Server Requirements and Transport Binding Profile

Review Draft, July 02, 2018

This version:
https://fidoalliance.org/specs/fido-v2.0-rd-20180702/fido-server-v2.0-rd-20180702.html

Previous Versions:
https://fidoalliance.org/specs/fido-v2.0-id-20180227/

Issue Tracking:
Github

Editors:
Adam Powers (FIDO Alliance)
Yuriy Ackermann (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 and Guidance 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.

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 crticial 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 electonrics 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 required 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 / platform / 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 precidence; however, there may be clarifications or additions in this specification that supercede 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 SHALL support registration. A registration request will take the form of sending a challenge to an authenticator and receiving a `CredentialCreationOptions` object (or similar) in response. The `response` attribute of the `PublicKeyCredential` will contain both a serialzed `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 SHALL 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 Attestation§

Servers SHALL validate attestation. [[!WebAuthn#registering-a-new-credential]] specifies how to validate 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 SHALL validate attestation certificate chains.

Servers MUST support the validation of attestation 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§

[[!WebAuthn#sctn-attestation-types]] defines multiple Attestation Types. A server MUST support one of the attestation formats.

- 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 [[!WebAuthn#defined-attestation-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. A server MUST support at least one attestation format.

- Servers MUST support Packed Attestation: [[!WebAuthn#packed-attestation]]
- Servers MUST support TPM Attestation: [[!WebAuthn#tpm-attestation]].
- Servers SHOULD support Android Key Attestation: [[!WebAuthn#android-key-attestation]]
- Servers MUST support U2F Attestation: [[!WebAuthn#fido-u2f-attestation]]
- Servers MUST support Android SafteyNet Attestation: [[!WebAuthn#android-safetynet-attestation]]
- 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 attesation using the "Validation Procedure" defined in [[!WebAuthn#packed-attestation]]

```
EXAMPLE 1
{
    "rawId": "sL39APyTmisrjh11vghaqNfuruLQmCfR0c1ryKtaQ81jkEhNa5u9xLTnkibvXC9YpzBLFwWEZ3k9CR_
xzm_pWYbBOtKxeZu9z2GT8b6QW4iQvRlyumCT3oENx_8401r",
    "id": "sL39APyTmisrjh11vghaqNfuruLQmCfR0c1ryKtaQ81jkEhNa5u9xLTnkibvXC9YpzBLFwWEZ3k9CR_sxz
_pWYbBOtKxeZu9z2GT8b6QW4iQvRlyumCT3oENx_8401r",
    "response": {
        "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJ1Vlg4OElnUmEwU1NyTUlSVF9xN2NSY2RmZ2ZSQnhDZ25fcG
twVUFuWEpLMnpPYjMwN3dkMU9MWFEwQXVOYU10QlIzYW1rNkhZenAtX1Z4SlRQcHdHdyIsIm9yaWdpbiI6Imh0dHBzOi8
d2ViYXV0aG4ub3JnIiwidG9rZW5CaW5kaW5nIjp7InN0YXR1cyI6Im5vdC1zdXBwb3J0ZWQifSwidHlwZSI6IndlYmF1d
huLmNyZWF0ZSJ9",
        "attestationObject": "o2NmbXRmcGFja2VkZ2F0dFN0bXSjY2FsZyZjc2lnWEgwRgIhAIsK0Wr9tmud-wa
IYoQw20UWi7DL_gDx_PNG3PB57eHLAiEAtRyd-4JI2pCVX-dDz4mbHc_AkvC3d_4qnBBa3n2I_hVjeDVjg1kCRTCCAkEw
gHooAMCAQICEBWfe8LNiRjxKGuTSPqfM-IwCgYIKoZIzj0EAwIwSTELMAkGA1UEBhMCQ04xHTAbBgNVBAoMFEZlaXRpYW
gVGVjaG5vbG9naWVzMRswGQYDVQQDDBJGZWl0aWFuIEZJRE8yIENBLTEwIBcNMTgwNDExMDAwMDAwWhgPMjAzMzA0MTAy
zU5NTlaMG8xCzAJBgNVBAYTAkNOMR0wGwYDVQQKDBRGZWl0aWFuIFRlY2hub2xvZ2llczEiMCAGA1UECwwZQXV0aGVudC
jYXRvciBBdHRlc3RhdGlvbjEdMBsGA1UEAwwURlQgQmlvUGFzcyBGSURPMiBVU0IwWTATBgcqhkjOPQIBBggqhkjOPQME
wNCAASABnVcWfvJSbAVqNIKkliXvoMKsu_oLPiP7aCQlmPlSMcfEScFM7QkRnidTP7hAUOKlOmDPeIALC8qHddvTdtdo4
JMIGGMB0GA1UdDgQWBBR6VIJCgGLYiuevhJglxK-RqTSY8jAfBgNVHSMEGDAWgBRNO9jEZxUbuxPo84TYME-daRXAgzAM
gNVHRMBAf8EAjAAMBMGCysGAQQBguUcAgEBBAQDAgUgMCEGCysGAQQBguUcAQEEBBIEEEI4MkVENzNDOEZCNEU1QTIwCg
IKoZIzj0EAwIDRwAwRAIgJEtFo76I3LfgJaLGoxLP-4btvCdKIsEFLjFIUfDosIcCIDQav04cJPILGnPVPazCqfkVtBuy
msBbx_v-ODn-JDAWQH_MIIB-zCCAaCgAwIBAgIQFZ97ws2JGPEoa5NI-p8z4TAKBggqhkjOPQQDAjBLMQswCQYDVQQGEw
DTjEdMBsGA1UECgwURmVpdGlhbiBUZWNobm9sb2dpZXMxHTAbBgNVBAMMFEZlaXRpYW4gRklETyBSb290IENBMCAXDTE4
DQxMDAwMDAwMFoYDzIwMzgwNDA5MjM1OTU5WjBJMQswCQYDVQQGEwJDTjEdMBsGA1UECgwURmVpdGlhbiBUZWNobm9sb2
pZXMxGzAZBgNVBAMMEkZlaXRpYW4gRklETzIgQ0EtMTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABI5-YAnswRZlzKD6
-lv5Qg7lW1XJRHrWzL01mc5V91n2LYXNR3_S7mA5gupuTO5mjQw8xfqIRMHVr1qB3TedY-jZjBkMB0GA1UdDgQWBBRNO9
EZxUbuxPo84TYME-daRXAgzAfBgNVHSMEGDAWgBTRoZhNgX_DuWv2B2e9UBL-kEXxVDASBgNVHRMBAf8ECDAGAQH_AgEA
A4GA1UdDwEB_wQEAwIBBjAKBggqhkjOPQQDAgNJADBGAiEA-3-j0kBHoRFQwnhWbSHMkBaY7KF_TztINFN5ymDkwmUCIC
rCkPBiMHXvYg-kSRgVsKwuVtYonRvC588qRwpLStZ7FkB3DCCAdgwggF-oAMCAQICEBWfe8LNiRjxKGuTSPqfM9YwCgYI
oZIzj0EAwIwSzELMAkGA1UEBhMCQ04xHTAbBgNVBAoMFEZlaXRpYW4gVGVjaG5vbG9naWVzMR0wGwYDVQQDDBRGZWl0aW
uIEZJRE8gUm9vdCBDQTAgFw0xODA0MDEwMDAwMDBaGA8yMDQ4MDMzMTIzNTk1OVowSzELMAkGA1UEBhMCQ04xHTAbBgNV
AoMFEZlaXRpYW4gVGVjaG5vbG9naWVzMR0wGwYDVQQDDBRGZWl0aWFuIEZJRE8gUm9vdCBDQTBZMBMGByqGSM49AgEGCC
GSM49AwEHA0IABJ3wCm47zF9RMtW-pPlkEHTVTLfSYBlsidz7zOAUiuV6k36PvtKAI_-LZ8MiC9BxQUfUrfpLY6klw344
wLq7POjQjBAMB0GA1UdDgQWBBTRoZhNgX_DuWv2B2e9UBL-kEXxVDAPBgNVHRMBAf8EBTADAQH_MA4GA1UdDwEB_wQEAw
BBjAKBggqhkjOPQQDAgNIADBFAiEAt7E9ZQYxnhfsSk6c1dSmFNnJGoU3eJiycs2DoWh7-IoCIA9iWJH8h-UOAaaPK66D
CLe6GIxdpIMv3kmd1PRpWqsaGF1dGhEYXRhWOSVaQiPHs7jIylUA129ENfK45EwWidRtVm7j9fLsim91EEAAAABQjgyRU
3M0M4RkI0RTVBMgBgsL39APyTmisrjh11vghaqNfuruLQmCfR0c1ryKtaQ81jkEhNa5u9xLTnkibvXC9YpzBLFwWEZ3k9
R_sxzm_pWYbBOtKxeZu9z2GT8b6QW4iQvRlyumCT3oENx_8401rpQECAyYgASFYIFkdweEE6mWiIAYPDoKz3881Aoa4sr
zkTm0aPKKYBvdIlggtlG32lxrang8M0tojYJ36CL1VMv2pZSzqR_NfvG88bA"
    }
};
```

