Standards Conformance

This document states, clause by clause, what csiPass implements, what it deliberately does not, and how each claim was checked. It exists so that a reader can tell an implemented feature from an intended one without reading the source.

csiPass is not certified. No FIDO Alliance certification has been applied for or granted, no interoperability testing against a real platform has been completed, and no side-channel or fault-injection review has been done. Every statement below is about what the code does, not about what any authority has attested. Where this document says a requirement is met, it means the code implements the clause and a test exercises it; it does not mean an independent party agrees.

Normative references

Short name	Document	Version used
CTAP 2.1	Client to Authenticator Protocol (CTAP)	Proposed Standard, 15 June 2021
WebAuthn L2	Web Authentication: An API for accessing Public Key Credentials, Level 2	W3C Recommendation, 8 April 2021
RFC 8949	Concise Binary Object Representation (CBOR)	STD 94, December 2020
RFC 9052	CBOR Object Signing and Encryption (COSE): Structures and Process	STD 96, August 2022
RFC 9053	COSE: Initial Algorithms	August 2022
FIPS 186-4	Digital Signature Standard (DSS)	July 2013
SP 800-186	Recommendations for Discrete Logarithm-Based Cryptography	February 2023
FIPS 180-4	Secure Hash Standard (SHS)	August 2015
FIPS 197	Advanced Encryption Standard (AES)	Updated May 2023
FIPS 198-1	The Keyed-Hash Message Authentication Code (HMAC)	July 2008
SP 800-38A	Block Cipher Modes of Operation	December 2001
SP 800-38D	Galois/Counter Mode (GCM) and GMAC	November 2007
RFC 5869	HMAC-based Extract-and-Expand Key Derivation Function (HKDF)	May 2010
RFC 4231	Test Vectors for HMAC-SHA-224/256/384/512	December 2005
RFC 3629	UTF-8, a transformation format of ISO 10646	November 2003
CTAPHID	CTAP 2.1 §11.2, USB Human Interface Device transport	Same document as CTAP 2.1
HID 1.11	USB Device Class Definition for Human Interface Devices	27 June 2001
U2F HID	FIDO U2F HID Protocol Specification	11 April 2017

How to read the verification column

Marker	Meaning
Host test	A native test in `firmware/waveshare/test/native` exercises it. Run with `firmware/waveshare/scripts/test-native.ps1`.
Vector	Checked against a published test vector from the referenced document.
Golden	The exact encoded bytes are pinned in a test and regenerated from the encoder, never hand-written.
Fuzzed	Reached by the native parser fuzzer with a padding-differential over-read detector.
Untested	Implemented, but nothing automated exercises it. Named explicitly rather than left implied.

Most of this document is verified on the host only. One end-to-end path has now run on hardware: a lab build registered a discoverable credential with a live relying party and then completed an assertion against it, signing in. That exercises MakeCredential, GetAssertion, CTAPHID framing, the encrypted store, and the trusted display together, and it is the difference between “implements the clause as read” and “a real platform agreed”. It covers one platform and one relying party, so it is a first data point rather than an interoperability matrix.

CTAP 2.1 — mandatory features

The CTAP 2.1 “Mandatory features” subsection lists what an authenticator claiming `FIDO_2_1` in the `versions` member must support. With `rk` true, as here, five apply.

#	Requirement	State	Where	Verification
1	`hmac-secret` extension	Implemented	`authenticator.cpp`, `webauthn_format.cpp`	Host test: input parsing, output encoding, per-credential seed round trip
2	`clientPin` or `uv` true	Implemented	`ctap_service.cpp`	Golden GetInfo vector. `clientPin` is false until a PIN is set, which is what the option means
3	`credMgmt` true, or the same function through a built-in UI	Implemented twice	`authenticator.cpp`, `ui.cpp`	Host test for the command; the on-device browser is the built-in UI the clause allows
4	`credProtect` where user verification exists	Implemented	`authenticator.cpp`	Host test: level parsing, hiding rather than refusing
5	`pinUvAuthToken` true	Implemented	`client_pin.cpp`	Golden GetInfo vector
6	`pinUvAuthProtocols` includes 2	Implemented	`pin_uv_auth_protocol.cpp`	Host test against the construction; HKDF against RFC 5869 vectors

`versions` therefore reports `FIDO_2_0`, `FIDO_2_1_PRE`, and `FIDO_2_1`. The preview version and command `0x41` remain accepted, which CTAP 2.1 states is OPTIONAL and provided precisely for platforms that have not moved to `FIDO_2_1`.

