Chapter 7: Post-Quantum Cryptographic Readiness

Learning Objectives

By the end of this chapter, students will be able to:

1. Explain why current public-key cryptography (RSA, ECDH, ECDSA) is vulnerable to quantum attack and quantify the threat timeline.
2. Describe the NIST Post-Quantum Cryptography standards (ML-KEM, ML-DSA, SLH-DSA) and their security properties.
3. Implement a TLS handshake inspector that classifies cipher suites by quantum resistance.
4. Model Harvest-Now-Decrypt-Later (HNDL) risk using Mosca's inequality and Monte Carlo simulation.
5. Estimate Grover's oracle resource requirements for symmetric-key search on constrained hardware.
6. Evaluate side-channel risks of deploying post-quantum algorithms on ARM Cortex-M class IoT devices.
7. Generate prioritized migration plans with cost estimates for heterogeneous device fleets.

Agentic Lens

This chapter gives you the essential tools for cryptographic migration. You should approach the material as a navigator—whether you are a careful analyst using Breakwater, an automated decision-maker, or both. The main goal is to figure out which systems need migration first, handle uncertainty with clear analysis, and focus on taking action rather than getting stuck in debates about the future of quantum computing.

To put this approach into practice, follow these steps:

1. Inventory all assets and extract current cryptographic configurations using automated tools.

2. Evaluate each system's quantum vulnerability and estimate risk using models such as HNDL and Mosca's inequality.

3. Classify assets based on data sensitivity, system lifetime, hardware constraints, and cryptographic agility.

4. Prioritize migration actions by combining risk scores, feasibility, and operational impact, while explicitly documenting uncertainties and confidence intervals in threat timelines.

5. Propose migration plans with fallback options for assets where uncertainty or hardware limitations prevent immediate transition.

> **Caution:** The techniques described in this chapter have operational boundaries. Always verify assumptions against the specific deployment environment before relying on any automated output.

6. Review recommendations with human oversight to challenge overconfident scheduling and ensure that exceptions and edge cases are flagged for manual analysis.

7. Update plans iteratively as new information becomes available, embedding uncertainty management as an explicit part of the decision process.

- **Agent role:** Rank cryptographic migration actions across a diverse device fleet.

- **Observations:** Consider negotiated cipher suites, protocol versions, asset lifetimes, data retention periods, hardware constraints, and available transition options.
- **Tools:** Use TLS inspection, HNDL models, side-channel risk estimates, and migration cost planners.
- **State:** Track exposure queues, migration dependencies, uncertainty notes, and exceptions requiring analyst review.
- **Verifier:** Conduct feasibility and compatibility checks, and clearly separate worked examples from measured fleet evidence.
- **Guardrails:** Avoid presenting speculative timelines as certain, and do not recommend unsupported hardware migrations.
- **Remember, if you focus only on exact numbers and ignore uncertainty, your migration plans might look strong on paper but fail in real situations**.

Threat Model and Assumptions

Figure 7.22: Quantum Threat Model Matrix

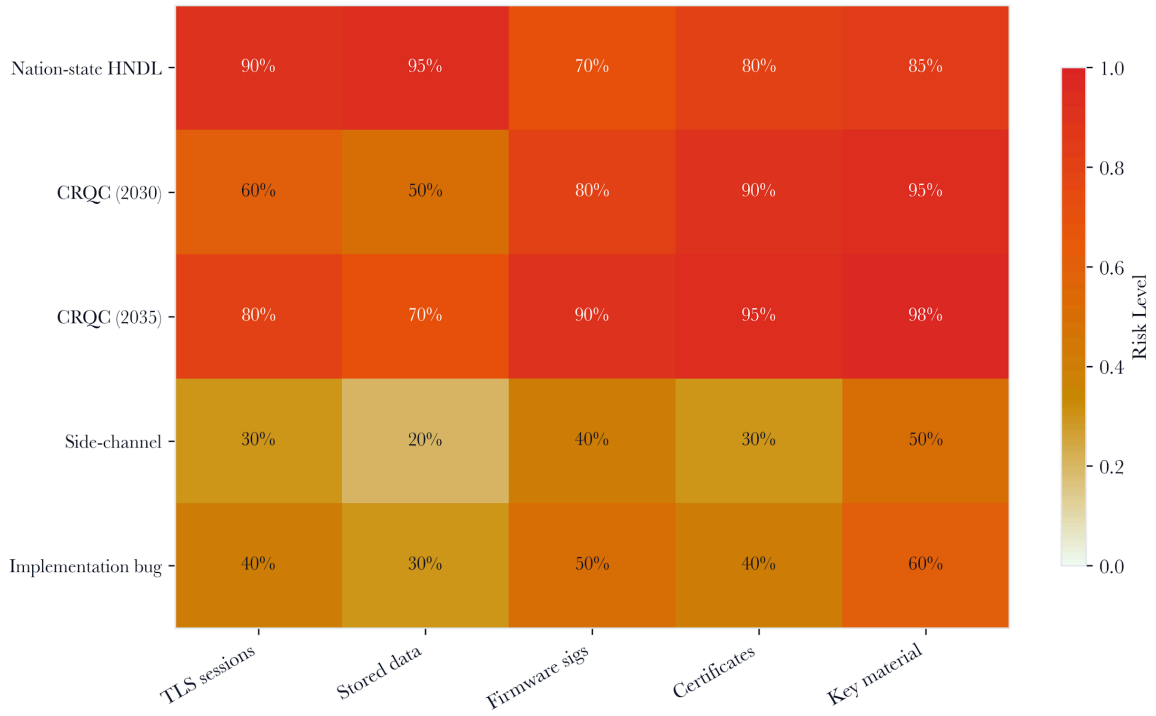

Figure 7.1: Threat model matrix. Chapter 7 separates cryptanalytic risk, HNDL exposure, side-channel deployment risk, and operational migration risk.

This chapter examines how cryptographic risks are evolving, showing how threats shift as technology and attackers become more advanced.

- **Threat model:** The primary concern is a patient adversary who collects encrypted traffic now and attempts to decrypt it when quantum capabilities improve.
- **Threat model.** A second concern is operational: rushed migration can create outage risk, side-channel regressions, or unusable devices if hardware constraints are ignored.

- **Assumptions:**
- **Inputs to the risk assessment include the current** protocol choices, expected device lifetimes, and possible migration options. The chapter does not assume access to precise forecasts for the arrival date of cryptographically relevant quantum computers (CRQC).
- The HNDL (Harvest-Now-Decrypt-Later) models prioritize which assets require migration by analyzing the relationships among data sensitivity, migration time, and potential quantum breakthroughs. Monte Carlo simulation timelines provide confidence intervals for CRQC arrival, supporting scenario-based planning rather than exact predictions. Side-channel risk estimates identify potential implementation vulnerabilities on constrained hardware, informing which migration strategies are feasible per device. All these models guide the prioritization of migration actions, but none of them predict a specific date for a quantum threat.
- **Assumption:** Some device fleets will remain hybrid for an extended period. The chapter evaluates cryptographic agility and transition feasibility, not only end-state solutions.
- **Scope note:** This chapter does not assume that adopting post-quantum cryptography is universally safe or urgent. Each decision should be based on specific data and mission requirements.

