

Quantum-Safe Cryptography: A Reading Guide for Tonight

Written for someone who needs to understand this field well enough to explain it, sell it, and hold a position in a conversation with a security team. Short chapters, no maths beyond one simple inequality, and everything is either a verified fact or marked as a judgement call.

Read it in order the first time, or jump to Chapter 12 if you want the pitch.

Contents

The sixty second version	3
Part One: the threat	3
Chapter 1. What a quantum computer actually is	3
Chapter 2. Which cryptography breaks, and which only weakens	4
Chapter 3. Harvest now, decrypt later	4
Chapter 4. Why the deadlines exist anyway	5
Part Two: the algorithms	5
Chapter 5. The toolkit in use today	5
Chapter 6. The replacements, and what they cost	6
Chapter 7. Hybrids, and why TLS 1.3 matters	7
Chapter 8. Crypto-agility, the thing that decides your timeline	7
Chapter 9. Misuse: the part most scanners miss	8
Part Three: how scanning actually works	8
Chapter 10. Why inventory is the hard part	8
Chapter 11. The six layers of an estate	9
Chapter 12. How a source scanner works, and where it lies	9
Chapter 13. How an endpoint scanner works, and the clever bit	10
Chapter 14. Artifacts, container images, and why they surprise people	11
Chapter 15. CBOM, SBOM, and being the place things land	11
Chapter 16. Scoring, and when a tool should refuse to score	12
Part Four: what this product does, capability by capability	12
Chapter 17. Every scan, what it reads, and what it produces	12
Chapter 18. The five Handbook areas, and what a tool can honestly claim	13
Chapter 19. What the tool deliberately does not do	14
Part Five: the market, the pitch, and the objections	14
Chapter 20. Singapore, precisely	15
Chapter 21. Who else is in the room	15
Chapter 22. The pitch at three lengths	16
Chapter 23. Objections, and the honest answers	16
Chapter 24. A practical path for your first five conversations	17
Chapter 25. Glossary, for the words that come up most	18
If you remember five things	19

The sixty second version

Today's internet is protected by two kinds of mathematics. One is public key cryptography, which is how two strangers agree on a secret and how a website proves it is really itself. The other is symmetric cryptography, which is how the bulk of the data is actually scrambled.

A sufficiently powerful quantum computer breaks the first kind completely. The second kind only gets weaker. That asymmetry is the whole story, and almost every practical decision follows from it.

Because data copied today can be decrypted later, the risk does not start when the quantum computer arrives. It starts when someone decides your traffic is worth storing. That is why governments have set deadlines, why the replacement algorithms are already standardised, and why the immediate work is not buying anything. It is finding out what you already depend on.

That last task, finding the cryptography, is what this product does.

Part One: the threat

Chapter 1. What a quantum computer actually is

A normal computer stores a bit, which is either 0 or 1. A quantum computer stores a qubit, which behaves like a 0 and a 1 in combination until you measure it. Measure it and you get one answer, with a probability attached to each possible outcome.

That alone would be a curiosity. What makes it powerful is two additional properties. Qubits can be entangled, so their outcomes are correlated in ways that classical bits cannot be. And quantum operations can be arranged so that the wrong answers cancel each other out, amplifying the right one. That trick, interference, is what a quantum algorithm actually exploits. It is not that the machine tries every answer at once. It is that it can be arranged so certain paths are cancelled.

The catch is noise. Qubits lose their state if disturbed, so real machines need error correction, and error correction needs many physical qubits to make one reliable logical qubit. The current ratio is measured in hundreds or thousands to one, depending on the hardware and the desired reliability. This is the single most important reason nobody can give you a date.

What to say if asked "how far away is it": nobody knows, and anyone who gives a confident year is guessing. The useful question is not when it arrives, but how long your data must stay secret after it does.

Chapter 2. Which cryptography breaks, and which only weakens

Two algorithms matter, and they do different things.

Shor's algorithm solves the mathematical problems that public key cryptography is built on: factoring large numbers, and finding discrete logarithms. Both become tractable on a sufficiently large quantum computer. This is a complete break, not a weakening. It affects:

- **RSA**, in every key size. 2048, 3072 and 4096 bits are all affected, because the problem is the structure, not the size.
- **Diffie-Hellman** and **DHE**, the classic way two parties agree on a shared key.
- **Elliptic curve cryptography**: ECDH, ECDSA, and the specific curves in use today such as P-256, P-384, secp256k1.
- **Ed25519** and **X25519**, which are modern and fast but still rest on discrete logarithms.
- **DSA**, the older US signature standard.

