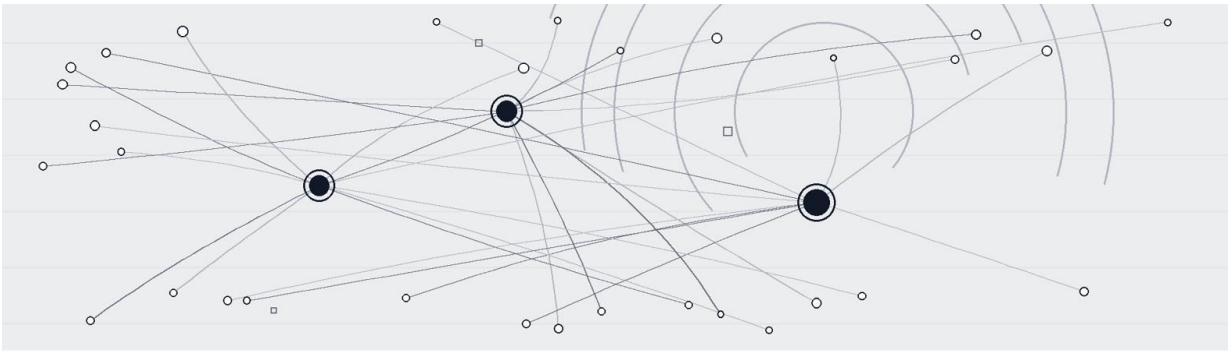
AUGUST 2026

APPLIED QUANTUM

THE APPLIED QUANTUM CCF TECHNICAL COMPANION



Obtaining the data: discovery, evidence, and what is not obtainable today

Version 1.0-RC – August 2026

Steve Vaile and Marin Ivezic, Applied Quantum

Disclosure: Applied Quantum provides post-quantum migration advisory services. This work is published openly under CC BY 4.0 and is computable without engaging Applied Quantum.

COPYRIGHT AND LICENSE

DISCLAIMER

This document is provided as professional guidance based on the authors' and Applied Quantum's practitioner experience and publicly available standards, regulations and industry research. It does not constitute legal, regulatory, financial or compliance advice. Organisations should consult qualified legal counsel, auditors and regulatory authorities for guidance specific to their jurisdiction, sector and circumstances.

The regulatory instruments, certification records, standards and vendor capabilities referenced in this document reflect the state of the landscape as of August 2026, and that landscape moves. Readers should verify current status against primary sources (regulators and supervisory authorities, NIST, EMVCo and PCI SSC, the CycloneDX specification, public certification listings, vendor documentation) before making decisions.

References to specific vendors, products or tools are for illustrative purposes and do not constitute endorsement.

This is a v1.0 release candidate: v1.0 final publishes in November 2026 after a pilot cycle. Open items are public on the roadmap at CCFramework.org, and corrections are published as dated entries on its errata page.

ABOUT THIS DOCUMENT

Document type	First-line technical companion
Parent document	The Cryptographic Concentration Framework – Universal – v1.0-RC (August 2026)
Intended audience	Security engineers, cryptographic inventory owners and tooling teams supplying data to second line
Assumed knowledge	The Universal Framework’s data requirements; working knowledge of the estate’s discovery and inventory tooling
Scope	Obtaining the data: discovery approaches, evidence grading, and what is not obtainable today. Deliberately outside the framework, so the method does not age with tooling.

Status	Version 1.0-RC – release candidate; v1.0 final November 2026 after pilot

VERSION HISTORY

Version	Date	Changes
1.0-RC	August 2026	First public release, as a release candidate; drafted through four external review rounds and a family-wide line-edit pass.
1.0-RC	7 September 2026	Section 6 cloud-custody guidance aligned to the Universal's unresolved-boundary determination.
1.0-RC	8 September 2026	Section 4.2: corroboration of a supplier attestation against a security policy recorded as a separate attribute, the evidence basis staying E4. Section 6: an undisclosed cloud module counted as one unit at its known population. Prose revised throughout.

