### FIDO Authenticator Security Requirements version DV 1.2.0 - Level 1 (>=L1)

<table>
<thead>
<tr>
<th>Section</th>
<th>SR No.</th>
<th>UAF/U2F/FIDO2,L</th>
<th>Description</th>
<th>Calibration</th>
<th>Vendor Instructions</th>
<th>Vendor Response</th>
<th>L1 Test Procedure</th>
<th>Security Secretariat Response and Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.1</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The vendor SHALL document an explicit Authenticator boundary. The Authenticator’s boundary SHALL include any hardware that performs or software that implements functionality used to fulfill FIDO Authenticator Security Requirements, or FIDO Relevant user verification, key generation, secure transaction confirmation display, or signature generation. If the Authenticator includes a software component, the boundary SHALL contain the processor that executes this software. If Transaction Confirmation Display is supported and the Metadata Statement related to this Authenticator claims Transaction Confirmation Display support with tDisplay including the flag TRANSACTION_CONFIRMATION_DISPLAY_PRIVILEGEDSOFTWARE (0x0002), then the Transaction Confirmation Display MAY be implemented outside of an AROE - even when the Authenticator aims for a certification at L2 and higher. However, in such case the vendor SHALL document where and how Transaction Confirmation Display is implemented. The Authenticator boundary as defined by FIDO is comprised of the hardware and software where the Authenticator runs. The Authenticator Application by definition, is always inside the authenticator boundary. The vendor MUST describe the operational environment for the Authenticator Application, including any specific hardware or operating system requirements to completely define this boundary. The Authenticator always comprises hardware and software and the vendor SHALL describe the boundary. An Authenticator typically belongs to one of the 4 categories: 1. Authenticator Application running on some HLOS without an effective 2. Authenticator Application running on some HLOS with an effective 3. Authenticator Application running on some HLOS without an effective 4. Authenticator Application running on some HLOS with an effective</td>
<td>No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, the Authenticator vendor SHALL declare and describe to which of the above mentioned categories the Authenticator Application belongs. At L1, the vendor SHALL also describe what portions of functionality the Authenticator uses from any underlying operating environment that belongs to the Authenticator but that is not included in the Authenticator Application.</td>
<td>This Authenticator belongs to Category 3 - as 1,2, but having the Secret Authenticator Security Parameters protected by an AROE. This Authenticator - Name of the authenticator: VoiceAuth - Hardware Type &amp; Version: Microphone, SecureElement with TRNG, BLE chip with an external button to activate the BLE, USB port for firmware update. Underlying Software Platform/OS: XYZ firmware Transaction Display is NOT implemented. Please refer to the logical representation of Authenticator boundary on the Device sheet.</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.2</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The vendor SHALL document all FIDO Relevant security and cryptographic functions implemented within the Authenticator, both those on the &quot;Allowed Cryptography List&quot; (FIDOAllowedCryopts) and those not on this list. Some algorithms may only be allowed for certain Security Certification Levels. For example, not all cryptographic algorithms that are acceptable for L1 may be acceptable for L3.</td>
<td>No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, the vendor SHALL mark the FIDO Relevant security and cryptographic functions implemented in the Authenticator but implemented outside the Authenticator Application (i.e. in the underlying OS or HW)</td>
<td>AES-GCM128 for key protection &amp; authentication, SHA-256 for Hash Algorithm, Hardware TRNG for Random Number (Fips140 2 certified), ECDSA on P-256 for Attestation Key Pair AES128 for symmetric encryption HMAC-SHA256 for data authentication</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.3</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The vendor SHALL document where Authenticator User Private Keys (Uauth.priv) are stored, the structure of all KeyIDs/CredentialIDs and Key Handles used by the Authenticator, and explain how these private keys are related to the KeyIDs/CredentialIDs and Key Handles used by the Authenticator.</td>
