TECHNICAL SPECIFICATION

version 8.4.1

INTERNET IDENTITY CARD ™

Cryptographic architecture and protocols of the IIC v8.4 system

Issuer	Https Card — Internet Identity Card Ltd
Company Registration	UK Companies House No. 09168431
Trademark	UK IPO UK00003166480 (2016)
Office	124 City Road, London EC1V 2NX, United Kingdom
File size	~336 KB (single self-contained HTML file)

Important notice

Legal & regulatory disclaimer

NATURE OF THE PRODUCT

Internet Identity Card ™ is a private software-based identity and verification platform developed by Https Card — Internet Identity Card Ltd. It is not a government-issued identity document and is not an officially recognised electronic identification scheme unless specifically recognised by applicable law in a given jurisdiction.

TRADEMARK NOTICE

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).

REGULATORY DISCLAIMER

References in this document to electronic signatures, eIDAS, international organisations, cybersecurity initiatives, or regulatory frameworks are provided for informational purposes only. Such references do not constitute certification, accreditation, endorsement, or legal recognition by any of the entities or frameworks mentioned. The cryptographic signatures produced by IIC are technical artifacts and are not inherently legally binding unless recognised under applicable law.

SECURITY DISCLAIMER

No software system can be guaranteed to be completely secure. The cryptographic properties described in this document are based on established algorithmic assumptions (NIST FIPS 197, SP 800-132, FIPS 186-4, RFC 6238) and are designed to substantially reduce — not eliminate — the risk of unauthorised access, modification, or impersonation. Users remain responsible for the operational security of their own devices, authentication factors, and exported card files.

NO ENDORSEMENT

Internet Identity Card ™ is an independent project. Participation in international discussions, public consultations, or working groups does not imply endorsement or adoption by the institutions mentioned.

1. Architecture overview

IIC v8.4 is a self-contained, single-file identity system. The generator and every exported card are standalone HTML files that run entirely in the browser, with no server, no external dependency, and no network call other than an optional self-integrity check.

1.1 System invariants

✓	**No backend**	All cryptography runs in the browser via WebCrypto and WebAssembly.
✓	**No telemetry**	No analytics, no fingerprinting, no metrics. The card connects to no third party.
✓	**Standalone artefacts**	The generator and each exported card are independent HTML files.
✓	**Reproducible**	Given the same identity and Card Passphrase, the same plaintext is produced.
✓	**Browser-native**	WebCrypto for AES-GCM and SHA-256; hash-wasm for Argon2id; no third-party JS at runtime.

1.2 File structure

```
IIC card HTML structure:
  <head>
    <meta charset="UTF-8">
    <meta name="iic-integrity-sha256" content="...">
    <meta http-equiv="Content-Security-Policy" content="default-src 'none'; ...">
    <meta http-equiv="Permissions-Policy" content="camera=(), microphone=(), ...">
    <style>...inline CSS...</style>
  </head>
  <body>
    <div class="scene"><div class="flip"><div class="front">...</div><div class="back">...</div></div>
    <!-- Below-card UX panels -->
    <script id="argon2lib">/* hash-wasm Argon2id, ~30KB */</script>
    <script>/* main runtime with var _iic = (function(){return{D, gT}})(); */</script>
    <script>/* integrity self-check */</script>
  </body>
```

2. Cryptographic specifications

2.1 Primitives

Symmetric cipher	AES-256-GCM (NIST SP 800-38D, IV 96 bits, tag 128 bits, AAD authenticated)
KDF for cards	Argon2id (RFC 9106) — memory 96 MiB, iterations 4, parallelism 4, hash 32 bytes

KDF for backups	PBKDF2-HMAC-SHA256, 600,000 iterations, salt 16 bytes
Hash	SHA-256 (FIPS 180-4)
Signature	ECDSA P-256 (SECP256R1, NIST FIPS 186-4)
Random source	crypto.getRandomValues (CSPRNG-backed Web Crypto)
Constant-time compare	Custom JavaScript _ctEq() routine (XOR + OR accumulator)

2.2 Dual-passphrase key architecture

IIC v8.4 separates two cryptographic domains:

1	**Memorable Word**	User-chosen, minimum 12 characters, 3 character classes. Stays on the issuer's device. Used to (a) unlock the local vault, (b) encrypt backups via PBKDF2 1.2M iterations, and (c) unlock the ECDSA private signature key.
2	**Card Passphrase**	Auto-generated at export time. Format xxxx-xxxx-xxxx-xxxx (rejection-sampled, ~95 bits entropy, chi-squared 65.67 < 80.23 critical value). Used only to derive the AES-256-GCM key that encrypts the identity payload of one specific exported card.

