TECHNICAL SPECIFICATION

version 10.0

INTERNET IDENTITY CARD ™

Architecture, cryptography & verification flows

Issuer	Https Card — Internet Identity Card Ltd
Company Registration	UK Companies House No. 09168431
Trademark	UK IPO UK00003166480 (2016) · INPI FR 4071419
US Copyright	TX 8-508-373 (2014/2017)
Office	124 City Road, London EC1V 2NX, United Kingdom
Disclosures	TDCommons #10079 · #10167 · #10394 · #10795 · #10796 · #11121 (CC BY 4.0)

This document supersedes the v9.0 edition. It specifies the vault key protection introduced in IIC v10.0, the corrected local signature chain, and the receipt status field that distinguishes a classical receipt from a failed hybrid attempt. The exported card format, the page integrity mechanism and the hybrid signature suite are unchanged from v9.0.

IIC v10.0 is a corrective release. It introduces no new capability: it repairs six defects found in the v9.0 implementation, four of which were reachable in normal use. Section references to v9.0 behaviour are retained where a reader of the previous edition would otherwise be misled.

1. Architecture overview

The IIC v10.0 is a single self-contained HTML file with zero runtime dependencies: no server, no PKI, no network access at verification time. All cryptography executes in the browser. Source modules are bundled deterministically into the distributed single file, so identical sources always produce an identical SHA-256.

- **Generator** — creates cards, holds the signing keys, exports standalone card files.
- **Issuer vault** — the generator's local state: issuer identity, private signing key, signature chain. Encrypted at rest under a key that is never stored. Specified in Section 2.2 and new as a documented component in v10.0.
- **Exported card** — a separate single-file document carrying its own verification engine (embedded via a base64 bootstrap), able to verify hybrid signatures fully offline.

2. Cryptographic specifications

2.1 Symmetric encryption

- AES-256-GCM for identity data at rest; random 96-bit IV per encryption.
- Key derivation: Argon2id, 96 MiB memory, t=4 iterations, p=4 lanes, random per-card salt.

2.2 Vault key protection (new in v10.0)

A random 256-bit state key encrypts the issuer vault under AES-256-GCM. The state key is not stored in any form. It is wrapped in two independent envelopes, either of which returns it:

```
envelope := { salt, iv, ct, kdf }
record   := { v, confirmed, word: envelope | null, recovery: envelope }

kek_i     = KDF(secret_i, salt_i)          // memorable word, recovery phrase
ct_i      = AES-256-GCM(kek_i, iv_i, stateKey)
stateKey  = AES-256-GCM-open(kek_i, iv_i, ct_i)   // either envelope
```

Both envelopes wrap the same state key. Rewrapping one envelope therefore leaves the vault contents, the signing key and the signature chain untouched: no key rotation is required or performed when a secret changes.

Current limitation. *The rewrap operation is implemented but is reachable only once, when a quick-mode vault is upgraded and a memorable word is set for the first time. There is no path to change a memorable word once it has been set, and no path to reissue a recovery phrase. A holder whose memorable word is compromised must create a new identity. This is a gap in the interface, not in the construction, and is recorded here rather than left for a reader to discover.*

Authentication is the GCM tag. No password hash is stored for the vault, so there is no offline target and no separate verification path. A failed unwrap is reported identically whichever secret was supplied; the reason is not disclosed.

KDF labelling. Each envelope records the derivation that produced it. Envelopes written by v10.0 carry `kdf: "argon2id"` with m=96 MiB, t=4, p=4. An envelope with no `kdf` field predates this release and is PBKDF2-SHA-256 at 600 000 iterations. The label is what makes the parameters revisable: without it, changing the derivation would render existing vaults unopenable and indistinguishable from a wrong passphrase.

Recovery phrase. Six blocks of four characters over a 62-character alphabet, drawn by rejection sampling to avoid modulo bias; approximately 142 bits. Generated at vault creation, displayed once, offered as a file, confirmed by transcription before the vault is written, and retained nowhere in any form. A vault created without a memorable word carries the recovery envelope only, and the phrase is then the sole means of access.

2.3 Hybrid signature suite

Unchanged from v9.0. Signatures use a crypto-agile suite registry; the active suite is `hybrid-v1`.