2.3.2. TPM Attestation§

Servers MUST validate a TPM attesation using the "Validation Procedure" defined in [[!WebAuthn#tpm-attestation]]

```
EXAMPLE 2
{
    "rawId": "hWzdFiPbOMQ5KNBsMhs-Zeh8F0iTHrH63YKkrxJFgjQ",
    "id": "hWzdFiPbOMQ5KNBsMhs-Zeh8F0iTHrH63YKkrxJFgjQ",
    "response": {
        "clientDataJSON": "ew0KCSJ0eXBlIiA6ICJ3ZWJhdXRobi5jcmVhdGUiLA0KCSJjaGFsbGVuZ2UiIDogIr
drNkxxRVhBTUFacHFjVFlsWTJ5b3I1RGppeUlfYjFneTluRE90Q0IxeUdZbm1fNFdHNFVrMjRGQXI3QXhUT0ZmUU1laWc
UnhPVExaTnJMeEN2Vl9RIiwNCgkib3JpZ2luIiA6ICJodHRwczovL3dlYmF1dGhuLm9yZyIsDQoJInRva2VuQmluZGluZ
IgOiANCgl7DQoJCSJzdGF0dXMiDogInN1cHBvcnRlZCINCgl9DQp9",
        "attestationObject": "o2NmbXRjdHBtaGF1dGhEYXRhWQFnlWkIjx7O4yMpVANdvRDXyuORMFonUbVZu4_
Xy7IpvdRFAAAAAAiYcFjK3EuBtuEw3lDcvpYAIIVs3RYj2zjEOSjQbDIbPmXofBdIkx6x-t2CpK8SRYI0pAEDAzkBACBZ
```

```
QDF2m9Nk1e94gL1xVjNCjFW0lTy4K2atXkx-YJrdH3hrE8p1gc1dNzleRDhmERJnY5CRwM5sXDQ1rUBq4jpwvTtMC5HGc
N6-iEJAPtm9_CJzCmGhtw9hbF8bcAys94RhN9xLLUaajhWqtPrYZXCEAi0o9E2QdTIxJrcAfJgZOf33JMr0--R1BAQxpC
GRDC8ss-tfQW9ufZLWw4JUuz4Z5Jz1sbfqBYB8UUDMWoT0HgsMaPmvd7T17xGvB-pvvDf-Dt96vFGtYLEZEgho8Yu26pr
CK_BOQ-2vX9N4MIYVPXNhogMGGmKYqybhM3yhye0GdBpZBUd5iOcgME6uGJ1_IUMBAAFnYXR0U3RtdKZjdmVyYzIuMGNh
Gc5__5jc2lnWQEAcV1izWGUWIs0DEOZNQGdriNNXo6nbrGDLzEAeswCK9njYGCLmOkHVgSyafhsjCEMZkQmuPUmEOMDKc
qxup_tiXQwG4yCW9TyWoINWGayQ4vcr6Ys-l6KMPkg__d2VywhfonnTJDBfE_4BIRD60GR0qBzTarthDHQFMqRtoUtuOs
F5jedU3EQPojRA5iCNC2naCCZuMSURdlPmhlW5rAaRZVF41ZZECi5iFOM2rO0UpGuQSLUvr1MqQOsDytMf7qWZMvwT_5_
BF6GNdB2l2VzmIJBbV6g8z7dj0fRkjlCXBp8UG2LvTq5SsfugrRWXOJ8BkdMplPfl0mz6ssU_n2N4NWOCWQS2MIIEsjCC
5qgAwIBAgIQEyidpWZzRxOSMNfrAvV1fzANBgkqhkiG9w0BAQsFADBBMT8wPQYDVQQDEzZOQ1UtTlRDLUtFWUlELTE1OT
ENEI2RUFGOThEMDEwNDg2NEI2OTAzQTQ4REQwMDI2MDc3RDMwHhcNMTgwNTIwMTYyMDQ0WhcNMjgwNTIwMTYyMDQ0WjAA
IIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvQ6XK2ujM11E7x4SL34p252ncyQTd3-4r5ALQhBbFKS95gUsuE
TG-48GBQwu48i06cckm3eH20TUeJvn4-pj6i8LFOrIK14T3P3GFzbxgQLq1KVm63JWDdEXk789JgzQjHNO7DZFKWTEikt
mBUPUA88TjQcXOtrR5EXTrt1FzGzabOepFann3Ny_XtxI8lDZ3QLwPLJfmk7puGtkGNaXOsRC7GLAnoEB7UWvjiyKG6HA
vVTgxcW5OQnHFb9AHycU5QdukXrP0njdCpLCRR0Nq6VMKmVU3MaGh-DCwYEB32sPNPdDkPDWyk16ItwcmXqfSBV5ZOr8i
vcXbCWUWwIDAQABo4IB5TCCAeEwDgYDVR0PAQH_BAQDAgeAMAwGA1UdEwEB_wQCMAAwbQYDVR0gAQH_BGMwYTBfBgkrBg
EAYI3FR8wUjBQBggrBgEFBQcCAjBEHkIAVABDAFAAQQAgACAAVAByAHUAcwB0AGUAZAAgACAAUABsAGEAdABmAG8AcgBt
CAAIABJAGQAZQBuAHQAaQB0AHkwEAYDVR0lBAkwBwYFZ4EFCAMwSgYDVR0RAQH_BEAwPqQ8MDoxODAOBgVngQUCAwwFaW
6MTMwEAYFZ4EFAgIMB05QQ1Q2eHgwFAYFZ4EFAgEMC2lkOjRFNTQ0MzAwMB8GA1UdIwQYMBaAFMISqVvO-lb4wMFvsVvc
zRHs3qjMB0GA1UdDgQWBBSv4kXTSA8i3NUM0q57lrWpM8p_4TCBswYIKwYBBQUHAQEEgaYwgaMwgaAGCCsGAQUFBzAChc
TaHR0cHM6Ly9hemNzcHJvZG5jdWFpa3B1Ymxpc2guYmxvYi5jb3JlLndpbmRvd3MubmV0L25jdS1udGMta2V5aWQtMTU5
WQ0YjZlYWY5OGQwMTA0ODY0YjY5MDNhNDhkZDAwMjYwNzdkMy8zYjkxOGFlNC0wN2UxLTQwNTktOTQ5MS0wYWQyNDgxOT
4MTguY2VyMA0GCSqGSIb3DQEBCwUAA4IBAQAs-vqdkDX09fNNYqzbv3Lh0vl6RgGpPGl-MYgO8Lg1I9UKvEUaaUHm845A
S8m7r9p22RCWO6TSEPS0YUYzAsNuiKiGVna4nB9JWZaV9GDS6aMD0nJ8kNciorDsV60j0Yb592kv1VkOKlbTF7-Z10jaa
x0CqhxEIUzEBb8y9Pa8oOaQf8ORhDHZp-mbn_W8rUzXSDS0rFbWKaW4tGpVoKGRH-f9vIeXxGlxVS0wqqRm_r-h1aZInt
0OOiL_S4367gZyeLL3eUnzdd-eYySYn2XINPbVacK8ZifdsLMwiNtz5uM1jbqpEn2UoB3Hcdn0hc12jTLPWFfg7GiKQ0h
9WQXsMIIF6DCCA9CgAwIBAgITMwAAAQDiBsSROVGXhwAAAAABADANBgkqhkiG9w0BAQsFADCBjDELMAkGA1UEBhMCVVMx
zARBgNVBAgTCldhc2hpbmd0b24xEDAOBgNVBAcTB1JlZG1vbmQxHjAcBgNVBAoTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbj
2MDQGA1UEAxMtTWljcm9zb2Z0IFRQTSBSb290IENlcnRpZmljYXRlIEF1dGhvcml0eSAyMDE0MB4XDTE3MDIwMTE3NDAy
FoXDTI5MTIzMTE3NDAyNFowQTE_MD0GA1UEAxM2TkNVLU5UQy1LRVlJRC0xNTkxRDRCNkVBRjk4RDAxMDQ4NjRCNjkwM0
0OEREMDAyNjA3N0QzMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA9IwUMSiQUbrQR0NLkKR-9RB8zfHYdlmD
0XN_m8qrNHKRJ__lBOR-mwU_h3MFRZF6X3ZZwka1DtwBdzLFV8lVu33bc15stjSd6B22HRRKQ3sIns5AYQxg0eX2PtWCJ
IhxdM_jDjP2hq9Yvx-ibt1IO9UZwj83NGxXc7Gk2UvCs9lcFSp6U8zzl5fGFCKYcxIKH0qbPrzjlyVyZTKwGGSTeoMMEc
Ziq-m_xIcrehYuHg-FAVaPLLTblS1h5cu80-ruFUm5Xzl61YjVU9tAV_Y4joAsJ5QP3VPocFhr5YVsBVYBiBcQtr5JFdJ
ZWWEgYcFLdAFUk8nJERS7-5xLuQIDAQABo4IBizCCAYcwCwYDVR0PBAQDAgGGMBsGA1UdJQQUMBIGCSsGAQQBgjcVJAYF
4EFCAMwFgYDVR0gBA8wDTALBgkrBgEEAYI3FR8wEgYDVR0TAQH_BAgwBgEB_wIBADAdBgNVHQ4EFgQUwhKpW876VvjAwV
xW90DNEezeqMwHwYDVR0jBBgwFoAUeowKzi9IYhfilNGuVcFS7HF0pFYwcAYDVR0fBGkwZzBloGOgYYZfaHR0cDovL3d3
y5taWNyb3NvZnQuY29tL3BraW9wcy9jcmwvTWljcm9zb2Z0JTIwVFBNJTIwUm9vdCUyMENlcnRpZmljYXRlJTIwQXV0aC
yaXR5JTIwMjAxNC5jcmwwfQYIKwYBBQUHAQEEcTBvMG0GCCsGAQUFBzAChmFodHRwOi8vd3d3Lm1pY3Jvc29mdC5jb20v
Gtpb3BzL2NlcnRzL01pY3Jvc29mdCUyMFRQTSUyMFJvb3QlMjBDZXJ0aWZpY2F0ZSUyMEF1dGhvcml0eSUyMDIwMTQuY3
0MA0GCSqGSIb3DQEBCwUAA4ICAQAKc9z1UUBAaybIVnK8yL1N1iGJFFFw_PpkxW76hgQhUcCxNFQskfahfFzkBD05odV
1DKyk2PyOle0G86FCmZiJa14MtKNsiu66nVqk2hr8iIcu-cYEsgb446yIGd1NblQKA1C_28F2KHm8YRgcFtRSkWEMuDiV
a0HDU8aI6ZHO04Naj86nXeULJSZsA0pQwNJ04-QJP3MFQzxQ7md6D-pCx-LVA-WUdGxT1ofaO5NFxq0XjubnZwRjQazy_
93dKWp19tbBzTUKImgUKLYGcdmVWXAxUrkxHN2FbZGOYWfmE2TGQXS2Z-g4YAQo1PleyOav3HNB8ti7u5HpI3t9a73xuE
y2gFcZQ24DJuBaQe4mU5I_hPiAa-822nPPL6w8m1eegxhHf7ziRW_hW8s1cvAZZ5Jpev96zL_zRv34MsRWhKwLbu2oOCS
YYh8D8DbQZjmsxlUYR_q1cP8JKiIo6NNJ85g7sjTZgXxeanA9wZwqwJB-P98VdVslC17PmVu0RHOqRtxrht7OFT7Z10ec
0tj9ODXrv5nmBktmbgHRirRMl84wp7-PJhTXdHbxZv-OoL4HP6FxyDbHxLB7QmR4-VoEZN0vsybb1A8KEj2pkNY_tmxHH
k87euM99bB8FHrW9FNrXCGL1p6-PYtiky52a5YQZGT8Hz-ZnxobTmhjZXJ0SW5mb1ih_1RDR4AXACIAC7xZ9N_ZpqQtw7
mr_LfDRmCa78BS2erCtbrsXYwa4AHABSsnz8FacZi-wkUkfHu4xjG8MPfmwAAAAGxWkjHaED549jznwUBqeDEpT-7xBMA
gALcSGuv6a5r9BwMvQvCSXg7GdAjdWZpXv6D4DH8VYBCE8AIgALAVI0eQ_AAZjNvrhUEMK2q4wxuwIFOnHIDF0Qljhf47
ncHViQXJlYVkBNgABAAsABgRyACCd_8vzbDg65pn7mGjcbcuJ1xU4hL4oA5IsEkFYv60irgAQABAIAAAAAAABAMXab02T
73iAvXFWM0KMVbSVPLgrZq1eTH5gmt0feGsTynWBwh03OV5EOGYREmdjkJHAzmxcNAitQGriOnC9O0wLkcZxw3r6IQkA-
b38InMKYaG3D2FsXxtwDKz3hGE33EstRpqOFaq0-thlcIQCLSj0TZB1MjEmtwB8mBk5_fckyvT75HUEBDGk6gZEMLyyz6
9Bb259ktbDglS7PhnknPWxt-oFgHxRQMxahPQeCwxo-a93tPXvEa8H6m-8N_4O33q8Ua1gsRkSCGjxi7bqmvkIr8E5D7a
f03gwhhU9c2GiAwYaYpirJuEzfKHJ7QZ0GlkFR3mI5yAwTq4YnX8"
    }
};
```

