Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 2 November 2020

D. Fett
yes.com
B. Campbell
Ping Identity
J. Bradley
Yubico
T. Lodderstedt
yes.com
M. Jones
Microsoft
D. Waite
Ping Identity
1 May 2020

OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer (DPoP)
draft-ietf-oauth-dpop-01

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 Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 November 2020.

Copyright Notice

Table of Contents

1. Introduction

This document outlines a relatively simple application-level mechanism for sender-constraining OAuth access and refresh tokens. It enables a client to demonstrate proof-of-possession of a public/private key pair by including the "DPoP" header in an HTTP request. Using that header, an authorization server is able to bind issued tokens to the public part of the client's key pair. Recipients of such tokens are then able to verify the binding of the token to the key pair that the client has demonstrated that it holds via the "DPoP" header, thereby providing some assurance that the client presenting the token also possesses the private key. In other words, the legitimate presenter of the token is constrained to be the sender that holds and can prove possession of the private part of the key pair.

The mechanism described herein can be used in cases where potentially stronger methods of sender-constraining tokens that utilize elements of the underlying secure transport layer, such as [RFC8705] or [I-D.ietf-oauth-token-binding], are not available or desirable. For example, due to a sub-par user experience of TLS client authentication in user agents and a lack of support for HTTP token binding, neither mechanism can be used if an OAuth client is a Single Page Application (SPA) running in a web browser.

DPoP can be used with public clients to sender-constrain access tokens and refresh tokens. With confidential clients, DPoP can be used in conjunction with any client authentication method to sender-constrain access tokens.

1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [RFC6749].

2. Main Objective

Under the attacker model defined in [I-D.ietf-oauth-security-topics], the mechanism defined by this specification aims to prevent token replay at a different endpoint.

More precisely, if an adversary is able to get hold of an access token or refresh token because it set up a counterfeit authorization server or resource server, the adversary is not able to replay the respective token at another authorization or resource server.

Secondary objectives are discussed in Section 9.

3. Concept

The main data structure introduced by this specification is a DPoP proof JWT, described in detail below, sent as a header in an HTTP request. A client uses a DPoP proof JWT to prove the possession of a private key corresponding to a certain public key. Roughly speaking, a DPoP proof is a signature over a timestamp and some data of the HTTP request to which it is attached.

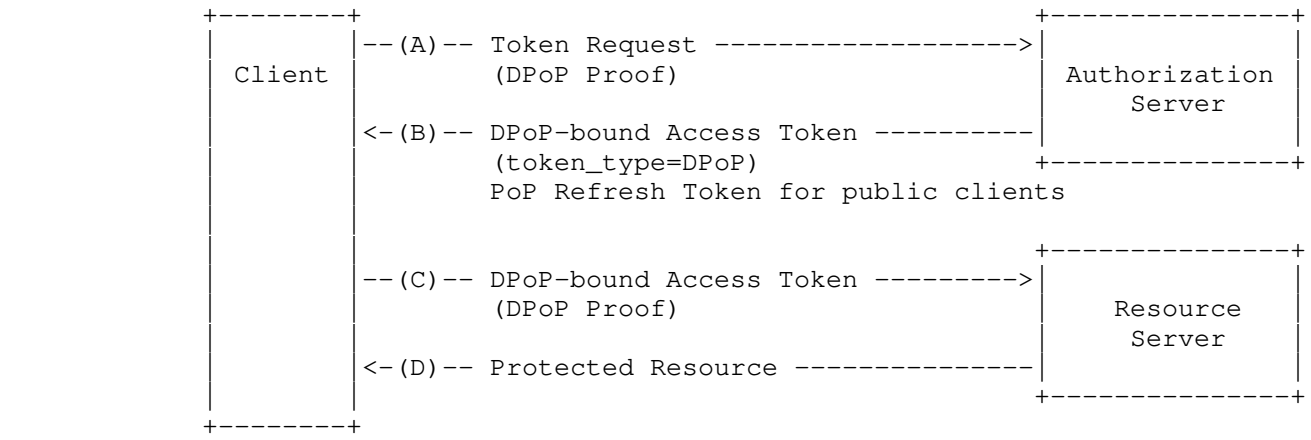

Figure 1: Basic DPoP Flow

The basic steps of an OAuth flow with DPoP are shown in Figure 1:

* (A) In the Token Request, the client sends an authorization code to the authorization server in order to obtain an access token (and potentially a refresh token). The client attaches a DPoP proof to the request in an HTTP header.