<td>No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, the private keys, KeyIDs/CredentialIDs etc. that are generated outside the Authenticator Application SHALL be documented, but their internal structure does not need to be explained in detail.</td>
<td>User Private keys: stored in the RawKeyHandle Private key’s relationship with keyhandle: UHash = SHA256 of (User Verification Template(UV) Rawkeyhandle = AES128 (SHA256(KeyID) + Uauth.priv) + KeyID KeyHandle = AES128-GCM(RawkeyHandle + Counter + UHash + SHA256(AppID)) )</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.4</td>
<td>UAIFIDO2; &gt;=L1</td>
<td>The vendor SHALL document an Authenticator as a first-factor Authenticator or a second-factor Authenticator: [UAFAuthnrCommands], Section 6.3.4 and FIDO2Glossary entries &quot;Authenticator, 1stF / First Factor&quot; and &quot;Authenticator, 2ndF / Second Factor&quot;. No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. This authenticator is a First-factor Authenticator</td>
<td>(A0) The Security Secretariat SHALL verify the requirement during Interoperability Testing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.5</td>
<td>UAIFIDO2; &gt;=L1</td>
<td>If the Authenticator is a second-factor Authenticator, then the Authenticator SHALL NOT store user names (UAIFIDO2) inside a Raw Key Handle [UAFAuthnrCommands], Section 5.1. A cryptographically wrapped Raw Key Handle is called Key Handle. No calibration required</td>
<td>Is this requirement applicable to the Authenticator? If No, then describe why. If Yes, Provide the Security Secretariat with a description of how the requirement above is met. N/A because it is a first-factor authenticator.</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.6</td>
<td>UAIFIDO2; &gt;=L1</td>
<td>Supporting Transaction Confirmation is OPTIONAL for Authenticators. If the Authenticator supports Transaction Confirmation Display, then it SHALL hash the Transaction Content using an Allowed Hashing Cryptographic Function [UAFAuthnrCommands], Section 6.3.4, [WebAuthn] Section 10.2 and 10.3. No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. This device supports Transaction Confirmation Display and the content of every transaction is hashed using SHA256.</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.7</td>
<td>UAIFIDO2; &gt;=L1</td>
<td>If the Authenticator uses the RHAccessToken method of binding keys to apps, then when responding to a &quot;Register&quot;, &quot;Sign&quot;, or &quot;Deregister&quot; command which includes the ApplyID/RP ID, the Authenticator SHALL use an Allowed Hashing or Data Authentication Cryptographic Function to mix the AIA provided RHAccessToken and ApplyID/RP ID. If the Authenticator uses an alternative method of binding keys to apps, the vendor SHALL describe why this method provides equivalent security. Equivalent security means, (1) it prevents other apps (not originating from the same RP) from using the key and (2) in the case of bound Authenticators, it prevents other FIDO Clients of triggering the use of that key; and (3) it may rely on the underlying HLOS platform to work as expected. No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. This authenticator implementation uses SHA256 to verify the ApplyID thereby preventing the use of a key that is not linked to the correct relying party.</td>
<td>(A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticator Definition and Derived Authenticator Requirements</td>
<td>1.9</td>
<td>UAIFIDO2; &gt;=L1</td>
<td>Supporting Transaction Confirmation is OPTIONAL for Authenticators. If the Authenticator supports Transaction Confirmation Display, then it SHALL display the transaction content supplied in the &quot;Sign&quot; command. [UAFAuthnrCommands], Section 6.3.4, [FIDO2Glossary], and [WebAuthn] Sections 10.2 and 10.3. No calibration required</td>
<td>Provide the Security Secretariat with a rationale of how the requirement above is met. This Authenticator supports Transaction Confirmation Display with metadata flag TRANSACTION_CONFIRMATION_DISPLAY_PRIVILEGED_SOFTWARE. The transaction content displayed is the content supplied in the &quot;sign&quot; command.</td>
<td>(A0) The Security Secretariat SHALL verify the requirement during Interoperability Testing.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Authenticator Definition and Derived Authenticator Requirements | 1.10 | L1AF + U2F + FIDO2; >=L1 | Authenticators SHALL validate data input to the Authenticator to defend against buffer overruns, stack overflows, integer under/overflow or other such invalid input-based attack vectors.  