2.3.3. Android SafetyNet Attestation Example

Servers MUST validate a Android Key attesation using the "Validation Procedure" defined in [[!WebAuthn#android-key-attestation]]

ISSUE 1 need an example of Android Key attestation.

2.3.4. Android SafetyNet Attestation Example

Servers MUST validate a Android SafetyNet attesation using the "Validation Procedure" defined in [[!WebAuthn#android-safetynet-attestation]]

EXAMPLE 3


```
{
    "rawId": "qCXEfJ-dEoBlWqIl0iq2p_gj13HSg7r_MA7xOcOiO8RkCrYNmQHIjV9yhZVASr87cUsflo7DNuuvGsr
lTl1ig",
    "id": "qCXEfJ-dEoBlWqIl0iq2p_gj13HSg7r_MA7xOcOiO8RkCrYNmQHIjV9yhZVASr87cUsflo7DNuuvGsnrlT
1ig",
    "response": {
        "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJEa1hCdWRCa2wzTzBlTUV5SGZBTVgxT2tRbHV4c2hjaW9WU3
dITVJMUlhtd044SXJldHg3cWJ0MWx3Y0p4d0FxWUU0SUxTZjVwd3lHMEhXSWtEekVMUT09Iiwib3JpZ2luIjoid2ViYXV
aG4ub3JnIiwiaGFzaEFsZyI6IlNIQS0yNTYifQ",
        "attestationObject": "o2hhdXRoRGF0YVjElWkIjx7O4yMpVANdvRDXyuORMFonUbVZu4_Xy7IpvdRAAAA
AAAAAAAAAAAAAAAAAAAAAAAAQKglxHyfnRKAZVqiJdIqtqf4I9dx0oO6_zAO8TnDojvEZAq2DZkByI1fcoWVQEq_O3FL
5aOwzbrrxrJ65U5dYqlAQIDJiABIVggh5OJfYRDzVGIowKqU57AnoVjjdmmjGi9zlMkjAVV9DAiWCDr0iSi0viIKNPMTI
N28gWNmkcwOr6DQx66MPff3Odm2NmbXRxYW5kcm9pZC1zYWZldHluZXRnYXR0U3RtdKJjdmVyaDEyNjg1MDIzaHJlc3Bv
nNlWRSnZXlKaGJHY2lPaUpTVXpJMU5pSXNJbmcxWXlJNld5Sk5TVWxGYVdwRFEwRXpTMmRCZDBsQ1FXZEpTVmxyV1c4MV
qQm5PRFpyZDBSUldVcExiMXBKYUhaalRrRlJSVXhDVVVGM1ZrUkZURTFCYTBkQk1WVkZRbWhOUTFaV1RYaElha0ZqUW1k
1ZrSkJiMVJHVldSMllqSmtjMXBUUWxWamJsWjZaRU5DVkZwWVNqSmhWMDVzWTNwRmJFMURUVWRCTVZWRlFYaE5ZMUl5TI
aYU1uaHNTVVZzZFdSSFZubGliVll3U1VWR01XUkhhSFpqYld3d1pWTkNTRTE2UVdWR2R6QjRUbnBGZVUxRVVYaE5la1Uv
GtST1lVWjNNSGhQUkVWNVRVUk5kMDFFUVhkTlJFSmhUVWQzZUVONlFVcENaMDVXUWtGWlZFRnNWbFJOVWsxM1JWRlpSRl
SVVVsRVFYQkVXVmQ0Y0ZwdE9YbGliV3hvVFZKWmQwWkJXVVJXVVZGSVJFRXhUbUl6Vm5Wa1IwWndZbWxDVjJGWFZqTk5V
zEzUlZGWlJGWlJVVXRFUVhCSVlqSTVibUpIVldkVFZ6VnFUVkp6ZDBkUldVUldVVkZFUkVKS2FHUklVbXhqTTFGMVdWYz
hMk50T1hCYVF6VnFZakl3ZDJkblJXbE5RVEJIUTFOeFIxTkpZak5FVVVWQ1FWRlZRVUUwU1VKRWQwRjNaMmRGUzBGdlNV
kJVVU5WYWpoM1dXOVFhWGhMWW1KV09ITm5XV2QyVFZSbVdDdGtTWE5HVkU5clowdFBiR2hVTUdrd1ltTkVSbHBMTW5KUC
FcGFNblZUVEZOV2FGbDJhWEJhVGtVelNFcFJXWFYxV1hkR2FtbDVLM2xyWm1GMFFVZFRhbEo2UmpGaU16RjFORE12TjI5
E5XcE5hRE5UTXpkaGJIZHFWV0k0UTFkcFZIaHZhWEJXVDFsM1MwdDZkVlY1YTNGRlEzUnFiR2hLTkVGclYyRkVVeXRhZU
0RmNVOWhaVGwwYmtOblpVaHNiRnBGTDA5U1oyVk5ZWGd5V0U1RGIwZzJjM0pVUlZKamEzTnFlbHBhY2tGWGVFdHpaR1oy
m5KWVRucERVamxFZUZaQlUzVkpOa3g2ZDJnNFJGTnNNa1ZQYjJ0aWMyRnVXaXNyTDBweFRXVkJRa1ptVUhkcWVYZHlZak
3Y2tWVmVUQndZV1ZXYzNWa0t6QndaV1Y0U3k4MUswVTJhM0JaUjBzMFdrc3libXR2Vmt4MVowVTFkR0ZJY2tGcU9ETlJL
UJQWW1KMlQzcFhZMFpyY0c1V1MzbHFielpMVVVGdFdEWlhTa0ZuVFVKQlFVZHFaMmRHUjAxSlNVSlJha0ZVUW1kT1ZraF
WVVZFUkVGTFFtZG5ja0puUlVaQ1VXTkVRVlJCWkVKblRsWklVa1ZGUm1wQlZXZG9TbWhrU0ZKc1l6TlJkVmxYTld0amJU
HdXa00xYW1JeU1IZGhRVmxKUzNkWlFrSlJWVWhCVVVWRldFUkNZVTFETUVkRFEzTkhRVkZWUmtKNlFVTm9hVVp2WkVoU2
wOXBPSFpqUjNSd1RHMWtkbUl5WTNaYU0wNTVUV2s1U0ZaR1RraFRWVVpJVFhrMWFtTnVVWGRMVVZsSlMzZFpRa0pSVlVc
lFVZEhTRmRvTUdSSVFUWk1lVGwyV1ROT2QweHVRbkpoVXpWdVlqSTVia3d3WkZWVk1HUktVVlZqZWsxQ01FZEJNVlZrUk
kUlYwSkNVVWM0U1hKUmRFWlNOa05WVTJ0cGEySXpZV2x0YzIweU5tTkNWRUZOUW1kT1ZraFNUVUpCWmpoRlFXcEJRVTFD
0VkQk1WVmtTWGRSV1UxQ1lVRkdTR1pEZFVaRFlWb3pXakp6VXpORGFIUkRSRzlJTm0xbWNuQk1UVU5GUjBFeFZXUkpRVk
oVFVKbmQwUkJXVXRMZDFsQ1FrRklWMlZSU1VaQmVrRkpRbWRhYm1kUmQwSkJaMGwzVFZGWlJGWlNNR1pDUTI5M1MwUkJi
zlEVTJkSmIxbG5ZVWhTTUdORWIzWk1NazU1WWtNMWQyRXlhM1ZhTWpsMlduazVTRlpHVGtoVFZVWklUWGsxYW1OdGQzZE
VVmxLUzI5YVNXaDJZMDVCVVVWTVFsRkJSR2RuUlVKQlJpOVNlazV1UXpWRWVrSlZRblJ1YURKdWRFcE1WMFZSYURsNlJX
kdXbVpRVERsUmIydHliRUZ2V0dkcVYyZE9PSEJUVWxVeGJGWkhTWEIwZWsxNFIyaDVNeTlQVWxKYVZHRTJSREpFZVRob2
rTkVja1pKTXl0c1Exa3dNVTFNTlZFMldFNUZOVkp6TW1ReFVtbGFjRTF6ZWtRMFMxRmFUa2N6YUZvd1FrWk9VUzlqYW5K
GJVeENUMGRMYTBWVk1XUnRRVmh6UmtwWVNtbFBjakpEVGxSQ1QxUjFPVVZpVEZkb1VXWmtRMFl4WW5kNmVYVXJWelppVV
OMk9GRkVialZQWkUxVEwxQnhSVEZrUldkbGRDODJSVWxTUWpjMk1VdG1XbEVyTDBSRk5reHdNMVJ5V2xSd1QwWkVSR2RZ
UN0TVowZFBjM2RvUld4cU9XTXpkbHBJUjBwdWFHcHdkRGh5YTJKcGNpOHlkVXhIWm5oc1ZsbzBTekY0TlVSU1RqQlFWVX
rT1hsUVUyMXFaeXRoYWpFcmRFaDNTVEZ0VVcxYVZsazNjWFpQTlVSbmFFOTRhRXBOUjJ4Nk5teE1hVnB0ZW05blBTSXNJ
zFKU1VWWVJFTkRRVEJUWjBGM1NVSkJaMGxPUVdWUGNFMUNlamhqWjFrMFVEVndWRWhVUVU1Q1oydHhhR3RwUnpsM01FSk
