

IETF 110, Prague
OAuth WG Virtual Interim
March 15, 2021

DPoP

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

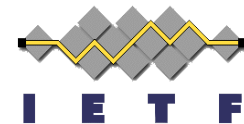
Brian Campbell
Daniel Fett
John Bradley
Michael Jones
David Waite
Torsten Lodderstedt

draft-ietf-oauth-dpop

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

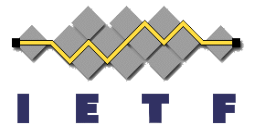
DPoP Overview



- 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

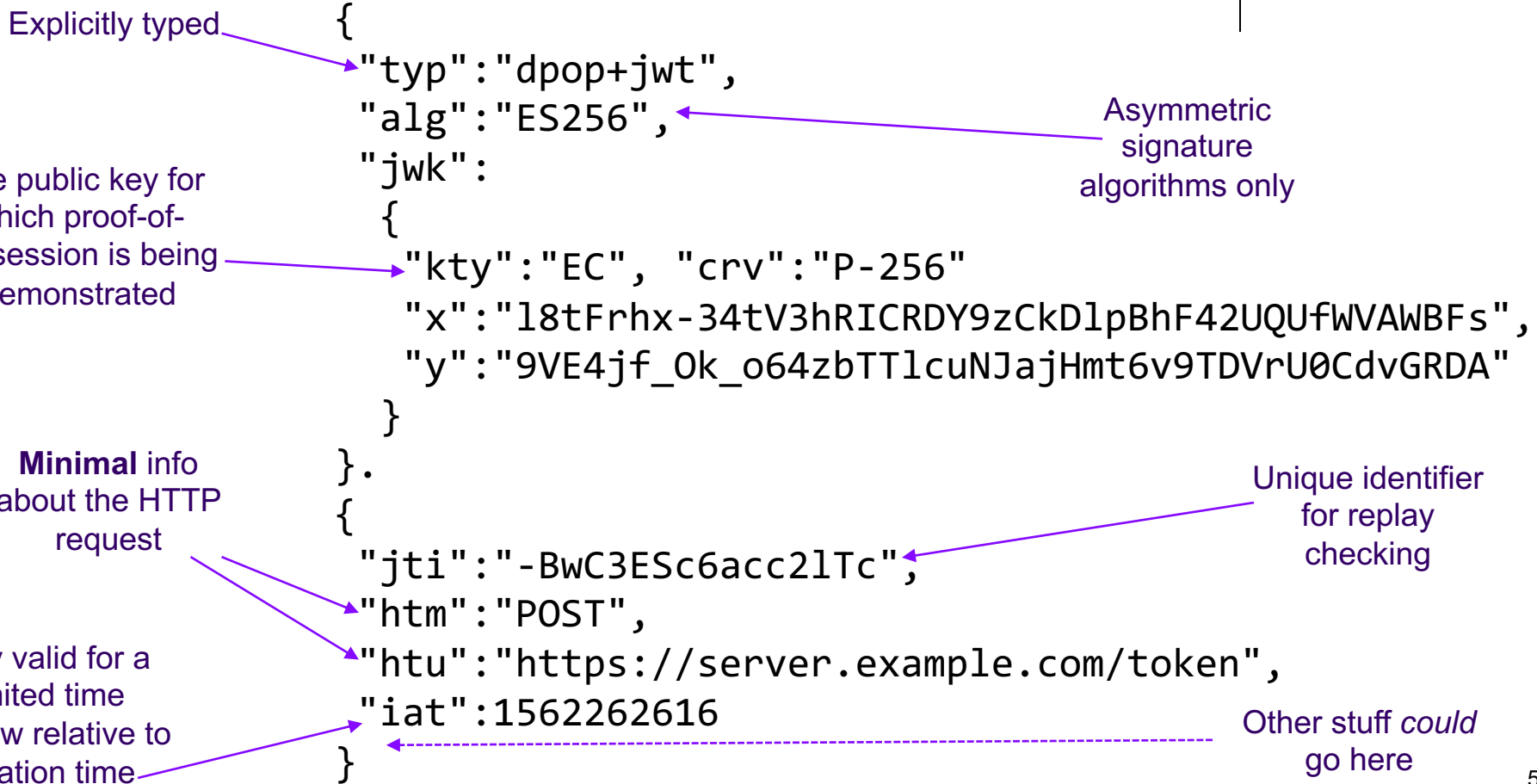


DPoP Proof JWT sent in DPoP HTTP Header

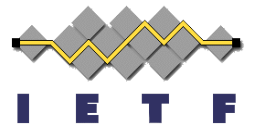


DPoP: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IktVMTJmU2IiwiaWdrIjpw7Imt0eSI6Ik
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
nMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
QSI6ImNydiI6I1AtMjU2In19.eyJqdGkiOiItQndDM0VTYzZhY2MybFRjIiwiaHRtIj
oiUE9TVCI6Imh0dSI6Imh0dHBzOi8vc2VydmVyLmV4YW1wbGUuY29tL3Rva2VuIiwia
WF0IjoxNTYyMjYyNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
4PtFLbdLXi0SsX0x7NVY-FNyJK70nfbV37xRZT3Lg

