~~IETF 109, Bangkok~~
OAuth WG Virtual Interim
November 2020

draft-ietf-oauth-dpop

DPoP

OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer

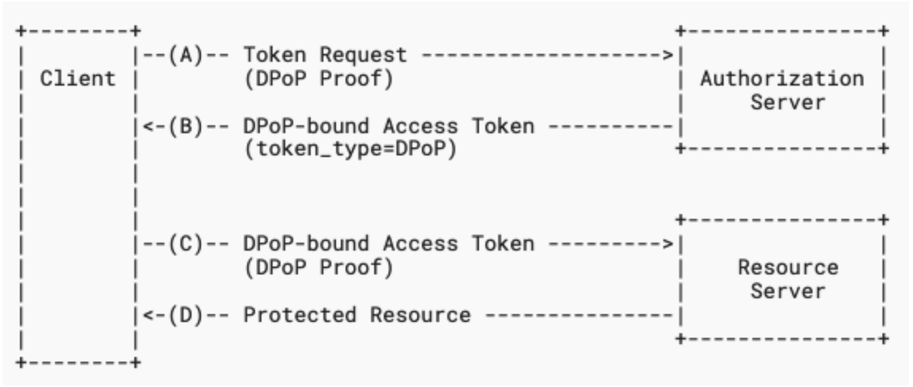
Brian Campbell, Daniel Fett, John Bradley, Michael Jones, David Waite, and Torsten Lodderstedt

DPoP: what it is & what it isn't

- It is:
  - Pragmatic application-level sender-constraining of access and refresh tokens by binding to a key pair (trust on first use style) controlled by the client
- It isn't:
  - An HTTP signature scheme
  - A client to AS authentication mechanism
  - A perfect or infallible solution

DPoP Overview

- DPoP Proof JWT sent as an HTTP header
  - Demonstrates a reasonable level of proof-of-possession in the context of the request
  - Sent the same way with the same syntax and semantics for both
    - token requests to the AS
    - protected resource requests
  - AS uses the proof to bind tokens
  - RS uses the proof to verify bound tokens



```
+--------+                                          +---------------+
|        |--(A)-- Token Request ------------------->|               |
| Client |        (DPoP Proof)                      | Authorization |
|        |                                          |     Server    |
|        |<-(B)-- DPoP-bound Access Token ----------|               |
|        |        (token_type=DPoP)                 +---------------+
|        |
|        |
|        |                                          +---------------+
|        |--(C)-- DPoP-bound Access Token --------->|               |
|        |        (DPoP Proof)                      |    Resource   |
|        |                                          |     Server    |
|        |<-(D)-- Protected Resource ---------------|               |
|        |                                          +---------------+
+--------+
```

DPoP Proof JWT sent in DPoP HTTP Header

```
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
 oiUE9TVCIsImh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
 WF0IjoxNTYyMjYyNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
 4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg
```

Anatomy of a DPoP Proof JWT

Explicitly typed

The public key for which proof-of-possession is being demonstrated

Asymmetric signature algorithms only

```
{
 "typ":"dpop+jwt",
 "alg":"ES256",
 "jwk":
  {
   "kty":"EC", "crv":"P-256"
   "x":"l8tFrhx-34tV3hRICRDY9zCkDlpBhF42UQUfWVAWBFs",
   "y":"9VE4jf_Ok_o64zbTTlcuNJajHmt6v9TDVrU0CdvGRDA"
  }
}.
{
 "jti":"-BwC3ESc6acc2lTc",
 "htm":"POST",
 "htu":"https://server.example.com/token",
 "iat":1562262616
}
```

Minimal info about the HTTP request

Unique identifier for replay checking

Only valid for a limited time window relative to creation time

Other stuff *could* go here

(code) Access Token Request

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
 oiUE9TVCIsImh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
 WF0IjoxNTYyMjYyNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
 4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg

grant_type=authorization_code
&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_verifier=bEaL42izcC-o-xBk0K2vuJ6U-y1p9r_wW2dFWIWgjz-
```

DPoP proof JWT
in HTTP header

Access Token Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-cache, no-store

{
  "access_token":" Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_NeO.gxU",
  "token_type":"DPoP",
  "expires_in":3600,
  "refresh_token":"Q..Zkm29lexi8VnWg2zPW1x-tgGad0Ibc3s3EwM_Ni4-g"
}
```