VWE5HUVVSQ1RVMVRRWGRJWjFsRVZsRlJURVY0WkVoaVJ6bHBXVmQ0VkdGWFpIVkpSa3AyWWpOUloxRXdSV2RNVTBKVFRX
EZWRTFDUlVkQk1WVkZRMmhOUzFJeWVIWlpiVVp6VlRKc2JtSnFSVlJOUWtWSFFURlZSVUY0VFV0U01uaDJXVzFHYzFVeW
HNWlha0ZsUm5jd2VFNTZRVEpOVkZWM1RVUkJkMDVFU21GR2R6QjVUVlJGZVUxVVZYZE5SRUYzVGtSS1lVMUdVWGhEZWtC
1FtZE9Wa0pCV1ZSQmJGWlVUVkkwZDBoQldVUldVVkZMUlhoV1NHSXlPVzVpUjFWblZraEtNV016VVdkVk1sWjVaRzFzYW
```

```
WWVRYaEtWR0ZxUW1RT1ZTSkJUVlJJUldSMlIqSmIjMXBUUWIwaWJsSmNZMjAxYkdSRFFTSmIxRkp2WWpUS2NHUkIhMmR
WsxM1oyZEZhVTFCTUVkRFUzRkhVMGxpTTBSUlJVSkJVVlZCUVRSSlFrUjNRWGRuWjBWTFFXOUpRa0ZSUkV0VmEzWnhTSF
2VDBwSGRXOHlia2xaWVU1V1YxaFJOVWxYYVRBeFExaGFZWG8yVkVsSVRFZHdMMnhQU2lzMk1EQXZOR2hpYmpkMmJqWkJF
Ul6UkZaNlpGRlBkSE0zUnpWd1NEQnlTbTV1VDBaVlFVczNNVWMwYm5wTFRXWklRMGRWYTNOWEwyMXZibUVyV1RKbGJVcF
NazRyWVdsamQwcExaWFJRUzFKVFNXZEJkVkJQUWpaQllXaG9PRWhpTWxoUE0yZzVVbFZyTWxRd1NFNXZkVUl5Vm5wNGIv
VliR3Q1VnpkWVZWSTFiWGMyU210TVNHNUJOVEpZUkZadlVsUlhhMDUwZVRWdlEwbE9USFpIYlc1U2Mwb3hlbTkxUVhGWl
xWlJUV012TjNONUt5OUZXV2hCVEhKV1NrVkJPRXRpZEhsWUszSTRjMjUzVlRWRE1XaFZjbmRoVnpaTlYwOUJVbUU0Y1VK
1RsRmpWMVJyWVVsbGIxbDJlUzl6UjBsS1JXMXFVakIyUmtWM1NHUndNV05UWVZkSmNqWXZOR2MzTW00M1QzRllkMlpwYm
VM1dsbFhPVGRGWm05UFUxRktaVUY2UVdkTlFrRkJSMnBuWjBWNlRVbEpRa3g2UVU5Q1owNVdTRkU0UWtGbU9FVkNRVTFE
VZsWmQwaFJXVVJXVWpCc1FrSlpkMFpCV1VsTGQxbENRbEZWU0VGM1JVZERRM05IUVZGVlJrSjNUVU5OUWtsSFFURlZaRV
zUlVJdmQxRkpUVUZaUWtGbU9FTkJVVUYzU0ZGWlJGWlNNRTlDUWxsRlJraG1RM1ZHUTJGYU0xb3ljMU16UTJoMFEwUnZT
Fp0Wm5Kd1RFMUNPRWRCTVZWa1NYZFJXVTFDWVVGR1NuWnBRakZrYmtoQ04wRmhaMkpsVjJKVFlVeGtMMk5IV1ZsMVRVUl
SME5EYzBkQlVWVkdRbmRGUWtKRGEzZEtla0ZzUW1kbmNrSm5SVVpDVVdOM1FWbFpXbUZJVWpCalJHOTJUREk1YW1NelFY
mpSM1J3VEcxa2RtSXlZM1phTTA1NVRXcEJlVUpuVGxaSVVqaEZTM3BCY0UxRFpXZEtZVUZxYUdsR2IyUklVbmRQYVRoMl
UTktjMHh1UW5KaFV6VnVZakk1Ymt3eVpIcGpha2wyV2pOT2VVMXBOV3BqYlhkM1VIZFpSRlpTTUdkQ1JHZDNUbXBCTUVK
ldtNW5VWGRDUVdkSmQwdHFRVzlDWjJkeVFtZEZSa0pSWTBOQlVsbGpZVWhTTUdOSVRUWk1lVGwzWVRKcmRWb3lPWFphZV
sNVdsaENkbU15YkRCaU0wbzFUSHBCVGtKbmEzRm9hMmxIT1hjd1FrRlJjMFpCUVU5RFFWRkZRVWhNWlVwc2RWSlVOMkoy
3pJMlozbEJXamh6YnpneGRISlZTVk5rTjA4ME5YTnJSRlZ0UVdkbE1XTnVlR2hITVZBeVkwNXRVM2hpVjNOdmFVTjBNbV
xZURsTVUwUXJVRUZxTWt4SldWSkdTRmN6TVM4MmVHOXBZekZyTkhSaVYxaHJSRU5xYVhJek4zaFVWRTV4VWtGTlVGVjVS
EpYVTJSMmRDdHViRkJ4ZDI1aU9FOWhNa2t2YldGVFNuVnJZM2hFYWs1VFpuQkVhQzlDWkRGc1drNW5aR1F2T0dOTVpITk
NeXQzZVhCMVprbzVkVmhQTVdsUmNHNW9PWHBpZFVaSmQzTkpUMDVIYkRGd00wRTRRMmQ0YTNGSkwxVkJhV2d6U21GSFQz
mpjR05rWVVOSmVtdENZVkk1ZFZsUk1WZzBhekpXWnpWQlVGSk1iM1Y2Vm5rM1lUaEpWbXMyZDNWNU5uQnRLMVEzU0ZRMF
GazRhV0pUTlVaRldteG1RVVpNVTFjNFRuZHpWbm81VTBKTE1sWnhiakZPTUZCSlRXNDFlRUUyVGxwV1l6ZHZPRE0xUkV4
lJuTm9SVmRtUXpkVVNXVXpaejA5SWwxOS5leUp1YjI1alpTSTZJbXhYYTBscWVEZFBOSGxOY0ZaQlRtUjJVa1JZZVhWUF
rMUdiMjVWWWxaYWRUUXZXSGszU1hCMlpGSkJRVUZCUVVGQlFVRkJRVUZCUVVGQlFVRkJRVkJRVkZMWjJ4
FNIbG1ibEpMUVZwV2NXbEtaRWx4ZEhGbU5FazVaSGd3YjA4MkwzcEJUemhVYmtSdmFuWkZXa0Z4TWtSYWEwSjVTVEZtWT
5WFZsRkZjUzlQTTBaTVNEVmhUM2Q2WW5KeWVISktOalZWTldSWmNXeEJVVWxFU21sQlFrbFdaMmRvTlU5S1psbFNSSHBX
jBsdmQwdHhWVFUzUVc1dlZtcHFaRzF0YWtkcE9YcHNUV3RxUVZaV09VUkJhVmREUkhJd2FWTnBNSFpwU1V0T1VFMVVTV1
PTWpoblYwNXRhMk4zVDNJMlJGRjROalpOVUdabU0wOWtiU3QxTm1WS2NVeENiREZJTWxNeWRISkJRa2hNYVc1cmJuTjVW
zFRYlM5Q1RsVldXakpLUm14eU9EQWlMQ0owYVcxbGMzUmhiWEJOY3lJNk1UVXlPRGt4TVRZek5ETTROU3dpWVhCclVHRm
hMkZuWlU1aGJXVWlPaUpqYjIwdVoyOXZaMnhsTG1GdVpISnZhV1F1WjIxeklpd2lZWEJyUkdsblpYTjBVMmhoTWpVMklo
2lTazlETTFWcmMyeHpkVlo2TVROObFQzQnVSa2s1UW5CTWIzRkNaemxyTVVZMlQyWmhVSFJDTDBkcVRUMGlMQ0pqZEhOUV
tOW1hV3hsVFdGMFkyZ2lPbVpoYkhObExDSmhjR3REWlhKMGFXWnBZMkYwWlVScFoyVnpkRk5vWVRJMU5pSTZXeUpIV0Zk
U9GaEdNM1pKYld3ekwwMW1ibTFUYlhsMVMwSndWRE5DTUdSWFlraFNVaTgwWTJkeEsyZEJQU0pkTENKaVlYTnBZMGx1ZE
WbmNtbDBlU0k2Wm1Gc2MyVXNJbUZrZG1salpTSTZJbEpGVTFSUFVrVmZWRTlmUmtGRFZFOVNXVjlTVDAwc1RFOURTMTlD
DA5VVRFOUJSRVZTSW4wLmlDRjZEMm9zOERZdURWT250M3pESkIybVNYblpqdFdKdGxfanpTRHg1TXJSQzlBMmZtRkJaNr
1a3BRWjJNaVE3b290ajlXa0hNZ3hxSWhyWDNkbGgyUE9IQXdrSVMzNHlTakxWTnNTUHByRTg0ZVpncVNGTE1FWVQwR1Iy
VZMSEFNUE44bjVSOEs2YnVET0dGM25TaTZHS3pHNTdabGw4Q1NvYjJ5aUFTOXI3c3BkQTZIMFRESC1OR3pTZGJNSUlkOC
aRDFkekZLTlFyNzdiNmxiSUFGZ1FiUlpCcm5wLWUtSDRpSDZkMjFvTjJOQVlSblI1WVVSYWNQNmtHR2oyY0Z4c3dFMjkv
Hd4djloaVlOS05vamVldThYYzRJdDdQYmhsQXVPN3l3aFFGQTgxaVBDQ0ZtMTFCOGNmVVhiV0E4bF8ydHROUEJFTUdNNj
aNlZ5UQ"
    }
};
```

