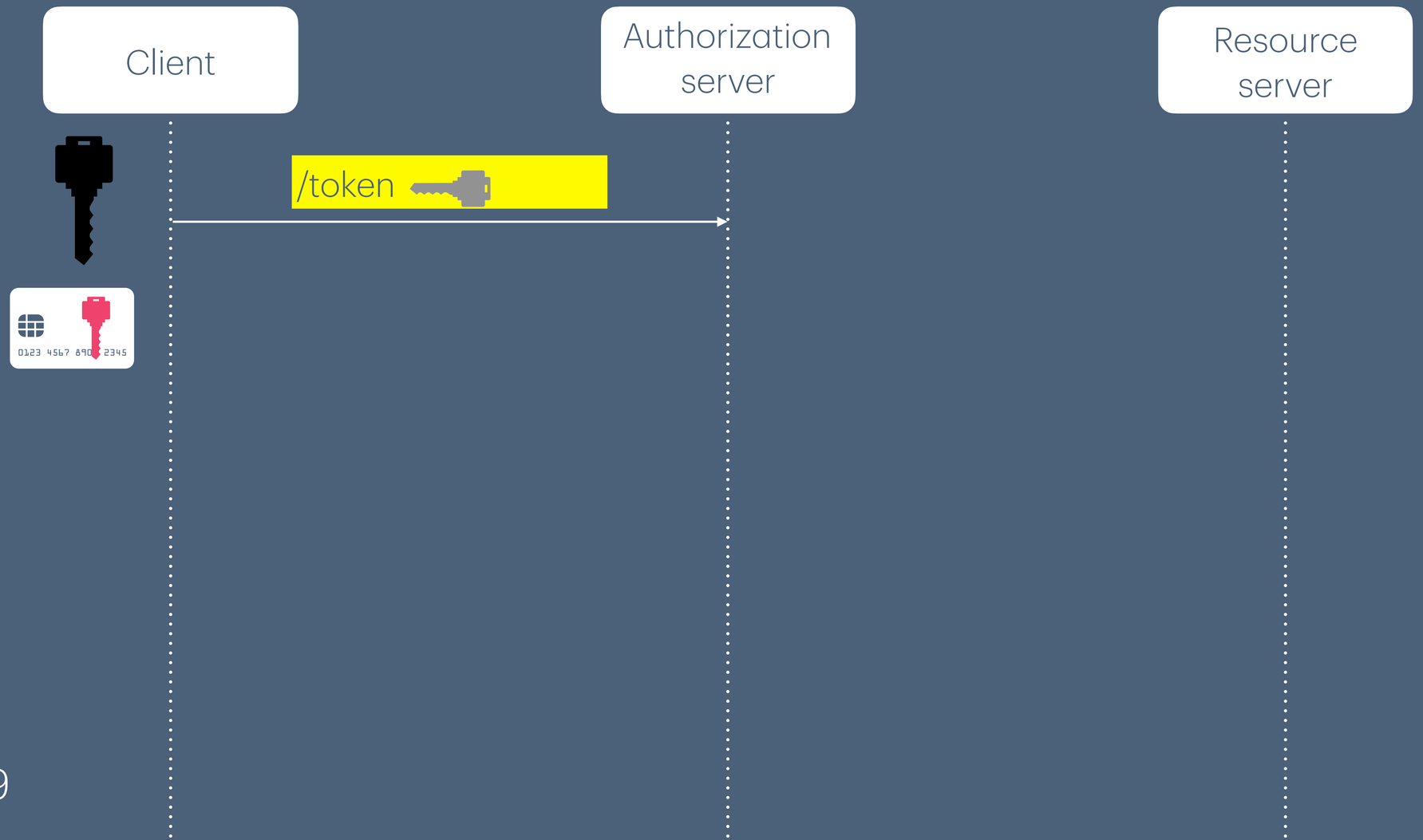
DPOP



RFC 9449

Demonstrating proof of possession

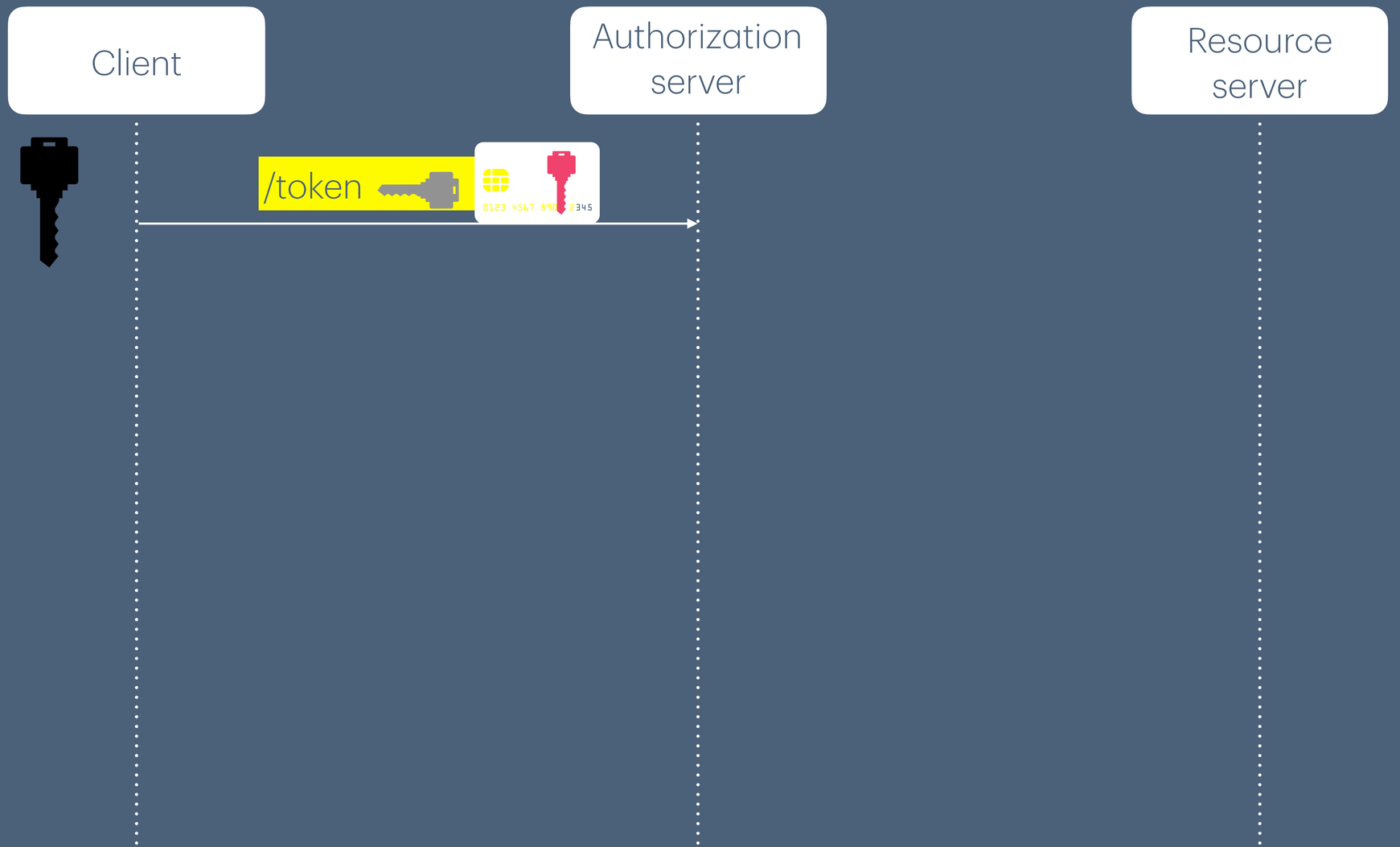