Note  
At L2, L3 and L3+ the entire AROE is likely to be within the authenticator boundary and thus part of the Authenticator.  
Examples of inputs directly related to the FIDO authenticator are FIDO protocol messages and FIDO authenticator configuration inputs.  
Examples of inputs to the AROE that are not directly related to FIDO are calls to configure the AROE itself or get status from the AROE itself. If the AROE can load and run an application like a signed ELF file, that signed ELF file is an input to the authenticator and the code for verifying and loading the ELF file are subject to this requirement. This is because a malicious ELF file could allow an attacker to compromise the AROE kernel and thus compromise FIDO code running on the AROE.  
At L2, L3 and L3+ the inputs to the Authenticator are primarily inputs that come from the less-secure or non-secure world outside the AROE. These are typically calls that come from the High-Level or Rich OS. Inputs between modules and subsystems within the AROE are not considered inputs for this requirement. Data read by the AROE from unsecured storage is also considered an input to the AROE. | Provide the Security Secretariat with a rationale of how the requirement above is met. | This authenticator implements input validation (eg. Type-length checks, etc.) to defend against input based attacks. | [A1] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |

| Key Management and Authenticator Security Parameters | 2.1.1 | L1AF + U2F + FIDO2; >=L1 | The vendor SHALL document all Authenticator Security Parameters (ASPs). Data parameters used by or stored within the Authenticator which are FIDO Relevant are called Authenticator Security Parameter. These SHALL, at minimum, include all FIDO user verification reference data, FIDO biometric data, Key Handle Access Tokens, User Verification Tokens (see [UAFAuthnrCommands], Section 5.3 and [FIDOGlossary]), signature or registration operation counters, FIDO Relevant cryptographic keys, and FIDO relevant Allowed Random Number Generator state data. Biometric data is defined as raw captures off the sensor, stored templates, candidate match templates, and any intermediate forms of biometric data. Biometric data not used with FIDO is excluded. | Provide the Security secretariat with a rationale of how the requirement above is met. | Please refer to “ASP’s Table” for the detailed documentation of all ASPs. | [A1] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |

| Key Management and Authenticator Security Parameters | 2.1.2 | L1AF + U2F + FIDO2; >=L1 | For each Authenticator Security Parameter, the vendor SHALL document the protections that are implemented for this parameter in order to support the FIDO Authenticator Security Goals or FIDO Authenticator Security Requirements, the location where this parameter is stored, how the parameter is protected in each storage location, how and when the parameter is input or output from the Authenticator, in what form the parameter is input or output, and when (if ever) the parameter is destroyed. Those Authenticator Security Parameters whose confidentiality MUST be protected in order to support the FIDO Security Goals or FIDO Authenticator Security Requirements SHALL be documented as “Secret Authenticator Security Parameters”, those SHALL, at minimum, include any of the following that are FIDO Relevant: secret and private keys, Allowed Random Number Generators’ state data, FIDO user verification reference data, and FIDO biometric data. | Provide the Security secretariat with a rationale of how the requirement above is met. | Please refer to “ASP’s Table” for the detailed documentation of all ASPs. | [A1] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
### Key Management and Authenticator Security Parameters

#### 2.1.3 UAF + U2F + FIDO2; >=L1

<table>
<thead>
<tr>
<th>Document</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>For each Authenticator Security Parameter that is a cryptographic key that is generated, used, or stored within the Authenticator, the vendor SHALL document how this key is generated, whether the key is unique to a particular Authenticator or shared between multiple Authenticators, and the key’s claimed cryptographic strength. This claimed cryptographic strength SHALL NOT be larger than the maximal allowed claimed cryptographic strength for the underlying algorithm, as specified in the “Allowed Cryptography List” [FIDOAllowedCrypto]. If the key is used with an algorithm not listed on the “Allowed Cryptography List” [FIDOAllowedCrypto], then the claimed cryptographic strength for this key SHALL be zero. <strong>Note</strong> This requirement interacts with requirement 5.4 as the cryptographic strength of a key might get degraded - depending on potential side channel attacks - slightly each time the key is used.</td>
</tr>
</tbody>
</table>

#### 2.1.4 UAF + U2F + FIDO2; >=L1

<table>
<thead>
<tr>
<th>Document</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.4</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The vendor SHALL document the Authenticator’s Overall Claimed Cryptographic Strength; the Overall Authenticator Claimed Cryptographic Strength SHALL be less than or equal to the claimed cryptographic strength of all the Authenticator Security Parameters that are cryptographic keys. <strong>Note</strong> The security strength is a number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system. It is specified in bits and it is often a value like 80, 112, 128, 192, 256.</td>
</tr>
</tbody>
</table>

