Safeguarding web applications from token theft

Fortifying security and trust in web browsers

Shikhar Kapoor

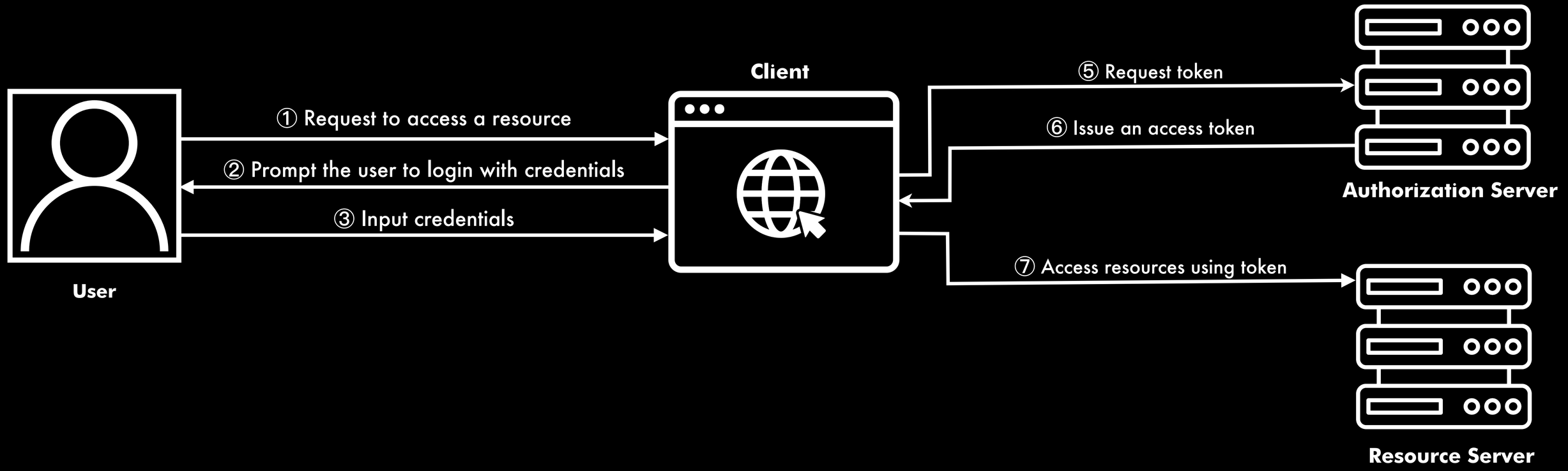
Architect, PhonePe

Authorization on the web

```
<Switch>
    <Route path="/public">
        <PublicPage />
    </Route>
    <Route path="/login">
        <LoginPage />
    </Route>
    <PrivateRoute path="/protected">
        <ProtectedPage />
    </PrivateRoute>
</Switch>
```

OAuth 2.0 & Bearer tokens

Bearer Token authorization



Problems?

Man-in-the-middle

Pass-the-cookie

“The global average cost of a data breach in 2023 was USD 4.45 million.”

IBM Cost of a data breach 2023 (https://www.ibm.com/reports/data-breach)

“In 2023, Attacks using stolen or compromised credentials increased by 71%. Making it one of the predominant attack vectors.”

IBM X-Force Threat Intelligence index 2024 (https://www.ibm.com/downloads/cas/L0GKXDWJ)

- Data breach
- Unauthorised access
- Service abuse
- Impersonation and social engineering
- Denial-of-Service
- Spread of malware
- Reputation damage
- Financial loss

Solution?