DPOP



RFC 9449

Demonstrating proof of possession

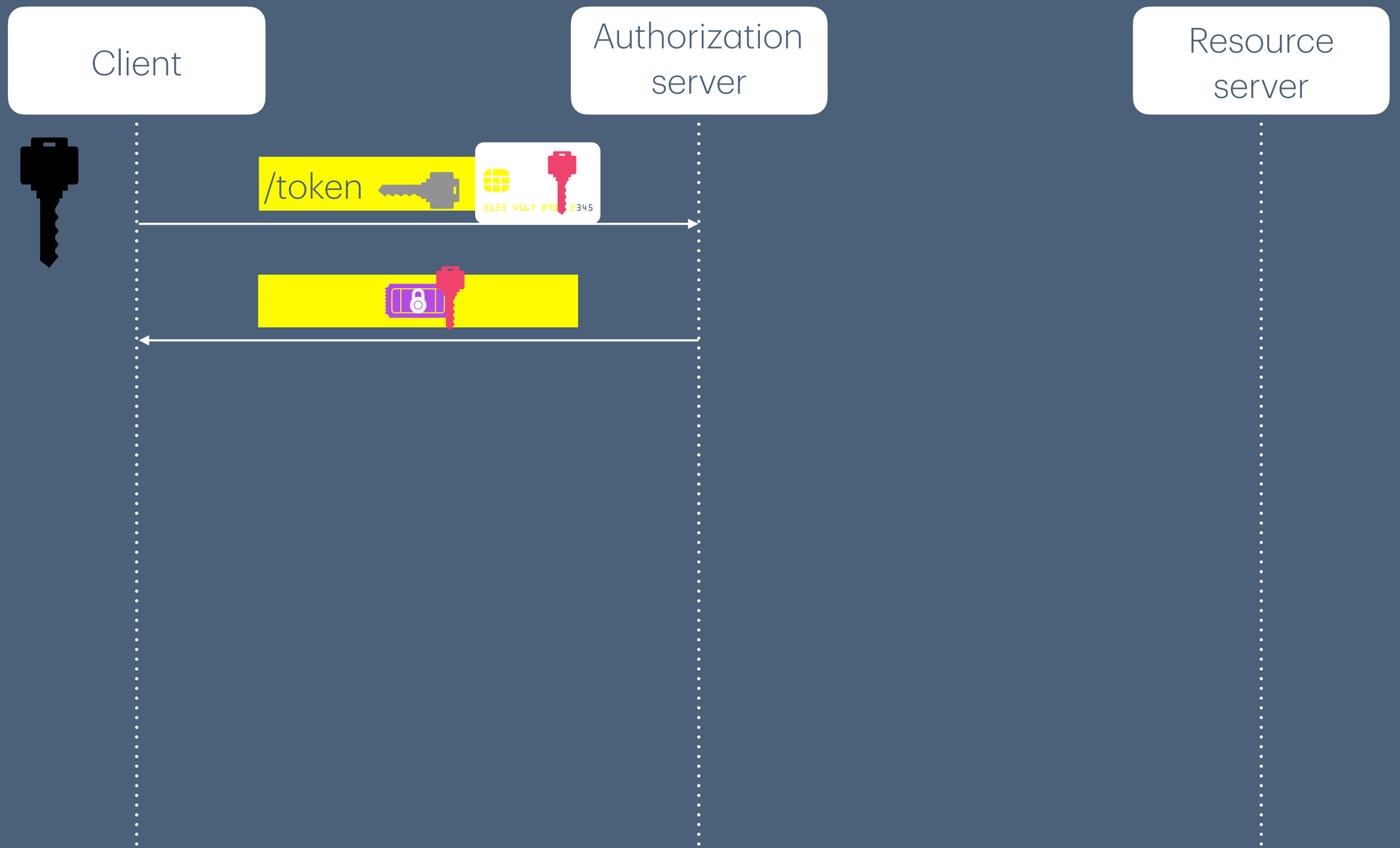