The practical consequence is uncomfortable: there is no "buy a bigger one" option. The answer is replacement.

Grover's algorithm is different. It gives a quadratic speedup for searching, which sounds dramatic but translates to halving the effective key length of a symmetric cipher. AES-128 becomes roughly as hard to break as AES-64 would be classically, which is why the advice is to prefer AES-256 for anything long lived. Grover does not break AES, and it does not break hashing or message authentication the way Shor breaks RSA.

There is a third category worth knowing, which is neither of these: cryptography that is already broken today. MD5, SHA-1, 3DES, RC4, and TLS 1.0 and 1.1 are problems right now, independent of quantum computing. A good scanner reports them in the same pass, and a good migration plan fixes them first because they are cheaper and more urgent.

Chapter 3. Harvest now, decrypt later

This is the argument that turns an abstract risk into a business decision, and it is the one to lead with.

An attacker with storage and patience does not need to break your encryption today. They copy the encrypted traffic now, keep it, and decrypt it when the machine exists. Encrypted data has a shelf life. The threat is not the day the quantum computer arrives, it is the day your data becomes readable.

The standard way to reason about it compares three numbers:

If the confidentiality you need (X) plus the time to migrate (Y) is greater than the time until the threat is real (Z), then the exposure is live today.

This is usually called Mosca's inequality, after Michele Mosca who popularised it. Work an example: medical records that must stay confidential for twenty years, in a system that would take four years to migrate, against an assumed eight years until a cryptographically relevant quantum computer. Twenty plus four is twenty-four, which is greater than eight. The data you are protecting this year is exposed.

The inequality is also why the CSA Handbook says that systems holding long-lived or highly sensitive data, financial and medical records in particular, are the primary targets of harvest-now-decrypt-later.

The practical value of this framing is that it converts "quantum" from a technology question into a data question, and data questions have owners, budgets and deadlines. The workbench in this product implements that calculation per system.

Chapter 4. Why the deadlines exist anyway

If the arrival date is unknown, why has Singapore legislated dates? Because the migration itself is the long part.

Replacing cryptography is not a software update. You have to find every use of it, decide what replaces it, verify performance and interoperability, re-issue certificates, update hardware that may have a ten year service life, renegotiate with vendors, and keep old signatures verifiable for as long as records must be trusted. The Handbook's own guidance is to align this with natural technology refresh cycles rather than run it as a separate programme, which is only possible if you start early.

The deadlines are therefore about starting, not about the quantum computer arriving on schedule.

Part Two: the algorithms

Chapter 5. The toolkit in use today

Before you can judge a finding, you need to recognise the vocabulary. Here is what a scan will surface and what each item is actually for.

Public key encryption and key agreement

Algorithm	What it does	Quantum position
RSA	Encryption and signatures. The workhorse since the 1970s.	Broken by Shor
Diffie-Hellman, DHE	Two parties agree a shared secret over an open channel	Broken by Shor

Algorithm	What it does	Quantum position
ECDH	The elliptic curve version, smaller and faster	Broken by Shor
ECDSA	Elliptic curve signatures, used in TLS certificates	Broken by Shor
Ed25519, X25519	Modern, fast, increasingly the default choice	Broken by Shor
DSA	Older US signature standard	Broken by Shor, and deprecated anyway

Symmetric encryption and hashing

Algorithm	What it does	Quantum position
AES-256	Scrambles data in blocks	Fine. Grover only halves it
AES-128	Same, shorter key	Acceptable, but little margin. Prefer 256
ChaCha20-Poly1305	Modern authenticated encryption, common on mobile	Fine
SHA-256 and stronger, SHA-3	Fingerprints and integrity	Fine
SHA-1, MD5	Older fingerprints	Already broken today
HMAC-SHA256	Message authentication using a shared secret	Fine. Shor does not apply
PBKDF2, bcrypt, scrypt, Argon2	Turning a password into a key	Fine. Review parameters separately
3DES, RC4, DES, Blowfish	Legacy ciphers	Already broken or deprecated

A useful mental shortcut: if it is **asymmetric** (a public key, a signature, a key exchange), quantum computing breaks it. If it is **symmetric** (a shared secret, a cipher, a hash, a MAC), it does not, though key sizes may need a margin.

Chapter 6. The replacements, and what they cost

The replacements are no longer theoretical. In August 2024 the US National Institute of Standards and Technology finalised three standards, and the rest of the world is building on them.