Demonstrating Proof-of-Possession

- IETF RFC (on its way to becoming a standard soon!)
- Mechanism to issue sender-constrained tokens

Internet Engineering Task Force (IETF)
Request for Comments: 9449
Category: Standards Track
Published: September 2023
ISSN: 2070-1721

D. Fett
Authlete
B. Campbell
Ping Identity
J. Bradley
Yubico
T. Lodderstedt
Tuconic
M. Jones
Self-Issued Consulting
D. Waite
Ping Identity

OAuth 2.0 Demonstrating Proof of Possession (DPoP)

Abstract

This document describes a mechanism for sender-constraining OAuth 2.0 tokens via a proof-of-possession mechanism on the application level. This mechanism allows for the detection of replay attacks with access and refresh tokens.

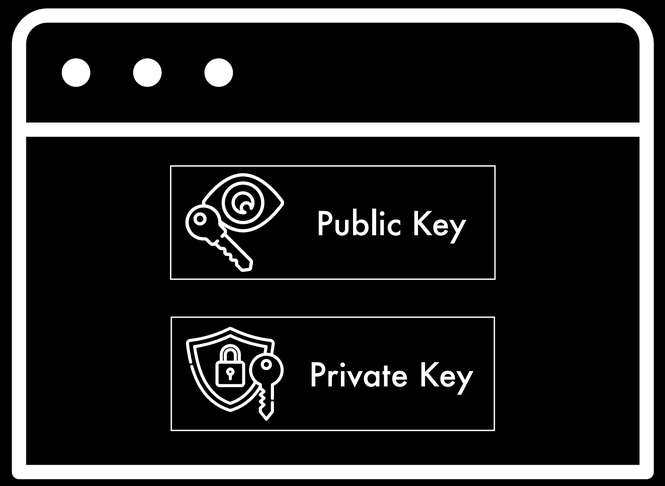
Status of This Memo

This is an Internet Standards Track document.

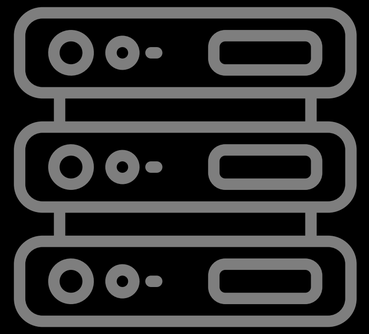
This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Step 1

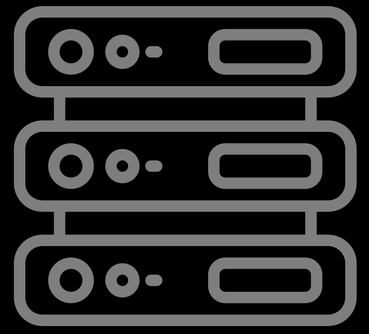
Generate an asymmetric key pair using web crypto APIs