Algorithm	Role	Public key	Signature
ECDSA P-256 (SHA-256)	Classical	33 B	64 B
ML-DSA-65 (FIPS 204)	Post-quantum	1,952 B	3,309 B

Verification rule. A hybrid signature is valid only if all non-deprecated components verify AND at least one post-quantum component verifies. Deprecating the classical component is a verification-time policy option requiring no change to already-signed documents.

```
SUITES = {
  'hybrid-v1':   ['ecdsa-p256', 'ml-dsa-65'],   // active
  'pqc-pure-v1': ['ml-dsa-65']                  // future
}
```

2.4 Signed core construction

Unchanged from v9.0. Both signing features sign a canonical byte string binding the payload to the public keys and a timestamp.

```
core   = 'IIC-CORE-v1|' || payload || '|PK|' || concat(pubkeys) || '|TS|' || timestamp
digest = SHA-256(core)
sig_i  = Sign_i(digest)   for each algorithm i in the suite
```

3. SHA-256 page integrity (Mode A)

Unchanged from v8.4.1. Sealing: assemble the full HTML with a 64-character placeholder zone; compute H = SHA-256 of the serialisation; replace the placeholder with H. Verification: read the embedded H_ref, re-serialise with the placeholder restored, recompute, compare. Any byte change outside the hash zone triggers fail-closed lockdown. Fully offline.

4. Card export flow

Unchanged from v9.0.

- Generate exportCID and card salt; encrypt identity (AES-256-GCM).
- Generate the hybrid keypair set at card finalisation.
- Hybrid-sign the card identity core (cid, fingerprint, holder fields, creation timestamp).
- Assemble the card HTML; embed the verification engine via base64 bootstrap.
- Seal page integrity (Section 3) and emit the standalone file.

The exported card carries the encrypted identity, the hybrid public keys, the identity signature block, the embedded verification engine (~54 KB) and the self-integrity hash. Typical secure card size: ~210 KB. Nothing in v10.0 alters the exported card format; cards produced by v9.0 are unaffected.

5. Document signing and receipt status

Receipts remain version 3.0 in structure. One field is added, and one behaviour changes.

```
receipt = {
  version:  '3.0',                       // '2.0' when no PQ suite is configured
  pqStatus: 'ok' | 'not-configured',     // new in v10.0
  payload:  { data, cid, fp, ts, nonce, seq, prevHash },
  hash:     SHA-256(payload),
  signature: [...],                      // ECDSA, v2.0-compatible legacy field
  publicKey: { JWK },                    // ECDSA public key, legacy field
  hybrid: {
    suite:      'hybrid-v1',
    signatures: { 'ecdsa-p256': ..., 'ml-dsa-65': ... },
    publicKeys: { 'ecdsa-p256': ..., 'ml-dsa-65': ... }
  }
}
```

Failure semantics (corrected in v10.0). The v9.0 specification stated that card creation never fails if the post-quantum module is absent. That principle is retained, but v9.0 implemented it by suppressing every error in the hybrid path, which conflated two situations that a verifier cannot tell apart. v10.0 separates them:

Condition	Result
No PQ keys or suite configured	Classical receipt, `version 2.0`, `pqStatus: 'not-configured'`. This is a mode, not a failure.
Hybrid signature produced	`version 3.0`, `pqStatus: 'ok'`.
PQ configured, signature not produced	**No receipt is emitted.** Signing is refused and the cause reported. Under v9.0 this case produced a receipt identical to the first row, while the holder believed a hybrid signature had been made.

Legacy ECDSA fields are preserved so that v2.0-only verifiers can still validate the classical signature. v2.0 receipts remain verifiable.