HOW TO USE THIS DOCUMENT

This document is the first-line companion to the CCF Universal Framework. It covers obtaining the data the framework consumes. It sits deliberately outside the framework so the method does not age with tooling. Second-line readers need the Universal's evidence standards, not this document.

ACCOMPANYING RESOURCES

The framework family publishes at CCFramework.org: the Universal Framework, the Payments and Financial Services extensions, the payments whitepaper, the CBOM Conformance Statement, the Technical Companion and the instruments, together with the public roadmap, the errata and corrections page, the provenance survey and the frontier-census methodology.

Every canonical figure in the family comes from the reference implementation, with ten canonical test vectors that fix each one exactly. Both are open under Apache-2.0 on the Applied Quantum GitHub, and any implementation that reproduces the fixtures is conformant.

The Applied Quantum CBOM Profile, the input data contract, publishes at CBOMProfile.org. The Applied Quantum PQC Migration Framework, whose sector taxonomy the extensions follow, publishes at PQCFramework.org. Practitioner background and adjacent analysis sits on PostQuantum.com, as further reading rather than a normative source.

TABLE OF CONTENTS

Companion to the CCF Universal Framework **v1.0-RC.** The framework is a second-line risk instrument. This document is a first-line engineering one. Field definitions are in the Applied Quantum CBOM Profile v1.0-RC or later.

1. WHY THIS IS A SEPARATE DOCUMENT

The engineering task is to establish what executes, what is trusted, where keys were generated and held, and what relevant paths actually do. The assessment team then resolves those facts under the Universal Framework. This separation allows discovery tooling to evolve without changing the calculation rules, and allows engineers to identify where an apparent numerical result exceeds the evidence available.

A risk framework that specified discovery tooling would become outdated and place execution with the wrong function. Model risk frameworks specify the data a model requires and the treatment of data that falls short; they do not prescribe how to build the data warehouse. CCF states its input contract. This companion explains how an engineering team satisfies it.

Four of the six layers can be populated well today from existing tooling. Layers 2 and 6 have no turnkey solution, no standard schema field and no public dataset mapping deployed estates to the lineage and design evidence they require. A commitment to complete CCF-3 data in the first cycle exceeds what the tooling supports. Plan these two layers as a supplier-engagement programme and use the disclosure response rate to measure progress.

Use this companion with the Data Request and Conformance Statement during collection and resolution. The CBOM Profile owns field definitions and native-structure conventions; the Universal Framework owns counting and calculations. The companion supplies engineering approaches and quality checks without prescribing a vendor or redefining the method.

The delivery is a versioned CBOM and the linked assessment supplement specified in the Conformance Statement. The supplement is necessary because the current Profile does not fully represent counted cohorts, every generation-origin edge, service-specific criticality or complete ordered-hop topology. It should remain linked to the factual inventory rather than become a competing inventory maintained independently.

A successful engineering response does not require every supplier to reveal all provenance. It requires accurate facts where obtainable, explicit uncertainty where not, and enough structure for the assessor to distinguish those states. A precise unresolved boundary is more useful than a guessed module identity.

2. OBTAINABILITY BY LAYER

Layer	Best achievable tier	Effort	Turnkey tooling
1 Algorithm	E1–E2	Low	Yes
2 Lineage	E4, occasionally E2	High	**No**
3 Trust root	E2–E3	Low	Yes
4 Custody	E3–E4	Medium	Partial
5 Protocol	E1	Medium	Yes
6 Key generation	E4–E5	High	**No**

Plan the programme against this table. Allocating equal effort to every layer overspends on layers 1 and 3 and leaves layers 2 and 6 unable to progress. This is why Universal Phase 0 warns against a CCF-3 first-cycle target.