Client



Authorization Server



Resource Server

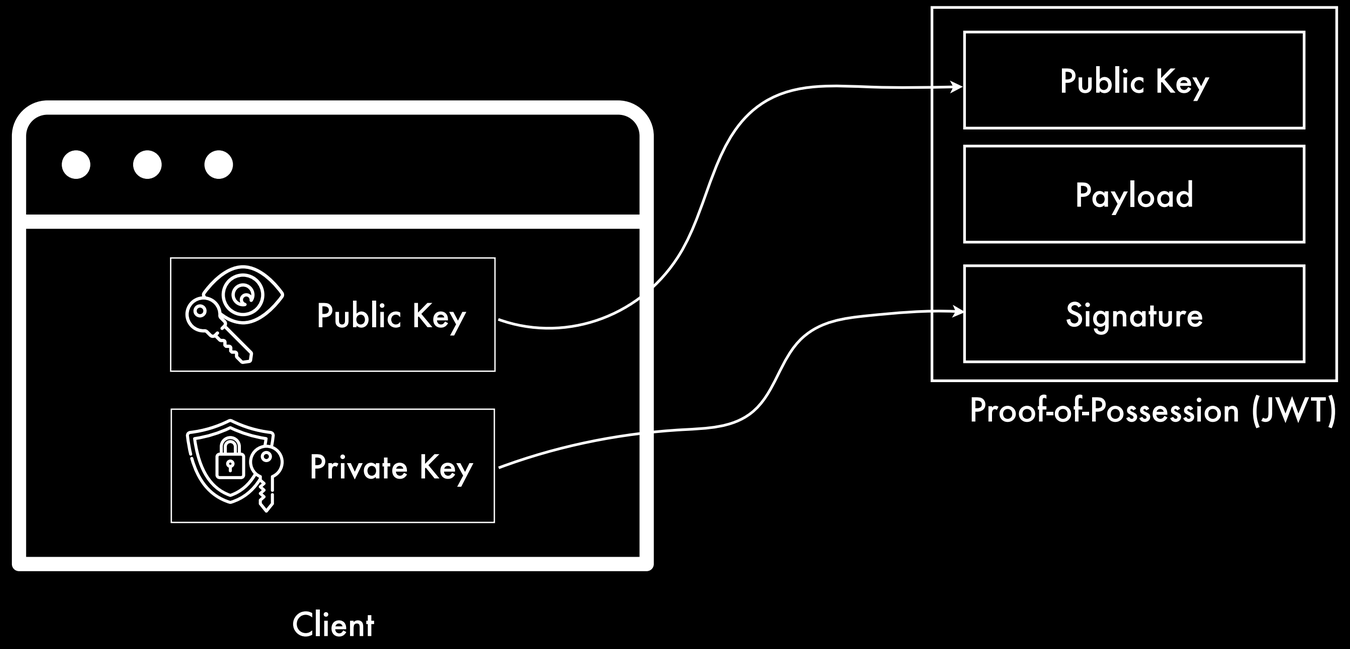
Syntax

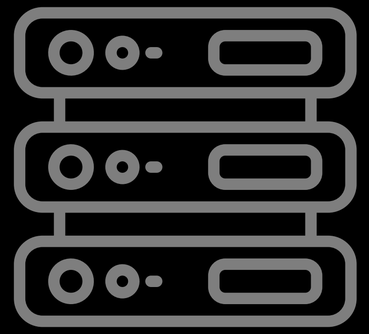
```
generateKey(algorithm, extractable, keyUsages)
```

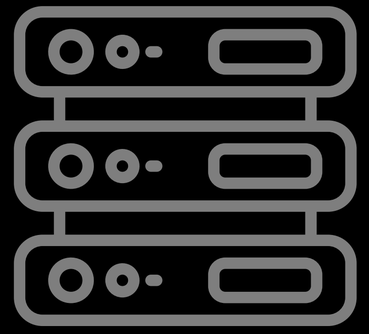
Example

```
generateKey('ECDSA', false, [‘sign’, ‘verify’])
```