CTAP 2.1 — commands

Command	Code	State	Verification
`authenticatorMakeCredential`	`0x01`	Implemented, ES256 only	Host test, Fuzzed, Golden authenticator data
`authenticatorGetAssertion`	`0x02`	Implemented	Host test, Fuzzed
`authenticatorGetNextAssertion`	`0x08`	Implemented	Host test
`authenticatorGetInfo`	`0x04`	Implemented	Golden, canonical-validated
`authenticatorClientPIN`	`0x06`	Implemented, protocols 1 and 2	Host test; the CBOR parser inside it is not yet fuzzed
`authenticatorReset`	`0x07`	Implemented	Host test
`authenticatorCredentialManagement`	`0x0A` and `0x41`	Implemented, all seven subcommands	Host test, Fuzzed
`authenticatorSelection`	`0x0B`	Implemented	Host test
`authenticatorConfig`	`0x0D`	`toggleAlwaysUv` and `setMinPINLength`	Host test
`authenticatorBioEnrollment`	`0x09`, and its `0x40` preview	Not implemented	No sensor exists; see `docs/TODO.md`
`authenticatorLargeBlobs`	`0x0C`	Not implemented	Out of scope for this profile
Vendor commands	`0x40` – `0xBF`	Only `0x41` is accepted, as the credential management preview	Host test

`enableEnterpriseAttestation` (`0x0D` / `0x01`) returns `CTAP2_ERR_INVALID_OPTION`: there is no attestation material to enable.

CTAP 2.1 — extensions

Extension	Clause	State	Verification
`credProtect`	§12.1 Credential Protection	Implemented, levels 1–3	Host test
`hmac-secret`	§12.5 HMAC Secret Extension	Implemented, one or two salts	Host test
`minPinLength`	§12.4 Minimum PIN Length Extension	Implemented and advertised, disclosed only to listed relying parties	Host test. It was implemented but missing from the GetInfo `extensions` array until a reading of CTAP 2.3 found it, so no platform could ask for it
`largeBlobKey`	§12.3 Large Blob Key	Not implemented	Depends on `authenticatorLargeBlobs`
`credBlob`	§12.2 Credential Blob	Not implemented	No use for it in this profile

CTAP 2.1 — PIN/UV auth protocols

Item	Clause	State	Verification
Protocol 1 key agreement and tag truncation	§6.5.6 PIN/UV Auth Protocol One	Implemented	Host test against the construction
Protocol 2 dual key derivation	§6.5.7 PIN/UV Auth Protocol Two	Implemented	Host test; HKDF: Vector (RFC 5869)
Random initialisation vector per protocol 2 message	§6.5.7	Implemented	Host test: two encryptions of one plaintext differ
`pinUvAuthToken` permissions	§6.5 authenticatorClientPIN	Implemented: `mc`, `ga`, `cm`, `acfg`	Host test
Retry counter, power-cycle blocking	§6.5 authenticatorClientPIN	Implemented	Host test
Minimum PIN length enforcement	§7.4 Set Minimum PIN Length	Implemented at set time and at token issuance	Host test
Forced PIN change	§7.4 Set Minimum PIN Length	Implemented, reported as `forcePINChange`	Host test
Always require user verification	§7.2 Always Require User Verification	Implemented for MakeCredential and presence-taking GetAssertion	Host tested: the rule is a pure function in `uv_policy`, and the test enumerates every combination of its seven inputs. What reaches it from a parsed request is Untested
`makeCredUvNotRqd`	§6.1 authenticatorMakeCredential	Implemented, forced false while `alwaysUv` is on	Host tested for both the reported option and the behavioural path, including that the exception does not extend to a discoverable credential

Protocol 2 key derivation is checked against the construction the specification describes, not against another implementation. No public test vectors for it appear to exist. See `docs/TODO.md`.