DPOP



RFC 9449

Demonstrating proof of possession

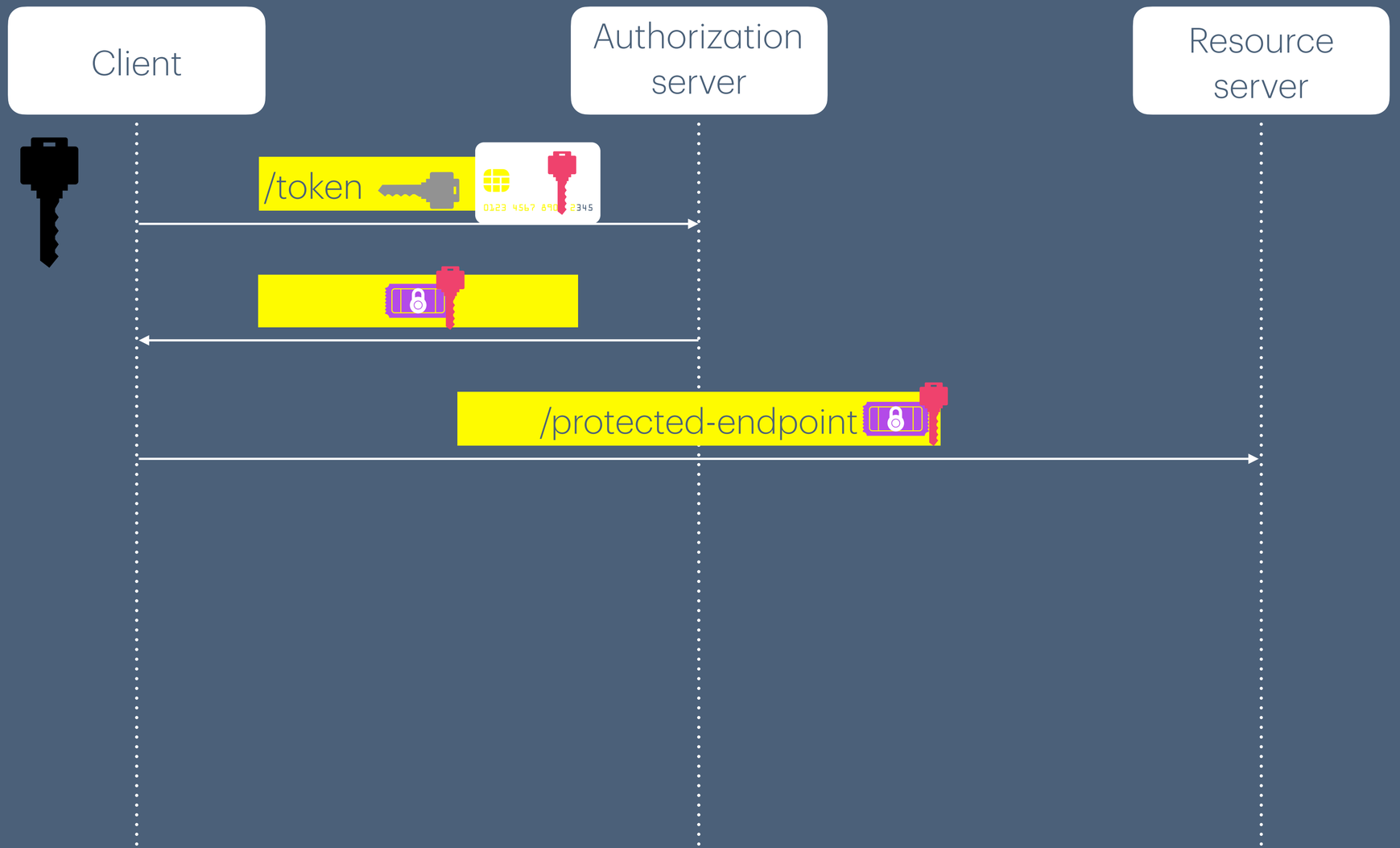
DPOP



RFC 9449