#### 2.1.5 UAF + U2F + FIDO2; >=L1

<table>
<thead>
<tr>
<th>Document</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.5</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>All Authenticator Security Parameters within the Authenticator SHALL be protected against modification and substitution.</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.6</td>
<td>L1AF + U2F + FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.7</td>
<td>L1AF + U2F + FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.8</td>
<td>L1AF + U2F + FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.9</td>
<td>L1AF + U2F + FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.10</td>
<td>L1AF + U2F + FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.11</td>
<td>LAF ; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.12</td>
<td>LAF + U2F; FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.13</td>
<td>LAF + U2F; FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.16</td>
<td>LAF + U2F; FIDO2; &gt;=L1</td>
</tr>
<tr>
<td>Key Management and Authenticator Security Parameters</td>
<td>2.1.18</td>
<td>LAF + U2F; FIDO2; &gt;=L1</td>
</tr>
</tbody>
</table>
| Key Management  | 2.1.19 | UAF + U2F + FIDO2; >=L1 | Any wrapped FIDO biometric data and FIDO user verification reference data that is output from the Authenticator SHALL only be able to be unwrapped by the Authenticator that produced this data.  
Note: Cryptographic Collision would be an exception. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. | In this authenticator, the user verification data is not stored, rather it is hashed and the hash is wrapped using AES-GCM and the key is stored in the Secure Element which can only be accessed and used by the authenticator that produced it. | {A1} The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
<p>| Key Management  | 2.1.20 | UAF + U2F + FIDO2; &gt;=L1 | Any wrapped Authenticator User Private Key (Unauth.priv) that is output from the Authenticator SHALL only be able to be unwrapped by the Authenticator that produced this data. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. | In this authenticator, the User Private Key is wrapped using AES-GCM and the key is stored in the Secure Element which can only be accessed and used by the authenticator that produced it. | {A1} The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
| Random Number Generation  | 2.2.1 | UAF + U2F + FIDO2; &gt;=L1 | An Allowed Random Number Generator or Allowed Key Derivation Function SHALL be used for all key generation resulting in an Authenticator Security Parameter and for any random input for FIDO Relevant signature generation. | | | | |
| N/A  | 2.2.2 | UAF + U2F + FIDO2; &gt;=L1 | The security strength (see the relevant Allowed Deterministic Random Number Generator specification document cited in the &quot;Allowed Cryptography List&quot; [FIDOAllowedCrypto]) of any Authenticator's Allowed Deterministic Random Number Generator SHALL be at least as large as the largest claimed cryptographic strength of any key generated or used. If the Authenticator generates a key with an Allowed Key Derivation Function, or uses a key with parameters generated by an Allowed Key Derivation Function (see the &quot;Allowed Cryptography List&quot; [FIDOAllowedCrypto]), then the security level of the Allowed Key Derivation Function SHALL be at least as large as the claimed cryptographic level of that key generated or used. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. | N/A because This authenticator utilizes a TRNG | {A1} The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.3 Random Number Generation</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>If the Authenticator adds Authenticator generated nonces and the nonces are produced randomly, then an Allowed Random Number Generator SHALL be used for nonce generation. Authenticators with unrestricted keys (i.e. Metadata Statement isKeyRestricted: false) do not exclusively control the to-be-signed message and hence have no need to generate a nonce. No calibration required. Provide the Security Secretariat with a rationale of how the requirement above is met. The authenticator doesn’t produce nonces. (A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
</tr>
<tr>
<td>2.3.1 Signature and Registration</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The vendor SHALL document whether the Authenticator supports Signature Counters and if they are supported, the vendor SHALL document whether one Signature Counter per authentication key is implemented or one (global) Signature Counter for all authentication keys (i.e. at least one counter covering multiple keys). No calibration required. Provide the Security Secretariat with a rationale of how the requirement above is met. LH: At LH, Authenticators not running in an Allowed Restricted Operating Environment (AROE) [FIDORestrictedOperatingEnv], SHALL support signature counter(s). Provide the Security Secretariat with a rationale of how the requirement above is met. We support one (global) Signature Counter for all authentication keys. (A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
</tr>
<tr>