2.3.5. U2F Attestation§

Servers MUST validate a U2F attesation using the "Validation Procedure" defined in [[!WebAuthn#fido-u2f-attestation]]

```
EXAMPLE 4
{
    "rawId": "Bo-VjHOkJZy8DjnCJnIc0Oxt9QAz5upMdSJxNbd-GyAo6MNIvPBb9YsUlE0ZJaaWXtWH5FQyPS6bT_e
98IirQ==",
    "id": "Bo-VjHOkJZy8DjnCJnIc0Oxt9QAz5upMdSJxNbd-GyAo6MNIvPBb9YsUlE0ZJaaWXtWH5FQyPS6bT_e698
irQ==",
    "response": {
        "attestationObject": "o2NmbXRoZmlkby11MmZnYXR0U3RtdKJjc2lnWEgwRgIhAO-683ISJhKdmUPmVbQ
uYZsp8lkD7YJcInHS3QOfbrioAiEAzgMJ499cBczBw826r1m55Jmd9mT4d1iEXYS8FbIn8MpjeDVjgVkCSDCCAkQwggEu
AMCAQICBFVivqAwCwYJKoZIhvcNAQELMC4xLDAqBgNVBAMTI1l1YmljbyBVMkYgUm9vdCBDQSBTZXJpYWwgNDU3MjAwNj
xMCAXDTE0MDgwMTAwMDAwMFoYDzIwNTAwOTA0MDAwMDAwWjAqMSgwJgYDVQQDDB9ZdWJpY28gVTJGIEVFIFNlcmlhbCA
DMyNTM0Njg4MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAESzMfdz2BRLmZXL5FhVF-F1g6pHYjaVy-haxILIAZ8sm5Rr
gRbDmbxMbLqMkPJH9pgLjGPP8XY0qerrnK9FDCaM7MDkwIgYJKwYBBAGCxAoCBBUxLjMuNi4xLjQuMS40MTQ4Mi4xLjUw
wYLKwYBBAGC5RwCAQEEBAMCBSAwCwYJKoZIhvcNAQELA4IBAQCsFtmzbrazqbdtdZSzT1n09z7byf3rKTXra0Ucq_QdJc
nFhTXRyYEynKleOMj7bdgBGhfBefRub4F226UQPrFz8kypsr66FKZdy7bAnggIDzUFB0-629qLOmeOVeAMmOrq41uxICr
whK0sunt9bXfJTD68CxZvlgV8r1_jpjHqJqQzdio2--z0z0RQliX9WvEEmqfIvHaJpmWemvXejw1ywoglF0xQ4Gq39qB5
De22zKr_cvKg1y7sJDvHw2Z4Iab_p5WdkxCMObAV3KbAQ3g7F-czkyRwoJiGOqAgau5aRUewWclryqNled5W8qiJ6m5RD
MQnYZyq-FTZgpjXaGF1dGhEYXRhWMRJlg3liA6MaHQ0Fw9kdmBbj-SuuaKGMseZXPO6gx2XY0EAAAAAAAAAAAAAAAAAAA
AAAAAAABABo-VjHOkJZy8DjnCJnIc0Oxt9QAz5upMdSJxNbd-GyAo6MNIvPBb9YsUlE0ZJaaWXtWH5FQyPS6bT_e698Ii
aUBAgMmIAEhWCA1c9AIeH5sN6x1Q-2qR7v255tkeGbWs0ECCDw35kJGBCJYIBjTUxruadjFFMnWlR5rPJr23sBJT9qexY
PCc9o8hmT",
        "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJWdTh1RHFua3dPamQ4M0tMajZTY24yQmdGTkxGYkdSN0txXI
hKSndRbm5hdHp0VVI3WElCTDdLOHVNUENJYVFtS3cxTUNWUTVhYXpOSkZrN05ha2dxQSIsImNsaWVudEV4dGVuc2lvbnM
Ont9LCJoYXNoQWxnb3JpdGhtIjoiU0hBLTI1NiIsIm9yaWdpbiI6Imh0dHBzOi8vbG9jYWxob3N0Ojg0NDMiLCJ0eXBlI
oid2ViYXV0aG4uY3JlYXRlIn0="
    }
};
```

3. Authentication and Assertions§

Servers SHALL support authentication.

Servers SHALL 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 SHALL validate assertion signatures.

Upon receiving an assertion response, the server SHALL validate the assertion response using the procedure defined in [[!WebAuthn#verifying-assertion]]

Servers SHALL validate TUP and / or other user verification.

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 must be configured that way when deployed in production.

Servers MAY support extensions.

Servers SHOULD support [[!WebAuthn#sctn-appid-extension]] for backwards 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§

Must observe the security requirements in [WebAuthn] Section 5.3.5

The `signature` is computed over the `rawData` field.

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: TBD (requested assignment -65535)
- Description: RSASSA-PKCS1-v1_5 w/ SHA-1
- Reference: Section 8.2 of[RFC8017]
- Status: Required

Name: RS256

- Value: TBD (requested assignment -257)
- Description: RSASSA-PKCS1-v1_5 w/ SHA-256
- Reference: Section 8.2 of[RFC8017]
- Status: Required

Name: RS384

- Value: TBD (requested assignment -258)
- Description: RSASSA-PKCS1-v1_5 w/ SHA-384
- Reference: Section 8.2 of[RFC8017]
- Status: Optional

Name: RS512

- Value: TBD (requested assignment -259)
- Description: RSASSA-PKCS1-v1_5 w/ SHA-512
- Reference: Section 8.2 of[RFC8017]
- 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
- Reference: [RFC8152]
- Status: Required

Name: ES384

- Value: -35
- Description: ECDSA using P-384 and SHA-384
- Reference: [RFC8152]
- Status: Recommended

Name: ES512

- Value: -36
- Description: ECDSA using P-521 and SHA-512
- Reference: [RFC8152]
- Status: Optional

Name: EdDSA

- Value: -8
- Description: EdDSA signature algorithms
- Reference: [RFC8037]
- Status: Recommended

Name: ES256K

- Value: TBD (requested assignment -43)
- Description: ECDSA using P-256K and SHA-256
- Reference: [SEC2V2]
- Status: Optional

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: [RFC8152]
- Status: Required

Name: P-384

- Value: 2
- Description: EC2 NIST P-384 also known as secp384r1
- Reference: [RFC8152]
- Status: Recommended

Name: P-521

- Value: 3
- Description: EC2 NIST P-521 also known as secp521r1
- Reference: [RFC8152]
- Status: Optional

Name: Ed25519

- Value: 6
- Description: Edwards-curve Digital Signature Algorithm on curve 25519
- Reference: [RFC8032]
- Status: Recommended

Name: Ed448

- Value: 6
- Description: Edwards-curve Digital Signature Algorithm on curve 448
- Reference: [RFC8032]
- Status: Optional

Name: P-256K

- Value: TBD - requested assignment 8
- Description: SECG secp256k1 curve
- Reference: [SEC2V2]
- 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].