* (B) The AS binds (sender-constrains) the access token to the public key claimed by the client in the DPoP proof; that is, the access token cannot be used without proving possession of the respective private key. This is signaled to the client by using the "token_type" value "DPoP".

* If a refresh token is issued to a public client, it is bound to the public key of the DPoP proof in a similar way. Note that for confidential clients, refresh tokens are required by [RFC6749] to bound to the "client_id" and associated authentication credentials, which is a sender-constraining mechanism that is more flexible than binding to a particular public key.

* (C) If the client wants to use the access token, it has to prove possession of the private key by, again, adding a header to the request that carries the DPoP proof. The resource server needs to receive information about the public key to which the access token is bound. This information is either encoded directly into the access token (for JWT structured access tokens), or provided at the token introspection endpoint of the authorization server (not shown). The resource server verifies that the public key to which the access token is bound matches the public key of the DPoP proof.

* (D) The resource server refuses to serve the request if the signature check fails or the data in the DPoP proof is wrong, e.g., the request URI does not match the URI claim in the DPoP proof JWT.

* When a refresh token that is sender-constrained using DPoP is used by the client, the client has to provide a DPoP proof just as in the case of a resource access. The new access token will be bound to the same public key.

The mechanism presented herein is not a client authentication method. In fact, a primary use case of DPoP is for public clients (e.g., single page applications) that do not use client authentication. Nonetheless, DPoP is designed such that it is compatible with "private_key_jwt" and all other client authentication methods.

DPoP does not directly ensure message integrity but relies on the TLS layer for that purpose. See Section 9 for details.

4. DPoP Proof JWTs

DPoP introduces concept of a DPoP proof JWT, which is used for binding public keys and proving knowledge about private keys. The DPoP proof JWT is sent with an HTTP request using the "DPoP" header field.

4.1. DPoP Proof JWT Syntax

A DPoP proof is a JWT ([RFC7519]) that is signed (using JWS, [RFC7515]) using a private key chosen by the client (see below). The header of a DPoP JWT contains at least the following parameters:

* "typ": type header, value "dpop+jwt" (REQUIRED).

* "alg": a digital signature algorithm identifier as per [RFC7518] (REQUIRED). MUST NOT be "none" or an identifier for a symmetric algorithm (MAC).

* "jwk": representing the public key chosen by the client, in JWK format, as defined in [RFC7515] (REQUIRED)

The body of a DPoP proof contains at least the following claims:

* "jti": Unique identifier for the DPoP proof JWT (REQUIRED). The value MUST be assigned such that there is a negligible probability that the same value will be assigned to any other DPoP proof used in the same context during the time window of validity. Such uniqueness can be accomplished by encoding (base64url or any other suitable encoding) at least 96 bits of pseudorandom data or by using a version 4 UUID string according to [RFC4122]. The "jti" SHOULD be used by the server for replay detection and prevention, see Section 9.1.

* "htm": The HTTP method for the request to which the JWT is attached, as defined in [RFC7231] (REQUIRED).

* "htu": The HTTP URI used for the request, without query and fragment parts (REQUIRED).

* "iat": Time at which the JWT was created (REQUIRED).

Figure 2 shows the JSON header and payload of a DPoP proof JWT.