- **ML-KEM** (FIPS 203, from CRYSTALS-Kyber) for key establishment. "Module Lattice Key Encapsulation Mechanism". This is what replaces RSA and ECDH for agreeing on a key.
- **ML-DSA** (FIPS 204, from CRYSTALS-Dilithium) for digital signatures. Replaces RSA and ECDSA signatures.
- **SLH-DSA** (FIPS 205, from SPHINCS+) for signatures based on hash functions rather than lattices. A different mathematical foundation, which matters if you want diversity, at the cost of much larger signatures and slower operation.
- **FN-DSA / FALCON** is a fourth family, compact signatures with harder implementation requirements.

Two of these families, ML-KEM and ML-DSA, rest on lattice problems. That is why the third exists: it would be poor practice to move the entire world onto a single new mathematical assumption in one step.

The costs are real and worth knowing before a customer raises them. Post-quantum keys and signatures are much larger than their elliptic curve equivalents. An ML-DSA signature can be several kilobytes where an ECDSA signature is around 70 bytes. That breaks assumptions in protocols with fixed-size fields: constrained tokens, QR codes, some smart card formats, and anything with a 1,500 byte packet budget. It is also why hybrid key exchange, covered next, is the practical near-term answer rather than a full switch.

Chapter 7. Hybrids, and why TLS 1.3 matters

If you send both a classical and a post-quantum proposal in the same handshake, the connection is no weaker than today even if the new algorithm turns out to have a flaw. That is the idea behind hybrid key exchange, and it is why a scan reports on a group name like **X25519MLKEM768** rather than a single algorithm: X25519 is the classical half, ML-KEM-768 the post-quantum half.

Hybrid key exchange is a TLS 1.3 feature. There are two consequences worth internalising:

1. **A host that will not negotiate TLS 1.3 cannot offer post-quantum key exchange at all.** That is a definitive answer, not an uncertainty, and it is why an endpoint scan checks TLS 1.3 support specifically.
2. Large parts of the internet already have this. Cloudflare, Google and others enabled hybrid key exchange as a default across their networks, so many sites support it without anyone having decided anything deliberately.

For the migration, the direction of travel is clear. Key agreement moves to ML-KEM, usually hybrid at first. Signatures move to ML-DSA, or SLH-DSA where diversity matters. Symmetric stays AES-256.

Chapter 8. Crypto-agility, the thing that decides your timeline

Agility is the ability to swap an algorithm without re-engineering a system. It is the difference between a migration measured in weeks and one measured in years, and it is why both the Handbook and the MAS advisory treat it as central rather than nice to have.

The test is one question: **if a new algorithm were mandated tomorrow, how many places would have to change, and how long would that take?** If the honest answer involves a code release across several applications and a round of regression testing, agility is not yet in place.

The workbench in this product has a twelve point agility checklist. The items that surprise people most often are the boring ones: algorithm choices duplicated across a codebase rather than centralised, hardcoded identifiers, and no test environment for cryptographic changes.

Chapter 9. Misuse: the part most scanners miss

A correct algorithm used badly is often worse than an old algorithm used correctly, because it creates false confidence. This is where a specialised scanner earns its place, and it is worth being able to name examples:

- **ECB mode**, where identical plaintext blocks produce identical ciphertext. It leaks structure and provides no integrity. Replace with AES-GCM or ChaCha20-Poly1305.
- **A reused IV or nonce** with the same key, which in some modes is catastrophic rather than merely weak.
- **TLS verification switched off**: `InsecureSkipVerify, verify=False, rejectUnauthorized: false`. This is a serious vulnerability today, quantum or not.
- **Hardcoded keys, salts and secrets** in source. A hardcoded key is not a secret.
- **Base64 mistaken for encryption**. It is reversible by anyone, and it appears in real code more often than anyone wants to admit.
- **SHA-1 still used for signatures**, which is broken today.
- **Static RSA key exchange**, meaning no forward secrecy: one compromised key decrypts all recorded traffic that used it.

None of these are quantum problems. All of them are found by the same scan, and fixing them is usually cheaper than the migration itself, which makes them the easiest early win to offer a customer.

Part Three: how scanning actually works

Chapter 10. Why inventory is the hard part

The Handbook's first no-regret move is to identify your most critical systems and run cryptographic asset discovery on those first, rather than attempting to inventory everything at once. The MAS advisory is even more specific: financial institutions should maintain an inventory recording the algorithm and key length, who owns each asset, and the specific system where it is used.