7. Transport Binding Profile§

This section is non-normative

7.1. Contents

7.2. Introduction

This document contains a non-normative, proposed REST API for FIDO2 servers. While this interface is not required, it is the interface that is used for the FIDO2 conformance test tools so that servers can receive and send messages in a standard way for those messages to be validated by the conformance test tools.

As with the FIDO2 specifications, the interfaces described here are highly dependent on the [WebAuthn] specification. The nomenclature of this document follows that of WebAuthn and reuses the Interface Definition Language (IDL) for defining the messages that are sent to / from the server.

This document is broken up into three sections: registration, authentication, and common. The registration and authentication sections contain the messages relevant to those operations, and the common section includes messgages and data formats that are common to both registration and authentication.

7.3. Registration§

This section includes a brief overview of the registration messages that are exchanged between a client and the server, followed by examples of those messages, and concluding with IDL definitions of the messages. Note that registration is also referred to as "credential creation" due to the WebAuthn nomenclature.

7.3.1. Registration Overview§

The registration flow takes part in two steps for a total of four messages. The first step is that a client retrieves "Credential Creation Options", which involves the client sending a `ServerPublicKeyCredentialCreationOptionsRequest` to the server and the server responding with a `ServerPublicKeyCredentialCreationOptionsResponse`. These options are intended to be used with WebAuthn's navigator.credentials.create(), especially the challenge which necessarily is generated by the server for the sake of Man in the Middle (MITM) protection. Upon completion of `navigator.credentials.create()` the dictionary that is created from that call is sent back to the server as the `ServerPublicKeyCredential` with `response` field set to `ServerAuthenticatorAttestationResponse`. Note that the `ServerAuthenticatorAttestationResponse` extends the generic `ServerAuthenticatorResponse`, which is described in the Common section below. The server will validate challenges, origins, signatures and the rest of the `ServerAuthenticatorAttestationResponse` according to the algorithm described in section 7.1 of the [Webauthn] specs, and will respond with the appropriate `ServerResponse` message.

7.3.2. Examples§

7.3.2.1. Example: Credential Creation Options§

Request:

- **URL:** /attestation/options
- **Method:** `POST`
- **URL Params:** None
- **Body:** `application/json` formatted `ServerPublicKeyCredentialCreationOptionsRequest`

```
{
    "username": "johndoe@example.com",
    "displayName": "John Doe",
    "authenticatorSelection": {
        "residentKey": false,
        "authenticatorAttachment": "cross-platform",
        "userVerification": "preferred"
    },
    "attestation": "direct"
}
```

Success Response:

- **HTTP Status Code:** `200 OK`
- **Body:** `application/json` formatted `ServerPublicKeyCredentialCreationOptionsResponse`

```
{
    "status": "ok",
    "errorMessage": "",
    "rp": {
        "name": "Example Corporation"
    },
    "user": {
        "id": "S3932ee31vKEC0JtJMIQ",
        "name": "johndoe@example.com",
        "displayName": "John Doe"
    },

    "challenge": "uhUjPNlZfvn7onwuhNdsLPkkE5Fv-lUN",
    "pubKeyCredParams": [
        {
            "type": "public-key",
            "alg": -7
        }
    ],
    "timeout": 10000,
    "excludeCredentials": [
        {
            "type": "public-key",
            "id": "opQf1WmYAa5aupUKJIQp"
        }
    ],
    "authenticatorSelection": {
        "residentKey": false,
        "authenticatorAttachment": "cross-platform",
        "userVerification": "preferred"
    },
    "attestation": "direct"
}
```


Error Response:

- **HTTP Status Code:** `4xx or 5xx`
- **Body:** `application/json` formatted `ServerResponse`


```
{
    "status": "failed",
    "errorMessage": "Missing challenge field!"
}
```


Sample JavaScript:

```
fetch('/attestation/options', {
    method  : 'POST',
    credentials : 'same-origin',
    headers : {
        'Content-Type' : 'application/json'
    },
    body: JSON.stringify({
        "username": "johndoe@example.com",
        "displayName": "John Doe",
        "authenticatorSelection": {
            "residentKey": false,
            "authenticatorAttachment": "cross-platform",
            "userVerification": "preferred"
        },
        "attestation": "direct"
    })
}).then(function (response) {
    return response.json();
}).then(function (json) {
    console.log(json);
}).catch(function (err) {
    console.log({ 'status': 'failed', 'error': err });
})
```

7.3.2.2. Example: Authenticator Attestation Response

Request:

- **URL:** /attestation/result
- **Method:** `POST`
- **URL Params:** None
- **Body:** `application/json` formatted `ServerPublicKeyCredential` with `response` field set to `ServerAuthenticatorAttestationResponse`

```
    {
        "id": "LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKqmDvr
uha6ywA",
        "rawId": "LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKqm
Dvruha6ywA",
        "response": {
            "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJOeHlab3B3VktiRmw3RW5uTWFlXzVGbmlyN1FKN1FXcDF
VRlVLakZIbGZrIiwiY2xpZW50RXh0ZW5zaW9ucyI6e30sImhhc2hBbGdvcml0aG0iOiJTSEEtMjU2Iiwib3JpZ2luIjoia
R0cDovL2xvY2FsaG9zdDozMDAwIiwidHlwZSI6IndlYmF1dGhuLmNyZWF0ZSJ9",
            "attestationObject": "o2NmbXRoZmlkby11MmZnYXR0U3RtdKJjc2lnWEcwRQIgVzzvX3Nyp_g9j9f
2B-tPWy6puW01aZHI8RXjwqfDjtQCIQDLsdniGPO9iKr7tdgVV-FnBYhvzlZLG3u28rVt10YXfGN4NWOBWQJOMIICSjCCA
KgAwIBAgIEVxb3wDANBgkqhkiG9w0BAQsFADAuMSwwKgYDVQQDEyNZdWJpY28gVTJGIFJvb3QgQ0EgU2VyaWFsIDQ1NzIw
DYzMTAgFw0xNDA4MDEwMDAwMDBaGA8yMDUwMDkwNDAwMDAwMFowLDEqMCgGA1UEAwwhWXViaWNvIFUyRiBFRSBTZXJpYWw
MjUwNTY5MjI2MTc2MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEZNkcVNbZV43TsGB4TEY21UijmDqvNSfO6y3G4ytnn
86ehjFK28-FdSGy9MSZ-Ur3BVZb4iGVsptk5NrQ3QYqM7MDkwIgYJKwYBBAGCxAoCBBUxLjMuNi4xLjQuMS40MTQ4Mi4xI
UwEwYLKwYBBAGC5RwCAQEEBAMCBSAwDQYJKoZIhvcNAQELBQADggEBAHibGMqbpNt2IOL4i4z96VEmbSoid9Xj--m2jJqp
RpqSOp1TO8L3lmEA22uf4uj_eZLUXYEw6EbLm11TUo3Ge-odpMPoODzBj9aTKC8oDFPfwWj6l1O3ZHTSma1XVyPqG4A57S
3YAjfrPbgj404xJns0mqx5wkpxKlnoBKqo1rqSUmonencd4xanO_PHEfxU0iZif615Xk9E4bcANPCfz-OLfeKXiT-1msix
zz8XGvl2OTMJ_Sh9G9vhE-HjAcovcHfumcdoQh_WM445Za6Pyn9BZQV3FCqMviRR809sIATfU5lu86wu_5UGIGI7MFDEYe
GSqzpzh6mlcn8QSIZoYXV0aERhdGFYxEmWDeWIDoxodDQXD2R2YFuP5K65ooYyx5lc87qDHZdjQQAAAAAAAAAAAAAAAAA
AAAAAAAAEAsV2gIUlPIHzZnNIlQdz5zvbKtpFz_WY-8ZfxOgTyy7f3Ffbolyp3fUtSQo5LfoUgBaBaXqK0wqqYO-u6FrrI
pQECAyYgASFYIPr9-YH8DuBsOnaI3KJa0a39hyxh9LDtHErNvfQSyxQsIlgg4rAuQQ5uy4VXGFbkiAt0uwgJJodp-Dymkc
crGsLtkI"
        },
        "type": "public-key"
    }
```


Success Response:

- **HTTP Status Code:** `200 OK`
- **Body:** `application/json` formatted `ServerResponse`

```
{
    "status": "ok",
    "errorMessage": ""
}
```

Error Response:

- **HTTP Status Code:** `4xx or 5xx`
- **Body:** `application/json` formatted `ServerResponse`

```
{
    "status": "failed",
    "errorMessage": "Can not validate response signature!"
}
```