This separation ensures that the Card Passphrase, which is shared with recipients, never gives access to the issuer's local vault or backups.

2.3 Argon2id parameters rationale

The Argon2id parameters (m=96 MiB, t=4, p=4) were chosen to:

1	**Resist GPU/ASIC**	Memory-hard design forces attackers to use RAM, not just compute.
2	**Run on mobile**	Below 128 MiB to fit on low-end mobile devices without OOM. Typical derivation time: 0.5–2 seconds on a 2020+ phone, 0.3–1 second on desktop.
3	**Match OWASP 2026**	Above OWASP Argon2id minimum recommendations of m=64 MiB, t=3.

3. SHA-256 page integrity (Mode A)

3.1 Algorithm

```
BUILD (in the generator):
  1. Assemble the full HTML, with a placeholder zone:
     <!--IIC-HASH-START-->[64 zeros]<!--IIC-HASH-END-->
  2. Compute H = SHA-256(html_with_placeholder)
  3. Replace the placeholder with H (64 hex chars)
```

```
  4. Output the resulting HTML

RUNTIME (when the card is opened):
  1. Read the embedded hash H_ref from the meta tag
  2. Fetch the full HTML of the page itself
  3. Replace the embedded H_ref with the 64-zero placeholder
  4. Compute H_live = SHA-256(canonical_html)
  5. If H_live === H_ref: ✓ verified; else: lockdown
```

3.2 Lockdown behaviour

On mismatch, the card displays a full-screen red warning, disables every <button> and <input> in the DOM, and exposes window._iicIntegrityFailed = true. Decryption is impossible from this state.

3.3 Honest limitations

✓	**Detects modifications**	Any byte-level change to the file outside the hash zone is detected.
✓	**Does not prove authenticity**	A determined attacker can also recompute and reinject the hash. SHA-256 integrity is not a substitute for ECDSA signing.
✓	**CORS-dependent**	Some browsers (Chrome, Safari) block fetch() from file:// URLs. The card then shows "verification unavailable" rather than failing closed.
✓	**Complemented by Bitcoin**	For external proof, the issuer can timestamp the card SHA-256 on the Bitcoin blockchain via OriginStamp.

4. Card export flow

```
Secure card export sequence:
  1. Generate exportCID = "IIC-" + random_4_blocks_alnum   // unique per export
  2. Generate cardSalt  = crypto.getRandomValues(16)        // per-card random salt
  3. Generate cardPass  = autoGenerate19CharsPassphrase()    // shown once to user
  4. rawKey = Argon2id(cardPass, cardSalt, m=96MiB, t=4, p=4, len=32)
  5. aesKey = importKey("AES-256-GCM", rawKey)
  6. aad    = JSON({v:"8.4", cid:exportCID, ts:now, alg:"AES-256-GCM/Argon2id-96MB-t4-p4"})
  7. iv     = crypto.getRandomValues(12)
  8. cipher = AES-GCM-Encrypt(aesKey, iv, aad, plaintext=JSON.stringify(identity))
  9. encId  = JSON({v:"8.4", iv, s:cardSalt, a:aad, d:cipher})
 10. _zeroize(rawKey); _zeroize(plaintextBuffer)
 11. html   = renderCardHTML(exportCID, encId, ...)
 12. html   = injectIntegrityHash(html)        // step 3.1 BUILD above
 13. saveAsFile("IIC-XXXX-XXXX-XXXX.html", html)
```