The reason both start here is that migration work is not blocked by knowing what to do. It is blocked by not knowing where to start. Cryptography is embedded in places nobody thinks of as cryptography: a configuration file, a base container image, a certificate that renews itself quietly, a dependency that happens to bundle its own OpenSSL.

So the product is not really a scanner that produces a report. It is an inventory builder. The report is how the inventory is presented.

Chapter 11. The six layers of an estate

It helps to think in layers, because each has different discovery techniques and different trust implications.

Layer	What lives there	How you find it
1. Repository	Source code, configuration, dependency manifests, keys committed by accident	Read the code with rules
2. Build artifact	What actually got compiled and packaged: JAR, WAR, wheel, package	Open the archive
3. Container image	Base OS libraries, runtime, OpenSSL build, certificates, packages	Read the image layers
4. Infrastructure	TLS and SSH configuration, load balancers, key services, hardware modules	Read configuration as code, or from inside the network
5. External endpoints	Public TLS, certificates, cipher suites	Perform a handshake
6. Data systems	Encryption at rest, connection encryption	Read configuration. Not the business data itself

Two points worth holding on to.

Source and artifact disagree, and the difference is the useful part. Source tells you what an application declares it depends on. The built artifact and the image tell you what is actually there. The classic case is an OpenSSL version that arrived through a base image and appears in no repository anywhere. This product labels that distinction as declared versus observed in the applications view.

We deliberately do not read layer 6. We read the cryptography around a database, such as a connection string that names TLS or a configuration file that names a keystore. We never connect to the database and we never read business data. That is a design choice, and it is a selling point: a security team is far more comfortable with a tool that structurally cannot see customer records.

Chapter 12. How a source scanner works, and where it lies

Point a scanner at a repository and it does five things: fetch, filter, match, classify, aggregate.

Fetch. Download the repository as a compressed archive over HTTPS. For a private repository, the same archive endpoint with an authorisation token, or the customer's own machine running the offline scanner.

Filter. Skip dependencies, build output, virtual environments and generated code. Skip binaries by looking for null bytes. Skip documentation. Keep the files that could plausibly contain a cryptographic decision.

Match. Apply a rule set. Each rule is a named pattern that means "this is an algorithm or a usage of one". In this engine there are 84 rules covering 366 individual patterns across 22 language and file type groups, reporting 81 distinct algorithms. The rules are written per language family rather than as one generic pattern, because `Cipher.getInstance("RSA")` in Java and `Aes.Create()` in C# and `:crypto.sign()` in Erlang do not look alike, and a single pattern broad enough to catch all three would produce noise.

Classify. Decide what each match means for quantum risk, and separate production code from tests, examples and documentation. That separation matters enormously: a test fixture with a 1024-bit RSA key is not the same finding as a 1024-bit RSA key protecting a payment endpoint.

Aggregate. Turn thousands of individual matches into a bill of materials: distinct algorithms, occurrence counts, file counts, and the recommended replacement for each.

Now the honest part, which you should say out loud rather than wait to be caught on. Pattern matching has two failure modes. It produces **false negatives**, because an algorithm name built at runtime, or loaded by reflection, or encrypted in a configuration file, will not match any pattern. And it produces **false positives**, because a mention of RSA in a comment is not a use of RSA. Any credible tool must therefore publish three things: what it read, what it skipped, and what it cannot see. This product prints all three in every report. A tool that overstates its coverage is more dangerous than no tool, because someone will make a decision on it.

Chapter 13. How an endpoint scanner works, and the clever bit

You cannot read the negotiated key exchange group from an ordinary web request. The browser does not tell you, and the `fetch` API does not expose it. So a real scanner has to perform the handshake itself.

Here is the trick used here, and it is worth being able to explain because it makes the approach sound as serious as it is. The scanner opens a raw TCP connection to port 443 and sends a handcrafted TLS 1.3 `ClientHello` that offers the post-quantum hybrid groups **first**, along with an X25519 key share. Under RFC 8446 the server must then do one of two things:

- Reply with a **HelloRetryRequest** naming the group it prefers, if it wants a group the client did not send a key share for. If it names a hybrid group, the server prefers post-quantum hybrid key exchange.
- Reply with a **ServerHello** naming the group it selected. If it says x25519, it does not.