Token type indicates that the **access token** is bound to the DPoP public key

(refresh) Access Token Request

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
 oiUE9TVCIsImh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
 WF0IjoxNTYyMjY1Mjk2fQ.pAqut2IRDm_De6PR93SYmGBPXpwrAk90e8cP2hjiaG5Qs
 GSuKDYW7_X620BxqhvYC8ynrrvZLTk41mSRroapUA

grant_type=refresh_token
&refresh_token=Q..Zkm29lexi8VnWg2zPW1x-tgGad0Ibc3s3EwM_Ni4-g
```

DPoP proof JWT
in HTTP header

Authorization Server Metadata

- dpop_signing_alg_values_supported:
  - A JSON array containing a list of the JWS alg values supported by the authorization server for DPoP proof JWTs.

DPoP Bound Access Token

JWT & Introspection Response

```
{
   ... other claims / members ...

   "cnf":
   {
     "jkt":"0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
   }
 }
```

Confirmation claim carries the SHA-256 JWK Thumbprint of the DPoP public key to which the access token is bound

Protected Resource Request

DPoP-bound (reference style) 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.eyJqdGkiOiJ1MWozVl9iS2ljOC1MQUVCIiwiaHRtIj
 oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
 WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.lNhmpAX1WwmpBvwhok4E74kWCiGB
 NdavjLAeevGy32H3dbF0Jbri69Nm2ukkwb-uyUI4AUg1JSskfWIyo4UCbQ
```

