Network Working Group
Internet-Draft
Intended status: Informational
Expires: 23 October 2024

T. Looker
MATTR
P. Bastian

21 April 2024

OAuth 2.0 Attestation-Based Client Authentication
draft-ietf-oauth-attestation-based-client-auth-02

Abstract

This specification defines a new method of client authentication for OAuth 2.0 [RFC6749] by extending the approach defined in [RFC7521]. This new method enables client deployments that are traditionally viewed as public clients to be able to authenticate with the authorization server through an attestation based authentication scheme.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://vcstuff.github.io/draft-ietf-oauth-attestation-based-client-auth/draft-ietf-oauth-attestation-based-client-auth.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-oauth-attestation-based-client-auth/.

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."

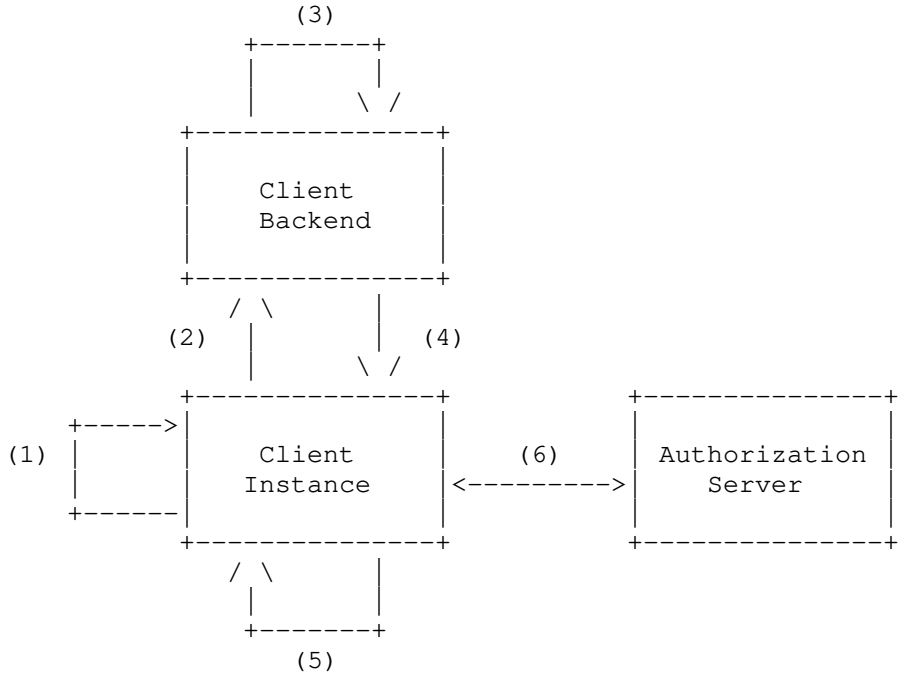
This Internet-Draft will expire on 23 October 2024.

Table of Contents

1. Introduction

[RFC7521] defines a way for a client to include an assertion in a token request to an authorization server for the purposes of client authentication. This specification uses this framework to define a new assertion type that provides a way for a client instance to authenticate itself with the authorization server through an assertion that is bound to a public key (for proof of possession). This assertion is designated with the name of Client Attestation in this draft.

The following diagram depicts the overall architecture and protocol flow.



The following steps describe this OAuth flow:

(1) The Client Instance generates a key (Client Instance Key) and optional further attestations (that are out of scope) to prove its authenticity to the Client Backend.

(2) The Client Instance sends this data to the Client Backend in request for a Client Attestation JWT.

(3) The Client Backend validates the Client Instance Key and optional further data. It generates a signed Client Attestation JWT that is cryptographically bound to the Client Instance Key generated by the Client. Therefore, the attestation is bound to this particular Client Instance.

(4) The Client Backend responds to the Client Instance by sending the Client Attestation JWT.

(5) The Client Instance generates a Proof of Possession (PoP) with the Client Instance Key.

(6) The Client Instance sends both the Client Attestation JWT and the Client Attestation PoP JWT to the authorization server, e.g. within a token request. The authorization server validates the Client Attestation and thus authenticates the Client Instance.