2.1 Plan collection by evidence source

Layer or claim	Principal sources	Main limitation to address
Algorithm and parameter identity	Source/binary analysis, certificate inventories, configuration and runtime/endpoint observations	Capability, installed code and actual use may differ
Executing implementation family	Build records, deployed artefacts, component inventories, provider configuration and runtime module information	Vendored code, static linking, replacements and mixed implementations
Implementation ancestry	Source/build provenance, relevant security-policy content and supplier engineering disclosure	Presence of a library does not by itself prove retained ancestry
Accepted trust anchor	Verifier configuration, chain-building behaviour, trust-	An installed or observed root may not be the path actually

Layer or claim	Principal sources	Main limitation to address
	store policy and certificate records	relied upon
Custody family/manufacturer	Module inventory, firmware records, applicable certificates and supplier disclosure	Cloud boundaries and reseller branding can obscure underlying identities
Observed protocol outcome	Endpoint/gateway logs, passive observations and appropriately scoped active tests	Samples may omit fallback populations or production behaviour
Generation point/design	Key-origin records, generator configuration, source/module provenance, entropy records and supplier disclosure	Current custody is not generation origin; a design class is not a shared design identity

This is a collection plan, not a universal ranking of tools or evidence. Obtainability differs across estates. Do not promise a first-cycle CCF-3 result solely because a scanner can emit CycloneDX.

3. LAYER 1 – ALGORITHM

Sources and tooling. Use source and binary scanning, certificate inventories, configuration and runtime observation. Relevant tooling categories include source and object-code scanners producing CBOM output, runtime analysers and filesystem scanners. Several emit CycloneDX directly. E2 is routinely achievable; E1 is available with runtime observation.

Collect algorithm, parameter and function information from the sources appropriate to the platform. Reconcile source declarations with the deployed build and, where available, observed use. A product's documentation can establish supported algorithms but cannot establish which one a particular service negotiated.

Normalise identifiers before counting. Preserve recognised curve aliases, OIDs, parameter sets and algorithm families in a controlled mapping tied to the chosen schema version. Record a canonical identifier when no applicable OID exists. Do not require mode or curve attributes on unrelated algorithm types merely to fill a common table.

Wrapping keys. Investigate asymmetric wrapping over symmetric key estates. A database's bulk encryption may be symmetric while a public-key wrapping or distribution mechanism remains in scope for the quantum scenario. Scanners report the symmetric estate and can omit the wrapping relationship. Query the key management system directly. Identify the wrapping relationship and relevant persistent key without falsely reclassifying the underlying symmetric algorithm.

Collection limitations. Firmware-resident cryptography lies outside source and filesystem scanning. Source scanners report intended algorithms, not what executes, so static-only collection cannot establish runtime use. Firmware, secure elements, platform modules and application-embedded cryptography may require other sources. Record inaccessible classes and their independently known or unknown populations.

Coverage treatment. An unobservable class is recorded as unknown coverage, never as absence. The absence of a scanner finding cannot establish the absence of cryptography in a class the method cannot inspect. Maintaining this distinction is the most consequential collection discipline in this document.

The Profile's `pqc:vulnerabilityStatus` is a classification assertion under the specified threat model. Have the relevant cryptography/risk owner approve rule-based enrichment and its parameter handling. A vendor's "quantum-safe" label should not substitute for that classification or for separate deployment assurance.

4. LAYER 2 – IMPLEMENTATION LINEAGE

Identify the implementation performing the operation. Record the actual library, provider, module or firmware implementation, with version/build and deployment context. Component identity should include a native `bom-ref`, name, type and the available supplier, hash or component identifier. Connect the implementation to its consumer and the algorithm capability it provides through the native dependency graph or a traceable supplement record.

For an operation using multiple simultaneous components, such as a hybrid, record all component roles and the combiner. The assessor uses the composite-unit rule for one primary vote per asset and retains each component in reach. Engineering output must not duplicate the asset merely to list both algorithms.

4.1 What does not work

Dependency manifests alone. Manifests identify declared dependencies but miss vendored code, statically linked cryptography and cryptography delivered in compiled form. They may also miss dynamically substituted code. Preserve the distinction between a declared dependency and the implementation performing the operation.

