Server Requirements (WebAuthn Level 3 and CTAP2.3)

Review Draft, October 23, 2025

This version:
http://fidoalliance.org/specs/fidoserver/fido-server-v2.3-rd-20251023.html

Previous Versions:
https://fidoalliance.org/specs/fidoserver/fido-server-v1.0-rd-20240116.html

Issue Tracking:
Github

Editors:
David Waite (Ping Identity)
John Bradley (Yubico)
Matthew Miller (Duo Security Inc.)
Tim Cappalli (Microsoft)

Former Editors:
Yuriy Ackermann (FIDO Alliance)
Adam Powers (FIDO Alliance)

Abstract

FIDO2 provides secure authentication through the use of authenticators that implement the Client-to-Authenticator Protocol (CTAP) and platforms or browsers that implement the W3C WebAuthn specifications. These authenticators are expected to communicate to servers that will validate registration and authentication requests. Many of the requirements for FIDO2 servers, such as assertion formats, attestation formats, optional extensions, and so forth, are contained in the W3C WebAuthn specification. This Server Requirements specification attempts to pull together all the requirements for servers in a single document that will be an aid to implementing a FIDO2 server, while leaving behind the details of authenticators and web browsers that do not pertain to servers.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current FIDO Alliance publications and the latest revision of this technical report can be found in the FIDO Alliance specifications index at https://fidoalliance.org/specifications/.

This document was published by the FIDO Alliance as a Review Draft Specification. This document is intended to become a FIDO Alliance Proposed Standard. If you wish to make comments regarding this document, please Contact Us. All comments are welcome.

This is a Review Draft Specification and is not intended to be a basis for any implementations as the Specification may change. Permission is hereby granted to use the Specification solely for the purpose of reviewing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this Specification for other uses must contact the FIDO Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The FIDO Alliance, Inc. and its Members and any other

contributors to the Specification are not, and shall not be held, responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS FIDO ALLIANCE SPECIFICATION IS PROVIDED “AS IS” AND WITHOUT ANY WARRANTY OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTY OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1. Introduction§

This specification provides a set of requirements and guidance for server implementers that draws heavily from the W3C [WebAuthn] specification. Servers are a critical piece of the FIDO ecosystem for making sure that implementations work together. There are many optional features of the various specifications, including different attestation formats (packed, Android, TPM, etc.), attestation modes (surrogate, full, ECDAA, etc.), cryptographic suites (RSA, ECDSA, etc.) and so on. The authenticators that typically implement these various features are typically consumer electronics devices that are memory and/or CPU-constrained, which limits their ability to implement multiple versions of these features. Therefore, it falls to servers to implement as many of these features as possible to ensure that servers are compatible with the broadest range of authenticators possible.

The WebAuthn specification is fairly simple in its concept: it provides a method for registering new authenticators with a server (`navigator.credentials.create()`) and another method for authenticating with previously registered authenticators (`navigator.credentials.get()`). During registration, an authenticator uses an attestation private key that was embedded in the authenticator during its manufacturing to create an attestation statement, thus providing a root of trust for the registration process. Registration creates a new key pair for each account that is registered and the private key of the registration is used to sign an assertion that is sent to the server to demonstrate valid authentication. The sections that follow describe the registration and attestation requirements, and the authentication and assertion requirements.

It should be noted that there is no specific protocol (REST, SOAP, carrier pigeon, quantum teleportation, etc.) required for the server (although there are requirements around having a secure communication channel). It is assumed that servers are receiving some form of the JavaScript objects that were created by the browser, the platform, or the authenticator. Note that these objects are signed over, so protocols MUST NOT alter the signed objects in ways that would cause the signature to be invalid, but otherwise, any form of transporting these objects to the server is acceptable. The requirements and guidelines laid out below do not make any requirements on how these objects are sent or received by the server.

In the case that this specification conflicts with the [WebAuthn] specification, the [WebAuthn] specification takes precedence; however, there may be clarifications or additions in this specification that supersede the [WebAuthn] specification and many of the descriptions of how to implement WebAuthn in a web browser are irrelevant to server implementers.

2. Registration and Attestations§

Servers MUST support registration. A registration request will take the form of sending a challenge to an authenticator and receiving a `PublicKeyCredential` object (or similar) in response. The `response` attribute of the `PublicKeyCredential` will contain both a serialized `clientDataJSON` attribute and a serialized `attestationObject` attribute. There is no requirement for the format of the serialization (e.g. - base64url encoding) except that when deserialized the underlying byte structure will remain the same as what was signed during attestation.