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

Figure 2: Example JWT content for "DPoP" proof header

Note: To keep DPoP simple to implement, only the HTTP method and URI are signed in DPoP proofs. The idea is sign just enough of the HTTP data to provide reasonable proof-of-possession with respect to the HTTP request. But that it be a minimal subset of the HTTP data so as to avoid the substantial difficulties inherent in attempting to normalize HTTP messages. Nonetheless, DPoP proofs can be extended to contain other information of the HTTP request (see also Section 9.4).

4.2. Checking DPoP Proofs

To check if a string that was received as part of an HTTP Request is a valid DPoP proof, the receiving server MUST ensure that

1. the string value is a well-formed JWT,

2. all required claims are contained in the JWT,

3. the "typ" field in the header has the value "dpop+jwt",

4. the algorithm in the header of the JWT indicates an asymmetric digital signature algorithm, is not "none", is supported by the application, and is deemed secure,

5. that the JWT is signed using the public key contained in the "jwk" header of the JWT,

6. the "htm" claim matches the HTTP method value of the HTTP request in which the JWT was received (case-insensitive),

7. the "htu" claims matches the HTTP URI value for the HTTP request in which the JWT was received, ignoring any query and fragment parts,

8. the token was issued within an acceptable timeframe (see Section 9.1), and

9. that, within a reasonable consideration of accuracy and resource utilization, a JWT with the same "jti" value has not been received previously (see Section 9.1).

Servers SHOULD employ Syntax-Based Normalization and Scheme-Based Normalization in accordance with Section 6.2.2. and Section 6.2.3. of [RFC3986] before comparing the "htu" claim.

5. Token Request (Binding Tokens to a Public Key)

To bind a token to a public key in the token request, the client MUST provide a valid DPoP proof JWT in a "DPoP" header. The HTTPS request shown in Figure 3 illustrates the protocol for this (with extra line breaks for display purposes only).

```
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-
```

Figure 3: Token Request for a DPoP sender-constrained token

The "DPoP" HTTP header MUST contain a valid DPoP proof JWT. If the DPoP proof is invalid, the authorization server issues an error response per Section 5.2 of [RFC6749] with "invalid_dpop_proof" as the value of the "error" parameter.

The authorization server, after checking the validity of the DPoP proof, associates the access token issued at the token endpoint with the public key. It then sets "token_type" to "DPoP" in the token response, which signals to the client that the access token was bound to its DPoP key and can used as described in Section 6.

If a refresh token is issued to a public client at the token endpoint and a valid DPoP proof is presented, the refresh token MUST be bound to the public key contained in the header of the DPoP proof JWT.

When a DPoP-bound refresh token is used at the token endpoint by a public client, the AS MUST ensure that the DPoP proof contains the same public key as the one the refresh token is bound to. The access token issued MUST be bound to the public key contained in the DPoP proof.

6. Resource Access (Proof of Possession for Access Tokens)

To make use of an access token that is bound to a public key using DPoP, a client MUST prove the possession of the corresponding private key by providing a DPoP proof in the "DPoP" request header.

A DPoP-bound access token is sent using the "Authorization" request header field per Section 2 of [RFC7235] using an authentication scheme of "DPoP". The syntax of the "Authorization" header field for the "DPoP" scheme uses the "token68" syntax defined in Section 2.1 of [RFC7235] (repeated below for ease of reference) for credentials. The Augmented Backus-Naur Form (ABNF) notation [RFC5234] syntax for DPoP Authorization scheme credentials is as follows:

```
token68    = 1*( ALPHA / DIGIT /
                 "-" / "." / "_" / "~" / "+" / "/" ) *"="

credentials = "DPoP" 1*SP token68
```

Figure 4: DPoP Authorization Scheme ABNF

For such an access token, a resource server MUST check that a "DPoP" header was received in the HTTP request, check the header's contents according to the rules in Section 4.2, and check that the public key of the DPoP proof matches the public key to which the access token is bound per Section 7.