Demonstrating proof of possession

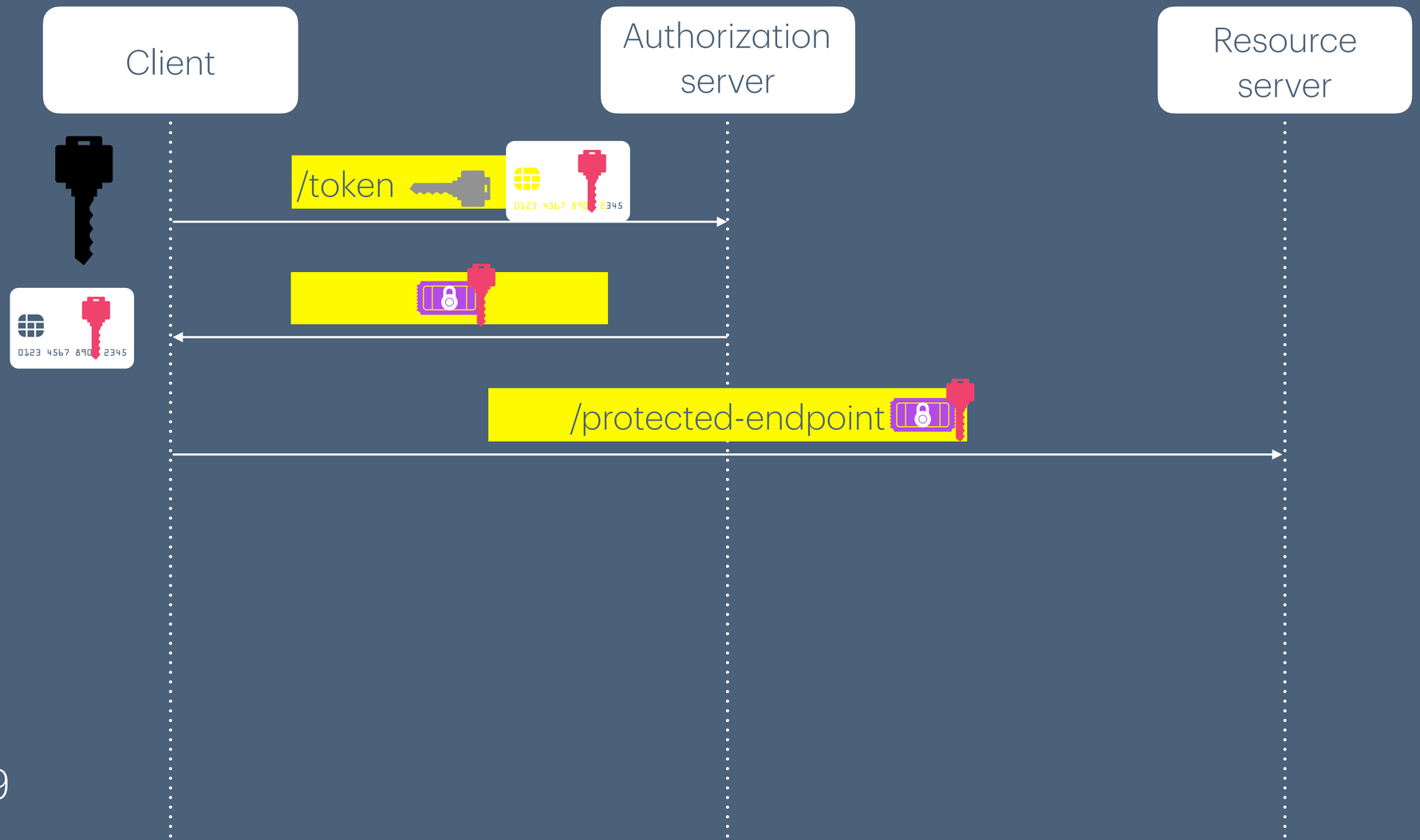
DPOP



RFC 9449

Demonstrating proof of possession