Step 2
Create a JWT proof

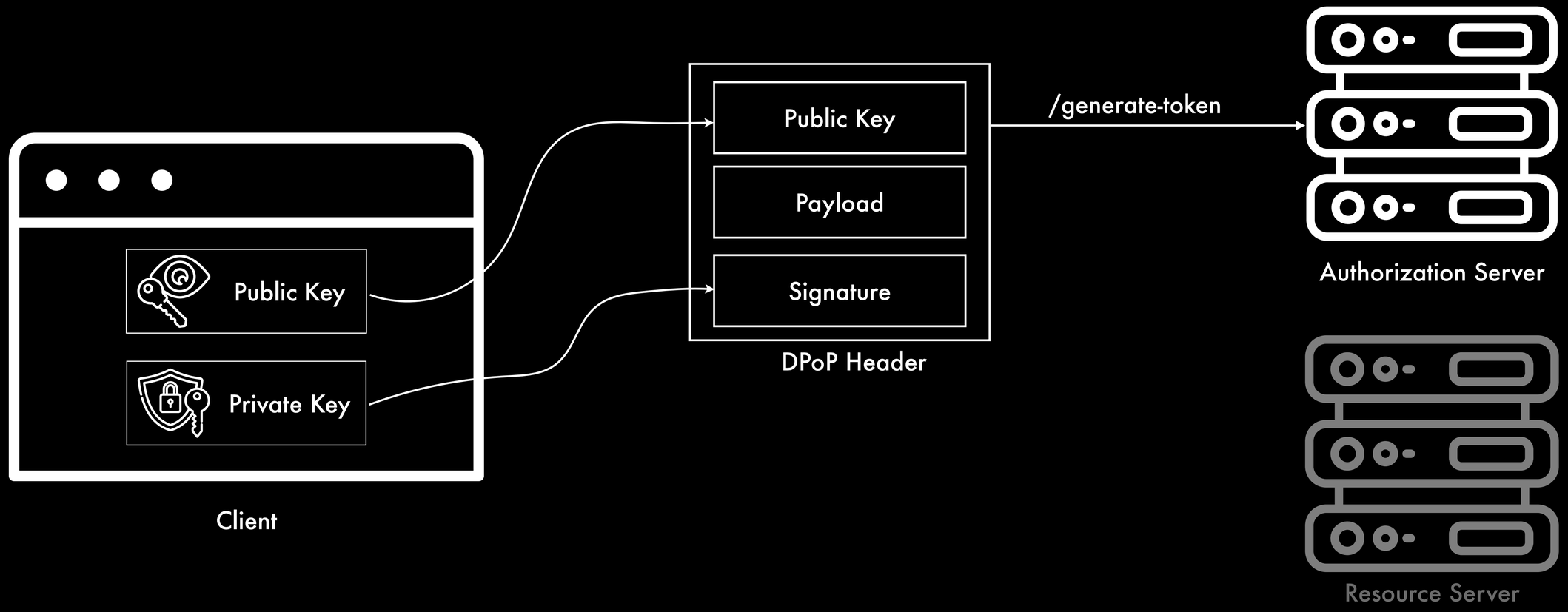






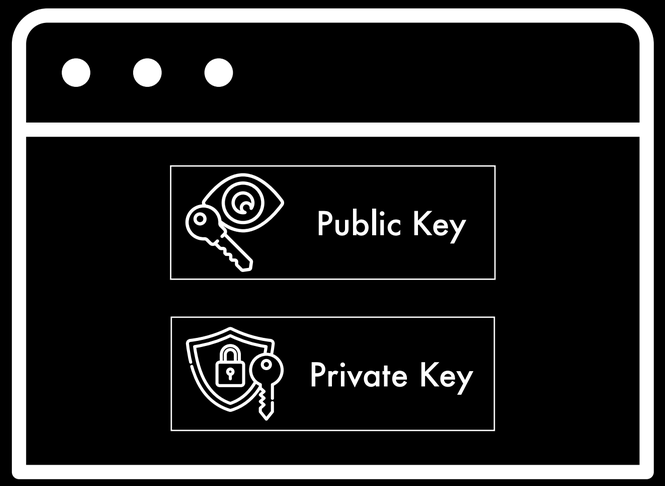
Step 3

Include the header in the authorization request

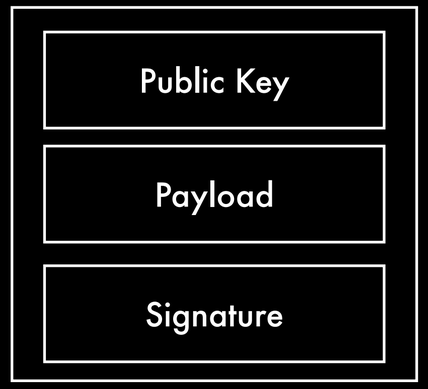


Step 4

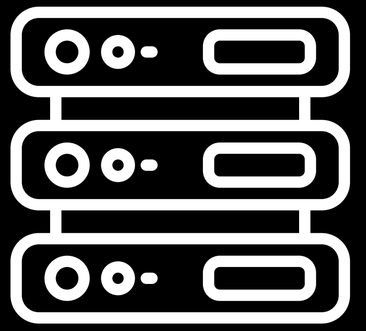
Authorization server verifies the header