<td>2.3.2 Signature and Registration</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>Authenticator SHALL ensure that the signature counter value contained in FIDO signature assertions related to one specific authentication key either 1- is (a) greater than “0” and always has been greater than “0” for any previously generated FIDO signature assertion related to the same authentication key and is (b) greater than the signature counter value contained in any previously generated FIDO signature assertion related to the same authentication key, or 2- is set to “0” indicating that the signature counter is not supported any longer (e.g. in the case of a counter error). No calibration required. Is this requirement applicable to the Authenticator? If no, then describe why. If yes, provide the Security Secretariat with a rationale of how the requirement above is met. If one signature counter per authentication key is implemented (recommended option), it SHALL be incremented by 1 per signature operation. If a global signature counter is implemented, it SHOULD be incremented by a positive random number per signature operation (see [UAFAuthnrCommands] [Section A Security Guidelines, entry SignCounter]). [U2FimplCons], [Section 2.6] and [UAFAuthnrCommands] [Section 6.3.4]. The requirement prevents remote attacks. The user has to confirm an action by pressing a button or providing some other gesture. No calibration required. Provide the Security Secretariat with a rationale of how the requirement above is met. Our authenticator supports a Global Signature Counter which is greater than ‘0’ and is incremented by a positive random number per signature operation. (A1) The Security Secretariat SHALL review the provided rationale to verify the requirement is met.</td>
</tr>
<tr>
<td>3.1 Authenticator’s Test for User Presence and User Verification</td>
<td>UAF + U2F + FIDO2; &gt;=L1</td>
<td>The Authenticator shall provide a mechanism to establish if the user authorizes a given action. (For a U2F, this is the “Test for User Presence”. Generically, the term “User Verification” may also refer to this “Test for User Presence”.). No calibration required. Provide the Security Secretariat with a rationale of how the requirement above is met. This Authenticator uses Biometric Voice recognition to verify the user. (A0) The Security Secretariat SHALL verify the requirement during Interoperability Testing.</td>
</tr>
</tbody>
</table>
| Authenticator's Test for User Presence and User Verification | L1AF + U2F + FIDO2; >=L1 | The Authenticator is not marked as a Silent Authenticator [FIDO glossary], the Authenticator SHALL NOT perform any authentication relevant operation without first establishing a user has requested the operation by verifying the user ([UAFAuthnrCommands], [section 6.2.4, 6.3.4]). An Authenticator without any keys for the specific user MAY allow the enrollment of new biometric reference data for that user without any additional user verification (bootstrapping user binding). Authentication relevant operations are:  
- Generating User Authentication Keys.  
- Producing signatures using such keys.  
- Adding any additional user verification methods.  
- Adding or changing user verification reference data sets (e.g. passwords or biometric templates). All such operations, with the exception of "Producing signatures using such keys" SHALL always require a fresh user verification (see requirement 3.4). With fresh user verification we mean a user verification that is performed at the time the respective operation to be approved by the user is triggered (and not before it). | 3.2 | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. | The authenticator enforces user verification for all authentication relevant operations [A1] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
| Authenticator's Test for User Presence and User Verification | L1AF + U2F + FIDO2; >=L1 | This requirement relates to "UserVerificationCaching" as specified in [UAFAuthnrCommands]. If not declared otherwise in the Metadata Statement: Once the Authenticator’s user verification / user presence check is successful, the user SHALL be deemed “verified” for no more than 10 seconds, or until the next operation which requires user verification, whichever comes first. Any provided User Verification Token SHALL NOT be valid after this time period. ([UAFAuthnrCommands], [Appendix A Security Guidelines] If declared otherwise in the Metadata Statement:  
1. The authenticator SHALL truthfully declare support of this user verification caching in the related Metadata Statement [FIDO Metadata Statement] [entry isFreshUserVerificationRequired=Yes].  
2. Once the Authenticator’s user verification / user presence check is successful, the user SHALL be deemed “verified” for no longer than the “maximum user verification caching time” as provided by the caller.  
3. If the caller has not specified a “maximum user verification caching time”, then the Authenticator SHALL NOT cache the user verification event. Any provided User Verification Token SHALL NOT be valid after this time period. Multiple authentication operations might be performed in this time. The authenticator MAY limit the number of acceptable authentications in this time.  
4. The authenticator SHALL add the “maximum user verification caching time” related to the specific Uauth key to the attestation statement.  
5. When performing a TransactionConfirmation operation, the | 3.4 | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. | A fresh user verification is required each time a user intends to do an operation. This authenticator does not support caching. [B1] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