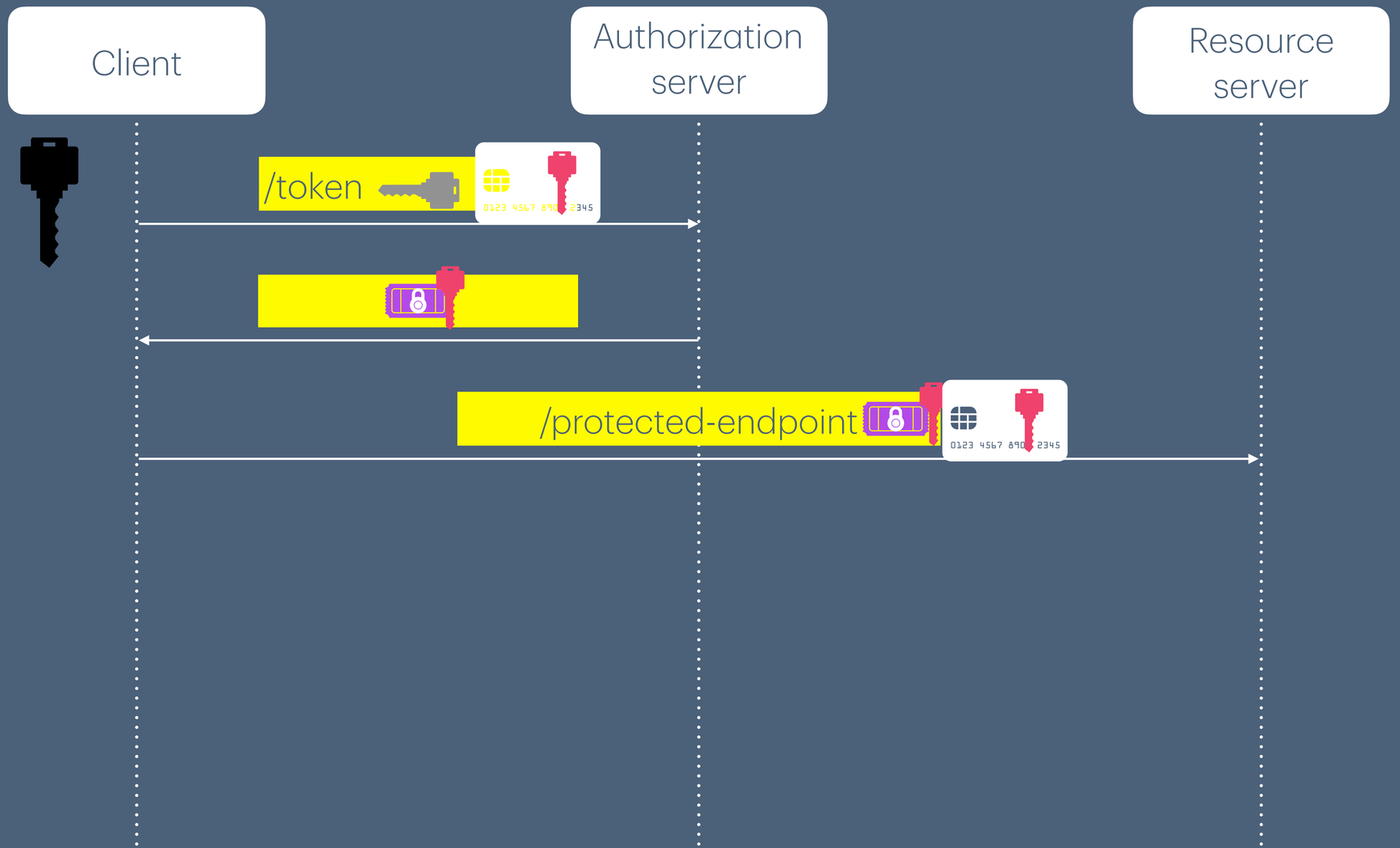
DPOP



RFC 9449

Demonstrating proof of possession