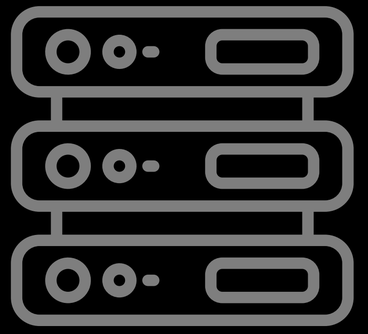
Client



DPoP Header (JWT)



Authorization Server

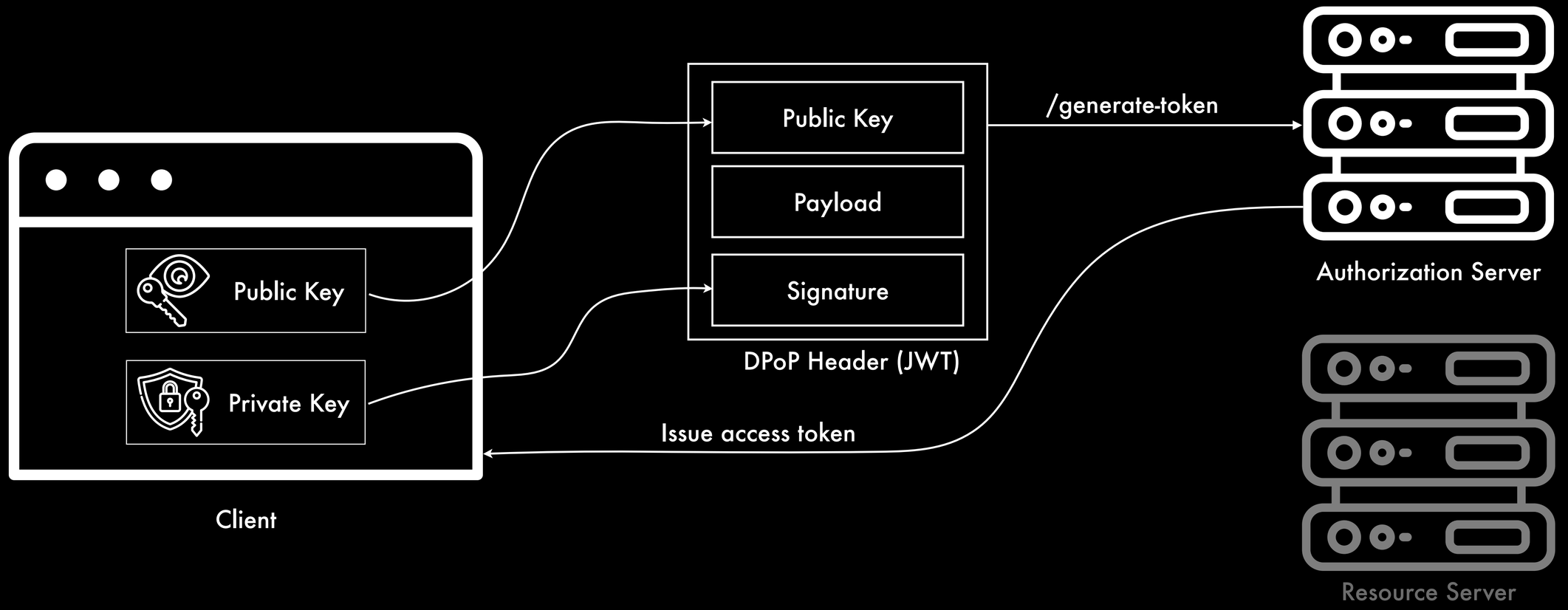


Resource Server

```
const verified = await crypto.subtle.verify("ECDSA", publicKey, signature, payload);
```