Either way the reply reveals the group, and the handshake never has to be completed, which means no key encapsulation mathematics is required. Four handshakes are run per host: TLS 1.3 for the key exchange, TLS 1.2 because its Certificate message is unencrypted and can be parsed, TLS 1.1 and 1.0 to detect legacy support, and a final probe offering only obsolete cipher suites.

Reading the certificate. TLS 1.2 sends the certificate chain in the clear, which means a small DER and X.509 parser can extract the subject, the issuer, the public key algorithm and size, the signature

algorithm, the chain length and the expiry dates. That is how an endpoint scan also produces certificate inventory, and why a report can tell you that a site uses an EC P-256 certificate signed with ecdsa-with-SHA384 that expires in 77 days.

Two limits, both of which the product states rather than hides. A hosted scanner running on Cloudflare cannot open a connection to Cloudflare's own edge, so Cloudflare-fronted hosts fall back to a labelled DNS inference instead of a measurement; the offline runner exists partly to solve that. And some hardened endpoints refuse a handshake that did not come from a real browser, in which case the honest answer is "could not confirm" rather than a guess.

Chapter 14. Artifacts, container images, and why they surprise people

An application source scan is reassuring and incomplete. The thing that runs in production is a built artifact inside a container image, and the image contains an operating system you did not write.

Artifacts are archives. A JAR is a ZIP, as are WAR, EAR, WHL and NUPKG. Scanning one means reading its index, listing every entry, finding bundled crypto libraries, certificates and keystores, reading the manifest and configuration text that survived the build, and recursing into nested archives, because a WAR contains JARs.

Container images stack in layers, and a later layer can delete a file from an earlier one using a whiteout entry. A scanner that ignores whiteouts will report files that are no longer in the image. This engine applies them.

The value is easiest to see in a concrete result. Scanning `nginx:1.27` straight from the registry, without a Docker daemon, found seven layers and 4,208 entries, and reported the bundled `libssl.so.3`, `libcrypto.so.3`, the `openssl` binary, `openssl.cnf`, and 141 Mozilla root certificates inside the CA bundle. Score: 25, Critical. Nobody chose most of that. It arrived through the base image.

Chapter 15. CBOM, SBOM, and being the place things land

An **SBOM** is a software bill of materials: a list of the components in a piece of software. A **CBOM** is a cryptography bill of materials: a structured inventory of cryptographic assets, meaning algorithms, keys, certificates and their relationships to software.

CycloneDX is the interchange format the ecosystem already speaks. Syft, cdxgen, Trivy and others can emit CycloneDX. Rather than competing with those tools, this product accepts their output, folds the crypto assets into its own inventory, and records which tool produced each item so provenance is never lost.

That is a deliberate strategic choice worth explaining clearly: the more scanners a customer already runs, the more they need one place that merges the output and converts it into a quantum position. Being the hub is a better business than being the eleventh scanner.

The same logic drives the offline runner's ability to detect tools already installed on a machine, run them, and merge what they produce. Nothing is bundled and nothing is downloaded for the customer: their licences stay theirs, they already installed it, and this product is not on the hook for the accuracy of someone else's rule set.

Chapter 16. Scoring, and when a tool should refuse to score

Customers want a number, so you need one, and you need to be able to defend it.

The method here weights each distinct algorithm by its category importance and by how often it appears in production code. Quantum-vulnerable algorithms count fully against the score, deprecated ones at 75 percent, adequate ones at 20 percent, and quantum-resistant or post-quantum ones at zero. Small bonuses apply for post-quantum signals and strong symmetric encryption. The full formula is published, which is the point: nobody should have to take the number on trust.

The more important design decision is what happens when there is not enough evidence. If a repository contains almost no cryptography, or if a host cannot be reached, the correct behaviour is to **decline to score** and say why. This engine returns "no cryptography detected" or "limited evidence" or "not measured", rather than a flattering 100 or a damning zero. It is a small thing that builds a large amount of credibility, and it is worth demonstrating rather than describing.