Binary composition analysis alone. A library string or signature match provides E2 evidence of presence, not ancestry. Backported fixes are not visible through that match, while stripping and obfuscation can defeat it. The result does not establish every retained code path.

Build provenance alone. Build provenance establishes "built from this commit", not "descends from this upstream". Use it where a vendor publishes it, although few do. Preserve the claim it supports rather than treating a source-commit reference as complete upstream derivation evidence.

4.2 Mining certification records

Published FIPS 140-3 non-proprietary security policies frequently name the upstream library in free text. The disclosure may be explicit or appear through integrity function names, module naming or repository URLs. These records form a public dataset that is not systematically used in financial services.

1. Enumerate every validated cryptographic module in the estate, from custody platform certificates, supplier attestations, and product documentation.
2. Retrieve each module's published security policy.
3. Extract upstream library mentions, integrity function names, and repository references.
4. Record as `appliedquantum:lineage:cmvpCertificate` with the extracted ancestry.

Identify applicable records for the product and version actually deployed. Link the relevant passage to the certificate reference and the underlying evidence record. A CMVP validation is not a general provenance assessment; NIST expressly identifies supply-chain and related commercial matters outside the programme's requirements. Its associated documentation can nevertheless contain useful supplier statements. A clear statement of derivation is different from an inference based on a function name or library-like string.

Certification programmes do not assess supply chain or source provenance. A security policy's mention of an upstream is therefore a voluntary disclosure, not a verified provenance attribute. Coverage is inconsistent, and an unnamed upstream leaves the lineage unresolved.

Grade E4 where the policy names an upstream. Where a supplier attestation and the security policy agree, record corroboration as a separate attribute; the evidence basis stays E4. Where they disagree, retain both assertions and raise the discrepancy. An analyst inference is E5 unless another actual evidence basis applies. Record scope differences and contradictions rather than selecting the more convenient source.

4.3 What establishes lineage

Scanning does not resolve layer-2 ancestry; supplier engagement does. Issue the Supplier Lineage Disclosure Request in the first week of the cycle, identifying the implementations and missing provenance that the supplier must address.

For first-party code, use the dependency graph, build provenance and a populated `pedigree` block. Populate ancestry at build time. Capturing this evidence in the build/release process is more repeatable than reconstructing it from deployed artefacts after an incident. An engineering assertion still needs a source record and a deployment link; internal ownership alone is not verification.

4.4 Populating pedigree

CycloneDX carries the structure in `pedigree.ancestors`, but cryptographic tooling does not populate it. Populate pedigree manually for cryptographic implementation components where ancestry is known. Record the chain from deployed component to intermediate to reference implementation. Derivation chains nest; flat entries represent a genuine merge.

Use native dependency/provision links for code that is called or bundled without derivation. Use variants where similarity is established but direction is not.

Retain version, commit/hash, relevant primitives and divergence evidence. A fork that replaced the affected code may not belong in a particular ancestor's failure set, while still having genuine historical ancestry. Conversely, a different component name or maintenance team is not evidence that the vulnerable code was replaced. Reach is resolved for a failure mode, not by treating every historical pedigree edge as equally relevant forever.

A populated field provides the strongest basis for an upstream standardisation discussion: it demonstrates the existing capability in use.

5. LAYER 3 – TRUST ROOT

Sources and tooling. Use certificate lifecycle management, certificate transparency logs, internal CA databases and trust store inventories. Certificate discovery and lifecycle tooling cover this layer well. E2 to E3 is routinely achievable.

Chain building is the step most often skipped. Tools record the issuer, while the dependency unit is the terminating anchor. Resolve every certificate to its anchor and record cross-signing separately; otherwise intermediates may be reported as independent dependencies.

Build the accepted chain for the relevant service and verifier. Retain the anchor reference, trust-store version, relevant policy and observation/configuration evidence.

Certificate repositories, certificate transparency and issuance databases can help enumerate certificates; they do not alone establish which chain a particular verifier accepts or prefers.