Step 5

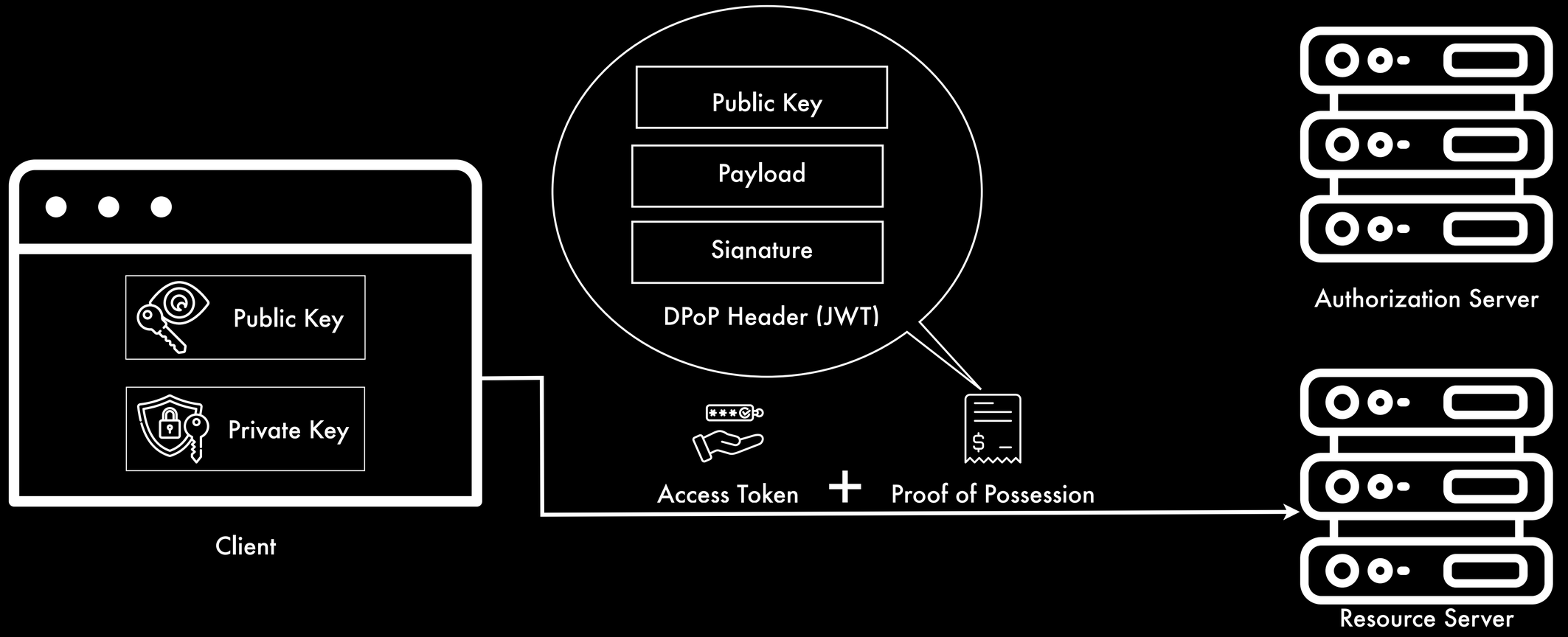
Generate an access token bound to the client’s public key