Anatomy of a DPoP Proof JWT



(code) Access Token Request

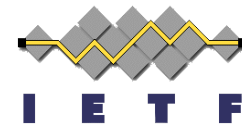


```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
DPoP: eyJ0eXAiOiJKcG9wK2p3dCI6ImFsZyI6IktVMjU2IiwiaWdrIjpw7Imt0eSI6Ik
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZlZlZlVdCR
nMiLCJ5Ijojoi0VZFNGpmX09rX282NHpiVFRsY3V0SmFqSG10NnY5VERWc1UwQ2R2R1JE
QSI6ImNydiI6IiAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZzhY2MybFRjIiwiaHRtIj
oiUE9TVCI6Imh0dSI6Imh0dHBzOi8vc2VydmlvYmV4YV1wbGUuY29tL3Rva2VuIiwia
WF0IjojNTYyMjYyYnNjE2fQ.2-GxA6T8lP4vfrg8v-FdWP0A0zdrj8igiMLvqRMUvwnQg
4PtFLbdLXiOSsX0x7NVY-FNyJK70nfbV37xRZT3Lg
```

```
grant_type=authorization_code
&code=Sp1xl0BeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&code_verifier=bEaL42izcC-o-xBk0K2vuJ6U-y1p9r_wW2dFWIWgjz-
```

DPoP proof JWT
in HTTP header

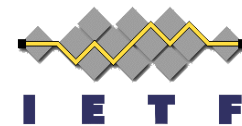
Access Token Response



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

```
{
  "access_token": "Kz~8mXK1EaIYznwH-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: eyJ0eXAiOiJKcG9wK2p3dCI6ImFsZyI6IkVMTmJGU2IiwiaWdrIjp7Imt0eSI6IkVDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZlZlVWVdCRnMiLCJ5IjoioVZFNGpmX09rX282NHpiVFRsY3V0SmFqSG10NnY5VERWc1UwQ2R2R1JEQSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiItQndDM0VTYzZHY2MybFRjIiwiaHRtIjoiUE9TVCI6Imh0dSI6Imh0dHBzOi8vc2VydMvYlMvV4YW1wbGUuY29tL3Rva2VuIiwiaWF0IjoxNTYyMjY1Mjk2fQ.pAqut2IRDm_De6PR93SYmGBPxpwrAk90e8cP2hjiaG5QsGSuKDYW7_X620BxqhVYC8ynrrvZLTk41mSRroapUA

grant_type=refresh_token

&refresh_token=Q..Zkm29lexi8VnWg2zPW1x-tgGad0Ibc3s3EwM_Ni4-g

DPoP proof JWT
in HTTP header

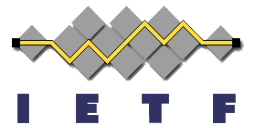


Authorization Server Metadata

- `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

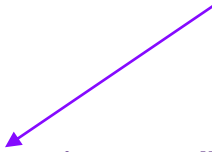
JWT & Introspection Response



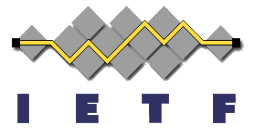
```
{
  ... other claims / members ...

