KavachQ

Post-quantum posture and cryptographic evidence for India's regulated estate

The Declaration Test
Five questions for any CBOM vendor

Every vendor now says “CERT-In compliant CBOM”. The phrase costs nothing to print. These five questions cost a vendor about ten minutes to answer honestly — and they separate a **document** from **evidence**, which is the only distinction that survives an audit. Each has a yes/no answer that can be demonstrated in the room. **Run them on us too.**

THE FIVE QUESTIONS

01 **Can I download a real CBOM right now, without a demo call?**

A CBOM is machine-readable output. Any vendor that generates one has thousands on disk and can send a redacted sample in seconds. Gating the artifact behind a booked call usually means the output is not ready, is thin, or is a screenshot.

`PASS` a parseable CycloneDX / SPDX file, before the meeting ends · `FAIL` “we'll walk you through it on the call”

02 **Is the CBOM signed? With what algorithm?**

CERT-In's minimum elements for CBOM/QBOM include an **Attestations** element — a digital signature over the bill of materials. An unsigned file has no integrity property: anyone in the chain can change a row and nobody downstream can tell.

`PASS` a named algorithm + an independently fetchable public key · `FAIL` unsigned, or “the platform secures it”

03 **Can it be verified offline, with your service unreachable?**

A submitted CBOM may be re-examined a year later, by a third party, after the vendor relationship has ended. Verification that requires calling the vendor's API is not verification. Offline verification needs a canonical serialisation (RFC 8785), a published or embedded public key, and a standalone verifier.

`PASS` “run the verifier on a disconnected laptop” · `FAIL` verification only through the vendor's platform

04 **Show me tamper detection. Change one byte.**

If 02 and 03 were answered honestly, this is a thirty-second demonstration. Ask for a documented, scriptable pass/fail result a pipeline can act on.

`PASS` live tamper demo + scriptable pass/fail · `FAIL` a green tick rendered by the same party that made the file

05 **What is the methodology behind your quantum timelines?**

NIST IR 8547 (final, October 2026; all quantum-vulnerable public-key algorithms, whatever the key size) deprecates RSA-2048 and ECC P-256 after **2030** and disallows them after **2035**. NSA **CNSA 2.0** sets required dates in the 2030–2033 range by category. Those are transition deadlines, not predictions that the algorithms break. A vendor rendering “2030” as “the year RSA breaks” has rounded off the source.

`PASS` deadlines cited to source, deprecation language intact · `FAIL` a break-year with no model behind it

OUR OWN ANSWERS

#	KAVACHQ	HOW TO CHECK IT
01	**Yes** — CycloneDX 1.7 CBOM, on request, before any call	Ask, or run the free scan and keep the output
02	**Yes** — ML-DSA-65 (FIPS 204) over an RFC 8785 JCS canonicalisation	Signature block embeds the public key and canonical hash; pin our published fingerprint
03	**Yes** — standalone verifier, no network, no KavachQ service	Run it on a disconnected machine
04	**Yes** — verifier returns PASS/FAIL; one edited byte fails it	Edit one byte in front of us
05	We cite IR 8547 and DST FY 2027–28 dates. **No break-year.**	Every date we print carries its source

WHAT WE DO, AND WHAT WE DELIBERATELY DO NOT

In scope today: a CBOM (CycloneDX 1.7) signed with ML-DSA-65 and verifiable offline, plus an SBOM in CycloneDX 1.6 and SPDX 2.3 (not signed on its own); signed QBOM, AIBOM and HBOM; VEX in OpenVEX 0.2.0 and CSAF 2.0; discovery across code, live endpoints, exports and imported BOMs. **Deliberately out of scope:** CVE reachability, live HSM enumeration (HSM key metadata is parsed from exports — PKCS#11, payShield) and SaaSBOM. Tell us early if your tender needs those.

HOW TO USE IT

Paste the five into your technical-evaluation annexure and require written answers — each maps to something CERT-In v2.0 already asks for. Asking costs nothing; not asking costs you the discovery, at submission time, that your artifact cannot be verified by the party receiving it.