CTAP 2.1 — GetInfo members

Reported: `versions` (`0x01`), `extensions` (`0x02`), `aaguid` (`0x03`), `options` (`0x04`), `maxMsgSize` (`0x05`), `pinUvAuthProtocols` (`0x06`), `maxCredentialCountInList` (`0x07`), `maxCredentialIdLength` (`0x08`), `transports` (`0x09`), `algorithms` (`0x0A`), `forcePINChange` (`0x0C`, only when true), `minPINLength` (`0x0D`), `firmwareVersion` (`0x0E`), `maxRPIDsForSetMinPINLength` (`0x10`, value 8), `remainingDiscoverableCredentials` (`0x14`), `attestationFormats` (`0x15`, `none` only), and `maxPINLength` (`0x16`, value 63).

Not reported, each for a stated reason: `maxSerializedLargeBlobArray` (`0x0B`) and `largeBlobs` (no large blob support); `maxCredBlobLength` (`0x0F`); `preferredPlatformUvAttempts` (`0x11`) and `uvModality` (`0x12`) (no built-in verification); `certifications` (`0x13`) — nothing here is certified, and the security baseline forbids implying otherwise.

The `aaguid` is sixteen zero bytes. That is a development identity, not an assigned one, and it means no relying party can allowlist this device.

CBOR encoding

Requirement	Clause	State	Verification
CTAP2 canonical CBOR form	CTAP 2.1 §8 Message Encoding	Enforced on input and produced on output	Host test, Golden, Fuzzed
Definite-length items only	CTAP 2.1 §8	Enforced	Host test
Minimal-length integer encoding	RFC 8949 §4.2.1	Enforced	Host test
Map keys sorted by major type, then length, then bytes	CTAP 2.1 §8	Enforced on input, produced on output	Host test
No duplicate map keys	RFC 8949 §5.6	Rejected	Host test
Rejection of tags	RFC 8949 §3.4	Major type 6 rejected outright	Host test, Fuzzed
Map keys limited to integers and strings	CTAP 2.1 §8	Complex keys rejected	Host test

Every response this firmware produces is validated against its own canonical validator in tests, so an encoder that drifted out of canonical order would fail the build rather than reach a platform.

COSE and cryptography

Item	Reference	State	Verification
ES256 signatures	RFC 9053 §2.1, FIPS 186-4 §6	Implemented, the only algorithm offered	On-device via PSA Crypto
P-256 curve	SP 800-186 §4.2.1.3	Implemented	On-device
COSE_Key EC2 encoding	RFC 9052 §7	Implemented for the credential public key and the platform key agreement key	Host test, Golden
ASN.1 DER signature encoding	WebAuthn L2 §6.5.6 Signature Formats	Implemented, including the high-bit padding case	Host test
SHA-256	FIPS 180-4	Used throughout	Vector (reference implementation)
HMAC-SHA-256	FIPS 198-1	Used for tokens, tags, and key derivation	Vector (RFC 4231)
AES-256-CBC	FIPS 197, SP 800-38A	PIN protocol message encryption	Vector (SP 800-38A)
AES-256-GCM	SP 800-38D	Credential vault at rest	Vector, including a case whose plaintext and additional data are both partial blocks
HKDF-SHA-256	RFC 5869	Vault key hierarchy and PIN protocol 2	Vector (RFC 5869)
UTF-8 validation of a PIN	RFC 3629, CTAP 2.1 §6.5.1 PIN Composition Requirements	Implemented, including surrogate and overlong rejection	Host test

The on-device implementations come from ESP-IDF's PSA Crypto provider. The vectors above prove the *host reference* implementation used as a test oracle, and — for HKDF and the PIN protocols — the production code compiled against that oracle. They do not prove the PSA implementations, which are the vendor's.