<p>| Authenticator's Test for User Presence and User Verification | 3.8 | UAF + U2F + FIDO | | UAF + U2F + FIDO |
| --- | --- | --- | --- |
| The Authenticator SHALL protect against injection or replay of FIDO user verification data (e.g. user presence status, PIN, or biometric data). | L1: At L1, the Authenticator Application SHALL follow best security practices specific to the underlying operating environment for protecting against injection or replay of FIDO user verification data. This especially means that the Authenticator Application SHALL NOT provide the Security Secretariat with a rationale of how the requirement above is met. | Provide the Security Secretariat with a rationale of how the requirement above is met. | The user verification data is SHA256 hashed and AES-encrypted. The encryption keys are stored in the Secure element and therefore depend on the SE for protection against injection/replay. | [A1] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
| Authenticators implementing user verification methods other than user presence check (FIDO specifications), SHALL rate-limit user verification attempts in order to prevent brute-force attacks. ([FIDO2MetaStatement], sections 3.1, 3.2, 3.3 and [UAFAuthnrCommands], Appendix A Security Guidelines, entry &quot;Matcher&quot;). The overarching requirement is based on an upper limit for the probability of a successful brute-force attack. The upper limits specified in &quot;calibration&quot; below. For the purposes of this requirement, a brute-force attack is defined as follows: The attacker tries all possible input combinations (e.g. passwords, PINs, patterns, biometrics...) in order to pass the user verification. In the case of biometric user verification, the attacker brings a potentially unlimited number of &quot;friends&quot; that can try whether their biometric characteristic is accepted (as false accept). In all cases the number of trials per time is limited by the verification speed of the authenticator and the integrity of the authenticator is not violated (e.g. no decapping of chips, no malware,...) - since there are other requirements dealing with such attacks. <strong>Note</strong> - The rate limiting requirement applies to all user verification methods (other than user presence check). - Implementing a more strict rate limiting method is allowed. - We recommend: -1. Allowing up to 3 failed user verification attempts without any penalty and then imposing a delay of at least 30 seconds before the 4th attempt. | L1: At L1, the time dependent probability of a successful brute-force attack on the authenticator SHALL be $P(t) \leq \min(170/10000, (24*t+16)/10000)$, with $t$ being the time in days. For a 4 digit PIN it means up to 170 non-biometric user verification attempts without any penalty and then imposing a delay of at least 30 seconds before the 4th attempt. | The authenticator implements user presence check, allowing up to 3 failed user verification attempts without any penalty and then imposing a delay of at least 30 seconds before the 4th attempt, increasing exponentially with each successive attempt (e.g., 1 minute before the 5th one, 2 minutes before the 6th one). After the 10th failed attempt, the device is reset to default mode and all resident keys are deleted. | [A0] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Privacy | 4.1 | FIDO2; &gt;=L1 | An Authenticator SHALL NOT have any Correlation Handle that is visible across multiple Relying Parties. If the authenticator uses a shared attestation key (e.g. Full Basic Attestation), the minimum number of Authenticators sharing this key MUST be at least 100,000. | No calibration required | The Security Secretariat SHALL review the provided rationale to verify the requirement is met. The implementation of our keyhandle is output from the authenticator in encrypted form and hence does not reveal any correlation handle that is visible across multiple Relying Parties. We use shared attestation keys that are shared across more than 100,000 authenticators. [AOI] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |
| Privacy | 4.2 | FIDO2; &gt;=L1 | An Authenticator SHALL NOT provide information to one Relying Party that can be used to uniquely identify that Authenticator instance to a different Relying Party. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, in addition to the rationale provided by the vendor, this requirement MUST be demonstrated to the Test Proctor during Interoperability Testing. Documentation is not required. [AOI] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Privacy | 4.3 | FIDO2; &gt;=L1 | An external party with two (AAID, KeyID) / (AAGUID, CredentialID) tuples produced using the Authenticator SHALL NOT be able to establish that they were produced using the same Authenticator. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. External party with two (AAID, KeyID) tuples produced using the Authenticator CANNOT establish that they were produced using the same Authenticator. The reason is because the keyhandle is encrypted &amp; hence not readable and the AAID is shared by 100,000 other devices of the same category. [AOI] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Privacy | 4.4 | FIDO2; &gt;=L1 | The Authenticator’s response to a “Deregister” command SHALL NOT reveal whether the provided KeyID was registered. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, in addition to the rationale provided by the vendor, this requirement MUST be demonstrated to the Test Proctor during Interoperability Testing. Documentation is not required. [AOI] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Attestation | 6.2 | FIDO2; &gt;=L1 | Each Authenticator being declared as the same model (i.e. having the same AAID, AAGUID or having at least one common attestationCertificateKeyIdentifier in the MetadataStatement), SHALL fulfill at least the security characteristics stated for that Authenticator model. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, in addition to the rationale provided by the vendor, this requirement MUST be demonstrated to the Test Proctor during Interoperability Testing. Documentation is not required. All authenticators of this model fulfill these security characteristics which are declared for the model. This has been demonstrated during the interoperability test. [AOI] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Attestation | 6.3 | FIDO2; &gt;=L1 | The Authenticator SHALL accurately describe itself in its provided metadata. The vendor SHALL provide all mandatory MetadataStatement fields see [FIDO_MetadataRequirements]. | No calibration required | Provide the Security Secretariat with a rationale of how the requirement above is met. At L1, in addition to the rationale provided by the vendor, this requirement MUST be demonstrated to the Test Proctor during Interoperability Testing. Documentation is not required. The Authenticator accurately describes itself in the metadata. This has been demonstrated during the interoperability test. [AOI] The Security Secretariat SHALL verify the requirement during Interoperability Testing. |
| Self-Tests and Firmware updates | 6.2 | FIDO2; &gt;=L1 | If the Authenticator mediates the update of its software, then the Authenticator SHALL use an Allowed Data Authentication or Signature Cryptographic Function, as required by the standard referenced in the “Allowed Cryptography List” [FIDOAllowedCrypto], to verify that the software being loaded has not been tampered with. If the loaded software does not pass, then the Authenticator SHALL NOT update the software. | No calibration required | (AOI) The Security Secretariat SHALL verify the requirement during Interoperability Testing. The authenticator update files are SHA256-HMAC'd to protect them from tampering. The authenticator verifies the integrity of the updates software. [AOI] The Security Secretariat SHALL review the provided rationale to verify the requirement is met. |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Step</th>
<th>Notes</th>
<th>Requirement Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td></td>
<td></td>
<td>If Authenticator Security Parameters which are cryptographic keys are generated during manufacturing, then these keys SHALL be generated as required by the standard referenced in the &quot;Allowed Cryptography List&quot; ([FIDOAllowedCrypto]) for that algorithm using an Allowed Random Number Generator. If Authenticator Security Parameters which are cryptographic keys are generated during manufacturing, then these keys SHALL be generated as required by the standard referenced in the &quot;Allowed Cryptography List&quot; ([FIDOAllowedCrypto]) for that algorithm using an Allowed Random Number Generator. Is this requirement applicable to the Authenticator? If No, then describe why. If Yes, provide the Security Secretariat with a rationale of how the requirement above is met. The key which is generated during manufacturing is the device attestation key &amp; it is generated according to the standard referenced in the “Allowed cryptography list” (FIPS 140-2).</td>
</tr>
<tr>
<td>9.4</td>
<td></td>
<td></td>
<td>A revision control system SHALL be implemented for the Authenticator and all of its components, and for all associated Authenticator documentation. This revision control system SHALL, at minimum, track changes to all software or hardware specifications, implementation files, and all tool chains used in the production of the final Authenticator. A revision control system SHALL be implemented for the Authenticator and all of its components, and for all associated Authenticator documentation. This revision control system SHALL, at minimum, track changes to all software or hardware specifications, implementation files, and all tool chains used in the production of the final Authenticator. Provide the Security Secretariat with a rationale of how the requirement above is met. We use SVN as the revision control system to track all software &amp; hardware specifications.</td>
</tr>
<tr>
<td>9.5</td>
<td></td>
<td></td>
<td>Each version of each configuration item that comprises the Authenticator and associated documentation SHALL be assigned a unique identification. Each version of each configuration item that comprises the Authenticator and associated documentation SHALL be assigned a unique identification. Provide the Security Secretariat with a rationale of how the requirement above is met. We use SVN to manage assignment of unique identification for all configuration items that are used in the authenticator. (eg source code, unit test results, guidance documents, etc.)</td>
</tr>
</tbody>
</table>