7.1 Introduction: The Cryptographic Cliff

Figure 7.1: Quantum-Readiness Analytics Taxonomy

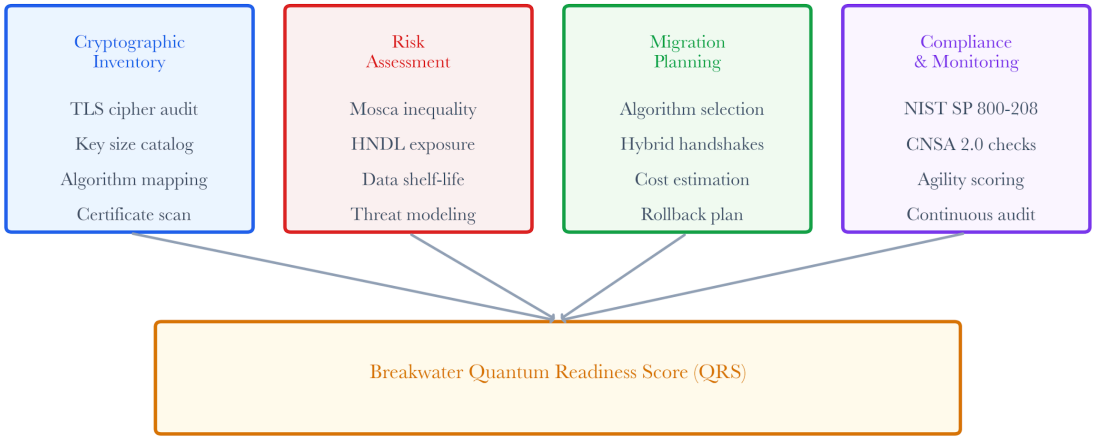

Figure 7.2: Analytics taxonomy for post-quantum readiness. Chapter 7 adds cryptographic exposure as a forward-looking risk signal that depends on present-day protocol evidence and uncertain future quantum capability.

All devices discussed in Chapters 1 through 6 use cryptographic algorithms to secure communications. TLS handshakes negotiate cipher suites, SSH sessions exchange keys, and VPN tunnels authenticate endpoints. These communications rely on two mathematical assumptions: the difficulty of factoring large integers (RSA) and the difficulty of computing discrete logarithms on elliptic curves. Although these assumptions have ensured security for decades, they are not expected to remain effective indefinitely.

Peter Shor demonstrated in 1994 that a sufficiently large quantum computer can factor an n-bit integer in O(n^3) time, reducing RSA-2048 from an estimated 2^112 classical operations to approximately 2^33 quantum

operations. The same algorithm applies to the elliptic curve discrete logarithm problem, rendering ECDH and ECDSA equally vulnerable. Lov Grover showed in 1996 that quantum search reduces symmetric key search from O(2^n) to O(2^(n/2)), effectively halving the security level of AES. The critical consideration is not only whether these algorithms will eventually be compromised, but also when organizations will experience the impact and whether adversaries are currently collecting encrypted traffic in anticipation of future quantum capabilities. This scenario is known as the Harvest-Now-Decrypt-Later (HNDL) threat: attackers may capture sensitive data today, store it at minimal cost, and decrypt it once quantum computers become available. For information requiring long-term confidentiality, such as medical records, military plans, or proprietary blueprints, this risk is already present.

Chapter 7 provides a step-by-step guide to assessing cryptographic security using the Breakwater platform. It analyzes each TLS handshake and sorts device cryptography as quantum-safe, quantum-vulnerable, or transitional. HNDL risk scores are calculated, and migration plans with cost estimates are made for each device. This approach turns quantum risk analysis into practical actions. While this is an instructional example, pilot studies show that this workflow helps find outdated cipher suites and supports efficient gateway proxy deployment for IoT devices. You are encouraged to run targeted scans, test migrations in safe environments, and connect your results with asset management systems to see real improvements.

Breakwater works best when its insights are built into your organization’s security processes. You can export quantum readiness and HNDL risk reports to SIEM systems, enabling automated alerts for devices at risk. Migration recommendations can be turned into IT tickets or incident response tasks, so you can track fixes until they are completed. Quantum status and migration progress appear on risk dashboards, giving analysts a clear view and supporting audits. By connecting Breakwater to ticketing systems, dashboards, and incident workflows, organizations can turn quantum risk insights into timely, effective actions.

Operational integration for Breakwater outputs can increase the utility and adoption of cryptographic readiness assessments within enterprise environments. A typical workflow for integrating Breakwater findings with existing risk dashboards and asset management systems includes the following steps:

1. Export Breakwater reports in a standard format (CSV, JSON, or via API) containing per-device quantum readiness scores, HNDL exposure, and migration recommendations.

2. Map device identifiers from Breakwater to asset records in the organization's asset management database.

3. Update risk classification fields within the asset management platform to include quantum exposure scores and migration status.

4. Surface prioritized migration actions and open vulnerabilities on the central risk dashboard, using tags such as "Quantum-Vulnerable," "Quantum-Safe," or "Transitional."

5. Trigger follow-up actions, such as automated ticket creation for high-risk devices or scheduling maintenance windows for prioritized updates.

6. Review and adjust migration progress through regular dashboard updates and coordination with IT and security teams.

Example checklist for integration:

- Validate device identifiers for mapping across systems
- Ensure secure API access between Breakwater and asset platforms
- Confirm alignment between migration recommendations and operational policies
- Schedule regular synchronization intervals for continuous risk posture updates
- Document each integration step. Following this process helps security teams bring quantum risk insights into daily work, respond quickly, and make sure all fixes are tracked and prioritized.

"47 devices still negotiate RSA-2048 key exchange. Estimated HNDL exposure: 12 TB of traffic with a 15-year sensitivity horizon. Estimated migration cost to ML-KEM: $23,400 across 3 firmware update cycles."

7.2 Quantum Computing Foundations

Evidence label: Literature-derived. Method note: The qubit counts, circuit costs, and CRQC timeline estimates here are taken from cited papers and public roadmaps, not from measurements in this repository. Assumptions: Error-correction overhead, hardware progress, and attack cost estimates are still uncertain and could change significantly. Boundary: Use these numbers for planning, not as firm dates.

7.2.1 Shor's Algorithm and Public-Key Cryptography

Figure 7.2: Shor's Algorithm Circuit Diagram

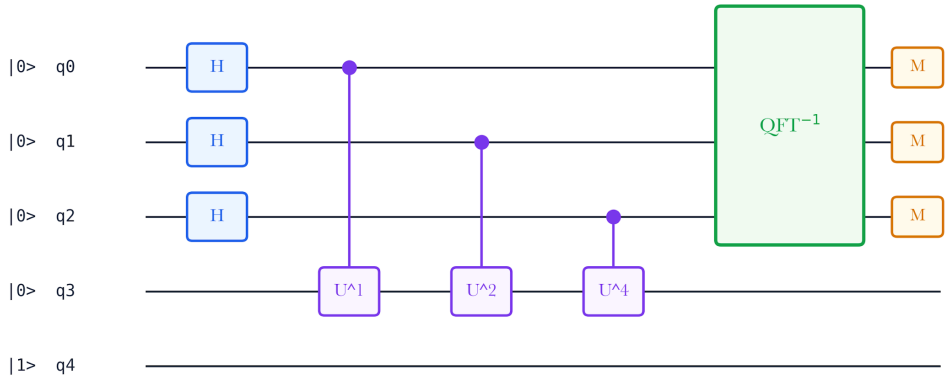

Figure 7.3: Short circuit structure. Period finding turns RSA and elliptic-curve assumptions into a quantum circuit problem once enough fault-tolerant logical qubits are available.