Part Four: what this product does, capability by capability

Chapter 17. Every scan, what it reads, and what it produces

Capability	What it reads	What you get	Availability
Endpoint scan	A public domain, by performing four TLS handshakes	Protocol support, preferred key exchange group, TLS 1.2 and 1.3 cipher suites, legacy ciphers still accepted, and the parsed certificate with key algorithm, size, signature algorithm and expiry	Free, no account
Bulk certificate scan	Up to twelve hostnames	A certificate register with algorithm, expiry and risk per host, importable into the workbench	Free, no account

Capability	What it reads	What you get	Availability
Public repository scan	A public GitHub or GitLab repository	Cryptographic bill of materials, readiness score, prioritised actions, every finding with file and line evidence, stated coverage limits	Free, no account
Private repository scan	A linked GitHub, GitLab or Bitbucket account, repositories you select	The same report, kept on your account	Paid plans
Offline runner	A directory, a public or private repository, a built artifact, a container image by registry tag or saved tar, or any local scanners already installed	The same report, produced inside your network. Nothing leaves except findings	One free repository scan, then paid. Endpoint scans always free
CBOM import	A CycloneDX CBOM or SBOM from another tool	The assets merged into your inventory, with the originating tool recorded	Free
Readiness workbench	Your own input, in the browser	Crown jewels with exposure arithmetic, inventory, governance register, migration waves, training plan, vendor register, then an exportable migration plan	Free
Applications	Reports attached to one product	A merged inventory with declared versus observed labelling, vulnerable count and mean score	Paid plans

The number that matters when someone asks how thorough it is: **84 detection rules, 366 patterns, 22 language and file type groups, 81 distinct algorithms**, and a coverage page generated from the rule set itself so the published list cannot drift from what the code does.

Chapter 18. The five Handbook areas, and what a tool can honestly claim

The CSA Quantum-Safe Handbook is organised into five areas. Being precise about what software can and cannot do in each is how you avoid overpromising.

Domain 1, risk assessment. The Handbook's no-regret move is to identify your most critical systems and run cryptographic asset discovery on those first. A tool can do the discovery. It cannot do your business impact analysis. The workbench provides the crown jewels register and the exposure arithmetic, and the judgement stays with the customer.

Domain 2, governance. The no-regret move is to decide who is accountable, because clear ownership upstream makes execution possible later. A tool can produce the owner register, the policy document, the milestone schedule and the agility checklist. It cannot enforce policy or sit in a governance forum.

Domain 3, technology. The no-regret move is to identify which quantum-safe algorithms can replace the vulnerable ones and to understand the performance trade-offs before committing. This is where a tool is strongest, and where the per-algorithm replacement guidance and the migration waves live.

Domain 4, training and capability. The no-regret move is to brief senior leadership now, because they approve the resources. A tool can generate the briefing outline and the capability register. It is not a training provider, and you should say so.

Domain 5, external engagement. The no-regret move is to contact key vendors for their post-quantum roadmaps, because their answers set your timeline. A tool can produce the vendor register and the request letter with the seven questions that actually change a timeline. It cannot make the call.

If someone asks whether a scan can prove compliance, the honest answer is no, and be sceptical of anyone who says otherwise. Automated scanning produces technical evidence. Compliance also needs attestation, documentation and human review. Saying this first is disarming rather than damaging.

Chapter 19. What the tool deliberately does not do

Knowing the boundaries is what makes the rest believable. This product does not:

- **Scan for CVEs.** Free tools such as Trivy and Gype win outright here, and a customer should run them.
- **Do whole-program dataflow analysis.** Semgrep and CodeQL follow a value from source to sink, which pattern matching cannot.
- **Detect secrets by entropy across git history.** gitleaks and trufflehog are better at it.
- **Grade TLS configuration in the depth SSL Labs does.** Use testssl.sh or sslyze alongside it.
- **Disassemble binaries.** Inside an image we report that a particular OpenSSL build shipped, not which functions inside it are called.
- **Connect to running systems, databases or hardware modules.**
- **Certify anything.** Nobody's scanner can.

What no combination of the free tools gives you is the joined-up answer: one bill of materials merging what source declares with what actually shipped, quantum classification with a per-algorithm replacement, misuse detection, post-quantum readiness as a positive signal, and a report a non-engineer can read. That is the position.

Part Five: the market, the pitch, and the objections

Chapter 20. Singapore, precisely

Two documents matter, and they come from different regulators. Learn these three dates verbatim, because they are the strongest thing you have in a sales conversation, and they are easy to check.

From the CSA Quantum-Safe Handbook V1, published 16 July 2026, for Critical Information Infrastructure owners:

By **31 Mar 2027**: Submission of CII owner's quantum-safe migration plan to CSA.

From **1 Jan 2028**: Procurement and implementation of new CII computer and computer systems that have a digital component should either support quantum-safe algorithms/technologies, and/or be quantum-safe ready.

By **31 Dec 2031**: Completion of quantum-safe migration across CII computer and computer systems, that is, vulnerable cryptography should no longer be used.