Note that the protocol for steps (2) and (4) and how the Client Instance authenticates to the Client Backend is out of scope of this specification. Note also that this specification can be utilized without the client having a backend server at all; in this case, each client instance will perform the functions described as being done by the backend for itself.

This specification defines the format of the Client Attestation that a Client Instance uses to authenticate in its interactions with an authorization server, which is comprised of two key parts:

1. A Client Attestation JWT - typically produced by the client backend.

2. A Client Attestation Proof of Possession (PoP) - produced by the client instance.

2. Conventions and Definitions

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.

3. Terminology

Client Attestation JWT: A JSON Web Token (JWT) generated by the client backend which is bound to a key managed by a client instance which can then be used by the instance for client authentication.

Client Attestation Proof of Possession (PoP) JWT: A Proof of Possession generated by the client instance using the key that the Client Attestation JWT is bound to.

Client Instance Key: A cryptographic, asymmetric key generated by the client instance and proven to the client backend. The public key is contained in the Client Attestation JWT and is used to sign the Client Attestation Proof of Possession.

4. Client Authentication

To perform client authentication using this scheme, the client instance uses the following parameter values and encodings.

The value of the "client_assertion_type" parameter (as defined in [RFC7521]) set to "urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation".

The value of the "client_assertion" parameter (as defined in [RFC7521]) set to a value containing two JWTs, separated by a '~' character. It MUST NOT contain more or less than precisely two JWTs separated by the '~' character. The first JWT MUST be the client attestation JWT defined in Section 4.1.1, the second JWT MUST be the client attestation PoP defined in Section 4.1.2.

The following example demonstrates client authentication using this scheme during the presentation of an authorization code grant in an access token request (with extra line breaks for display purposes only):

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-client-attestation&
client_assertion=eyJhbGciOiJSUzI1NiIsImtpZCI6IjIyIn0.
eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...]~eyJzI1NiIsImtphbGciOimtpZCI6IjIyIn0.
IjIyIn0[...omitted for brevity...].
iOiJSUzI1[...omitted for brevity...]
```

4.1. JWT Format and Processing Requirements

In order to authenticate the client using this scheme, the authorization server MUST validate BOTH the JWTs present in the "client_assertion" parameter according to the criteria below.

It is RECOMMENDED that the authorization server validate the Client Attestation JWT prior to validating the Client Attestation PoP.

4.1.1. Client Attestation JWT

The following rules apply to validating the client attestation JWT. Application of additional restrictions and policy are at the discretion of the authorization server.

1. The JWT MUST contain an "iss" (issuer) claim that contains a unique identifier for the entity that issued the JWT. In the absence of an application profile specifying otherwise, compliant applications MUST compare issuer values using the Simple String Comparison method defined in Section 6.2.1 of [RFC3986].

2. The JWT MUST contain a "sub" (subject) claim with a value corresponding to the "client_id" of the OAuth client.

3. The JWT MUST contain an "exp" (expiration time) claim that limits the time window during which the JWT can be used. The authorization server MUST reject any JWT with an expiration time that has passed, subject to allowable clock skew between systems.

4. The JWT MUST contain an "cnf" claim conforming [RFC7800] that conveys the key to be used for producing the client attestation pop for client authentication with an authorization server. The key MUST be expressed using the "jwk" representation.

5. The JWT MAY contain an "nbf" (not before) claim that identifies the time before which the token MUST NOT be accepted for processing.

6. The JWT MAY contain an "iat" (issued at) claim that identifies the time at which the JWT was issued.

7. The JWT MAY contain other claims.

8. The JWT MUST be digitally signed using an asymmetric cryptographic algorithm. The authorization server MUST reject the JWT if it is using a Message Authentication Code (MAC) based algorithm. The authorization server MUST reject JWTs with an invalid signature.

9. The authorization server MUST reject a JWT that is not valid in all other respects per "JSON Web Token (JWT)" [RFC7519].

The following example is the decoded header and payload of a JWT meeting the processing rules as defined above.