Shor's algorithm reduces integer factorization to period-finding via the Quantum Fourier Transform. For RSA-N, the resource estimate is approximately 2N+3 logical qubits (Beauregard, 2003), translating to roughly 4,000 logical qubits for RSA-2048. With the current error-correction overhead (~5,000:1 physical-to-logical qubit ratio), this requires approximately 20 million physical qubits. Current quantum hardware operates with ~1,000 noisy physical qubits, but IBM, Google, and others project to have millions of physical qubits by the mid-2030s.

The same algorithm applies to the elliptic curve discrete logarithm problem. Roetteler et al.(2017) estimate that breaking P-256 (the curve used in most ECDH deployments) requires approximately 2,330 logical qubits – fewer than RSA-2048.

7.2.2 Grover's Algorithm and Symmetric Ciphers

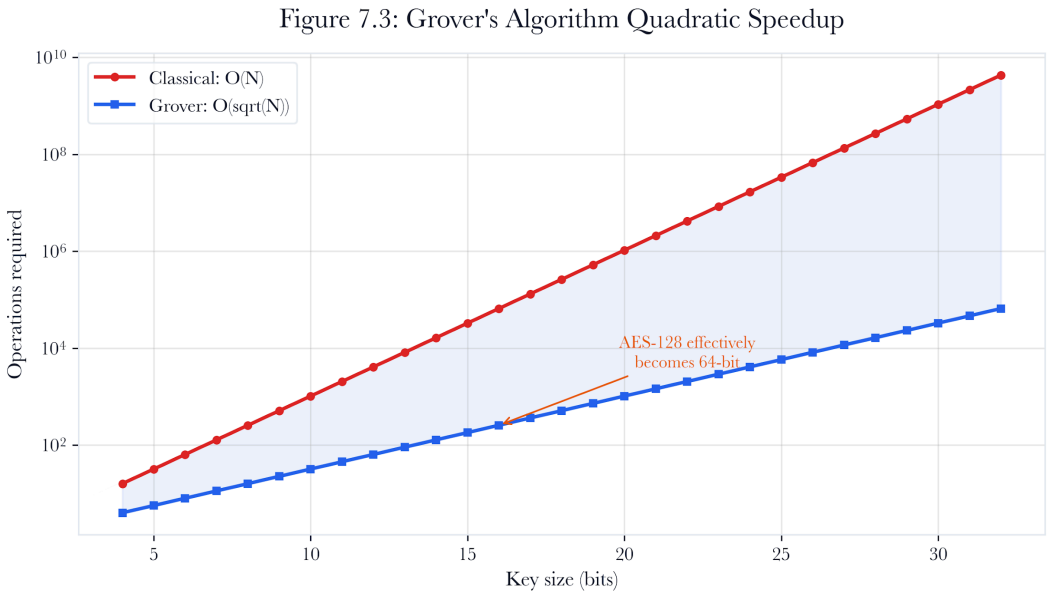

Figure 7.4: Grover speedup. Symmetric-key search receives a quadratic quantum advantage, reducing the effective security of n-bit keys to approximately n/2 bits.

Grover's algorithm provides a quadratic speedup for unstructured search, reducing the effective security of an n-bit symmetric key to n/2 bits. For AES-128, this means an effective security level of 64 bits – below the NIST minimum of 128 bits for long-term security. AES-256 retains 128-bit security post-Grover and remains quantum-safe.

The Breakwater implementation quantifies this precisely in grover_oracle.py:

```
# From apps/api/app/scanning/quantum/grover_oracle.py
QUBIT_REQUIREMENTS: dict[str, dict[str, Any]] = {
"AES-128": {
"attack": "grover",
"logical_qubits": 2953,
"circuit_depth": 2**64,
"oracle_calls": 2**64,
"quantum_security_bits": 64,
"reference": "Grassl et al. 2016 / NIST IR 8309",
},
"AES-256": {
"attack": "grover",
"logical_qubits": 6681,
"circuit_depth": 2**128,
"oracle_calls": 2**128,
"quantum_security_bits": 128,
"reference": "Grassl et al. 2016",
},
}
```

The qubit estimates come from Grassl et al. (2016), who calculated the exact circuit cost for implementing the AES S-box as a quantum oracle.

7.2.3 NIST Post-Quantum Standards

Figure 7.20: Classical vs Post-Quantum Key Sizes

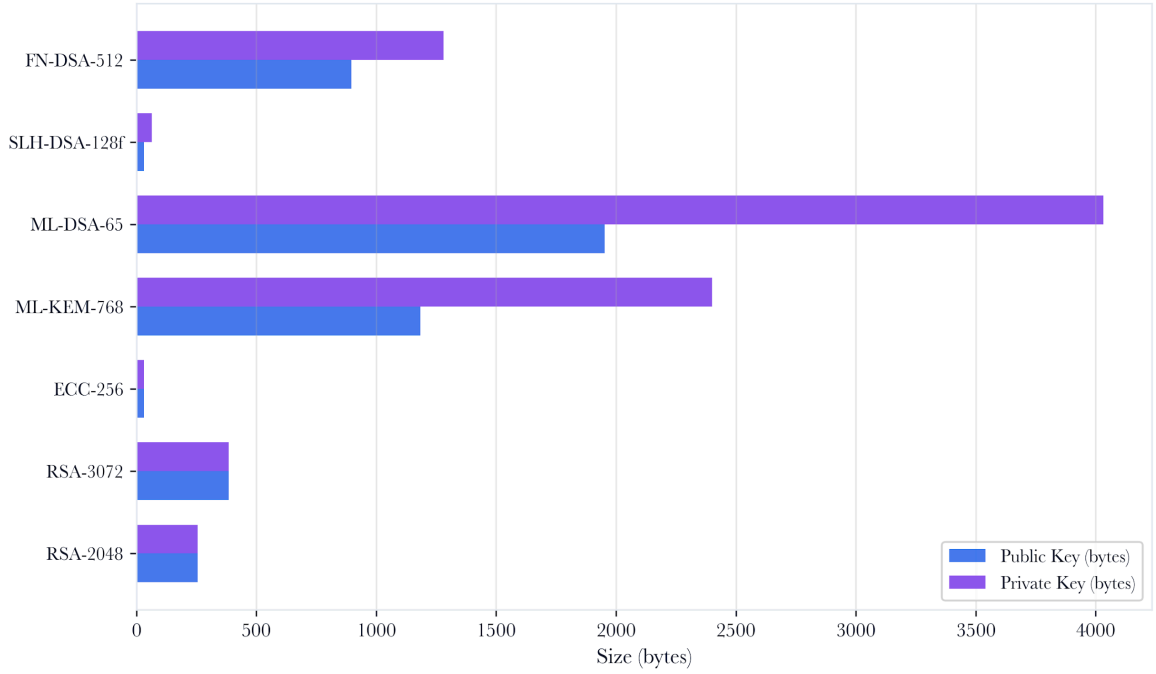

Figure 7.5: Key-size comparison. Post-quantum algorithms change bandwidth, certificate, and firmware constraints, especially on IoT and OT devices with narrow update budgets.