Servers MUST use random challenges for each registration request. While determining the randomness of a challenge is beyond the scope of this specification (see [FIDOSecRef] for more details), using the same challenge, monotonically increasing challenges, or other simple challenges is unacceptable and insecure and it is expected that a cryptographically secure random number generator is used for generating challenges.

2.1. Validating Attestations§

Servers MUST validate attestations. *Web Authentication* § 7.1 Registering a New Credential specifies how to validate an attestation. Requirements for the `Relying Party` are normative for servers. Note that the fields in the AttestationResponse MAY NOT match the field names or formats in the [WebAuthn] specification -- applications and servers may negotiate their own field formats and names. The names and formats described in [WebAuthn] are for convenience only.

Servers MUST validate attestation certificate chains.

Servers MUST support the validation of attestations through the FIDO Metadata Service [FIDOMetadataService].

Servers MAY have policies to allow, disallow, require additional authentication factors, or perform risk analysis for authenticators based on their metadata attributes.

2.2. Attestation Types§

Web Authentication § 6.5.3 Attestation Types defines multiple Attestation Types.

- Servers MUST support basic attestation
- Servers MUST support self-attestation
- Servers MAY support Privacy CA attestation
- Servers MAY support Elliptic Curve Direct Anonymous Attestation (ECDAA)

2.3. Attestation Formats§

The *Web Authentication* § 8 Defined Attestation Statement Formats defines multiple attestation formats, and the [WebAuthn-Registries] registry may be updated from time to time to add additional attestation formats as the ecosystem evolves.

- Servers MUST support Packed Attestation: *Web Authentication* § 8.2 Packed Attestation Statement Format
- Servers MUST support TPM Attestation: *Web Authentication* § 8.3 TPM Attestation Statement Format.
- Servers MAY support Android Key Attestation: *Web Authentication* § 8.4 Android Key Attestation Statement Format
- Servers MUST support U2F Attestation: *Web Authentication* § 8.6 FIDO U2F Attestation Statement Format
- Servers MUST support Android SafetyNet Attestation: *Web Authentication* § 8.5 Android SafetyNet Attestation Statement Format
- Servers MAY support other attestation formats as defined by [WebAuthn-Registries], which may be updated from time to time. If authenticators or servers create new attestation formats, they SHOULD be registered with the [WebAuthn-Registries] registry.

2.3.1. Packed Attestation§

Servers MUST validate a Packed attestation using the "Validation Procedure" defined in *Web Authentication* § 8.2 Packed Attestation Statement Format

2.3.2. TPM Attestation§

Servers MUST validate a TPM attestation using the "Validation Procedure" defined in *Web Authentication* § 8.3 TPM Attestation Statement Format

2.3.3. Android Key Attestation§

Servers MAY validate an Android Key attestation using the "Validation Procedure" defined in *Web Authentication* § 8.4 Android Key Attestation Statement Format. This comes in the form of a DER-encoded x.509 certificate.

2.3.4. Android SafetyNet Attestation§

Servers MUST validate an Android SafetyNet attestation using the "Validation Procedure" defined in *Web Authentication* § 8.5 Android SafetyNet Attestation Statement Format

2.3.5. U2F Attestation§

Servers MUST validate a U2F attestation using the "Validation Procedure" defined in *Web Authentication* § 8.6 FIDO U2F Attestation Statement Format

3. Authentication and Assertions§

Servers MUST support authentication.

Servers MUST use random challenges for each authentication request. While determining the randomness of a challenge is beyond the scope of this specification (see [FIDOSecRef] for more details), using the same challenge, monotonically increasing challenges, or other simple challenges is unacceptable and insecure and it is expected that a cryptographically secure random number generator is used for generating challenges.

Servers MUST validate assertion signatures.

Upon receiving an assertion response, the server MUST validate the assertion response using the procedure defined in *Web Authentication* § 7.2 Verifying an Authentication Assertion

Servers SHALL validate UP and/or UV flags.

4. Communication Channel Requirements§

If servers are implementing TLS and Token Binding is available they SHOULD implement [TokenBindingProtocol] using [TokenBindingOverHttp].

5. Extensions§

A server MUST have a mode of operation that allows it to perform registration and authentication without any extensions present. Although there is no requirement that it needs to be configured that way when deployed in production.

Servers MAY support extensions.

Servers SHOULD support *Web Authentication* § 10.1.1 FIDO AppID Extension (appid) for backward compatibility with FIDO U2F. Note that browsers, platforms, and other clients may or may not support extensions.

If a server implements a new extension, it SHOULD be registered in the [WebAuthn-Registries] registry.