Sample Call:

```
    fetch('/attestation/result', {
        method  : 'POST',
        credentials : 'same-origin',
        headers : {
            'Content-Type' : 'application/json'
        },
        body: JSON.stringify({
            "id": "LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKo
mDvruha6ywA",
            "rawId": "LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6it
MKqmDvruha6ywA",
            "response": {
                "clientDataJSON": "eyJjaGFsbGVuZ2UiOiJOeHlab3B3VktiRmw3RW5uTWFlXzVGbmlyN1FKN1
FXcDFVRlVLakZIbGZrIiwiY2xpZW50RXh0ZW5zaW9ucyI6e30sImhhc2hBbGdvcml0aG0iOiJTSEEtMjU2Iiwib3JpZ2lu
joiaHR0cDovL2xvY2FsaG9zdDozMDAwIiwidHlwZSI6IndlYmF1dGhuLmNyZWF0ZSJ9",
                "attestationObject": "o2NmbXRoZmlkby11MmZnYXR0U3RtdKJjc2lnWEcwRQIgVzzvX3Nyp_g
9j9f2B-tPWy6puW01aZHI8RXjwqfDjtQCIQDLsdniGPO9iKr7tdgVV-FnBYhvzlZLG3u28rVt10YXfGN4NWOBWQJOMIICS
CCATKgAwIBAgIEVxb3wDANBgkqhkiG9w0BAQsFADAuMSwwKgYDVQQDEyNZdWJpY28gVTJGIFJvb3QgQ0EgU2VyaWFsIDQ1
zIwMDYzMTAgFw0xNDA4MDEwMDAwMDBaGA8yMDUwMDkwNDAwMDAwMFowLDEqMCgGA1UEAwwhWXViaWNvIFUyRiBFRSBTZX1
YWwgMjUwNTY5MjI2MTc2MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEZNkcVNbZV43TsGB4TEY21UijmDqvNSfO6y3G4y
nnjP86ehjFK28-FdSGy9MSZ-Ur3BVZb4iGVsptk5NrQ3QYqM7MDkwIgYJKwYBBAGCxAoCBBUxLjMuNi4xLjQuMS40MTQ4M
4xLjUwEwYLKwYBBAGC5RwCAQEEBAMCBSAwDQYJKoZIhvcNAQELBQADggEBAHibGMqbpNt2IOL4i4z96VEmbSoid9Xj--mZ
Jqg6RpqSOp1TO8L3lmEA22uf4uj_eZLUXYEw6EbLm11TUo3Ge-odpMPoODzBj9aTKC8oDFPfwWj6l1O3ZHTSma1XVyPqG4
579f3YAjfrPbgj404xJns0mqx5wkpxKlnoBKqo1rqSUmonencd4xanO_PHEfxU0iZif615Xk9E4bcANPCfz-OLfeKXiT-I
sixwzz8XGvl2OTMJ_Sh9G9vhE-HjAcovcHfumcdoQh_WM445Za6Pyn9BZQV3FCqMviRR809sIATfU5lu86wu_5UGIGI7MI
EYeVGSqzpzh6mlcn8QSIZoYXV0aERhdGFYxEmWDeWIDoxodDQXD2R2YFuP5K65ooYyx5lc87qDHZdjQQAAAAAAAAAAAAAA
AAAAAAAAAAAAEAsV2gIUlPIHzZnNIlQdz5zvbKtpFz_WY-8ZfxOgTyy7f3FfbolyP3fUtSQo5LfoUgBaBaXqK0wqqYO-u0
rrLApQECAyYgASFYIPr9-YH8DuBsOnaI3KJa0a39hyxh9LDtHErNvfQSyxQsIlgg4rAuQQ5uy4VXGFbkiAt0uwgJJodp-D
mkoBcrGsLtkI"
            },
            "type": "public-key"
        })
    }).then(function (response) {
        return response.json();
    }).then(function (json) {
        console.log(json);
    }).catch(function (err) {
        console.log({ 'status': 'failed', 'error': err });
    })
```

7.3.3. Registration Primary IDL

7.3.3.1. ServerPublicKeyCredentialCreationOptionsRequest

```
    dictionary ServerPublicKeyCredentialCreationOptionsRequest {
        required DOMString              username;
        required DOMString              displayName;
        AuthenticatorSelectionCriteria  authenticatorSelection;
        AttestationConveyancePreference attestation = "none";
    };
```

- **required** `username` - A human-readable name for the entity. For example, "alexm", "alex.p.mueller@example.com" or "+14255551234".
- **required** `displayName` - A human-friendly name for the user account, intended only for display. For example, "Alex P. Müller" or " ".

- `authenticatorSelection` - a dictionary containing AuthenticatorSelectionCriteria described in WebAuthn specification
- `attestation` - can be set to "none", "indirect", "direct". More in WebAuthn specification. Default set to none

7.3.3.2. ServerPublicKeyCredentialCreationOptionsResponse

```
dictionary ServerPublicKeyCredentialCreationOptionsResponse : ServerResponse {
    required PublicKeyCredentialRpEntity         rp;
    required ServerPublicKeyCredentialUserEntity user;

    required DOMString                              challenge;
    required sequence<PublicKeyCredentialParameters>  pubKeyCredParams;

    unsigned long                                  timeout;
    sequence<ServerPublicKeyCredentialDescriptor> excludeCredentials = [];
    AuthenticatorSelectionCriteria                authenticatorSelection;
    AttestationConveyancePreference               attestation = "none";
    AuthenticationExtensionsClientInputs          extensions;
};
```

- required `rp` - a dictionary defined as PublicKeyCredentialRpEntity described in WebAuthn specification
- required `user` - a dictionary defined as **ServerPublicKeyCredentialUserEntity**, described in this document
- required `challenge` - a random base64url encoded challenge, that is minumum 16 bytes long, and maximum 64 bytes long
- required `pubKeyCredParams` - sequence of PublicKeyCredentialParameters described in WebAuthn specification
- `timeout` - timeout(ms)
- `excludeCredentials` - a sequence of **ServerPublicKeyCredentialDescriptor** described in this document
- `authenticatorSelection` - a dictionary set AuthenticatorSelectionCriteria described in WebAuthn specification
- `attestation` - can be set to "none", "indirect", "direct". More in WebAuthn specification. Default set to none
- `extensions` - a dictionary set to AuthenticationExtensionsClientInputs described in WebAuthn specs
- Extends **ServerResponse** described in this document

7.3.3.3. ServerAuthenticatorAttestationResponse

Generally the same as AuthenticatorAttestationResponse from WebAuthn, but uses `base64url` encoding for fields that were of type `BufferSource`.

```
dictionary ServerAuthenticatorAttestationResponse : ServerAuthenticatorResponse {
    required DOMString      clientDataJSON;
    required DOMString      attestationObject;
};
```

- required `clientDataJSON` - base64url encoded clientDataJSON buffer
- required `attestationObject` - base64url encoded attestationObject buffer

7.3.4. Registration Supporting IDL

7.3.4.1. ServerPublicKeyCredential

Generally the same as PublicKeyCredential from WebAuthn, but uses `base64url` formatting for fields that are defined as `BufferSource` in WebAuthn.

```
dictionary ServerPublicKeyCredential : Credential {
    required DOMString                    rawId;
    required ServerAuthenticatorResponse  response;
    AuthenticationExtensionsClientOutputs getClientExtensionResults;
};
```

- required `id` - This attribute is inherited from Credential, though ServerPublicKeyCredential overrides it with base64url encoding of the authenticator credId
- required `rawId` - same as `id`
- required `response` - a dictionary defined as **ServerAuthenticatorAttestationResponse** or by **ServerAuthenticatorAssertionResponse**, described in this document
- required `type` - This attribute is inherited from Credential, though ServerPublicKeyCredential overrides it with **"public-key"**
- `getClientExtensionResults` - a map containing extension identifier, which contain client extension output entries produced by the extension's client extension processing.
- Extends Credential described in Credential Management API specification

7.3.4.2. ServerPublicKeyCredentialUserEntity

Generally the same as the PublicKeyCredentialUserEntity from WebAuthn, but uses `base64url` formatting instead of `BufferSource` for `id`.

```
    dictionary ServerPublicKeyCredentialUserEntity : PublicKeyCredentialEntity {
        required DOMString   id;
        required DOMString   displayName;
    };
```

- required `id` - base64url encoded id buffer
- required `displayName` - A human-friendly name for the user account, intended only for display. For example, "Alex P. Müller" or " ". Corresponding to ServerPublicKeyCredentialCreationOptionsRequest.displayName
- Extends PublicKeyCredentialEntity described in WebAuthn specification

7.3.4.3. ServerPublicKeyCredentialDescriptor

Generally the same as PublicKeyCredentialDescriptor from WebAuthn, but uses `base64url` formatting instead of `BufferSource` for `id`.

```
    dictionary ServerPublicKeyCredentialDescriptor {
        required PublicKeyCredentialType      type;
        required DOMString                    id;
        sequence<AuthenticatorTransport>      transports;
    };
```

- required `type` - a dictionary defined as PublicKeyCredentialType described in WebAuthn specification
- required `id` - contains base64url encoded credential ID of the public key credential that the caller is referring to.

- transports - a sequence of AuthenticatorTransport described in WebAuthn specification

7.4. Authentication§

This section starts with an overview of the messages exchanged with the server for authentication, then proceeds to show examples of those messages, and concludes with the specific IDL definitions of those messages. Note that "authentication" is sometimes referred to as "getting credentials", a "credential request", or "getting an authentication assertion" due to the terminology used in WebAuthn.

7.4.1. Authentication Overview§

Similar to the communication flow described for Registration, the Authentication flow requires four messages to be exchanged with the server. The first pair of messages are a request from the client to the server in the format of `ServerPublicKeyCredentialGetOptionsRequest`and the server returns a corresponding `ServerPublicKeyCredentialGetOptionsResponse` to the client. This `ServerPublicKeyCredentialGetOptionsResponse` is intended to be used as the parameters to the WebAuthn `navigator.credentials.get()` call. The results of `navigator.credentials.get()` are formatted by the client in to a `ServerPublicKeyCredential` with `response` field set to `ServerAuthenticatorAssertionResponse` and sent to the server. The server validates the assertion according the section 7.2 of the [WebAuthn] specification, and returns the corresponding `ServerResponse`.