Figure 7.4: NIST PQC Algorithm Comparison (Security Level 3)

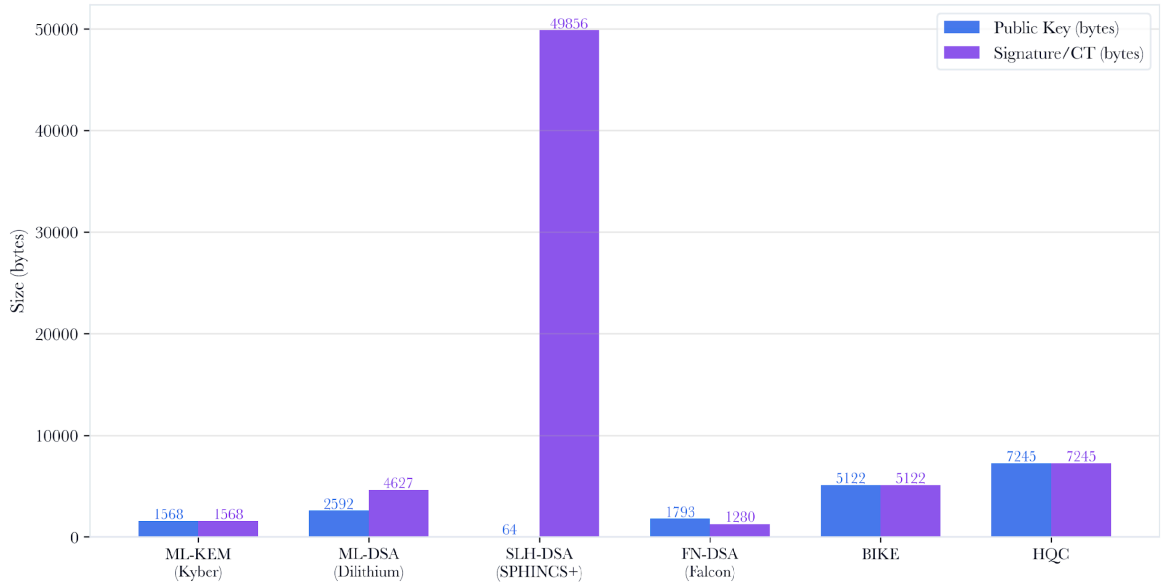

Figure 7.6: NIST post-quantum standards comparison. ML-KEM, ML-DSA, and SLH-DSA solve different key-establishment and signature-migration problems with varying sizes and performance trade-offs.

Chapter 7: Post-Quantum Cryptographic Readiness

In August 2024, NIST published the first three post-quantum cryptography standards:

ML-KEM (FIPS 203)	KEM	Key encapsulation	CRYSTALS-Kyber	800-1568 B	N/A
ML-DSA (FIPS 204)	Signature	Digital signature	CRYSTALS-Dilithium	1312-2592 B	2420-4627 B
SLH-DSA (FIPS 205)	Signature	Hash-based signature	SPHINCS+	32-64 B	7856-49856 B

ML-KEM is the primary replacement for ECDH/RSA key exchange. ML-DSA replaces ECDSA/RSA signatures. SLH-DSA provides a conservative hash-based alternative with minimal assumptions.

Figure 7.9: ML-DSA Digital Signature Flow

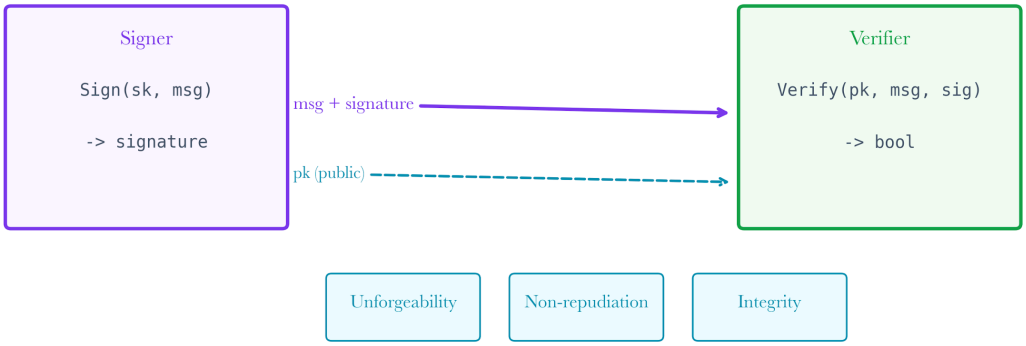

Figure 7.7: Digital signature flow. Post-quantum signatures protect authentication and software trust chains, not just encrypted transport confidentiality.

Figure 7.8: ML-KEM Key Encapsulation Flow

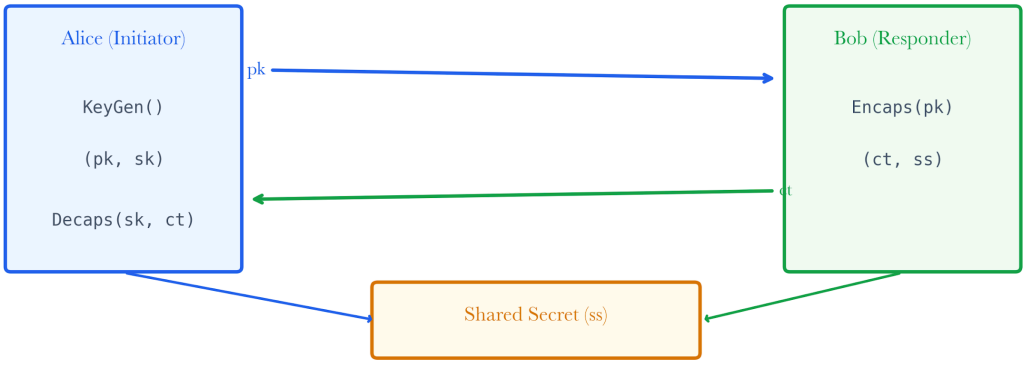

Figure 7.8: Key encapsulation flow. ML-KEM replaces classical key exchange by encapsulating a shared secret in the recipient's public key and decapsulating it with the recipient's private key.

7.3 Architecture: Quantum Readiness Assessment Pipeline

```
graph TD
A[Phase 1-6 Scan Results] --> B[TLS Deep Inspector]
B --> C[Crypto Classifier]
C --> D[HNDL Risk Modeler]
D --> E[Monte Carlo Simulator]
C --> F[Cryptographic Agility Prober]
C --> G[Side-Channel Estimator]
F --> H[Migration Planner]
D --> H
G --> H
H --> I[HYDRA Stream Q]

style B fill:#e1f5fe
style C fill:#e8f5e9
style D fill:#fff3e0
style E fill:#fff3e0
style F fill:#f3e5f5
style G fill:#fce4ec
style H fill:#e0f2f1
style I fill:#ffebee
```

The pipeline uses seven engines, each running for every host:

1. **TLS Deep Inspector** (crypto_scanner.py) – active TLS handshake analysis
2. **Crypto Classifier** (pq_classifier.py) – quantum vulnerability scoring
3. **HNDL Risk Modeler** (hndl_engine.py) – harvest-now-decrypt-later risk
4. **Monte Carlo Simulator** (threat_simulator.py) – probabilistic HNDL confidence intervals
5. **Cryptographic Agility Prober** (agility_scanner.py) – PQ negotiation testing
6. **Side-Channel Estimator** (side_channel_estimator.py) – implementation risk on IoT

Figure 7.9: Side-channel attack surface. PQ migration can introduce new leakage paths on constrained hardware, so algorithm choice and implementation quality must be assessed together.