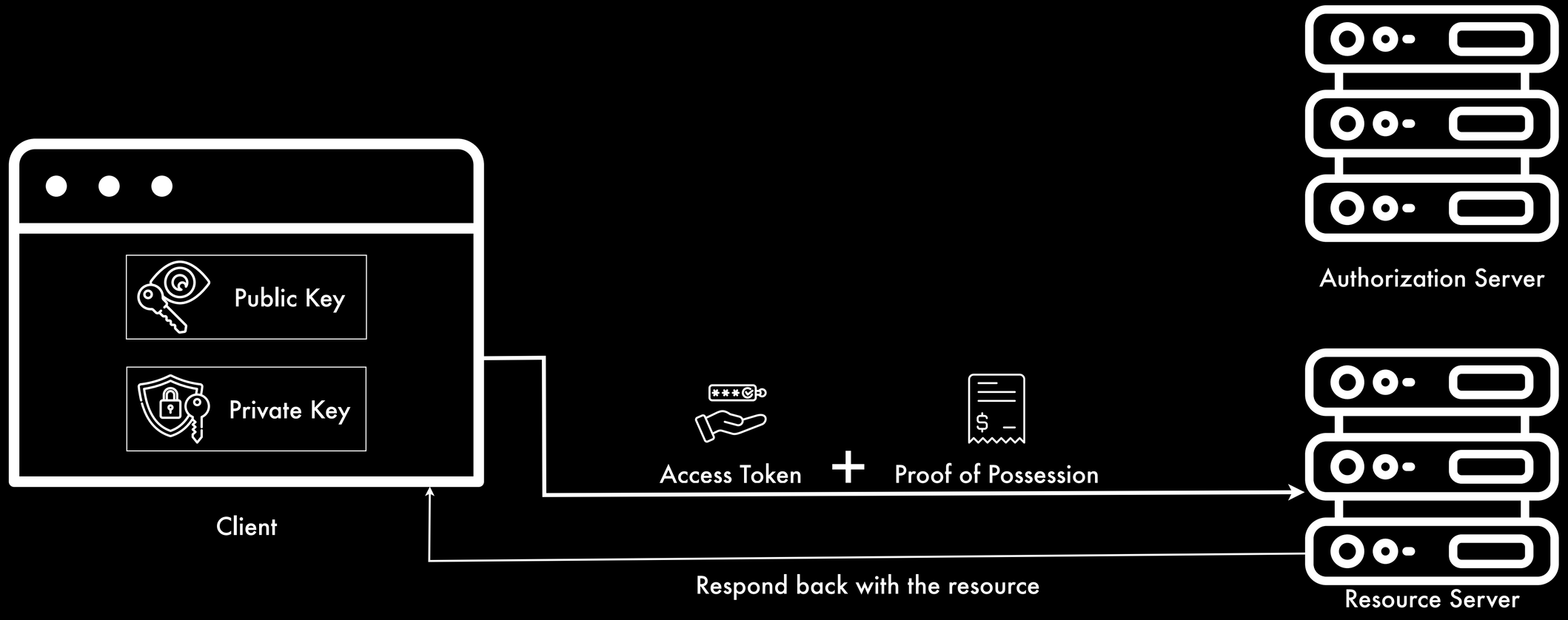
(Payload, PublicKey) === Signature in DPoP Header