7.4.2. Authentication Examples§

7.4.2.1. Authentication Example: Credential Get Options§

Request:

- **URL:** /attestation/options
- **Method:** `POST`
- **URL Params:** None
- **Body:** `application/json` encoded `ServerPublicKeyCredentialGetOptionsRequest`

```
{
    "username": "johndoe@example.com",
    "userVerification": "required"
}
```

Success Response:

- **HTTP Status Code:** `200 OK`
- **Body:** `applicaiton/json` encoded `ServerPublicKeyCredentialGetOptionsResponse`

```
{
    "status": "ok",
    "errorMessage": "",
    "challenge": "6283u0svT-YIF3pSolzkQHStwkJCaLKx",
    "timeout": 20000,
    "rpId": "https://example.com",
    "allowCredentials": [
        {
            "id": "m7xl_TkTcCe0WcXI2M-4ro9vJAuwcj4m",
            "type": "public-key"
        }
    ],
    "userVerification": "required"
}
```

Error Response:

- **HTTP Status Code:** `4xx or 5xx`
- **Body:** `applicaiton/json` encoded `ServerResponse`

```
    {
        "status": "failed",
        "errorMessage": "User does not exists!"
    }
```

Sample Call:

```
    fetch('/attestation/options', {
        method  : 'POST',
        credentials : 'same-origin',
        headers : {
            'Content-Type' : 'application/json'
        },
        body: JSON.stringify({
            "username": "johndoe@example.com",
            "userVerification": "required"
        })
    }).then(function (response) {
        return response.json();
    }).then(function (json) {
        console.log(json);
    }).catch(function (err) {
        console.log({ 'status': 'failed', 'error': err });
    })
```

7.4.2.2. Authentication Example: Authenticator Assertion Response

Request:

- **URL:** /assertion/result
- **Method:** `POST`
- **URL Params:** None
- **Body:** `application/json` encoded `ServerPublicKeyCredential` with `response` field set to `ServerAuthenticatorAssertionResponse`

```
    {
        "id":"LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKqmDvru
ha6ywA",
        "rawId":"LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKqmD
vruha6ywA",
        "response":{
            "authenticatorData":"SZYN5YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2MBAAAAAA",
            "signature":"MEYCIQCv7EqsBRtf2E4o_BjzZfBwNpP8fLjd5y6TUOLWt5l9DQIhANiYig9newAJZYTzC
1i5lwP-YQk9uXFnnDaHnr2yCKXL",
            "userHandle":"",
            "clientDataJSON":"eyJjaGFsbGVuZ2UiOiJ4ZGowQ0JmWDY5MnFzQVRweTBrTmM4NTMzSmR2ZExVcHFZ
UDh3RFRYX1pFIiwiY2xpZW50RXh0ZW5zaW9ucyI6e30sImhhc2hBbGdvcml0aG0iOiJTSEEtMjU2Iiwib3JpZ2luIjoiaH
0cDovL2xvY2FsaG9zdDozMDAwIiwidHlwZSI6IndlYmF1dGhuLmdldCJ9"
        },
        "type":"public-key"
    }
```


Success Response:

- **HTTP status code:** `200 OK`
- **Body:** `application/json` encoded `ServerResponse`


```
{
    "status": "ok",
    "errorMessage": ""
}
```


Error Response:

- **HTTP status code:** `4xx or 5xx`
- **Body:** `application/json` encoded `ServerResponse`


```
    {
        "status": "failed",
        "errorMessage": "Can not validate response signature!"
    }
```


Sample Call:

```
    fetch('/assertion/result', {
        method  : 'POST',
        credentials : 'same-origin',
        headers : {
            'Content-Type' : 'application/json'
        },
        body: JSON.stringify({
            "id":"LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itMKqr
Dvruha6ywA",
            "rawId":"LFdoCFJTyB82ZzSJUHc-c72yraRc_1mPvGX8ToE8su39xX26Jcqd31LUkKOS36FIAWgWl6itM
KqmDvruha6ywA",
            "response":{
                "authenticatorData":"SZYN5YgOjGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2MBAAAAAA",
                "signature":"MEYCIQCv7EqsBRtf2E4o_BjzZfBwNpP8fLjd5y6TUOLWt5l9DQIhANiYig9newAJ
YTzG1i5lwP-YQk9uXFnnDaHnr2yCKXL",
                "userHandle":"",
                "clientDataJSON":"eyJjaGFsbGVuZ2UiOiJ4ZGowQ0JmWDY5MnFzQVRweTBrTmM4NTMzSmR2ZEx
VcHFZUDh3RFRYX1pFIiwiY2xpZW50RXh0ZW5zaW9ucyI6e30sImhhc2hBbGdvcml0aG0iOiJTSEEtMjU2Iiwib3JpZ2lu
oiaHR0cDovL2xvY2FsaG9zdDozMDAwIiwidHlwZSI6IndlYmF1dGhuLmdldCJ9"
            },
            "type":"public-key"
        })
    }).then(function (response) {
        return response.json();
    }).then(function (json) {
        console.log(json);
    }).catch(function (err) {
        console.log({ 'status': 'failed', 'error': err });
    })
```

7.4.3. Authentication IDL§

7.4.3.1. ServerPublicKeyCredentialGetOptionsRequest§

```
    dictionary ServerPublicKeyCredentialGetOptionsRequest {
        required DOMString              username;
        UserVerificationRequirement     userVerification = "preferred";
    };
```

- **required** `username` - A human-readable name for the entity. For example, "alexm", "alex.p.mueller@example.com" or "+14255551234".
- `userVerification` - can be set to "required", "preferred", "discouraged". More inWebAuthn specification. Default set to "preferred"

7.4.3.2. ServerPublicKeyCredentialGetOptionsResponse§

```
    dictionary ServerPublicKeyCredentialGetOptionsResponse : ServerResponse {
        required DOMString                  challenge;
        unsigned long                       timeout;
        USVString                           rpId;
        sequence<ServerPublicKeyCredentialDescriptor> allowCredentials = [];
        UserVerificationRequirement         userVerification = "preferred";
        AuthenticationExtensionsClientInputs extensions;
    };
```

- required `challenge` - a random base64url encoded challenge, that is minumum 16 bytes long, and maximum 64 bytes long
- `timeout` - timeout(ms)
- `rpId` - This optional member specifies the relying party identifier claimed by the caller. If omitted, its value will be the CredentialsContainer object's relevant settings object's origin's effective domain.
- `excludeCredentials` - a sequence of **ServerPublicKeyCredentialDescriptor** described in this document
- `userVerification` - can be set to "required", "preferred", "discouraged". More in WebAuthn specification. Default set to "preferred". Corresponds to **ServerPublicKeyCredentialGetOptionsRequest.userVerification**
- `extensions` - a dictionary set to AuthenticationExtensionsClientInputs described in WebAuthn specs
- Extends **ServerResponse** described in this document

7.4.3.3. ServerAuthenticatorAssertionResponse§

```
dictionary ServerAuthenticatorAssertionResponse : ServerAuthenticatorResponse {
    required DOMString      clientDataJSON;
    required DOMString      authenticatorData;
    required DOMString      signature;
    required DOMString      userHandle;
};
```

- required `clientDataJSON` - base64url encoded clientDataJSON buffer
- required `authenticatorData` - base64url encoded authenticatorData buffer
- required `signature` - base64url encoded signature buffer
- required `userHandle` - base64url encoded userHandle buffer. Corresponding to registered user **ServerPublicKeyCredentialUserEntity.id**

7.5. Common§

7.5.1. Common IDL§

7.5.1.1. ServerResponse§

```
    dictionary ServerResponse {
        required Status     status;
        required DOMString  errorMessage = "";
    }
```

- required `status` - Describing the status of the response. Can be set to either **"ok"** or **"failed"**.
- required `errorMessage` - If `status` is set to **"failed"** this field MUST NOT be empty

Index§

Terms defined by reference§

[credential-management-1] defines the following terms:

CredentialCreationOptions

References§

Normative References§

[CREDENTIAL-MANAGEMENT-1]
Mike West. Credential Management Level 1. 4 August 2017. WD. URL:https://www.w3.org/TR/credential-management-1/

[FIDOMetadataService]
R. Lindemann; B. Hill; D. Baghdasaryan. FIDO Metadata Service v1.0. Implementation Draft. URL: https://fidoalliance.org/specs/fido-v2.0-rd-20180702/fido-metadata-service-v2.0-rd-20180702.html

[FIDOPrivacyPrinciples]
FIDO: Privacy Principles. Feb 2014. URL: https://fidoalliance.org/wp-content/uploads/2014/12/FIDO_Alliance_Whitepaper_Privacy_Principles.pdf

[RFC8017]
K. Moriarty, Ed.; et al. PKCS #1: RSA Cryptography Specifications Version 2.2. November 2016. Informational. URL: https://tools.ietf.org/html/rfc8017

[RFC8032]
S. Josefsson; I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). January 2017. Informational. URL: https://tools.ietf.org/html/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://tools.ietf.org/html/rfc8037

[RFC8152]
J. Schaad. CBOR Object Signing and Encryption (COSE). July 2017. Proposed Standard. URL: https://tools.ietf.org/html/rfc8152

[RFC8230]
M. Jones. Using RSA Algorithms with CBOR Object Signing and Encryption (COSE) Messages September 2017. Proposed Standard. URL: https://tools.ietf.org/html/rfc8230

[SEC2V2]
SEC2: Recommended Elliptic Curve Domain Parameters, Version 2.0 URL: http://www.secg.org/sec2-v2.pdf

[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; et al. Web Authentication: An API for accessing Public Key Credentials Level 1. March 2018. CR. URL: https://www.w3.org/TR/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; D. Baghdasaryan; B. Hill. FIDO Security Reference. Implementation Draft. URL: https://fidoalliance.org/specs/fido-v2.0-rd-20180702/fido-security-ref-v2.0-rd-20180702.html

Issues Index§

ISSUE 1 need an example of Android Key attestation.↵