7. **Migration Planner** (migration_planner.py) – prioritized remediation

7.4 TLS Deep Inspector

Figure 7.7: TLS Cipher Suite Classification

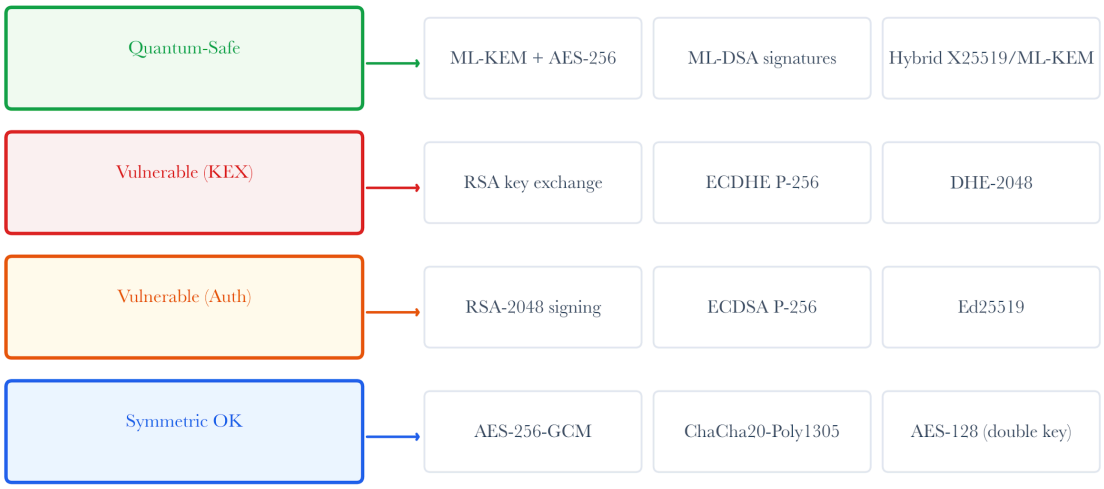

Figure 7.10: TLS cipher classification. The scanner separates quantum-vulnerable, transitional, and quantum-safe negotiation evidence before computing per-device readiness.

The crypto_scanner.py module connects to each host via Python's ssl module to extract the negotiated cipher suite, TLS version, certificate details, and supported cipher list.

```
# From apps/api/app/scanning/quantum/crypto_scanner.py
_TLS_PORTS: list[int] = [443, 8443, 8883, 993, 995, 465, 636, 5671]

def extract_cipher_info(ssl_info: dict[str, Any]) ->list[CipherAssessment]:
"Parse SSL connection info into a list of CipherAssessment.

ssl_info contains keys from ssl.SSLSocket: 'cipher' (negotiated)
Moreover, 'shared_ciphers' (all supported).
"""
assessments: list[CipherAssessment] = []
negotiated = ssl_info.get("cipher")
if negotiated and isinstance(negotiated, (tuple, list)):
name = str(negotiated[0])
catalog = classify_cipher(name)
assessments.append(CipherAssessment(
cipher_name=name,
key_exchange=catalog.get("kex", ""),
authentication=catalog.get("auth", ""),
quantum_safe=catalog.get("quantum_safe", False),
vulnerability_rating=catalog.get("vulnerability_rating", "unknown"),
))
return assessments
```

The inspector probes all 8 standard TLS ports on each host. The cipher_catalog.py module maps IANA cipher suite names to their component algorithms and quantum safety classification.

7.5 Quantum Vulnerability Scoring

The pq_classifier.py module computes a composite quantum readiness score (0-100) per device using five weighted factors:

```
# From apps/api/app/scanning/quantum/pq_classifier.py
_WEIGHT_KEX =0.40# Key exchange (most impactful for HNDL)
_WEIGHT_SIG =0.20# Signature / authentication algorithm
_WEIGHT_CERT =0.20# Certificate key type and size
_WEIGHT_PROTOCOL =0.10# TLS protocol version
_WEIGHT_CIPHER_MODE =0.10# Bulk cipher mode (AEAD preferred)
```

Algorithm classification uses three sets:

```
QUANTUM_SAFE_KEXES: set[str] = {
"ML-KEM-512", "ML-KEM-768", "ML-KEM-1024",
"X25519Kyber768", "X25519MLKEM768",
"SNTRUP761", "SNTRUP761X25519",
"FRODOKEM-640", "FRODOKEM-976", "FRODOKEM-1344",
"BIKE-L1", "BIKE-L3", "BIKE-L5",
"HQC-128", "HQC-192", "HQC-256",
}

QUANTUM_VULNERABLE_KEXES: set[str] = {
"RSA", "DHE", "ECDHE", "DH", "ECDH",
"X25519", "X448", "SECP256R1", "SECP384R1",
...
}

TRANSITIONAL_KEXES: set[str] = {
"TLS13",
"ECDHE+ML-KEM-768",# hybrid draft
}
```

Key exchange gets the highest weight (40%) because it has the biggest impact on HNDL risk. If an attacker can break the key exchange, they can obtain the session key and decrypt all previously stored traffic.

7.6 The HNDL Threat Model

Figure 7.6: Harvest-Now-Decrypt-Later Risk Matrix

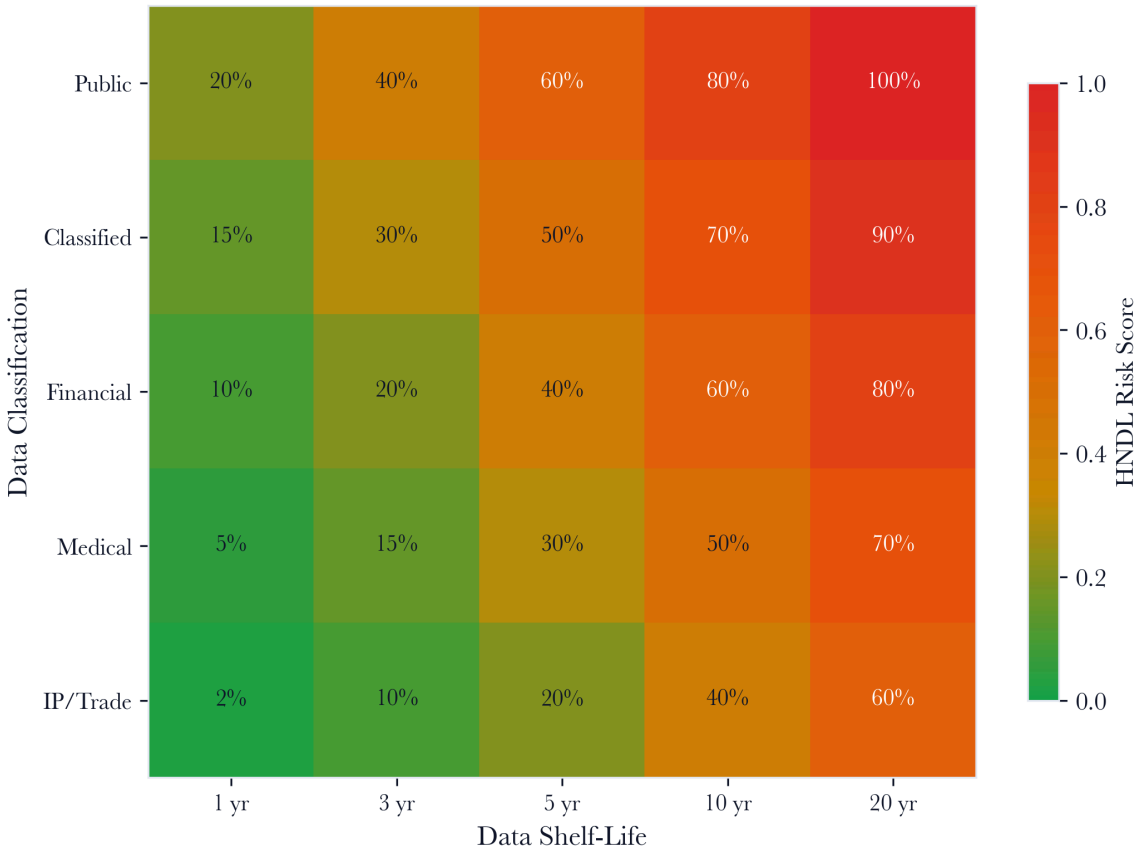

Figure 7.11: HNDL risk matrix. Harvest-now-decrypt-later exposure depends on both cryptographic weakness and the sensitivity horizon of captured data.