The resource server MUST NOT grant access to the resource unless all checks are successful.

```
GET /protectedresource HTTP/1.1
Host: resource.example.org
Authorization: DPoP eyJhbGciOiJFUzI1NiIsImtpZCI6IkJlQUxrYiJ9.eyJzdWI
 iOiJzb21lb25lQGV4YW1wbGUuY29tIiwiaXNzIjoiaHR0cHM6Ly9zZXJ2ZXIuZXhhbX
 BsZS5jb20iLCJhdWQiOiJodHRwczovL3Jlc291cmNlLmV4YW1wbGUub3JnIiwibmJmI
 joxNTYyMjYyNjExLCJleHAiOjE1NjIyNjYyMTYsImNuZiI6eyJqa3QiOiIwWmNPQ09S
 Wk5ZeS1EV3BxcTMwalp5SkdIVE4wZDJIZ2xCVjN1aWd1QTRJIn19.vsFiVqHCyIkBYu
 50c69bmPJsj8qYlsXfuC6nZcLl8YYRNOhqMuRXu6oSZHe2dGZY0ODNaGg1cg-kVigzY
 hF1MQ
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik
 VDIiwieCI6Imw4dEZyaHgtMzR0VjNoUklDUkRZOXpDa0RscEJoRjQyVVFVZldWQVdCR
 nMiLCJ5IjoiOVZFNGpmX09rX282NHpiVFRsY3VOSmFqSG10NnY5VERWclUwQ2R2R1JE
 QSIsImNydiI6IlAtMjU2In19.eyJqdGkiOiJlMWozVl9iS2ljOC1MQUVCIiwiaHRtIj
 oiR0VUIiwiaHR1IjoiaHR0cHM6Ly9yZXNvdXJjZS5leGFtcGxlLm9yZy9wcm90ZWN0Z
 WRyZXNvdXJjZSIsImlhdCI6MTU2MjI2MjYxOH0.lNhmpAX1WwmpBvwhok4E74kWCiGB
 NdavjLAeevGy32H3dbF0Jbri69Nm2ukkwb-uyUI4AUg1JSskfWIyo4UCbQ
```

Figure 5: Protected Resource Request with a DPoP sender-constrained access token

Upon receipt of a request for a URI of a protected resource within the protection space requiring DPoP authorization, if the request does not include valid credentials or or does not contain an access token sufficient for access to the protected resource, the server can reply with a challenge using the 401 (Unauthorized) status code ([RFC7235], Section 3.1) and the "WWW-Authenticate" header field ([RFC7235], Section 4.1). The server MAY include the "WWW-Authenticate" header in response to other conditions as well.

In such challenges:

* The scheme name is "DPoP".

* The authentication parameter "realm" MAY be included to indicate the scope of protection in the manner described in [RFC7235], Section 2.2.

* A "scope" authentication parameter MAY be included as defined in [RFC6750], Section 3.

* An "error" parameter ([RFC6750], Section 3) SHOULD be included to indicate the reason why the request was declined, if the request included an access token but failed authorization. Parameter values are described in Section 3.1 of [RFC6750].

* An "error_description" parameter ([RFC6750], Section 3) MAY be included along with the "error" parameter to provide developers a human-readable explanation that is not meant to be displayed to end-users.

* An "algs" parameter SHOULD be included to signal to the client the JWS algorithms that are acceptable for the DPoP proof JWT. The value of the parameter is a space-delimited list of JWS "alg" (Algorithm) header values ([RFC7515], Section 4.1.1).

* Additional authentication parameters MAY be used and unknown parameters MUST be ignored by recipients

For example, in response to a protected resource request without authentication:

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

Figure 6

And in response to a protected resource request that was rejected because the confirmation of the DPoP binding in the access token failed:

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

Figure 7

7. Public Key Confirmation

It MUST be ensured that resource servers can reliably identify whether a token is bound using DPoP and learn the public key to which the token is bound.

Access tokens that are represented as JSON Web Tokens (JWT) [RFC7519] MUST contain information about the DPoP public key (in JWK format) in the member "jkt" of the "cnf" claim, as shown in Figure 8.