Note the wording of the middle date. It says "should", and it is about new procurement, not everything already running. Being precise about that distinguishes you from vendors who overstate it.

The MAS advisory, circular MAS/TCRS/2024/01, dated 20 February 2024, was issued to the chief executives of all financial institutions. It asks for crypto-agility, a cryptographic inventory recording algorithm and key length, ownership, and where each asset is used, and a strategy for the transition.

The strategic insight for a supplier. If you are neither a CII owner nor a financial institution, none of this is addressed to you directly. It reaches you through your customers. A bank or a government agency cannot meet its own obligations while depending on suppliers who cannot answer questions about cryptography, so the questions arrive in procurement documents and security questionnaires. That is the tailwind behind the entire mid-market opportunity.

Also worth knowing: the Handbook and the Quantum Readiness Index were jointly developed by CSA, GovTech and IMDA, and the QRI is a self-assessment at go.gov.sg/qri covering five domains across four maturity levels. We are not the QRI, we do not administer it, and we do not reproduce it.

Chapter 21. Who else is in the room

Three groups, and you should place yourself accurately rather than pretend the market is empty.

Free and open source tools. Trivy, Gripe and Syft for dependencies and CVEs. Semgrep and CodeQL for static analysis. gitleaks for secrets. testssl.sh, sslyze and SSL Labs for TLS. CBOMkit and cdxgen for cryptography discovery. Individually excellent, and collectively they do not produce a quantum readiness position or a report a procurement team can read.

Specialist quantum-safe vendors, including Singapore based ones. PQStation, QuantumSecure, ST Engineering with Quantizant, Theos, Qlavis, and others. Several offer cryptographic discovery, CBOM generation, dashboards and migration planning. Assume they are credible. Your differentiation cannot be "we also find cryptography".

Enterprise platforms. IBM, Thales, Entrust, Palo Alto. They sell hardware security modules, key management, PKI and network encryption, with the services to deploy them. For a large bank, that is the right purchase. It simply is not the first purchase, because none of it answers "where is our cryptography and which of it is quantum vulnerable".

The honest position, and the one on the compare page: the evidence layer comes before the enforcement layer, and the evidence layer is where a small, fast, affordable product can win.

Chapter 22. The pitch at three lengths

Thirty seconds, for a chance meeting.

"Singapore requires critical infrastructure owners to submit a quantum-safe migration plan by March 2027. You cannot write that plan until you know where your cryptography is. We find it, in your code, your container images and your public endpoints, and produce the inventory and the plan. The endpoint scan is free, so you can see the quality before we talk about money."

Two minutes, for a technical evaluator.

Add the mechanism and the honesty. "There are 84 detection rules across 22 language and file type groups, covering 81 distinct algorithms, plus misuse patterns that most tools miss. For endpoints we perform the TLS handshake ourselves, so we can tell you the negotiated key exchange group and whether hybrid post-quantum key exchange is in play. For images we read the layers, honour whiteouts, and report the OpenSSL build that arrived through your base image. Every report states what it scanned, what it skipped and what it cannot see. The offline runner scans inside your network so your source never leaves; the hosted option trades that for convenience. Both produce the same report."

A CISO conversation, where they will test you.

Lead with what you do not do: "We are not a CVE scanner and we are not a replacement for Semgrep. We do not connect to your databases and we do not read business data. What we do is produce the cryptographic inventory and the migration evidence, merge it across source, artifacts, images, endpoints and whatever your existing scanners produce, and classify all of it against the quantum model. Then you get a plan and an evidence pack."

That framing tends to earn more trust than a longer list of features, because it shows you know the difference between detection and judgement.

Chapter 23. Objections, and the honest answers

"Quantum computers are years away." Probably true, and it is the wrong question. The risk is data with a long confidentiality requirement being copied today. Work the arithmetic with them: how many years must this stay secret, how long would migration take, how long until the machine. If the first two exceed the third, the exposure is already live.

"Free tools can already find cryptography." Correct, and you should agree. Some of them are very good. They do not give you a merged inventory, a quantum classification with a per-algorithm replacement, a prioritised plan, or a report you can hand to a customer. Run them too. We will import their output.

"We are not a CII owner, so this does not apply to us." It reaches you through your customers. If you sell to banks, government or critical infrastructure, you will be asked, and the answer will be contractual rather than advisory.

"Can this make us compliant?" No, and be sceptical of any tool that claims it. This produces technical evidence. Compliance needs attestation, documentation and human review.