```
{
  "alg": "ES256",
  "kid": "11"
}
.
{
  "iss": "https://client.example.com",
  "sub": "https://client.example.com",
  "nbf":1300815780,
  "exp":1300819380,
  "cnf": {
    "jwk": {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  }
}
```

4.1.2. Client Attestation PoP JWT

The following rules apply to validating the Client Attestation JWT. Application of additional restrictions and policy are at the discretion of the Authorization Server.

1. The JWT MUST contain an "iss" (issuer) claim with a value corresponding to the "client_id" of the OAuth client.

2. The JWT MUST contain an "exp" (expiration time) claim that limits the time window during which the JWT can be used. The authorization server MUST reject any JWT with an expiration time that has passed, subject to allowable clock skew between systems. Note that the authorization server may reject JWTs with an "exp" claim value that is unreasonably far in the future.

3. The JWT MUST contain a "jti" (JWT ID) claim that provides a unique identifier for the token. The authorization server MAY ensure that JWTs are not replayed by maintaining the set of used "jti" values for the length of time for which the JWT would be considered valid based on the applicable "exp" instant.

4. The JWT MUST contain an "aud" (audience) claim containing a value that identifies the authorization server as an intended audience. The [RFC8414] issuer identifier URL of the authorization server MUST be used as a value for an "aud" element to identify the authorization server as the intended audience of the JWT.

5. The JWT MAY contain an "nonce" claim containing a String value that is provided by the authorization server to associate the Client Attestation PoP JWT with a particular transaction and prevent replay attacks.

6. The JWT MAY contain an "nbf" (not before) claim that identifies the time before which the token MUST NOT be accepted for processing.

7. The JWT MAY contain an "iat" (issued at) claim that identifies the time at which the JWT was issued. Note that the authorization server may reject JWTs with an "iat" claim value that is unreasonably far in the past.

8. The JWT MAY contain other claims.

9. The JWT MUST be digitally signed using an asymmetric cryptographic algorithm. The authorization server MUST reject the JWT if it is using a Message Authentication Code (MAC) based algorithm. The authorization server MUST reject JWTs with an invalid signature.

10. The public key used to verify the JWT MUST be the key located in the "cnf" claim of the corresponding client attestation JWT.

11. The authorization server MUST reject a JWT that is not valid in all other respects per "JSON Web Token (JWT)" [RFC7519].

The following example is the decoded header and payload of a JWT meeting the processing rules as defined above.