  "cnf":
  {
    "jkt": "0ZcOCORZNYy-DWpqq30jZyJGHTN0d2Hg1BV3uiguA4I"
  }
}
```

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



Protected Resource Request



GET /protectedresource HTTP/1.1

Host: resource.example.org

Authorization: DPoP Kz~8mXK1EalYznwH-LC-1fBAo.4Ljp~zsPE_Ne0.gxU

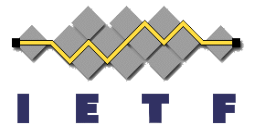
DPoP: eyJ0eXAiOiJkcG9wK2p3dCI6ImFsZyI6IktVMjU2IiwiaWdrIjpw7Imt0eSI6Ik
VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUk1DUkRZOXpDa0RscEJoRjQyVVFVZldwQVdCR
nMiLCJ5IjoioVZFNzGpmX09rX282NHpiVFRsY3V0SmFqSG10NnY5VERWclUwQ2R2R1JE
QSI6ImNydiI6IiAtMjU2In19.eyJqdGkiOiJlMwozVl9iS2ljOC1MQUVCIiwiaHRtIj
oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlM9yZy9wcm90ZWNoZ
WRyZXNvdXJjZSI6Im1hdCI6MTU2MjU2MjYxOH0.1NhmpAX1WwmpBvwhok4E74kWCiGB
NdavjLAeevGy32H3dbF0Jbri69Nm2ukkwb-uyUI4AUG1JSSskfWIyo4UCbQ

DPoP-bound
(reference
style) access
token

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"

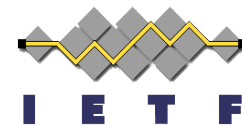
Status Update

- Published changes since the last interim:



Bangkok's abandoned Ghost Tower is representative of the amount of publishing on the draft since the last meeting, which was one of a series of interims held in place of the meeting that would have been in Bangkok, if not for the global pandemic. 13

No new draft?!



Rifaat: Is any against staying the way the draft is?

Brian: I will summarize the options and post **to the list.**

Confirmation Bias

Brian: reviewed slide

Justin: there are lots of other similar issues where the client may not perform all of its checks.

Daniel: valid point – but by making specification very explicit on what to check and the proper order an implementation should be able to avoid these issues

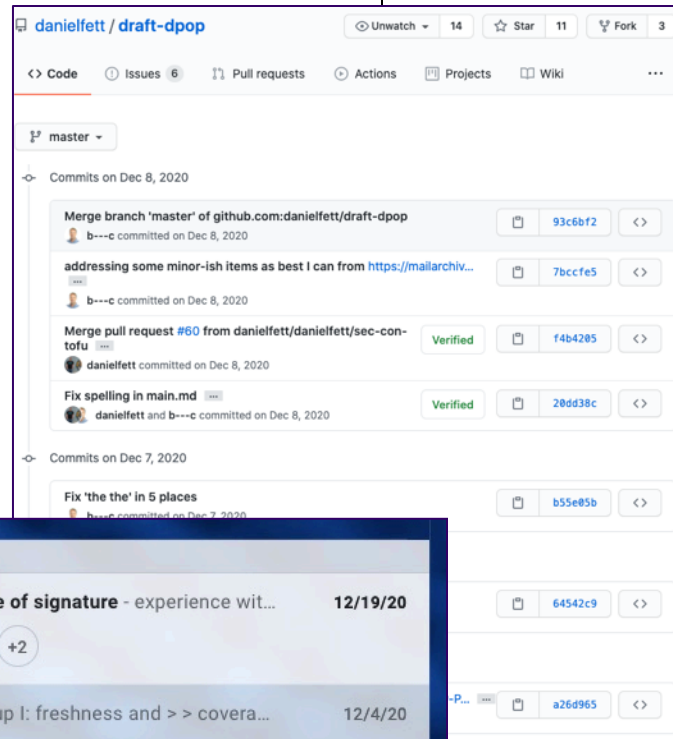
Brian: I will take this issue **to the list** and provide the choices.

Does the World need a new OAuth client to AS Authentication method?

Brian: reviewed slide.

Rifaat: out of time. Please take all three of these issues to the list.

Brian: will take **to list.**

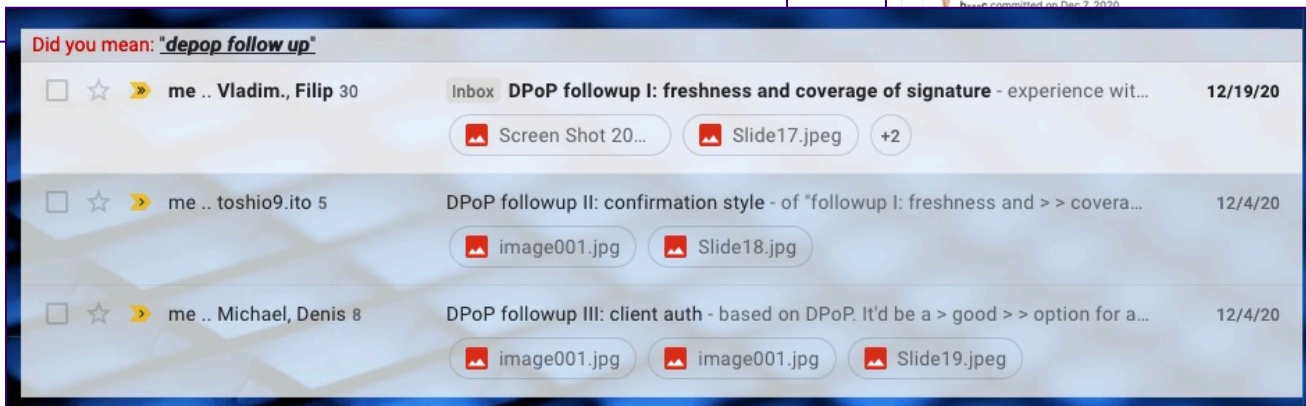


Consensus
has been
somewhat
elusive

?

?

X



Freshness & Signature Coverage (for lack of a better name)



- 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 (maybe via challenge) some nonce like contribution to the proof
 - Feels awkward within the current design (including AS vs RS differences)
 - A challenge per call seems untenable (need to amortize but then how does that work?)
 - Others...

Proposed Path Forward



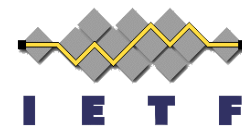
- Let XSS Nihilism Prevail
 - “But if XSS is game over, let's not bother with trying to patch one particular scenario with a hash.”
- No protocol changes
- Some editorial changes in the form of yet-to-be-published considerations