4.1 Recipient decryption

```
Recipient flow:
  1. Open the .html file in a browser
  2. Integrity self-check runs (section 3)
  3. User types cardPass into the front-face input (with paste auto-normalization)
  4. uI() is triggered:
     - obj    = JSON.parse(EI)
     - rawKey = Argon2id(cardPass, obj.s, m=96MiB, t=4, p=4, len=32)   // ~1-3s
     - aesKey = importKey("AES-256-GCM", rawKey)
```

```
- plain  = AES-GCM-Decrypt(aesKey, obj.iv, obj.a, obj.d)
- data   = JSON.parse(plain)
- render(data) → photo, name, contacts, social, payment addresses
- _zeroize(rawKey)
```

5. Quick Card mode

Quick Cards reuse the entire Secure Card structure (3D flip layout, SHA-256 integrity, compact footer) but embed the identity directly in the HTML, with no encryption. The identity is serialized as a JSON literal into the runtime JavaScript at build time:

```
var _QUICK_DATA = {fn:"Alice", ln:"Dupont", email:"alice@example.com", ...};
window._ud = _QUICK_DATA;
window._isQuickCard = true;
// then either:
//  (a) hijack _iic.D to return the embedded data, call uI() for full styling
//  (b) fallback inline renderer that writes directly into #ph and #dt
```

The back face shows an upgrade hint inviting the issuer to switch to Secure mode for encrypted messages and ECDSA signatures. Quick Cards still benefit from SHA-256 page integrity, providing tamper-evidence even without confidentiality.

6. Storage & state management

6.1 Vault encryption (local)

The browser localStorage vault is encrypted with a per-browser random AES-256-GCM key generated at first run. This protects against incidental localStorage dumps but is not designed to resist an attacker with full device access.

6.2 Backup encryption (portable)

When the user exports a backup (.iic file), the vault is re-encrypted with PBKDF2-SHA-256 (600,000 iterations) using the Memorable Word, then AES-256-GCM. Without the Memorable Word, the backup is indistinguishable from random data.

7. Security analysis

7.1 Threat model — what IIC protects against

✓	**Passive eavesdropping**	A network observer cannot read the identity without the Card Passphrase.
✓	**Server compromise**	There is no server. There is nothing to compromise on a backend.
✓	**Modification in transit**	SHA-256 integrity detects post-export tampering.

✓	**Offline brute force**	Argon2id at 96 MiB makes large-scale GPU/ASIC attacks economically prohibitive against high-entropy passphrases.
✓	**Memory disclosure**	Derived keys and plaintext are zeroed immediately after use.
✓	**Replay across cards**	The AAD includes a per-card cid and timestamp, binding ciphertext to one export.

7.2 Out of scope

✓	**Compromised device**	Malware or keyloggers on the issuer's or recipient's device.
✓	**Weak passphrases**	If the user chooses a passphrase outside the auto-generator, weak choices weaken the system.
✓	**Physical coercion**	Forced disclosure of the Memorable Word or Card Passphrase.
✓	**Trusted third parties**	There are none. The issuer is the sole trust anchor.
✓	**Browser bugs**	IIC inherits the security of the host browser's WebCrypto implementation.

7.3 Cryptographic strength estimates

Card Passphrase entropy	~95 bits (rejection-sampled, $chi^2 = 65.67 < 80.23$ critical)
AES-256-GCM strength	256 bits (post-quantum still ~128 bits via Grover)
Argon2id GPU resistance	At 96 MiB, a single GPU thread requires 96 MiB of dedicated memory; commodity GPUs run only tens to a few hundred parallel threads.
SHA-256 collision	128-bit collision resistance, 256-bit preimage resistance
ECDSA P-256	128-bit security level (NIST B suite)

8. About this specification

This specification documents the technical architecture of IIC v8.4.1. The underlying cryptographic constructions are disclosed as open defensive prior art in two publications on Technical Disclosure Commons (Elsevier), released under CC BY 4.0:

• TDCommons **#10079** (12 May 2026) — Cryptographic Identity Document System Using TOTP-Derived Symmetric Keys.
https://www.tdcommons.org/dpubs_series/10079

• TDCommons **#10167** (May 2026) — Self-Verifying Single-File Cryptographic Documents with Dual-Passphrase Architecture.
https://www.tdcommons.org/dpubs_series/10167

Issuer: Https Card — Internet Identity Card Ltd, UK company No. 09168431 (incorporated 2014), holder of UK trademark UK00003166480 (filed 25 May 2016).