Keep certificate signature and subject-key roles distinct in the native representation. For CycloneDX 1.7, the newer relation array contains objects, while the legacy fields carry string references. Test the mapping rather than performing a textual replacement of one property name with another.

Record cross-signing, alternate paths and any policy authority whose actions could affect several roots. The assessor will distinguish root withdrawal from root compromise. Demonstrating an alternative valid chain supports an availability question, but does not prove resistance to forgery while another compromised chain remains accepted.

Also enumerate trust stores on endpoints and appliances, including the anchors trusted by counterparty-facing paths that the institution does not operate. For externally operated anchors, establish the relevant service relationship rather than treating all public CAs as blocking counterparties. A certificate-provider migration plan is not proof that the deployed signature chain has reached the target state.

6. LAYER 4 – KEY CUSTODY

Sources and tooling. Use custody management planes, cloud key inventory APIs, procurement and asset records, and certification records. Cloud KMS APIs expose key inventory, algorithm, key specification, origin and cluster association. E3 is achievable for the inventory; E4 is the expected basis for the firmware-family and manufacturer claims that determine resolution.

Use platform management records and supplier evidence to map each applicable key or endpoint to its custody component. Native `securedBy` can describe a mechanism and an algorithm reference; it does not automatically identify a module. Preserve the actual custody link separately where required.

Firmware family and manufacturer origin are generally not exposed through management APIs. A managed cloud service's underlying module is often undisclosed. Identify firmware family using the supplier's canonical family and version information, with its issuer. Preserve product and original manufacturer independently. A reseller's invoice, module certificate and firmware record may identify different organisations for legitimate reasons; do not collapse them into a single vendor field.

Use the following evidence sequence:

1. **Certification records.** Validated module certificates and their security policies identify the module and often the manufacturer. Match those records to the deployed platform.
2. **Supplier request.** Use Section B of the Supplier Lineage Disclosure Request. Ask which other products share the firmware family under any brand.
3. **Procurement records.** A reseller is not the manufacturer. Use the records to distinguish their identities.

Where a managed cloud service conceals the module, retain the provider as the unresolved visible boundary, with its relevant account, service and deployment scope. That boundary is counted as one unit at its known population. Leave the lineage, firmware-family, manufacturer and generation fields unpopulated, with the reason, rather than filling them with the provider's name. Second line computes the reach bounds. Report one relationship producing bounded readings at layers 2, 4 and 6 as a finding.

If a key was imported, its current custody does not identify where it was generated. Make imported/generated origin explicit in the supplement even where the platform inventory does not expose a complete earlier provenance chain.

7. LAYER 5 – PROTOCOL AND NEGOTIATION

Sources and tooling. Use passive traffic observation, active scanning, load balancer and gateway logs, and configuration. Passive analysis and TLS fingerprinting record what was negotiated; active scanners record what is offered. Both are needed because they answer different questions. E1 is routinely achievable here; this is the only layer where the best tier is routine.

Record the outcome, not the capability. Observe the outcome for the relevant cryptographic function. For TLS key establishment, retain the protocol/version/group and the associated configuration population. For authentication or payment-specific profiles, retain a function-qualified outcome and the selected authentication method. A message-format name or generic protocol version may not be sufficient to identify the mechanism assessed.

Active testing can establish what a probe negotiated or what a configuration offers, depending on the test. It does not necessarily establish the distribution of production outcomes. Passive or endpoint observation can establish production behaviour within its visibility and sample. Where encrypted transport metadata limits passive observation, use endpoint logs at the terminating gateway, not configuration as a substitute for the outcome.

Create first-class path-state records with unique `bom-ref` values. Use `pathId` for the logical route, `observationWindow` for the period, and the supplement for service, hop,

outcome or population distinctions beyond the scalar properties. Preserve separate observations rather than overwriting a less secure outcome with a later or more common one.