Step 6

Use the token to access the protected resource



Step 7
Validate the DPoP header and the access token



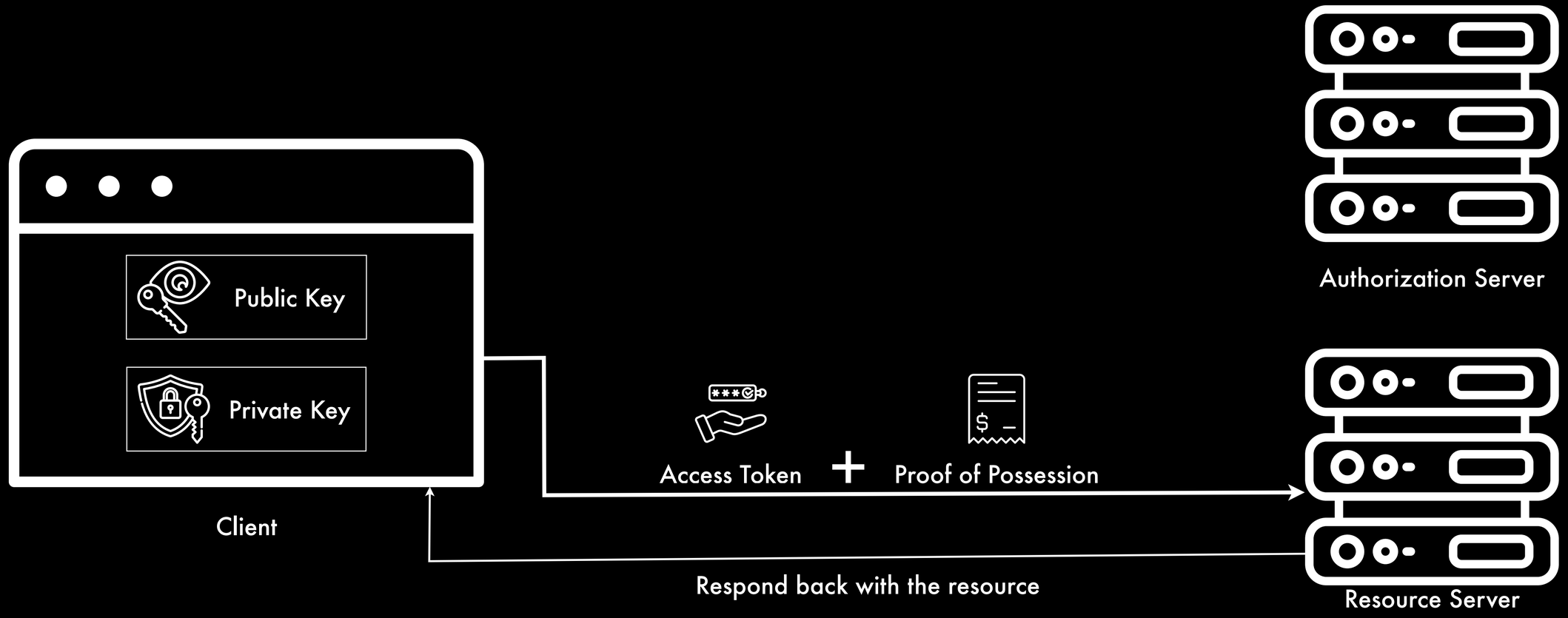
```
GET /protectedresource HTTP/1.1

Host: resource.example.org

Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU

DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik\
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR\
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE\
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj\
 oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z\
 WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOCwiYXRoIjoiZlVIeU8ycjJaM0RaNTNF\
 c05yV0JiMHhXWG9hTnk1OUlpS0NBcWtzbVFFbyJ9.2oW9RP35yRqzhrtNP86L-Ey71E\
 OptxRimPPToA1plemAgR6pxHF8y6-yqyVnmcw6Fy1dqd-jfxSYoMxhAJpLjA
```

Recap


Public Key
Private Key
Client
Access Token
+
Proof of Possession
Authorization Server
Resource Server
Respond back with the resource

Thieves out. Party on!

Not yet.

Replay attacks

Token pre-generation

Key extraction attacks

Replay attacks

Repurposing the same DPoP to repeat the same action over and over again.

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "typ": "dpop+jwt",
  "alg": "ES256",
  "jwk": {
    "kty": "EC",
    "x": "l8tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs",
    "y": "9VE4jf_Ok_o64zbTTlcuNJajHmt6v9TDVrU0CdvGRDA",
    "crv": "P-256"
  }
}
```


PAYLOAD: DATA


```
{
  "jti": "-BwC3ESc6acc2lTc",
  "htm": "POST",
  "htu": "https://server.example.com/token",
  "iat": 1562262616
}
```


Source: https://jwt.io

Token Pre-Generation

Generate DPoP proofs ahead of time and use them later.

Key extraction attacks

Gain control of the CDN and modify JS to make the private key extractable

Replay attacks

Short lived tokens

Token pre-generation

Time sensitive nonce

Key extraction attacks

Sub-resource Integrity

Thank you

Shikhar Kapoor

𝕏 @kapoorshikhar