

FIELD REPORT · POST-QUANTUM CRYPTOGRAPHY

Cryptography as Code

We said hybrid post-quantum would arrive through your providers, and that governance should lead. It did, and it should. What we learned is where the years actually go.

A dogfooding report. In 2025 we told security leaders how to get ready for post-quantum cryptography. This year we held our own infrastructure to that advice and measured what happened.

What we said

In 2025 I told security leaders how to get ready for post-quantum cryptography. The algorithms were already chosen. NIST finalized ML-KEM for key establishment and ML-DSA for signatures. The advice was about everything else:

- 1 **Treat it as governance and risk, not a crypto project.** Put it under operational resilience. Let the governance teams lead: the business continuity and resilience people already know the vendors, the critical applications and where the data lives.
- 2 **Start with the inventory.** Find the systems that depend on quantum-vulnerable algorithms, and prioritize by the data they protect and how long it has to stay secret.
- 3 **Put quantum risk in the registers.** Add it as a contributing factor in the risk and threat registers you already run.
- 4 **Talk to your vendors.** Most of your cryptography lives inside products you don't control. Ask for their roadmaps.
- 5 **Hybrid is the path.** Pair classical key exchange with ML-KEM, so neither a classical break nor a quantum break alone exposes a session. It will arrive through your providers, your libraries and your protocols. Keep classical fallback for peers that aren't ready.
- 6 **Pilot and test.**
- 7 **Make cryptography lifecycle management part of service delivery,** so every product, component and protocol that can be updated, is.

I also said it would take time, and to start now.

That was the advice. This is what happened when we held our own infrastructure to it.

What happened at the edge

Nothing dramatic. That's the first finding.

Hybrid post-quantum key exchange reached our public edge the way we said it would: through provider defaults. The reverse proxy we run picked it up when its runtime made hybrid the default. Our content

delivery network turned it on at its own edge. The forwarding service in front of our apex domain did too. Nobody wrote a line of cryptography.

On September 25, 2026, we tested every public endpoint we run with three kinds of client: one that offers only the hybrid group, one with default settings, and one that speaks only classical cryptography.

WHAT THE CLIENT OFFERED	WHAT OUR ENDPOINTS DID
Hybrid only (X25519 + ML-KEM-768)	Negotiated hybrid. Every endpoint.
A normal modern client	Chose hybrid on its own. Every endpoint.
Classical only	Still connected, on classical. No one locked out.

Hybrid by default. Classical fallback intact. Across our marketing site, our product, and our source-control platform.

Our certificates are still classical, by design. Public web certificates move to post-quantum signatures when public authorities and browsers trust them. Key exchange is the part harvest-now attacks go after, and at our edge, key exchange is done.

Why it was easy: lifecycle

It was easy because everything at our edge was current.

The parts we run are declared as code and rebuilt from that code on every deploy, so they take their runtime's updates as a routine, not a project. The parts we don't run are services whose providers keep them current for us. When hybrid shipped, there was no system at our edge waiting on a maintenance window, no appliance three releases back, no service pinned to a library from 2019.

That points at where the years in a post-quantum migration actually go. Not the cryptography. The algorithms are standard and the providers are shipping them. **The years go to lifecycle debt:**

- ◆ systems that don't take updates, or take them a year late;
- ◆ patches deferred because nobody is sure what depends on what;
- ◆ legacy and end-of-life platforms with no upgrade path;
- ◆ vendor products whose roadmap is "later";
- ◆ cryptography hard-coded into applications instead of taken from the platform.

Everything that is current gets hybrid for free. Everything that is behind is your migration.

You could fairly say: you're a young company, of course your edge is current. That's the point. The difference between our edge and yours is not cryptography. It's lifecycle debt.

And some of that debt you can step around today. A current reverse proxy, declared as code, can be dropped in front of an existing deployment and terminate TLS with hybrid key exchange for every client that supports it. The system behind it keeps speaking what it speaks, on a private hop you control, until it can be upgraded or retired. That's the gateway pattern we recommended in 2025 for systems without native support, and it's how our own edge works: a current, declared proxy in front of the services behind it.

So the inventory we told you to start with has a sharper job than listing algorithms: **it has to tell you what is behind.** And the cryptography lifecycle management we told you to fold into service delivery turns out to be the whole program.

A call-out to ITSM

That makes post-quantum readiness an IT service management question, and most service management can't answer it.

Ask a typical CMDB which systems will negotiate hybrid key exchange. It can't tell you. It knows what someone entered, not what is running, which version, or which library. It is hand-fed, so it is always behind, and "behind" is exactly the thing you are trying to find. The inventory the migration needs can't be a survey or a spreadsheet, for the same reason.

Governance as Code turns that around. When the estate is declared as code and continuously compared with what is actually running:

- ◆ **The configuration record is a read of the estate**, not a ledger someone maintains. It can't go stale.
- ◆ **"Is it current?" is a difference** between what is declared and what is observed, computed continuously. A system that falls behind shows up as drift.
- ◆ **An upgrade is a change like any other**: proposed, tested, reviewed, merged and deployed, with the record written as a byproduct.
- ◆ **A system that can't move is a risk**, in the same register as every other risk, owned and treated, not a line in a migration spreadsheet.