6. Other§

Servers MUST observe the security requirements in [WebAuthn] Section 5.3.5.

Servers MUST implement the algorithms below marked as Required and MAY implement those marked as Recommended and Optional. Servers MAY also implement other algorithms.

Name: RS1

- Value: -65535
- Description: RSASSA-PKCS1-v1_5 w/ SHA-1
- Reference: Section 2 of [RFC8812]
- Status: Required

Name: RS256

- Value: -257
- Description: RSASSA-PKCS1-v1_5 w/ SHA-256
- Reference: Section 2 of [RFC8812]
- Status: Required

Name: RS384

- Value: -258
- Description: RSASSA-PKCS1-v1_5 w/ SHA-384
- Reference: Section 2 of [RFC8812]
- Status: Optional

Name: RS512

- Value: -259

- Description: RSASSA-PKCS1-v1_5 w/ SHA-512
- Reference: Section 2 of [RFC8812]
- Status: Optional

Name: PS256

- Value: -37
- Description: RSASSA-PSS w/ SHA-256
- Reference: [RFC8230]
- Status: Optional

Name: PS384

- Value: -38
- Description: RSASSA-PSS w/ SHA-384
- Reference: [RFC8230]
- Status: Optional

Name: PS512

- Value: -39
- Description: RSASSA-PSS w/ SHA-512
- Reference: [RFC8230]
- Status: Optional

Name: ES256

- Value: -7
- Description: ECDSA using P-256 and SHA-256
- Fully Specified By: ESP256
- Reference: [RFC9053]
- Status: Required

Name: ES384

- Value: -35
- Description: ECDSA using P-384 and SHA-384
- Fully Specified By: ESP384
- Reference: [RFC9053]
- Status: Recommended

Name: ES512

- Value: -36
- Description: ECDSA using P-521 and SHA-512
- Fully Specified By: ESP512
- Reference: [RFC9053]
- Status: Optional

Name: ESP256

- Value: -9
- Description: ECDSA using P-256 and SHA-256
- Reference: [FullySpecifiedAlgorithms]
- Status: Recommended

Name: ESP384

- Value: -51
- Description: ECDSA using P-384 and SHA-384
- Reference: [FullySpecifiedAlgorithms]
- Status: Recommended

Name: ESP512

- Value: -52
- Description: ECDSA using P-521 and SHA-512
- Reference: [FullySpecifiedAlgorithms]
- Status: Optional

Name: EdDSA

- Value: -8
- Description: EdDSA using Ed25519 curve
- Fully Specified By: Ed25519
- Reference: [RFC8037]
- Status: Required

Name: Ed25519

- Value: -19
- Description: EdDSA using Ed25519 curve
- Reference: [FullySpecifiedAlgorithms]
- Status: Recommended

Name: Ed448

- Value: -53
- Description: EdDSA using Ed448 curve
- Reference: [FullySpecifiedAlgorithms]
- Status: Recommended

Name: ES256K

- Value: -47
- Description: ECDSA using secp256k1 curve and SHA-256
- Reference: [RFC8812]
- Status: Optional

Name: ML-DSA-44

- Value: -48
- Description: ML-DSA with ML-DSA-44 parameter set

- Reference: [DilithiumAlgs]
- Status: Recommended

Name: ML-DSA-65

- Value: -49
- Description: ML-DSA with ML-DSA-65 parameter set
- Reference: [DilithiumAlgs]
- Status: Recommended

Name: ML-DSA-87

- Value: -50
- Description: ML-DSA with ML-DSA-87 parameter set
- Reference: [DilithiumAlgs]
- Status: Recommended

Algorithms `ES256`, `ES384`, `ES512`, and `EdDSA` are considered "polymorphic" algorithms, in the sense that the algorithm identifier itself is insufficient to negotiate compatibility and fully specify usage.

These algorithms were deprecated by [FullySpecifiedAlgorithms] in favor of identifiers that fully specify their usage (`ESP256`, `ESP384`, `ESP512`, and `Ed25519` respectively).

Web Authentication and FIDO have previously defined the polymorphic algorithms to have the behavior of the new fully-defined algorithms. For compatibility with future clients and authenticators, servers SHOULD support the fully-specified identifiers when the corresponding polymorphic identifier is supported. The server MUST request the polymorphic identifier for backward-compatibility when requesting the fully-specified identifier variant.

Servers MUST implement the curves below marked as Required and MAY implement those marked as Recommended and Optional. Servers MAY also implement other curves.