"Do you need our source code?" Only if you choose the hosted path. The offline runner scans inside your network and uploads findings only. For private repositories, the managed option fetches and scans on our infrastructure, and we say so plainly rather than hiding it behind "secure".

"How accurate is it?" Somewhere between useful and incomplete, and we publish exactly where. Pattern matching misses dynamic construction and encrypted configuration. That is why every report lists coverage limits rather than implying completeness.

"Why not just wait and buy hardware later?" Because the inventory is what tells you which hardware to buy. Buying an HSM without knowing what you are migrating produces an expensive shelf ornament.

Chapter 24. A practical path for your first five conversations

1. **Lead with the free endpoint scan.** Type their domain into the site while you are on the call. It takes seconds and it is a real measurement, not a demo. If they are behind Cloudflare, use the offline runner instead and explain why that matters. Being unable to measure something, and saying so, is more credible than a confident wrong answer.
2. **Run the public repository scan on something they own.** It is free and public, so there is nothing to approve. Most engineering teams have never seen a cryptographic bill of materials for their own code.
3. **Ask about the 2027 and 2031 dates.** Not rhetorically. Ask who in the organisation owns the answer, and whether an inventory exists today. The gap in the conversation is the gap in their evidence.
4. **Offer the smallest useful next step.** Not the platform. A fixed price discovery assessment across a handful of applications, with a report they can hand to their own customers.

5. **Leave them the coverage page.** It is generated from the tooling and it states the limits. Technical buyers read it, and it does more for credibility than any claim on a landing page.

Chapter 25. Glossary, for the words that come up most

CBOM: cryptography bill of materials. A structured inventory of cryptographic assets. Compare **SBOM**, which lists software components.

CII: Critical Information Infrastructure. The systems whose disruption would seriously affect Singapore. Their owners carry the CSA deadline obligations.

CRQC: cryptographically relevant quantum computer. One powerful enough to break real world cryptography. The phrase exists because "quantum computer" alone is ambiguous.

Crypto-agility: the ability to change algorithms without re-engineering systems.

CycloneDX: the open interchange format for SBOMs and CBOMs. This is how we accept other tools' output.

DER and X.509: the binary encoding and the format of digital certificates. Reading them is how a scan reports what a certificate is actually made of.

FIPS 203, 204, 205: the NIST standards for ML-KEM, ML-DSA and SLH-DSA respectively, finalised in August 2024.

Grover's algorithm: gives a quadratic speedup, halving effective symmetric key strength. It does not break RSA.

Harvest now, decrypt later: recording encrypted traffic today in order to decrypt it once a quantum computer exists.

HelloRetryRequest: a TLS 1.3 reply that tells the client which key exchange group the server prefers. The signal this product reads from endpoints.

Hybrid key exchange: using a classical and a post-quantum algorithm together, such as X25519MLKEM768.

ML-DSA, ML-KEM, SLH-DSA: the three standardised post-quantum algorithms. Lattice based for the first two, hash based for the third.

Mosca's inequality: if confidentiality required plus migration time exceeds time to threat, you are already exposed.

PQC: post-quantum cryptography.

QRI: the Quantum Readiness Index, CSA's self-assessment questionnaire. Five domains, four levels. Not us, and not something we reproduce.

Shor's algorithm: breaks RSA, Diffie-Hellman, ECDSA and Ed25519 outright. The reason migration exists.

TLS 1.3: the protocol version that makes hybrid key exchange possible. A host refusing it cannot offer post-quantum key exchange at all.

Whiteout: a container layer entry that deletes a file from an earlier layer. Ignoring these produces findings for files that no longer exist.

If you remember five things

1. Quantum computing breaks public key cryptography outright and only weakens symmetric cryptography.
 2. The risk starts now, not when the machine arrives, because traffic can be copied today.
 3. The hard part of migration is inventory, and inventory is what this product produces.
 4. The Singapore dates are March 2027, January 2028 and December 2031, and they are checkable facts.
 5. Our position is not "we scan better". It is "we produce the joined-up evidence, we state our limits, and we merge with whatever you already run".
-

Everything factual in this guide was verified during the build: the CSA Handbook dates and the QRI structure from the Handbook and CSA's press release, the MAS advisory from the circular itself, the FIPS standards and TLS 1.3 behaviour from the standards. Judgements about competitors and market position are marked as such. We are not affiliated with CSA, GovTech, IMDA or MAS.