```
{
  "alg": "ES256"
}
.
{
  "iss": "https://client.example.com",
  "aud": "https://as.example.com",
  "nbf":1300815780,
  "exp":1300819380,
  "jti": "d25d00ab-552b-46fc-ae19-98f440f25064"
}
```

5. Implementation Considerations

5.1. Reuse of a Client Attestation JWT

Implementers should be aware that the design of this authentication mechanism deliberately allows for a client instance to re-use a single Client Attestation JWT in multiple interactions/requests with an authorization server, whilst producing a fresh Client Attestation PoP JWT. Client deployments should consider this when determining the validity period for issued Client Attestation JWTs as this ultimately controls how long a client instance can re-use a single Client Attestation JWT.

5.2. Refresh token binding

Authorization servers issuing a refresh token in response to a token request using the "urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation" client authentication method MUST bind the refresh token to the client instance, and NOT just the client as specified in section 6 [RFC6749]. To prove this binding, the client instance MUST authenticate itself to the authorization server when refreshing an access token using the "urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation" authentication method. The client MUST also use the same key that was present in the "cnf" claim of the client attestation that was used for client authentication when the refresh token was issued.

5.3. Rotation of Client Instance Key

This specification does not provide a mechanism to rotate the Client Instance Key in the Client Attestation JWT's "cnf" claim. If the Client Instance needs to use a new Client Instance Key for any reason, then it MUST request a new Client Attestation JWT from its Client Backend.

6. Privacy Considerations

6.1. Client Instance Tracking Across Authorization Servers

Implementers should be aware that using the same client attestation across multiple authorization servers could result in correlation of the end user using the client instance through claim values (including the public key in the cnf claim). Client deployments are therefore RECOMMENDED to use different client attestations across different authorization servers.

7. Security Considerations

The guidance provided by [RFC7519] and [RFC8725] applies.

7.1. Replay Attack Detection

The following mechanisms exist within this client authentication method in order to allow an authorization server to detect replay attacks for presented client attestation PoPs:

* The client uses "jti" (JWT ID) claims for the Client Attestation PoP JWT and the authorization server maintains a list of used (seen) "jti" values for the time of which the JWT would be considered valid based on the applicable "exp" claim. If any Client Attestation PoP JWT would be replayed, the authorization server would recognize the "jti" and respond with an authentication error.

* The authorization server provides a nonce for the particular transaction and the client uses it for the "nonce" claim in the Client Attestation PoP JWT. The authorization server validates that the nonce matches for the transaction. This approach may require an additional roundtrip in the protocol. The authorization server MUST ensure that the nonce provides sufficient entropy.

* The authorization server may expect the usage of a nonce in the Client Attestation PoP JWT, but instead of providing the nonce explicitly, the client may implicitly reuse an existing artefact, e.g. the authorization code. The authorization server MUST ensure that the nonce provides sufficient entropy.

The approach using a nonce explicitly provided by the authorization server gives stronger replay attack detection guarantees, however support by the authorization server is OPTIONAL to simplify mandatory implementation requirements. The "jti" method is mandatory and hence acts as a default fallback.

8. Appendix A IANA Considerations

8.1. Sub-Namespace Registration of urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation

This section registers the value "client-assertion-type:jwt-client-attestation" in the IANA "OAuth URI" registry established by "An IETF URN Sub-Namespace for OAuth" [RFC6755].

* URN: urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation

* Common Name: OAuth 2.0 Attested Key-Based Client Authentication

* Change Controller: IESG

* Specification Document: TBC

8.2. Registration of attest_jwt_client_auth Token Endpoint Authentication Method

This section registers the value "attest_jwt_client_auth" in the IANA "OAuth Token Endpoint Authentication Methods" registry established by OAuth 2.0 Dynamic Client Registration Protocol [RFC7591].

* Token Endpoint Authentication Method Name: "attest_jwt_client_auth"

* Change Controller: IESG

* Specification Document(s): TBC

9. References

9.1. Normative References

[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/rfc/rfc2119>.

[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/rfc/rfc3986>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <https://www.rfc-editor.org/rfc/rfc6755>.

[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/rfc/rfc7519>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/rfc/rfc7591>.

[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/rfc/rfc7800>.

[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/rfc/rfc8174>.

[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/rfc/rfc8414>.

[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/rfc/rfc8725>.

9.2. Informative References

[ARF] "The European Digital Identity Wallet Architecture and Reference Framework", n.d..

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

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <https://www.rfc-editor.org/rfc/rfc7521>.

Appendix A. Additional Examples

A.1. Wallet Instance Attestation

This non-normative example shows a client attestations used as an wallet instance attestation in the context of eIDAS 2.0 [ARF], e.g. to secure a Type-1 configuration credential. The additional claims describe the wallet's device binding und user binding capabilities and the achievable level of assurance.

```
{
        "typ": "wallet-attestation+jwt",
        "alg": "ES256",
        "kid": "1"
}
.
{
        "iss": "https://attestation-service.com",
        "sub": "https://wallet-provider.com",
        "iat": 1541493724,
        "exp": 1516247022,
        "attested_security_context" : "https://eu-trust-list.eu/asc/high",
        "cnf": {
                "jwk" : {
                        "kty": "EC",
                        "crv": "P-256",
                        "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
                        "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
                },
                "key_type" : "STRONGBOX",
                "user_authentication" : "SYSTEM_PIN"
        }
}
```

Appendix B. Document History

-02

* Add text on rotation of the confirmation key

-01

* Updated eIDAS example in appendix

* Removed text around jti claim in client attestation, refined text for its usage in the client attestation pop

* Refined text around cnf claim in client attestation

* Clarified how to bind refresh tokens to a client instance using this client authentication method

* Made it more explicit that the client authentication mechanism is general purpose making it compatible with extensions like PAR

* Updated acknowledgments

* Simplified the diagram in the introduction

* Updated references

* Added some guidance around replay attack detection

-00

* Initial draft

Acknowledgments

We would like to thank Brian Campbell, Francesco Marino, Guiseppe De Marco, Kristina Yasuda, Michael B. Jones, Takahiko Kawasaki and Torsten Lodderstedt for their valuable contributions to this specification.

Authors' Addresses

Tobias Looker
MATTR
Email: tobias.looker@mattr.global

Paul Bastian
Email: paul.bastian@bdr.de