The value in "jkt" MUST be the base64url encoding [RFC7515] of the JWK SHA-256 Thumbprint (according to [RFC7638]) of the public key to which the access token is bound.

```
{
  "sub":"someone@example.com",
  "iss":"https://server.example.com",
  "aud":"https://resource.example.org",
  "nbf":1562262611,
  "exp":1562266216,
  "cnf":{
      "jkt":"0ZcOCORZNYy-DWpqq30jZyJGHTN0d2HglBV3uiguA4I"
  }
}
```

Figure 8: Example access token body with "cnf" claim

When access token introspection is used, the same "cnf" claim as above MUST be contained in the introspection response.

Resource servers MUST ensure that the fingerprint of the public key in the DPoP proof JWT equals the value in the "jkt" claim in the access token or introspection response.

8. Authorization Server Metadata

This document introduces the following new authorization server metadata [RFC8414] parameter to signal the JWS "alg" values the authorization server supports for DPoP proof JWTs:

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

9. Security Considerations

In DPoP, the prevention of token replay at a different endpoint (see Section 2) is achieved through the binding of the DPoP proof to a certain URI and HTTP method. DPoP does not, however, achieve the same level of protection as TLS-based methods such as OAuth Mutual TLS [RFC8705] or OAuth Token Binding [I-D.ietf-oauth-token-binding] (see also Section 9.1 and Section 9.4). TLS-based mechanisms can leverage a tight integration between the TLS layer and the application layer to achieve a very high level of message integrity and replay protection. Therefore, it is RECOMMENDED to prefer TLS-based methods over DPoP if such methods are suitable for the scenario at hand.

9.1. DPoP Proof Replay

If an adversary is able to get hold of a DPoP proof JWT, the adversary could replay that token at the same endpoint (the HTTP endpoint and method are enforced via the respective claims in the JWTs). To prevent this, servers MUST only accept DPoP proofs for a limited time window after their "iat" time, preferably only for a relatively brief period. Servers SHOULD store the "jti" value of each DPoP proof for the time window in which the respective DPoP proof JWT would be accepted and decline HTTP requests for which the "jti" value has been seen before. In order to guard against memory exhaustion attacks a server SHOULD reject DPoP proof JWTs with unnecessarily large "jti" values or store only a hash thereof.

Note: To accommodate for clock offsets, the server MAY accept DPoP proofs that carry an "iat" time in the near future (e.g., up to a few seconds in the future).

9.2. Signed JWT Swapping

Servers accepting signed DPoP proof JWTs MUST check the "typ" field in the headers of the JWTs to ensure that adversaries cannot use JWTs created for other purposes.

9.3. Signature Algorithms

Implementers MUST ensure that only asymmetric digital signature algorithms that are deemed secure can be used for signing DPoP proofs. In particular, the algorithm "none" MUST NOT be allowed.

9.4. Message Integrity

DPoP does not ensure the integrity of the payload or headers of requests. The signature of DPoP proofs only contains the HTTP URI and method, but not, for example, the message body or other request headers.

This is an intentional design decision to keep DPoP simple to use, but as described, makes DPoP potentially susceptible to replay attacks where an attacker is able to modify message contents and headers. In many setups, the message integrity and confidentiality provided by TLS is sufficient to provide a good level of protection.

Implementers that have stronger requirements on the integrity of messages are encouraged to either use TLS-based mechanisms or signed requests. TLS-based mechanisms are in particular OAuth Mutual TLS [RFC8705] and OAuth Token Binding [I-D.ietf-oauth-token-binding].

Note: While signatures on (parts of) requests are out of the scope of this specification, signatures or information to be signed can be added into DPoP proofs.

10. IANA Considerations

10.1. OAuth Access Token Type Registration

This specification requests registration of the following access token type in the "OAuth Access Token Types" registry [IANA.OAuth.Params] established by [RFC6749].