That is cryptography lifecycle management, done by the system that runs the estate rather than by a program layered on top of it.

Governance leads, the pipeline does the work

This is how the three parts of our platform line up with the advice.

Everything as Code is the inventory and the lifecycle. The estate is declared, observed and reconciled, so the question "what is behind, and what does it protect?" is a query, not a project.

Integrated Cyber Risk Governance is where the governance teams lead from. Quantum exposure gets no special track. It shows up the way any exposure does, as a scored finding, and it sits in the risk register if it ranks, with an owner and a treatment. Controls read their evidence from what the system produces. A regression is a finding with an owner, not a surprise at the next assessment.

The AI-SDLC pipeline does the work. Each change is authored, tested, reviewed and released through the same governed pipeline as everything else we ship. The agent carries the labor. People hold the merge and the release.

We ran our own edge work this way. Declaring the key exchange, pinning the proxy and adding the check were each ordinary reviewed changes through that pipeline. No special lane for cryptography.

What our system does now

The providers gave us hybrid. They did not give us proof that it stays that way. A default is a decision someone else made, and someone else can change it. So the system now declares, verifies and records it.

Declared. The key exchange our proxies offer is written in their configuration, hybrid first and classical after:

```
key exchange, in order of preference
X25519MLKEM768  hybrid: classical X25519 + ML-KEM-768
X25519           classical fallback
P-256           classical fallback
```

The proxy version is pinned too, so the next upgrade is a change someone reviews, not a default that arrives on its own.

Verified after every deploy, and every day. The same three-client test runs against every public endpoint after each release and once a day, including the endpoints our providers serve, where we can't declare anything and can only observe. It also checks that each certificate is valid and unexpired, and that no endpoint still accepts TLS 1.0 or 1.1. The list of endpoints is derived from the configuration itself, and a test fails if a new site appears that the check doesn't cover.

One record. Each result lands in the same control register as every other control, signed. A passing check is evidence that transmitted information is protected. A failing check, or a check that stops running, withdraws that evidence and raises a finding. Silence never reads as a pass.

What we learned about downgrades

We said to watch for downgrades. Doing it taught us which downgrade matters.

A client that can't speak hybrid falls back to classical. That's the fallback working as designed, and it's the exposure of whoever runs that client. The downgrade that belongs to us is the other one: our own edge quietly stopping offering hybrid, so every capable client falls back with it. That's the one the check catches, on every endpoint, after every change.

Seeing which key exchange each individual connection got is still worth having, and it isn't available everywhere yet. Our proxy's access logs record the protocol version and the cipher, not the key exchange, and neither do our CDN's. For the proxy, we proposed the field to the open-source project; its maintainers agreed the design, and it is being implemented. Our CDN's logs have no such field. Neither is on the critical path, because the downgrade that matters is already watched.

What this means for you

The 2025 advice stands. Governance leads, the inventory comes first, and hybrid is the path. What a year of running it adds:

- 1 **Test your edge today.** With OpenSSL 3.5 or later it takes one line per endpoint: `openssl s_client -connect your.site:443 -groups X25519MLKEM768`. A hybrid endpoint completes the handshake and reports the group. Run it again with `-groups X25519` to prove you haven't locked out classical clients. You probably have more hybrid than you think.
- 2 **Point the inventory at lifecycle, and make it a read of the estate.** What is behind, what can't be updated, what depends on a vendor with no roadmap. That list is your migration, and a hand-fed record can't produce it, because being behind is exactly what it misses.
- 3 **Put a current edge in front of what can't move yet.** A declared, pinned proxy gives those clients hybrid today, while the system behind it waits its turn in the lifecycle.
- 4 **Declare it, verify it on every change, and put the result where your other evidence lives,** including what you can't configure. A provider's default is still your exposure, and a regression should be a finding with an owner, not silence.

For everything that is current, the migration has already happened. For everything that isn't, the migration is lifecycle management. Either way, it is a posture you can prove, not a program you hope finishes.

What's next

Signatures. Where the trust root is ours, private certificates and code signing can move to ML-DSA now. Public certificates wait for the ecosystem.

The inside. Everything above is our public edge. Service-to-service traffic gets the same treatment: measure it, declare it, verify it.

And per-connection visibility, upstream, where it belongs.

We'll report back the same way. Measured, not promised.

About Xiaotime Labs

Xiaotime Labs builds ICRG, Integrated Cyber Risk Governance, and the AI-SDLC Pipeline: one platform where governance is expressed as code and enforced as the system runs. We work with regulated organizations whose engineering teams ship under scrutiny and whose buyers and regulators are beginning to ask for evidence, not attestation.

Xiaotime Labs was built by Stephan Hundley and Craig George, two former CISOs who have known each other for nearly fifteen years. They built the platform because the gap was real and the existing tools were not built for the speed at which engineering ships.

This report is the argument, and the proof is the running system. If it holds up against your environment, we should talk.

xiaotimelabs.ai