An asset using several relevant paths must be linked to all of them through appropriate native references or the supplement. A single-valued `pathId` cannot encode that set. Do not create duplicate counted assets to work around the field's cardinality. The same restriction applies when one route has several required cryptographic hops.

Sampling is acceptable and its basis must be stated. Sample by path class, record observed paths at E1 and leave values outside the sample unpopulated. Retain the sampling frame, selection, period and limitations; any extrapolation is separate from directly observed data. Universal minimum population or transaction counts are not prescribed here because relevant populations differ, but a bare assertion that sampling occurred is not a reproducible basis. Include known fallback and critical configurations in the design and explain any omission.

For populations, preserve joint segment information. A table showing that 80% of devices support one condition and 70% support another cannot identify exactly which devices satisfy both. Supply the overlap evidence or retain bounds; do not take the minimum as a measured joint result.

8. LAYER 6 – KEY GENERATION

No tooling exists for this layer. Obtain the data manually through a chain of published certificates and documents.

1. Identify the key generation point.

Begin with the persistent key or a homogeneous key cohort and identify the point that produced it. Trace its relevant entropy-source, deterministic generator and generation-algorithm components. Record the dates, versions and conditions applicable when the material was generated, not only the platform's current configuration.

2. Identify the validated module certificate that covers it.
3. Follow the module to its referenced entropy source certificate.

Entropy source certificates are standalone records, publicly listed and referenceable by number across modules. Each covers one source in one stated operating environment. Record whether the deployment matches that environment. A validated source can support the evidence chain without establishing every link to the institution's deployment.

4. Retrieve the public-use document

It describes the noise source and the deterministic generator.

5. Record the design, not the certification profile.

Two products conforming to one profile may or may not share a design. Record both and let the framework compute the bound. Preserve supplier, source identity and design specificity.

Where the chain breaks, use Section C of the Supplier Lineage Disclosure Request. Identify the particular missing link: generation location, module identity, source design, library origin or operating conditions.

Distinguish design classes from identities. "Ring oscillator", a DRBG specification and a certification profile describe different levels of abstraction. Several vendors meeting one profile do not thereby use one design, and different labels do not prove different designs. Record class-only knowledge and apply the Universal Framework's evidenced and bounded states to the supported and possible relationships. Refer unresolved design membership to its bound treatment rather than asserting a common design.

The generation point remains the primary CCI unit. Shared designs are reach relationships. A known generation point with unknown design can still support a generation-point distribution, whereas a vendor name with no resolved generation relationship may not. A broad design-only inventory should not be represented as complete generation-point resolution.

Sample by generation-point class and state the basis. The homogeneity and completeness of the class must be evidenced. Cover every generation-point class and cover classes supporting flagged critical assets in full. Do not extrapolate across distinct designs without evidence.

Evidence tiers are ordered within a claim type. Under Universal A.6, E4 supplier attestation is the strongest routinely admissible basis for key-generation design, not a deficiency. Expect E4 or E5 across most estates. An institution reporting E2 entropy data across a large estate has probably recorded a certification profile as a design.

9. STATING A DEFENSIBLE COVERAGE CLAIM

Build the denominator before the numerator. Enumerate the estate from the configuration management database (CMDB), service map and application portfolio, then measure what discovery reached. Key-management inventories and issued-estate records contribute to that enumeration. Do not define the estate as whatever one tool discovered. Preserve counted classes, duplicate-resolution rules and known unquantified populations.

State the observable perimeter for each asset class. Classes the method cannot see are recorded as unknown coverage, never folded into the residual. Unknown whole-estate completeness is compatible with a useful scoped fraction, provided the two are not conflated.

Report per service. A blended figure can conceal the service most likely to produce the finding. Expect reported coverage to fall before it rises: a better second-cycle denominator can lower the fraction by exposing the incompleteness of the original enumeration.

The assessment supplement should link each material claim to its component or relationship reference, property/function, source, evidence basis, date and scope. Namespace-level evidence overrides and component defaults are useful for routine homogeneous data. Exceptions are required when an E1 protocol observation, an E4 ancestry statement and an E5 generation inference concern the same component.