* Type name: "DPoP"

* Additional Token Endpoint Response Parameters: (none)

* HTTP Authentication Scheme(s): "DPoP"

* Change controller: IESG

* Specification document(s): [[this specification]]

10.2. HTTP Authentication Scheme Registration

This specification requests registration of the following scheme in the "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" [RFC7235][IANA.HTTP.AuthSchemes]:

* Authentication Scheme Name: "DPoP"

* Reference: [[Section 6 of this specification]]

10.3. Media Type Registration

[[Is a media type registration at [IANA.MediaTypes] necessary for "application/dpop+jwt"? There is a "+jwt" structured syntax suffix registered already at [IANA.MediaType.StructuredSuffixs] by Section 7.2 of [RFC8417], which is maybe sufficient? A fullblown registration of "application/dpop+jwt" seems like it'd be overkill. The "dpop+jwt" is used in the JWS/JWT "typ" header for explicit typing of the JWT per Section 3.11 of [RFC8725] but it is not used anywhere else (such as the "Content-Type" of HTTP messages).

Note that there does seem to be some precedence for [IANA.MediaTypes] registration with [I-D.ietf-oauth-access-token-jwt], [I-D.ietf-oauth-jwsreq], [RFC8417], and of course [RFC7519]. But precedence isn't always right.]]

10.4. JWT Confirmation Methods Registration

This specification requests registration of the following value in the IANA "JWT Confirmation Methods" registry [IANA.JWT] for JWT "cnf" member values established by [RFC7800].

* Confirmation Method Value: "jkt"

* Confirmation Method Description: JWK SHA-256 Thumbprint

* Change Controller: IESG

* Specification Document(s): [[Section 7 of this specification]]

10.5. JSON Web Token Claims Registration

This specification requests registration of the following Claims in the IANA "JSON Web Token Claims" registry [IANA.JWT] established by [RFC7519].

HTTP method:

* Claim Name: "htm"

* Claim Description: The HTTP method of the request

* Change Controller: IESG

* Specification Document(s): [[Section 4.1 of this specification]]

HTTP URI:

* Claim Name: "htu"

* Claim Description: The HTTP URI of the request (without query and fragment parts)

* Change Controller: IESG

* Specification Document(s): [[Section 4.1 of this specification]]

10.6. HTTP Message Header Field Names Registration

This document specifies the following new HTTP header fields, registration of which is requested in the "Permanent Message Header Field Names" registry [IANA.Headers] defined in [RFC3864].

* Header Field Name: "DPoP"

* Applicable protocol: HTTP

* Status: standard

* Author/change Controller: IETF

* Specification Document(s): [[this specification]]

10.7. Authorization Server Metadata Registration

This specification requests registration of the following values in the IANA "OAuth Authorization Server Metadata" registry [IANA.OAuth.Parameters] established by [RFC8414].

* Metadata Name: "dpop_signing_alg_values_supported"

* Metadata Description: JSON array containing a list of the JWS algorithms supported for DPoP proof JWTs

* Change Controller: IESG

* Specification Document(s): [[Section 8 of this specification]]

11. Normative References

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

12. Informative References

[I-D.ietf-oauth-token-binding]
Jones, M., Campbell, B., Bradley, J., and W. Denniss, "OAuth 2.0 Token Binding", Work in Progress, Internet-Draft, draft-ietf-oauth-token-binding-08, 19 October 2018, <https://tools.ietf.org/html/draft-ietf-oauth-token-binding-08>.

[I-D.ietf-oauth-access-token-jwt]
Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-access-token-jwt-07, 27 April 2020, <https://tools.ietf.org/html/draft-ietf-oauth-access-token-jwt-07>.