6. Local signature chain

The generator maintains a hash-linked log of signing events. Three corrections in v10.0 make it verifiable in the cases where v9.0 silently was not.

```
entry  := { type: 'sig'|'rot', seq, hash, ts, data, prevHash }
state  := { sigChain: [entry], sigSeq, sigAnchor, sigRotations }
anchor := { seq, hash, droppedTotal }
```

- **Sequence number.** Taken from `sigSeq`, a counter persisted with the vault and monotonic over its lifetime. In v9.0 it was derived from the array length, which the retention cap froze: every entry past the fiftieth carried seq 51.
- **Truncation anchor.** Retaining the most recent 50 entries records the sequence number and hash of the last dropped entry, plus the running total dropped. The oldest surviving entry links to that anchor, so the retained chain still verifies to a known point. In v9.0 the entry referenced by the oldest survivor was itself discarded, leaving the chain linked to nothing.
- **Key rotation.** A change of signing key is recorded as an attested entry: a signature by the outgoing key over the canonical form of the incoming public key, produced before the outgoing key is discarded. Rotations are archived in `sigRotations`, which is never truncated, so key history outlives the retention cap. The attestation message is defined only over EC P-256 keys and includes field lengths, so that it is injective over every input it accepts.

The chain is a local audit trail. It carries no authority beyond the keys that signed its entries and is not a revocation mechanism.

7. Storage schema

Key	Contents
`iic-v6`	Vault ciphertext: AES-256-GCM under the session state key.
`iic-v6-env`	Envelope record (Section 2.2). New in v10.0.
`iic-v6-k`	**Removed.** In v9.0 this held the 32-byte vault key in unencrypted, beside the ciphertext it protected. It is deleted at migration and never written again.

Migration. A v9.0 vault is migrated at next open, without a defer option. The order of writes is the protection: the record is written unconfirmed while the old key remains in place; both envelopes are then proven to return the original key; only then is the record confirmed and the unencrypted key removed. A record left unconfirmed is discarded at next open and the migration restarts. The legacy unencrypted state format is detected, migrated, and written back encrypted.

Removing an entry from browser storage does not guarantee that the bytes leave the disk. Migration protects a vault against future profile access; it cannot retroactively protect one whose profile has already been copied.

8. Offline verification in the exported card

The exported card embeds the full verification engine, carried as a base64 payload and activated at load time by a bootstrap that decodes it and attaches it as an inline script element (no eval; compatible with the card's CSP). The card verifies hybrid receipts and its own identity signature with zero network access, on any modern browser, indefinitely.

Known limitation. Verification engines embedded in cards exported before v10.0 do not read `pqStatus`. Such an engine reports a receipt by its version field alone and would treat a malformed receipt claiming version 3.0 with no hybrid block as classical rather than invalid. Updating the embedded engine is deferred to a later release because it changes what reaches recipients.

9. Blockchain anchoring

Package-level and per-document SHA-256 digests are anchored on Bitcoin and Ethereum. An anchor proves the artefact existed at the anchor time; combined with the hybrid signature it supports verification even after a future deprecation of the classical component. Anchoring is performed by the issuer per release. It is not an automated credential function and cannot occur offline.

10. Compatibility & versioning

- v8.4.1 cards and v2.0 receipts remain verifiable; the ECDSA path is preserved.
- The exported card format is unchanged: no change in v10.0 touches how cards are produced or verified.
- v9.0 vaults migrate on first open. Envelopes written under the previous derivation continue to open, selected by the `kdf` label.
- Legacy memorable-word hashes computed without a per-vault salt are accepted once and upgraded on success, so the compatibility path removes itself rather than persisting.
- Deterministic build: same sources produce the same file hash, enabling reproducible anchoring.

11. Verification status

The v10.0 build has been exercised on a single machine, in a single browser, against test vaults. Six defects were identified by code review; four further defects were found only by executing the build, in paths that review had passed over, and two more during final journey testing. The release has not been examined by a third party and carries no formal evaluation.

12. Disclaimer

This document is provided for informational purposes only. It describes intended behaviour and properties under the assumptions stated in it; where those assumptions do not hold, the described protections may not apply. No software system can be guaranteed to be free of defects or completely secure. Conformance to a cryptographic standard in code is distinct from formal evaluation: nothing here constitutes a FIPS 140 CMVP validation, an ANSSI qualification, or a certification of any kind, each of which requires assessment by an accredited third party. Internet Identity Card — Prove and Protect Your Online Identity ™ is a trademark of Https Card — Internet Identity Card Ltd, registered with the UK Intellectual Property Office (UK00003166480).