For a shared asset, retain service-specific criticality and target-state determinations where they differ. A single boolean on a component cannot safely represent different service tolerances. Likewise, a single counterparty readiness boolean cannot represent both key-establishment and signature targets without a defined function context.

9.1 Quality checks before hand-off

Confirm that references resolve and `bom-ref` values are unique; that native relationship fields retain their schema meanings; that implementation pedigree is not attached to an abstract algorithm as though the algorithm were source code; and that layer/coverage populations do not count supporting records as additional assets.

Validate single-valued and repeated properties per carrier. Check that booleans and fractions are encoded as permitted strings, that unknowns are not silently false, and that population percentages have a denominator and date. Check that a new format conversion has not dropped multiple service references or changed a certificate relation array into a string.

Reconcile the CCF layer definitions with the engineering resolution: executing family rather than ancestor at layer 2; firmware family rather than vendor at layer 4; generation point rather than generic design class at layer 6. Where the supplement cannot establish these facts, state the gap and its affected population.

Retain a release manifest identifying input files and hashes, exact schema/Profile versions, tooling versions, transformations, manually reviewed exceptions and tests not executed. The hand-off should permit independent reproduction, not merely provide a spreadsheet of final figures.

10. STAGED IMPLEMENTATION

Stage 1, months 0–3. Establish CBOM generation and certificate discovery, and populate layers 1, 3 and 5. Build the service mapping and issue lineage disclosure requests to the largest suppliers in week one. Make existing inventory and observation

data usable through consistent identities, version-correct CBOMs and the assessment supplement.

Stage 2, months 3–9. Build the certification-record mining pipeline. Resolve custody platforms to firmware family and manufacturer, and establish the entropy evidence chain. Follow up the supplier requests and record the disclosure response rate.

Stage 3, months 9–18. Add runtime and passive observation to lift layers 1 and 5 to E1. Populate `pedigree` at build time for first-party components. Add provenance questions to supplier due diligence and maintain the relationships through build, supplier and platform change processes.

Continuous. Obtain notification of changes in ancestry, firmware family, manufacturer or entropy design, contracted where possible.

The schedule supports planning; it is not a promise that a particular supplier estate will be fully resolved within a fixed period. Advance when the necessary data, ownership and controls are in place, with material interim gaps reported.

11. WHAT DOES NOT EXIST

The implementation plan must account for six limitations:

1. **Lineage representation.** There is no crypto-specific lineage profile and no deployed tooling populating ancestry. CycloneDX's `pedigree` structure supplies the semantics, but no cryptography profile mandates its use.
2. **Binary-to-lineage tooling.** The authors know of no tool that reliably establishes cryptographic code lineage from a deployed binary.
3. **Device-to-design data.** The authors know of no comprehensive public dataset mapping deployed devices to entropy source designs.
4. **Certification provenance fields.** Certification records have no structured provenance field.
5. **Supplier questionnaires.** There is no standard supplier questionnaire asking for cryptographic implementation provenance. The Supplier Lineage Disclosure Request addresses that gap.
6. **Sector inventory schema.** There is no financial-sector cryptographic inventory schema. The CBOM Profile addresses that gap.

Each limitation is a constraint on the assessment, not a finding against the inventory team.

RESOLVED AT V1.0-RC

Three questions carried from v0.9 review, closed August 2026.

1. The certification-record mining technique stays as Section 4.2 describes it – the method at category level, graded honestly. No reference implementation ships with the framework: tooling stays outside it by standing decision, and the frontier census remains Applied Quantum's own application of the technique.
2. The document stays at tool-category level. Named products would date it within a year, and a framework that measures vendors' shared dependencies does not endorse vendors.
3. Entropy and protocol sampling need no minimum sample basis. Stating the basis is the requirement, per Sections 7 and 8 – a stated basis is gradable and challengeable, where a numeric floor would be false precision.