Transport

Item	Reference	State	Verification
FIDO HID usage page `0xF1D0`	U2F HID	Implemented	Untested outside a build
64-byte reports	U2F HID	Implemented	Host tested, including that a declared length past the buffer is refused rather than truncated and that exactly the maximum is accepted
`CTAPHID_INIT`, `PING`, `CBOR`, `CANCEL`, `KEEPALIVE`, `ERROR`	CTAP 2.1 §11.2 USB	Implemented	Reassembly is host tested and fuzzed as a packet stream, with the assembler guarded to catch a write past its message buffer. Command dispatch above it, and the busy-channel rule, still run only on the device
`CTAPHID_WINK`	CTAP 2.1 §11.2	Implemented	Untested
`CTAPHID_MSG`	CTAP 2.1 §11.2	Implemented. Carries CTAP1/U2F, which the specification says an authenticator SHOULD support	The APDU layer, the command execution, the approval window and the counter lease are host tested; the signatures are verified against real curve arithmetic. No real client has been through it
`CTAPHID_LOCK`	CTAP 2.1 §11.2	Not implemented	Not applicable until implemented
Channel busy, transaction timeout	CTAP 2.1 §11.2	Implemented	Untested
USB HID descriptors	HID 1.11	Implemented via TinyUSB	Untested

The USB device descriptor declares vendor `0xCAFE` and product `0x4014`, which is the TinyUSB example identity assigned to nobody. This must be replaced before any distribution; see `docs/TODO.md`.

WebAuthn

csiPass is an authenticator, so WebAuthn applies to it indirectly: the platform implements WebAuthn and speaks CTAP to the device. The clauses that reach the device directly are these.

Item	Clause	State
Authenticator data layout, flags, counter	§6.1 Authenticator Data	Implemented
Attested credential data	§6.5.1 Attested Credential Data	Implemented
Extension outputs in authenticator data	§6.1	Implemented
`none` attestation	§8.7 None Attestation Statement Format	Implemented, and the only form offered
Packed, TPM, and other attestation formats	§8 Defined Attestation Statement Formats	Not implemented
Signature counter	§6.1	Always zero, which explicitly declares that clone detection is not supported

A zero signature counter is a deliberate statement, not an omission. A counter that can be rolled back gives a false guarantee, and persisting a real one would cost a flash erase per authentication.

Credential storage

Item	Reference	State	Verification
Discoverable credentials	CTAP 2.1 §6.1	64 slots in the encrypted vault, enumerable through `credMgmt`, deletable on the device and from a host	Host tested through the vault suite
Non-discoverable credentials	CTAP 2.1 §6.1	Carried inside their own credential ID and never stored as vault secrets, so they are unlimited and cost no credential slot. A nonce, the policy, and a keyed tag over both with the RP ID hash; the key is derived on use. A site-held registration journal records RP/account metadata only (no secrets) – see `docs/TODO.md` and `docs/security.md`	Host tested: the published point is the one the derived scalar names, an ID opens only for the relying party it was made for, one flipped bit anywhere fails, and a bumped epoch invalidates every one of them at once
`credProtect` on a credential that is not stored	CTAP 2.1 §12.1	Carried in the ID in the clear and covered by the tag, so a host can read it and cannot change it	Host tested: rewriting the policy byte to claim a weaker level invalidates the tag, so the credential stops being ours rather than becoming a weaker one
`hmac-secret` on a credential that is not stored	CTAP 2.1 §12.5	The per-credential seed is derived from the same nonce, so the secret is stable with nothing stored. Whether the extension was requested at creation is a flag in the ID, so a relying party that never asked cannot obtain the secret later by sending salts	Host tested for stability, for separation between two credentials of one site, and for refusal under the wrong relying party
Exclude list	CTAP 2.1 §6.1.2	Covers both kinds. An ID this device issued is recognised whether or not it was stored	Untested outside a build
Cross-protocol separation	–	A U2F key handle and a CTAP2 credential ID are the same construction under different labels, and neither opens as the other	Host tested from a shared seed, so it is a statement about the labels rather than about the inputs happening to differ

A credential that is not stored cannot be deleted or listed. Revoking one means bumping the epoch, which revokes all of them, and the device's browse screen shows only the credentials it holds. This is the honest cost of not spending a slot, and it is the relying party's own choice: a site that wants a passkey the device manages asks for a discoverable one.