Token is bound to the key in proof

DPoP proof

401 W/ WWW-Authenticate Challenge

Response To A Protected Resource Request Without A Token

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: DPoP realm="WallyWorld", algs="ES256 PS256”
```

Response To A Protected Resource Request With An Invalid Token

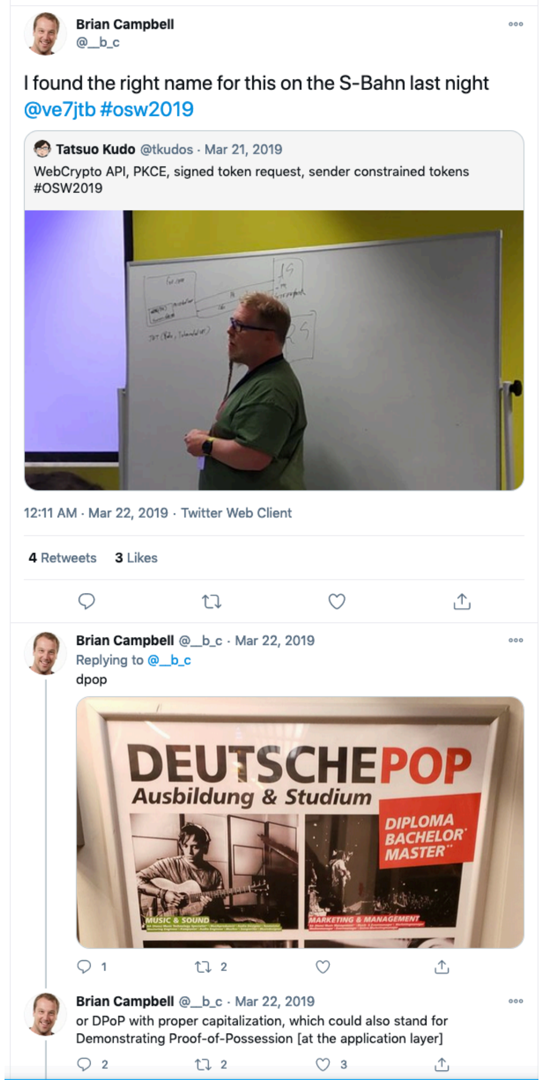
```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: DPoP realm="WallyWorld", error="invalid_token",
   error_description="Invalid DPoP key binding", algs="ES256"
```

History, Status, Updates, etc.

Traveled through Frankfurt retuning from the 4th OAuth Security Workshop in Stuttgart where DPoP was largely conceived thereby justifying the use of this photo

The DPoP Era

- Ideated (more or less) during the Stuttgart 2019 OAuth Security Workshop, March 20–22
- draft-fett-oauth-dpop-00 published March 27
- Presented in Prague 104, March 28
- -01 (-fett) published April 2
- -02 (-fett) published July 8
- Presented in Montreal 105, July 26
- -03 (-fett) published October 30
- Presented in both sessions in Singapore 106, November 16-22
- -04 (-fett) published March 4, 2020
- Pre virtual 107 WG interim on March 9
- Call for adoption started March 17
- draft-ietf-oauth-dpop-00 WG draft published on April 1st (no joke)
- -01 published on May 1
- Post virtual 107 WG interim on May 4
- -02 published on Nov 18
- Post virtual 109 WG interim on Nov 30



Brian Campbell
@_b_c

I found the right name for this on the S-Bahn last night @ve7jtb #osw2019

Tatsuo Kudo @tkudos · Mar 21, 2019
WebCrypto API, PKCE, signed token request, sender constrained tokens #OSW2019

12:11 AM · Mar 22, 2019 · Twitter Web Client

4 Retweets 3 Likes

Brian Campbell @_b_c · Mar 22, 2019
Replying to @_b_c
dpop

1 2

Brian Campbell @_b_c · Mar 22, 2019
or DPoP with proper capitalization, which could also stand for Demonstrating Proof-of-Possession [at the application layer]

2 2 3

What's new in -02

- Lots of editorial updates and additions including **expanding on the objectives**, better defining the key confirmation representations, example updates and additions, **better describing mixed or transitional bearer/DPoP token type deployments**, clarify RT binding only being done for public clients and why, **more clearly allow for a bound RT but with bearer AT**, explain/justify the choice of SHA-256 for key binding, and more
- Require that a protected resource simultaneously supporting bearer and DPoP must reject an access token received as bearer, if that token is DPoP-bound
- Remove the case-insensitive qualification on the htm claim check
- Relax the jti tracking requirements a bit and qualify it by URI

IETF 110, originally planned for Prague in March 2021, would have been the next face-to-face meeting but has been changed to be all virtual due to the pandemic. Here's a photo from Prague anyway for this transition slide.

[some] Points of Discussion

Freshness & Signature Coverage

- Issue:
  - Malicious XSS code executed in the context of the browser-based client can create DPoP proofs with timestamp values in the future and exfiltrate them (along with tokens)
  - These stolen artifacts can later be used together to access protected resources or acquire new access tokens (independent of the client application)
  - Future DPoP proofs could be created for tokens not yet issued
- Current Situation:
  - \`iat\` doesn't prevent pre-computation by an adversary who can use the private key but not steal it (e.g., via XSS)
  - No server contribution to the DPoP proof
  - Token not covered by the DPoP proof
  - Not having a challenge/response (for the proof) was an intentional design choice aimed at simplicity and less overhead

- Some options/ideas … ?
  - It's sufficiently okay as is
    - discussed in draft with key rotation recommended as means to reduce impact
    - Attack vector allows for direct use anyway (reductio ad XSS nihilism)
  - Incorporate (a hash of) the access token into the DPoP proof (and maybe authorization code, refresh token, and other grants too)
  - Allow server to provide, probably via challenge, some contribution to the proof
    - Feels awkward within the current design
    - A challenge per call seems untenable (need to amortize)
  - Other?

Confirmation Bias

- Issue:
  - It's been suggested that, for resource access, having the JWK the proof makes it too easy to just use that key to validate the signature miss checking the binding to the AT's cnf/jkt hash
  - Compared to "alg":"none" (which is the worst hyperbole in the history of time)
  - But not entirely wrong…
- Current Situation:
  - Full JWK in proof
  - JWK hash in AT's confirmation
  - Foot gun?
  - Only one person advocating
- Options:
  - It's fine as is (AS/RS symmetry is nice, similar to MTLS/TB, & kinda fundamental)
  - Put the full JWK in the AT's confirmation and omit it from the proof for resource access (less error prone & no hash function needed for confirmation)

Does the world need a new OAuth client to AS authentication method?

- Issue:
  - DPoP is ***not*** a client authentication method
  - But current posiblity for app layer asymmetric authentication and binding (DPoP + private_key_jw) seems a little odd
- Current Situation
  - But with pre-registered/configured client keys (jwks / jwks_uri) and conveying the client identifier in the request to the AS, DPoP could be a client auth method
- Some options/ideas … ?
  - Add it to this document (ew)
  - Do it in a new document
  - Shut up and never speak of this again

Gratuitous closing slide featuring the city where will meet together next *

* *Maybe* Summer 2021 for #111 in San Francisco