7.6.1 Mosca's Inequality

Figure 7.5: Mosca's Inequality Timeline

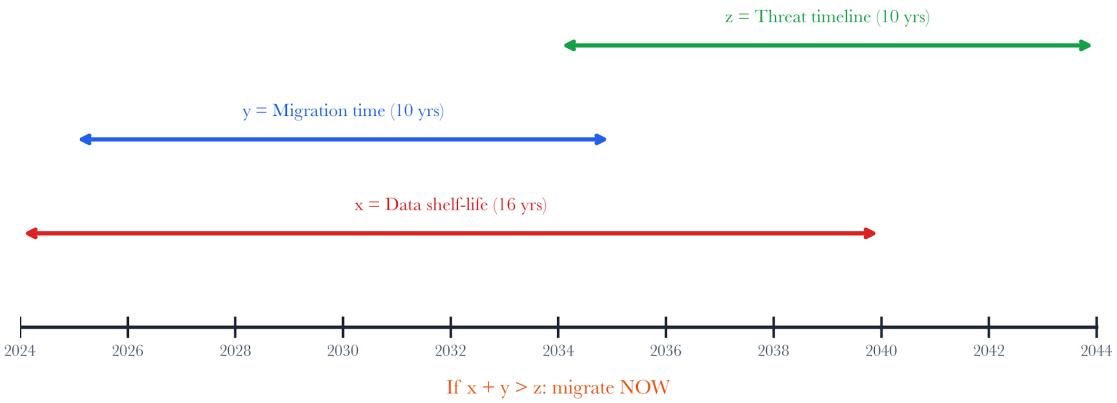

Figure 7.12: Mosca inequality timeline. Migration urgency rises when the sum of data lifetime and migration time exceeds the time remaining before the relevant quantum capability.

Michele Mosca formalized the HNDL risk in a simple inequality:
If **X + Y > Z**, the data is at risk.

Where: - **X** = shelf life of the data (how long it remains sensitive) - **Y** = migration time (how long it takes to deploy PQ cryptography) - **Z** = years until a CRQC becomes available

If data will still be sensitive when quantum computers become available, and migration cannot finish before then, the data is already at risk from HNDL attacks.

7.6.2 Timeline Models

Figure 7.21: Cryptographic Algorithm Timeline

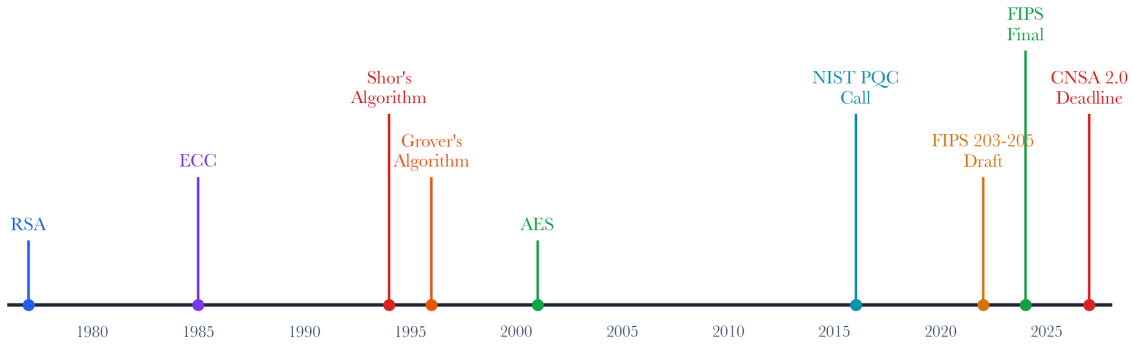

Figure 7.13: Algorithm timeline. Classical, transitional, hybrid, and post-quantum states coexist for years, so the planner treats migration as staged risk reduction rather than a single cutover.

The hndl_engine.py module implements three CRQC timeline models:

```
# From apps/api/app/scanning/quantum/hndl_engine.py
QUANTUM_TIMELINE_MODELS: dict[str, dict[str, Any]] = {
"pessimistic": {
"crqc_year": 2040,
"confidence": 0.30,
"label": "Assumes significant hardware challenges delay CRQC until 2040",
},
"moderate": {
"crqc_year": 2033,
"confidence": 0.50,
"label": "Mainstream projection; CRQC around 2033",
},
"optimistic": {
"crqc_year": 2029,
"confidence": 0.20,
"label": "Aggressive state-actor timeline; CRQC as early as 2029",
},
}
```

7.6.3 Data Sensitivity Classification

Each data flow is classified by sensitivity level, which determines the HNDL risk score. The HNDL risk score uses Mosca's inequality, a model of how storage costs drop by about 20% each year, and an estimate of the attacker's return on investment. calculation.