[I-D.ietf-oauth-jwsreq]
Sakimura, N. and J. Bradley, "The OAuth 2.0 Authorization Framework: JWT Secured Authorization Request (JAR)", Work in Progress, Internet-Draft, draft-ietf-oauth-jwsreq-21, 19 April 2020, <https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-21>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/info/rfc8705>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[IANA.MediaTypes] IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

[RFC8417] Hunt, P., Ed., Jones, M., Denniss, W., and M. Ansari, "Security Event Token (SET)", RFC 8417, DOI 10.17487/RFC8417, July 2018, <https://www.rfc-editor.org/info/rfc8417>.

[IANA.JWT] IANA, "JSON Web Token Claims", <http://www.iana.org/assignments/jwt>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[IANA.OAuth.Params] IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[IANA.MediaType.StructuredSuffixs] IANA, "Structured Syntax Suffix Registry", <https://www.iana.org/assignments/media-type-structured-suffixs>.

[IANA.Headers] IANA, "Message Headers", <https://www.iana.org/assignments/message-headers>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

[IANA.HTTP.AuthSchemes] IANA, "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry", <https://www.iana.org/assignments/http-authschemes>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

[I-D.ietf-oauth-security-topics] Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", Work in Progress, Internet-Draft, draft-ietf-oauth-security-topics-15, 5 April 2020, <https://tools.ietf.org/html/draft-ietf-oauth-security-topics-15>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

Appendix A. Acknowledgements

We would like to thank Filip Skokan, Mike Engan, Justin Richer, Michael Peck, Vladimir Dzhuvinov, Rob Otto, Dominick Baier, Jim Willeke, Annabelle Backman, Bjorn Hjelm, Steinar Noem, Aaron Parecki, Neil Madden, Paul Querna, Dick Hardt, Dave Tonge, Jared Jennings, Mark Haine and others (please let us know, if you've been mistakenly omitted) for their valuable input, feedback and general support of this work.

This document resulted from discussions at the 4th OAuth Security Workshop in Stuttgart, Germany. We thank the organizers of this workshop (Ralf Kusters, Guido Schmitz).

Appendix B. Document History

[[To be removed from the final specification]]

-01

* Editorial updates

* Attempt to more formally define the DPoP Authorization header scheme

* Define the 401/WWW-Authenticate challenge

* Added "invalid_dpop_proof" error code for DPoP errors in token request

* Fixed up and added to the IANA section

* Added "dpop_signing_alg_values_supported" authorization server metadata

* Moved the Acknowledgements into an Appendix and added a bunch of names (best effort)

-00 [[Working Group Draft]]

* Working group draft

-04

* Update OAuth MTLS reference to RFC 8705

* Use the newish RFC v3 XML and HTML format

-03

* rework the text around uniqueness requirements on the jti claim in the DPoP proof JWT

* make tokens a bit smaller by using "htm", "htu", and "jkt" rather than "http_method", "http_uri", and "jkt#S256" respectively

* more explicit recommendation to use mTLS if that is available

* added David Waite as co-author

* editorial updates

-02

* added normalization rules for URIs

* removed distinction between proof and binding

* "jwk" header again used instead of "cnf" claim in DPoP proof

* renamed "Bearer-DPoP" token type to "DPoP"

* removed ability for key rotation

* added security considerations on request integrity

* explicit advice on extending DPoP proofs to sign other parts of the HTTP messages

* only use the jkt#S256 in ATs

* iat instead of exp in DPoP proof JWTs

* updated guidance on token_type evaluation

-01

* fixed inconsistencies

* moved binding and proof messages to headers instead of parameters

* extracted and unified definition of DPoP JWTs

* improved description

-00

* first draft

Authors' Addresses

Daniel Fett
yes.com

Email: mail@danielfett.de

Brian Campbell
Ping Identity

Email: bcampbell@pingidentity.com

John Bradley
Yubico

Email: ve7jtb@ve7jtb.com

Torsten Lodderstedt
yes.com

Email: torsten@lodderstedt.net

Michael Jones
Microsoft

Email: mbj@microsoft.com
URI: https://self-issued.info/

David Waite
Ping Identity

Email: david@alkaline-solutions.com