TECHNICAL SPECIFICATION

version 9.0

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 (CC BY 4.0)

This document supersedes the v8.4.1 edition. It documents the hybrid post-quantum posture introduced in IIC v9.0: signatures combine the classical ECDSA P-256 algorithm with the post-quantum ML-DSA-65 (FIPS 204) algorithm, following ANSSI transition guidance.

1. Architecture overview

The IIC v9.0 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. v9.0 introduces a modular build: source modules (crypto suite, multi-zone sealing, document signing, card identity) 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.
- **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 Hybrid signature suite (new in 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 (e.g., after a quantum break of ECDSA) is a verification-time policy option that requires no change to already-signed documents.

Crypto-agility. Algorithms are registry entries with a uniform keygen/sign/verify interface. Suites are named lists of algorithms; migrating from *hybrid-v1* to a pure post-quantum suite (*pqc-pure-v1*) is a one-word configuration change, as recommended by ANSSI transition guidance.

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

Implementation: @noble/post-quantum (pure JavaScript, no WASM). Pure JavaScript means the existing Content-Security-Policy (script-src 'unsafe-inline') suffices; no wasm-unsafe-eval is required.

2.3 Signed core construction

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 serialization; replace the placeholder with H. Verification: read the embedded H_ref, re-serialize with the placeholder restored, recompute, compare. Any byte change outside the hash zone triggers fail-closed lockdown. Fully offline.

4. Card export flow (v9.0)

- 1. Generate exportCID and card salt; encrypt identity (AES-256-GCM).
- 2. Generate the hybrid keypair set (ECDSA P-256 + ML-DSA-65) at card finalization.
- 3. Hybrid-sign the card identity core (cid, fingerprint, holder fields, creation timestamp).
- 4. Assemble the card HTML; embed the verification engine via base64 bootstrap.
- 5. 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.

5. Document signing (hybrid receipts v3.0)

The card's document-signing feature produces chained receipts. v9.0 receipts are version 3.0:

```
receipt = {
  version: '3.0',
  payload: { data, cid, fp, ts, nonce, seq, prevHash },
  hash: SHA-256(payload),
  signature: [...],       // ECDSA signature (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': ... }
  }
}
```

The legacy ECDSA fields are preserved so that v2.0-only verifiers can still validate the classical signature; hybrid-aware verifiers validate the hybrid block, which co-signs the same payload hash with both algorithms.

The chain (seq, prevHash) is preserved from v2.0. Verification recomputes the payload hash, then verifies every suite component. v2.0 (ECDSA-only) receipts remain verifiable for backward compatibility; v3.0 is produced whenever hybrid keys are present.

6. Offline verification in the exported card

The exported card embeds the full verification engine. The engine is 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 can therefore verify hybrid receipts and its own identity signature with zero network access, on any modern browser, indefinitely.

7. Blockchain anchoring

Package-level and per-document SHA-256 digests are anchored on Bitcoin and Ethereum. An anchor proves the signed artifact existed at the anchor time; combined with the hybrid signature it supports verification even after a future deprecation of the classical component (the anchor evidences that signing predated the deprecation).

8. Compatibility & versioning

- v8.4.1 cards and v2.0 receipts remain verifiable (ECDSA path preserved).
- v9.0 additions are strictly additive; card creation never fails if the PQC module is absent.
- Deterministic build: same sources → same file hash, enabling reproducible anchoring.