7.7 Grover Oracle Resource Estimation

Figure 7.14: Quantum Resource Estimates for Breaking Classical Crypto

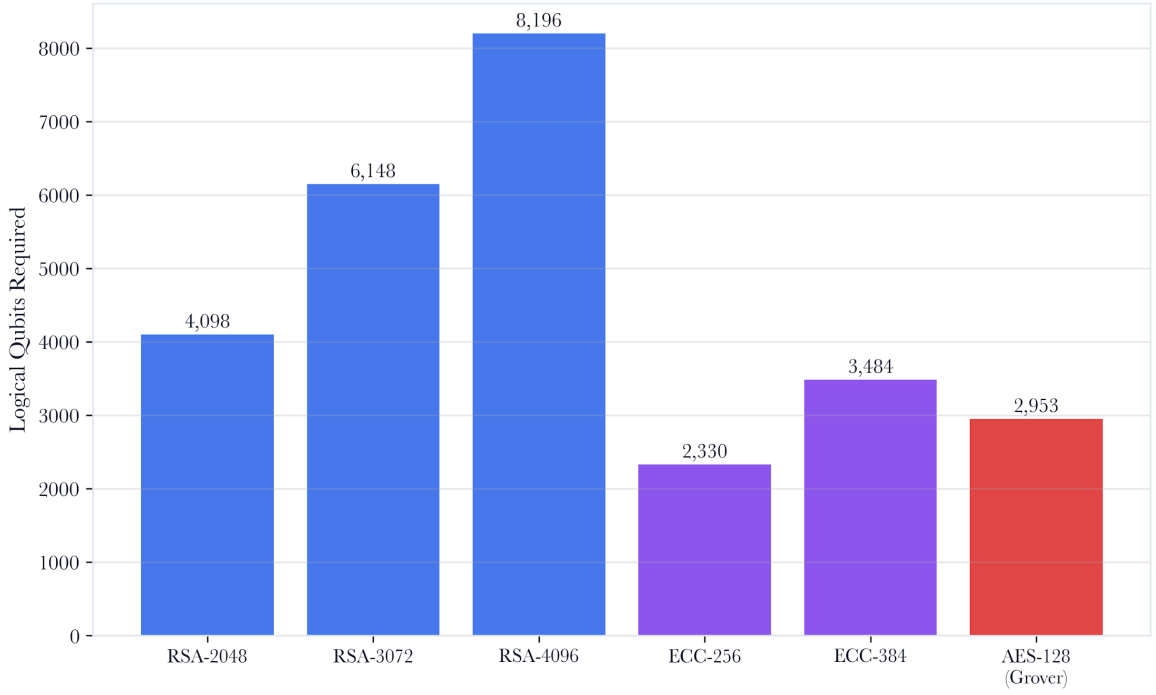

Figure 7.14: Quantum resource estimates. Logical qubits, circuit depth, and oracle calls translate abstract quantum attacks into bounded planning inputs.

The grover_oracle.py module goes beyond the standard "Grover halves the key length" heuristic to provide precise quantum resource estimates per cipher:

```
graph LR
A[Cipher Suite] --> B{Asymmetric?}
B -->|Yes| C[Shor Resource Estimate]
B -->|No| D[Grover Resource Estimate]
C --> E[Logical Qubits + T-gate depth]
D --> E
E --> F[Years Until Threat]
F --> G[Qubit Roadmap Projection]
```

For each cipher, the module computes: - Logical qubits required – based on published circuit analyses - Circuit depth – the number of quantum operations (T-gates) - Oracle calls – the number of Grover iterations needed - Quantum security bits – the effective security level post-quantum - Years until threat – projected using IBM/Google qubit scaling roadmaps.

Figure 7.19: Grover's Oracle and Diffusion Operator

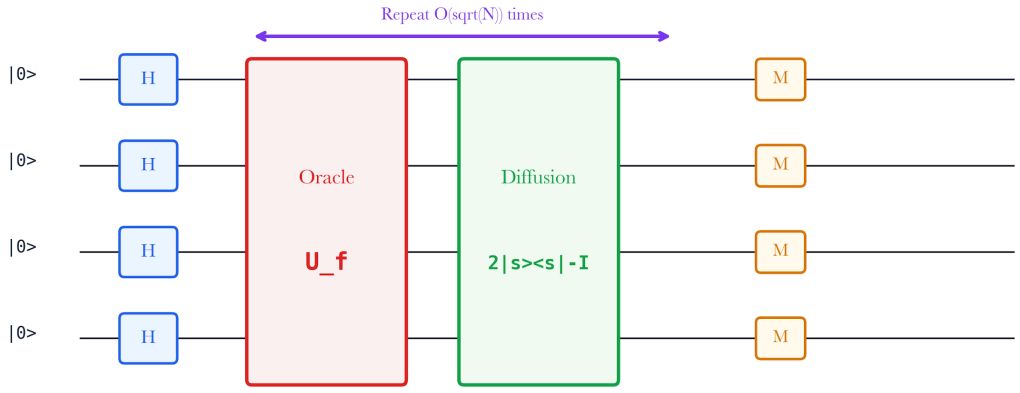

Figure 7.15: Grover oracle circuit. The oracle-cost model exposes why symmetric cryptography is weakened by quantum search but not broken in the same way as RSA or ECDH.

For example, AES-128 needs 2,953 logical qubits and a circuit depth of 2^64. Even with expected progress in quantum hardware, reaching this circuit depth is unlikely before 2040. Since its effective security is 64 bits, well below the NIST standard, AES-128 is no longer considered strong. That is why AES-256 is now the better choice for quantum resistance.

One important but often overlooked aspect of PQ migration is the security of PQ algorithms when running on limited hardware. The side_channel_estimator.py module links device CPU types to known side-channel weaknesses:

```
# From apps/api/app/scanning/quantum/side_channel_estimator.py
ARCHITECTURE_PROFILES: dict[str, dict[str, Any]] = {
"arm-m4": {
"label": "ARM Cortex-M4",
"leakage_risk_base": 0.70,
"known_weaknesses": [
"DPA on NTT butterfly operations",
"EMFI fault injection on comparison operations",
"Timing leakage in polynomial arithmetic",
"Cache timing attacks on SHA-3/SHAKE implementations",
],
"typical_devices": ["IoT sensors", "wearables", "embedded controllers"],
},
"arm-m33": {
"label": "ARM Cortex-M33 (TrustZone)",
"leakage_risk_base": 0.45,
...
},
"arm-a53": {
"label": "ARM Cortex-A53",
"leakage_risk_base": 0.30,
...
},
}
```

ML-KEM and ML-DSA rely heavily on the Number Theoretic Transform (NTT), which involves butterfly operations whose power consumption is operand-dependent on platforms without hardware masking. Ravi et al. (2023) demonstrated single-trace attacks against CRYSTALS-Dilithium running on a Cortex-M4. Primas et al.(2024) showed that even masked lattice-based implementations leak through higher-order statistical analysis.

Figure 7.13: Lattice vs Hash-Based Tradeoff Space

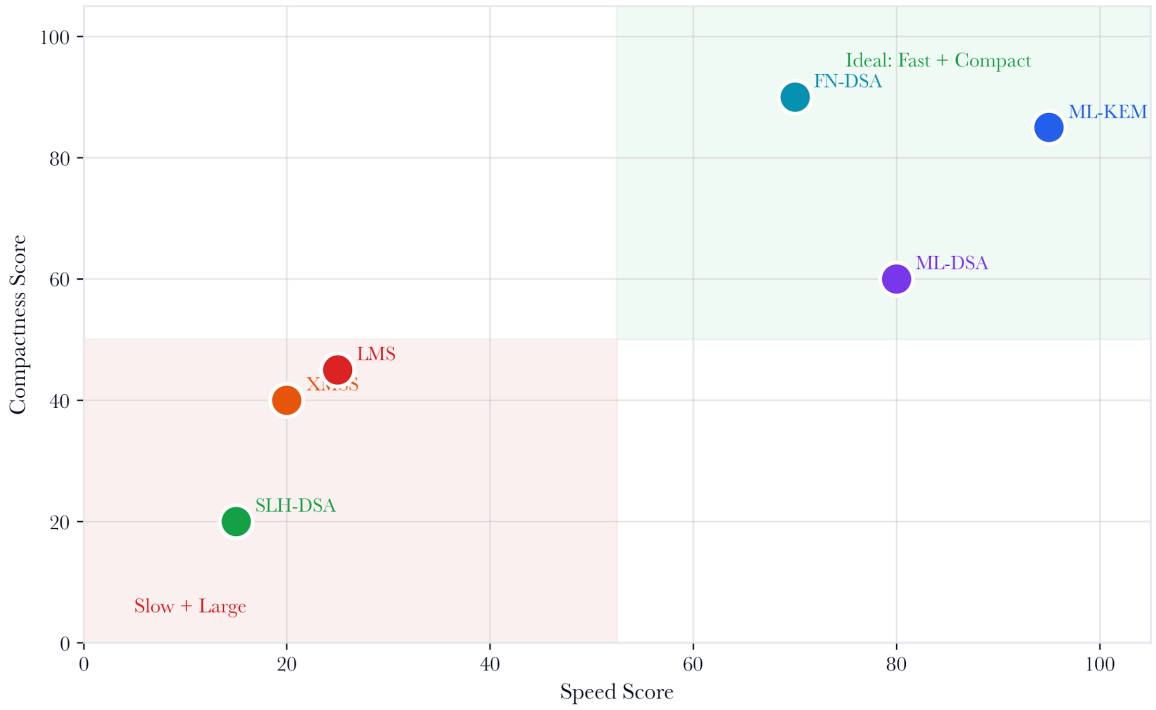

Figure 7.16: Lattice versus hash tradeoff. Lattice schemes usually provide better practical performance, while hash-based signatures offer conservative assumptions at larger signature sizes.