Name: P-256

- Value: 1
- Description: EC2 NIST P-256 also known as secp256r1
- Reference: [RFC9053]
- Status: Required

Name: P-384

- Value: 2
- Description: EC2 NIST P-384 also known as secp384r1
- Reference: [RFC9053]
- Status: Recommended

Name: P-521

- Value: 3
- Description: EC2 NIST P-521 also known as secp521r1
- Reference: [RFC9053]
- Status: Optional

Name: Ed25519

- Value: 6

- Description: Edwards-curve Digital Signature Algorithm on curve 25519
- Reference: [RFC8032]
- Status: Required

Name: Ed448

- Value: 7
- Description: Edwards-curve Digital Signature Algorithm on curve 448
- Reference: [RFC8032]
- Status: Optional

Name: secp256k1

- Value: 8
- Description: SECG secp256k1 curve
- Reference: [RFC8812]
- Status: Optional

Note that, by design, only algorithms and curves actually being used by authenticators as of the time of this writing are included in the list of Required algorithms and curves.

Servers wanting to be prepared in advance for possible future cryptographic developments ought to consider implementing the Recommended algorithms and curves in addition to the Required ones.

Servers MUST comply with the FIDO privacy principles[FIDOPrivacyPrinciples].

Index§

Terms defined by reference§

[WEBAUTHN-3] defines the following terms:

PublicKeyCredential

References§

Normative References§

[DilithiumAlgs]
M. Prorock; et al. *ML-DSA for JOSE and COSE*. April 1, 2025. URL:https://datatracker.ietf.org/doc/draft-ietf-cose-dilithium/06/

[FIDOMetadataService]
B. Jack; R. Lindemann; Y. Ackermann. *FIDO Metadata Service*. 21 May 2025. Proposed Standard. URL: https://fidoalliance.org/specs/mds/fido-metadata-service-v3.1-ps-20250521.html

[FIDOPrivacyPrinciples]
FIDO: Privacy Principles. Feb 2021. URL: https://media.fidoalliance.org/wp-content/uploads/2021/02/FIDO-Privacy-Principles.pdf

[FullySpecifiedAlgorithms]
M. B. Jones; O. Steele. *Fully-Specified Algorithms for JOSE and COSE*. May 13, 2025. URL: https://datatracker.ietf.org/doc/draft-ietf-jose-fully-specified-algorithms/13/

[RFC8032]
S. Josefsson; I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. January 2017. Informational. URL: https://www.rfc-editor.org/rfc/rfc8032

[RFC8037]
I. Liusvaara. *CFRG Elliptic Curve Diffie-Hellman (ECDH) and Signatures in JSON Object Signing and Encryption (JOSE)*. January 2017. Proposed Standard. URL:https://www.rfc-editor.org/rfc/rfc8037

[RFC8230]
M. Jones. *Using RSA Algorithms with CBOR Object Signing and Encryption (COSE) Messages* September 2017. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc8230

[RFC8812]
Michael B. Jones. *CBOR Object Signing and Encryption (COSE) and JSON Object Signing and Encryption (JOSE) Registrations for Web Authentication (WebAuthn) Algorithms*. August 2020. IETF Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc8812

[RFC9053]
J. Schaad. *CBOR Object Signing and Encryption (COSE): Initial Algorithms* August 2022. RFC. URL: https://www.rfc-editor.org/rfc/rfc9053.html

[TokenBindingOverHttp]
A. Popov; et al. *Token Binding over HTTP*. December 7, 2018. URL:https://tools.ietf.org/html/draft-ietf-tokbind-https-17

[TokenBindingProtocol]
A. Popov; et al. *The Token Binding Protocol Version 1.0* May 23, 2018. URL:https://tools.ietf.org/html/draft-ietf-tokbind-protocol-19

[WebAuthn]
Dirk Balfanz (Google); et al. *Web Authentication: An API for accessing Public Key Credentials Level 2* 8 April 2021. TR. URL: https://www.w3.org/TR/webauthn-2/

[WEBAUTHN-3]
Tim Cappalli; et al. *Web Authentication: An API for accessing Public Key Credentials - Level 3* URL: https://w3c.github.io/webauthn/

[WebAuthn-Registries]
Jeff Hodges; G. Mandyam; Michael B. Jones. *Registries for Web Authentication (WebAuthn)*. March 24, 2017. Draft. URL: https://tools.ietf.org/html/draft-hodges-webauthn-registries

Informative References§

[FIDOSecRef]
R. Lindemann; et al. *FIDO Security Reference*. 23 May 2022. Proposed Standard. URL: https://fidoalliance.org/specs/common-specs/fido-security-ref-v2.1-ps-20220523.html