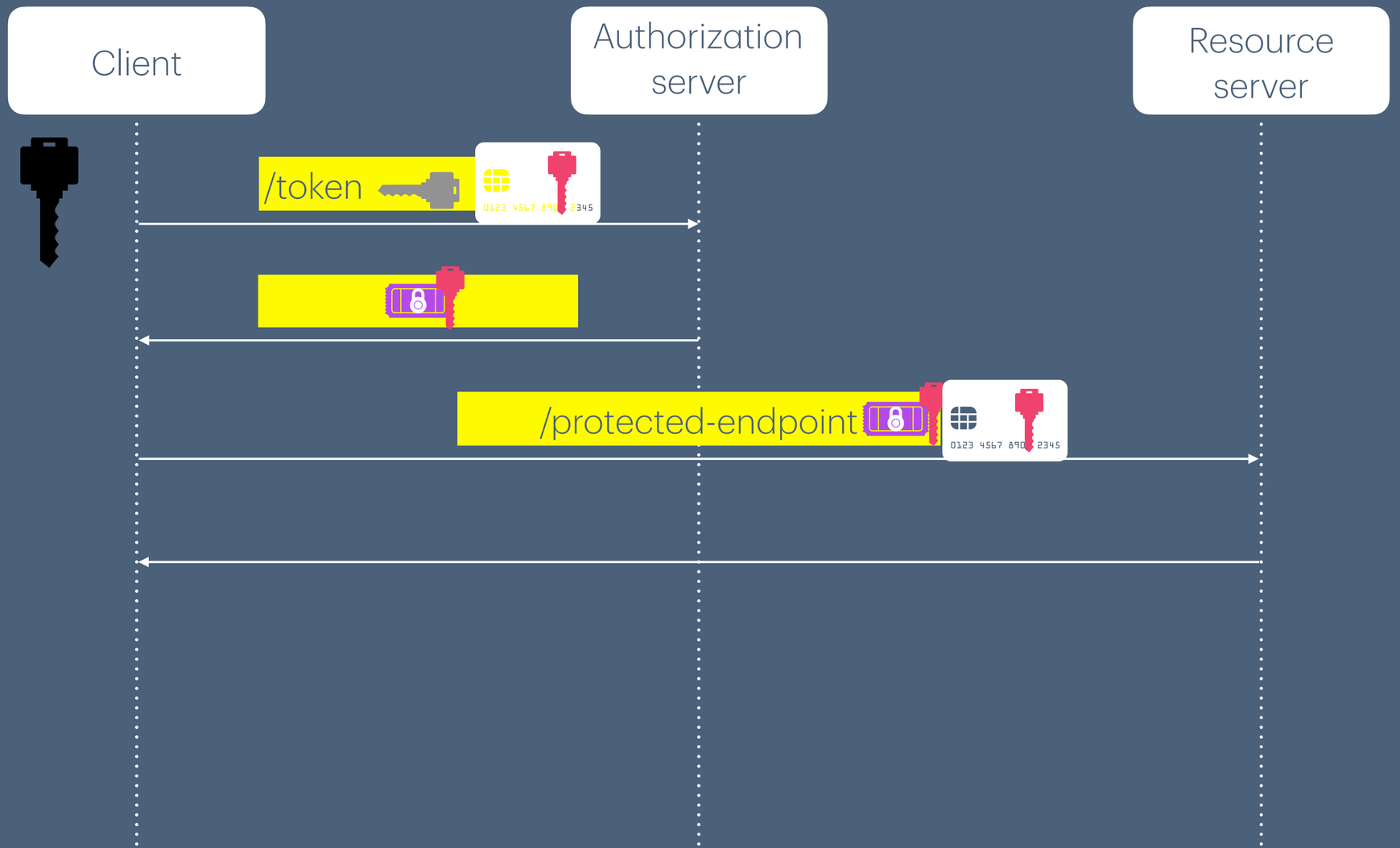
DPOP



RFC 9449

Demonstrating proof of possession

DPOP



RFC 9449

Thank you!

hello@woofie.dev

Questions? Feedback?
Opportunities? Beer?

Socials, PGP, etc. at
woofie.dev/contact

aidn

aidn

aidn