---

**Note:** "Configuration Item" stands for all the objects managed by the configuration management system during the product development. These may be either parts of the product (e.g., source code) or objects related to the development of the product (like guidance documents, development tools, tests results, etc.)
Authenticator Boundary

**HLOS (High Level Operating System)**
- Display

**ROE (Restricted Operating Environment)**
- Authenticator Application
- Firmware
- Secure Element:
  - Hardware Crypto Engine
  - Hardware TRNG
- Secure Storage
- BLE chip
- Microphone

UVHash = SHA256 of (User Verification Template(UV))

RawkeyHandle = AES128 (SHA256(KeyID + PrivKey) + KeyID)

Key Handle = AES128-GCM(RawKeyHandle + Counter + SHA256(AppID) + UVHash)

**Signing Operation**
1. Device receives the Keyhandle with a browser supplied AppID
2. Prompts user for voice verification & hashes the captured data.
3. Decrypts keyhandle AES-GCM
4. Verifies UVhash in the keyhandle with the hash of voice verification data in No.2
5. If user is verified, hash the received AppID and compare with the AppID hash that is inside the KeyHandle.
6. Validate counter
7. Access AES key for raw keyhandle using KeyID
8. Decrypt the encrypted part of KawkeyHandle
9. Verify the keyID hash
10. Sign
<table>
<thead>
<tr>
<th>ASP Description</th>
<th>Data</th>
<th>Description</th>
<th>Secret ?</th>
<th>Strength</th>
<th>Where this is stored.</th>
<th>How this is protected.</th>
<th>How this is generated.</th>
<th>Input/Output</th>
<th>When this is destroyed.</th>
<th>Unique or Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Keys/ Uauth Private keys</td>
<td>ECDSA P-256 Curve</td>
<td>Private key used for signing operation</td>
<td>Yes</td>
<td>128</td>
<td>in the Keyhandle</td>
<td>wrapped and exported to RP in AES-GCM 128</td>
<td>Generated using TRNG in SE</td>
<td>during registration and authentication</td>
<td>N/A</td>
<td>Unique</td>
</tr>
<tr>
<td>User verification reference data</td>
<td>Biometric voice data hash</td>
<td>sample used for user verification before any operation</td>
<td>Yes</td>
<td>N/A</td>
<td>in the Keyhandle</td>
<td>wrapped and exported to RP in AES-GCM 128</td>
<td>during user verification</td>
<td>during registration and authentication</td>
<td>N/A</td>
<td>Unique</td>
</tr>
<tr>
<td>Key handle</td>
<td>array</td>
<td>Contains all information necessary to authenticate a user to the RP</td>
<td>No</td>
<td>128</td>
<td>with relying party</td>
<td>AES-GCM</td>
<td>during user registration</td>
<td>during registration and authentication</td>
<td>N/A</td>
<td>Unique</td>
</tr>
<tr>
<td>Device Attestation Key</td>
<td>Device Root key</td>
<td>Device Root key inserted at manufacturing</td>
<td>Yes</td>
<td>128</td>
<td>in the SE</td>
<td>it relies on the SE for protection AES128</td>
<td>during manufacturing</td>
<td>Never</td>
<td>Never</td>
<td>Shared with 100,000 other authenticators shared for all keys on a device</td>
</tr>
<tr>
<td>Global Signature Counter</td>
<td>int 32</td>
<td>Keeps track of the signature done by the device</td>
<td>No</td>
<td>NA</td>
<td>in the Keyhandle</td>
<td>AES128</td>
<td>in the SE during device instantiation</td>
<td>during registration and authentication</td>
<td>Device Reset</td>
<td></td>
</tr>
<tr>
<td>Raw Key Handle</td>
<td>Key ID + PrivKey</td>
<td>concatenation of the AppID and PrivKey</td>
<td>Yes</td>
<td>128</td>
<td>in the keyhandle</td>
<td>AES128</td>
<td>during user registration</td>
<td>during registration and authentication</td>
<td>N/A</td>
<td>unique per user Registration</td>
</tr>
<tr>
<td>Secret Key</td>
<td>AES128 key</td>
<td>Symmetric key used to protect PrivKey</td>
<td>Yes</td>
<td>128</td>
<td>in the SE</td>
<td>it relies on the SE for protection</td>
<td>in the SE during user registration</td>
<td>Never</td>
<td>Device Reset</td>
<td>unique per user Registration</td>
</tr>
<tr>
<td>Device Key</td>
<td>AES128-GCM key</td>
<td>Symmetric key to protect Keyhandle</td>
<td>Yes</td>
<td>128</td>
<td>in the SE</td>
<td>it relies on the SE for protection</td>
<td>in the SE during device instantiation</td>
<td>Never</td>
<td>Device Reset</td>
<td>Unique per authenticator device</td>
</tr>
</tbody>
</table>