

PQ Authentication in Optical Transport Networks: Deployment Experience and Open Questions

ciaran.mullan, richard.zink@advasecurity.com

Adva Network Security GmbH

September 3, 2026

Talk Submission: IAB Workshop on Accelerating the Deployment of Post-Quantum Authentication

Adva Network Security GmbH develops high-assurance encryptors for carrier and critical networks, securing Layer-2 (IEEE 802.1AE MACsec) and OTN (ITU-T G.709). Products are long-lived (up to 20+ years), difficult to update remotely, and subject to national security certifications (e.g. BSI, FIPS), in regulated deployments. As part of **Adtran, Inc.**, we apply our cryptography expertise to carrier and critical-infrastructure networking.

Our PQC migration is well underway. Data-in-transit has been protected with hybrid key establishment (classical + PQC) since 2021 already including national security certification. The management plane (NET-CONF, SSH, HTTPS, etc.) can follow as updated protocol stacks emerge. Our current PQC migration efforts are focused on *authentication*.

We propose a talk covering the topics below, grounded in our real-world experience: what is working, the engineering decisions and trade-offs we made, the mistakes we have made (and may still make), and the open questions we are still dealing with.

PQ link authentication

Protocol design and formal analysis for link/peer authentication are complete and rollout is underway. We opted for KEM-based authentication.

Some engineering questions we faced along the way:

- How many authentication keys? The wire cost may be small. But two long-term static keys? Double enrollment ceremonies, with asynchronous algorithm-sunset paths? If a single static authentication key is preferred today, would it be classical or PQ? If ML-KEM is doing the key exchange, should a different KEM be used for authentication? Should KEM-based authentication combine two KEMs?
- Re-authentication cadence on a continuously rekeying link. Two extremes: re-authenticate at every rekey event ('static' shared secrets become ephemeral), or never re-authenticate (static shared secrets act more like PSKs). Unlike DH, there is no relationship between static KEM keypairs and static shared secrets. Is there a middle way?

Long-lived PQ identities

It's possible that our devices outlive the current generation of PQ signature algorithms. Knowing that, how should we provision a long-lived device with a cryptographic identity?

IEEE P802.1AReg amends IEEE Std 802.1AR (Secure Device Identity) to bind per-device identifiers (DevIDs) to ML-DSA. PQ-capable DevIDs are expected to become a US government requirement as early as January 2027 under the US Quantum Computing Cybersecurity Act.

This raises several engineering questions for long-lived devices. Here, *eFuses* are one-time-programmable hardware elements used to store device-specific keys or other root-of-trust data.

- What should be provisioned in the eFuses?
- Should certificate lifetimes be chosen to span the expected lifetime of the device, or should identities be designed for cryptographic rollover?
- Should a device eventually stop trusting a legacy (non-PQ) signature scheme?
- How should the root of trust support migration if the PQ signature scheme used for the initial DevID is itself eventually deprecated?

We can share our experiences with RSA-based identities and vision for future PQ/hybrid identities. Discussion is especially welcome on this topic as the decisions we make now will be deployed in the field throughout the 2030s and beyond.

PQ software authentication

This is probably familiar territory to the audience. Yet it has been by far our most time-consuming and error-prone part of the transition: HSM migration, new ML-DSA key-generation ceremonies, and new firmware and container layouts, all happening at once.

The first thing that broke was the container format. Our verifiers parse signed artefacts at fixed byte offsets with hard-coded lengths. Not only larger signatures and public keys, but also larger hash fields (SHA-256 $\rightarrow$ SHA-512).

Our new HSMs currently support ML-DSA only through vendor-defined mechanisms, which are incompatible with off-the-shelf tools such as OpenSSL. The vendor anchor remains classical, as does mutual authentication to the HSM, leading us to the basic question:

- Where, exactly, does the PQ boundary begin and end for code-signing?

And how much of the chain do we actually need to make post-quantum by when? For many of our devices, the boot ROM implements RSA only. For existing devices, PQ protection can therefore only be introduced above that immutable root, i.e. in the bootloader.