CTAP1 / U2F

Item	Reference	State	Verification
`U2F_REGISTER`	U2F Raw Message Formats §4.1, §4.3	Implemented	Host tested. The response layout is pinned byte by byte, and the registration signature is verified against the curve under the key the certificate carries – and checked not to verify under the credential key, which is the only thing that tells the two apart from outside
`U2F_AUTHENTICATE`	U2F Raw Message Formats §4.2, §4.4	Implemented in all three control modes	Host tested. The signature verifies under the key the handle names and stops verifying if the counter or the challenge moves by one value. Silent signing (control `0x08`) is refused: this device has no way to sign without a press
`U2F_VERSION`	U2F Raw Message Formats §4.5	Implemented	Host tested
`U2F_V2` in `versions`	CTAP 2.1 §10.1	Advertised only while CTAP1 can be answered	Host tested, including that a device that cannot answer it does not claim it
Key handles	U2F Raw Message Formats §4.1	Derived, not stored. Scalar and a keyed tag from the vault root with the epoch as HKDF salt	Host tested against real curve arithmetic: the published point is the one the scalar names, a handle opens only for the application it was made for, and one flipped bit anywhere in it fails
Attestation certificate	U2F Raw Message Formats §4.3	Per-device, self-signed, derived rather than stored and rebuilt on demand. X.509 v3, `basicConstraints CA:FALSE` critical, fifty-year validity written as UTCTime and GeneralizedTime as RFC 5280 requires either side of 2050	The exact bytes are pinned as a golden vector, produced once and checked with OpenSSL, which parsed it and verified its own self-signature. Host tested for byte-identical rebuilds and for surviving an epoch bump that invalidates every key handle. **A per-device attestation key is a cross-site tracking vector**; FIDO's answer is a batch certificate shared by at least a hundred thousand devices, which needs a manufacturing process this project does not have

Item	Reference	State	Verification
Signature counter	U2F Raw Message Formats §4.4	Implemented, leased in blocks of thirty-two	Host tested across simulated power loss at every position in a lease: no value is ever issued twice, and a failed reservation refuses to sign rather than signing at a value storage does not know was used. CTAP2 assertions still carry zero, which CTAP2 allows and U2F does not
`alwaysUv` interaction	CTAP 2.1 §10.1	Every U2F command, the version query included, is refused while user verification is required	Host tested
User presence	U2F Raw Message Formats §4.2	An approval window rather than a held request, because U2F clients poll	Host tested: a press with no prompt on screen approves nothing, an approval is consumed exactly once, and a clock that goes backwards expires the window rather than widening it

The approval screen cannot name the caller. U2F sends only the SHA-256 of the origin, and there is no way back to a name, so the screen says a site is asking using the older protocol rather than showing a hash nobody can read or a name the device would be inventing. The trusted display is this project's central argument and it is structurally weaker here; saying so is the only honest option available.

Deliberately out of scope

Item	Reason
NFC and BLE transports	The ESP32-S3 has no NFC radio; BLE would need its own pairing and privacy analysis
Enterprise attestation	There is no attestation material and the AAGUID is zeros
Certification of any kind	Nothing has been submitted to any body

Known gaps in this document's own claims

These are stated here rather than only in `docs/TODO.md`, because a conformance document that hides its own limits is worse than none.

1. **Interoperability is one data point, not a matrix.** One platform and one relying party have completed a registration and an assertion. Nothing else has been tried, and none of the optional paths – PIN, credential management, `hmac-secret`, protocol 2 – has been exercised by a real platform.

2. **The host-versus-hardware comparison now passes, over one credential.** A lab board's record, PIN, and index sectors were read back and all authenticate and decode under the host format. Getting there took two fixes, both on the host: the oracle's GHASH mishandled a partial final block, and the inspector compared the superseded copy of the index. The firmware, which uses the platform's GCM, was correct throughout. What has not been exercised is several occupied slots, a rewritten slot, or a torn write on real hardware.
3. **PIN/UV auth protocol 2 has no second implementation to agree with.** See the PIN protocol section.
4. **The CTAPHID framing layer is not fuzzed.** It is entangled with TinyUSB, so the fuzzer reaches the CBOR layer only.
5. **The Client PIN CBOR parser is not fuzzed.** It needs the crypto backend, and whether the host crypto stand-in is acceptable for parser fuzzing has not been decided.
6. **No side-channel or fault-injection review.** Constant-time comparison is used where it matters, but nothing has been measured.