```
Showing 1 changed file with 48 additions and 0 deletions.

48 main.md

@@ -812,6 +812,43 @@ Note: To accommodate for clock offsets, the server MAY accept DPoP
812 812     proofs that carry an 'iat' time in the reasonably near future (e.g., a few
813 813     seconds in the future).
814 814
815 + ## Untrusted Code in the Client Context
816 +
817 + If an adversary is able to run code in the client's execution context,
818 + the security of DPoP is no longer guaranteed. Common issues in web
819 + applications leading to the execution of untrusted code are cross-site
820 + scripting and remote code inclusion attacks.
821 +
822 + If the private key used for DPoP is stored in such a way that it
823 + cannot be exported, e.g., in a hardware or software security module,
824 + the adversary cannot exfiltrate the key and use it to create arbitrary
825 + DPoP proofs. The adversary can, however, create new DPoP proofs as
826 + long as the client is online, and use these proofs (together with the
827 + respective tokens) either on the victim's device or on a device under
828 + the attacker's control to send arbitrary requests that will be
829 + accepted by servers.
830 +
831 + To send requests even when the client is offline, an adversary can try
832 + to pre-compute DPoP proofs using timestamps in the future and
833 + exfiltrate these together with the access or refresh token.
834 +
835 + An adversary might further try to associate tokens issued from the
836 + token endpoint with a key pair under the adversary's control. One way
837 + to achieve this is to modify existing code, e.g., by replacing
838 + cryptographic APIs. Another way is to launch a new authorization grant
839 + between the client and the authorization server in an iframe. This
840 + grant needs to be "silent", i.e., not require interaction with the
841 + user. With code running in the client's origin, the adversary has
842 + access to the resulting authorization code and can use it to associate
843 + their own DPoP keys with the tokens returned from the token endpoint.
844 + The adversary is then able to use the resulting tokens on their own
845 + device even if the client is offline.
846 +
847 + Therefore, protecting clients against the execution of untrusted code
848 + is extremely important even if DPoP is used. Besides secure coding
849 + practices, Content Security Policy (CSP) can be used as a second
850 + layer of defense against cross-site scripting.
851 +
```

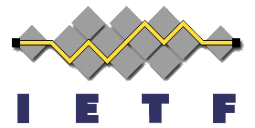

Confirmation Bias



- Issue:
 - It's been suggested that, for resource access, having the JWK in the header of the DPoP proof JWT makes it too easy to just use that key to validate the signature and 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 really 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)



A Decent Proposal



- Remove the foot gun
 - full JWK in the access token confirmation and omit it from the proof on resource access

proof

```
{
  "typ": "dpop+jwt",
  "alg": "ES256"
}.
{
  "jti": "-BwC3ESc6acc2lTc",
  "htm": "POST",
  "htu": "https://rs.example.io/stuff",
  "iat": 1562262616
}
```


access token

```
{
  [... other claims / members ...]
  "cnf":
  {
    "jwk":
    {
      "kty": "EC",
      "crv": "P-256",
      "x": "l8tFrhx-34tV3hRICRDY9zCkDlPbBhF42UQUfWVAWBFs",
      "y": "9VE4jf_ok_o64zbTTlCuNJajHmt6v9TDVrU0CdvGRDA"
    }
  }
}
```

A red arrow points from the 'jwk' field in the 'proof' JSON to the 'jwk' field in the 'access token' JSON. A large red 'X' is drawn over the 'cnf' field in the 'access token' JSON, indicating that the full JWK should be removed from the confirmation.

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

* “IETF 111 San Francisco ... seems highly unlikely that an in-person meeting can go ahead” - IETF Executive Director



[IETF 111 San Francisco >](#)
IETF 111 starts Saturday 24 July and runs through Friday afternoon, 30 July.
| San Francisco