The side-channel estimator delivers a risk score that guides the migration planner in recommending hardware upgrades and smarter algorithm choices for the most vulnerable devices.

7.9 Migration Planning

Figure 7.10: PQ Migration Roadmap

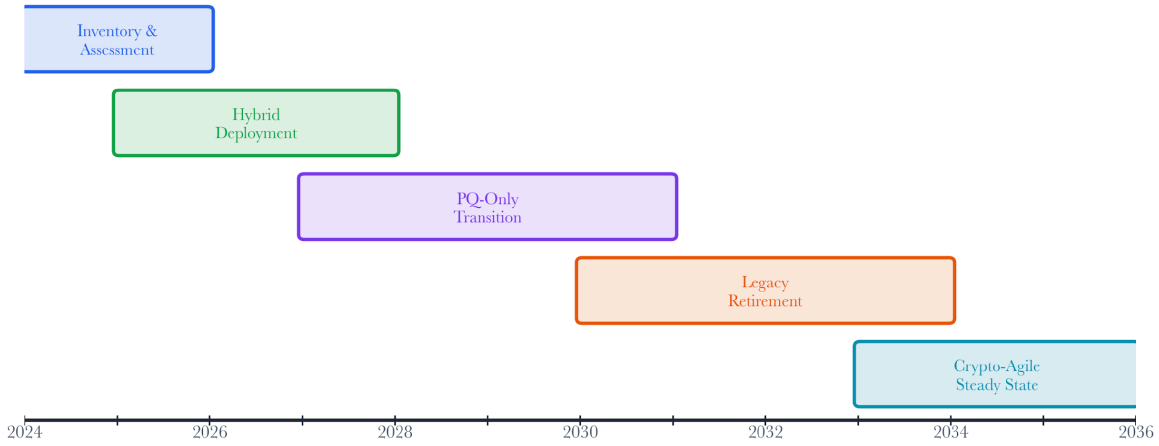

Figure 7.17: Migration roadmap. Devices move through inventory, exposure ranking, compatibility testing, compensating controls, firmware updates, and certificate rotation.

The migration_planner.py module generates per-device migration actions with cost estimates:

```
# From apps/api/app/scanning/quantum/migration_planner.py
DEVICE_MIGRATION_STRATEGIES: dict[str, list[str]] = {
"server": ["firmware_update", "cert_rotation"],
"router": ["firmware_update", "vpn_overlay"],
"camera": ["vpn_overlay", "gateway_proxy"],
"iot_sensor": ["gateway_proxy", "vpn_overlay"],
"plc": ["gateway_proxy", "vpn_overlay"],
...
}

MIGRATION_COST_DB: dict[str, dict[str, float]] = {
```

Figure 7.17: Migration Cost Curve: Proactive vs Reactive

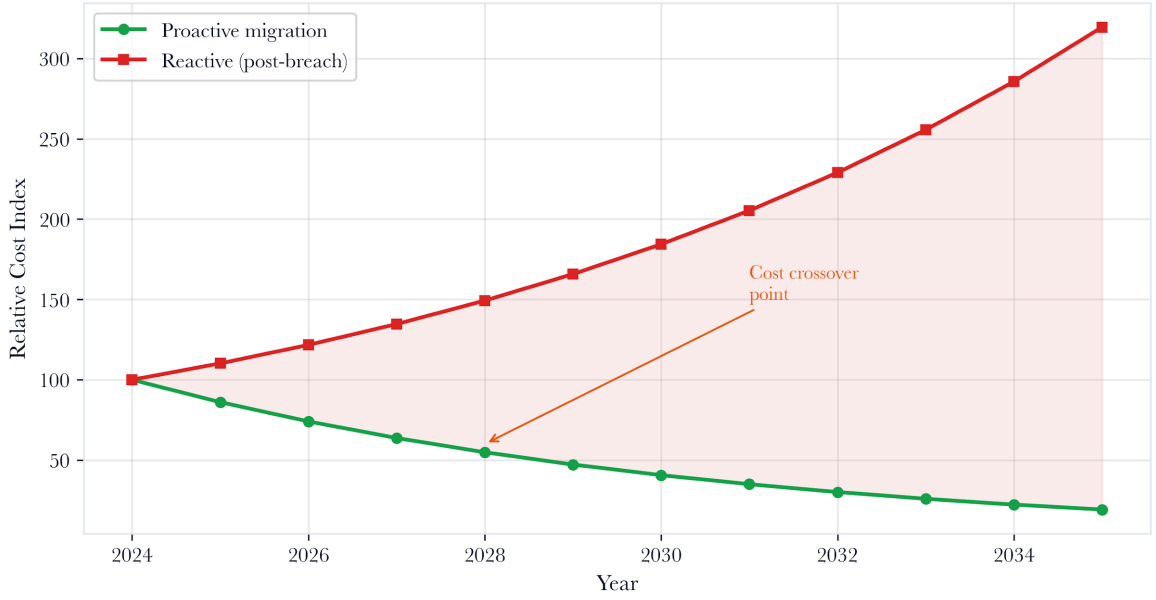

Figure 7.18: Migration cost curve. Direct firmware updates, overlays, gateways, and replacement programs have different cost profiles and residual-risk outcomes.

```
"firmware_update": {"total_per_device": 520.0},
"vpn_overlay": {"total_per_device": 430.0},
"gateway_proxy": {"total_per_device": 990.0},
"cert_rotation": {"total_per_device": 270.0},
}
```

A significant consideration is that many IoT devices, such as cameras, sensors, and programmable logic controllers (PLCs), cannot be upgraded to support ML-KEM. For these devices, the migration planner recommends using gateway proxies for post-quantum TLS or VPN overlays to route traffic through quantum-secured tunnels. Gateway proxies enable legacy devices to achieve quantum-safe communication, but may introduce latency and require careful placement and failover strategies. VPN overlays require minimal changes to devices but can increase network complexity and introduce additional encryption overhead. Each approach presents distinct security trade-offs: proxies may become single points of failure if not properly secured, while overlays protect all traffic but necessitate robust key and tunnel management. Migration steps are prioritized by HNDL risk and grouped by vendor or firmware to enhance efficiency. The planner organizes migration into three phases: immediate (addressing urgent HNDL risk), short-term (1–2 years), and long-term (3–5 years).

Vendor coordination is a crucial part of the migration process, especially for devices that cannot be directly updated. Analysts should follow structured steps to influence and expedite vendor support for post-quantum cryptography:

Figure 7.11: Crypto-Agility Layer Architecture

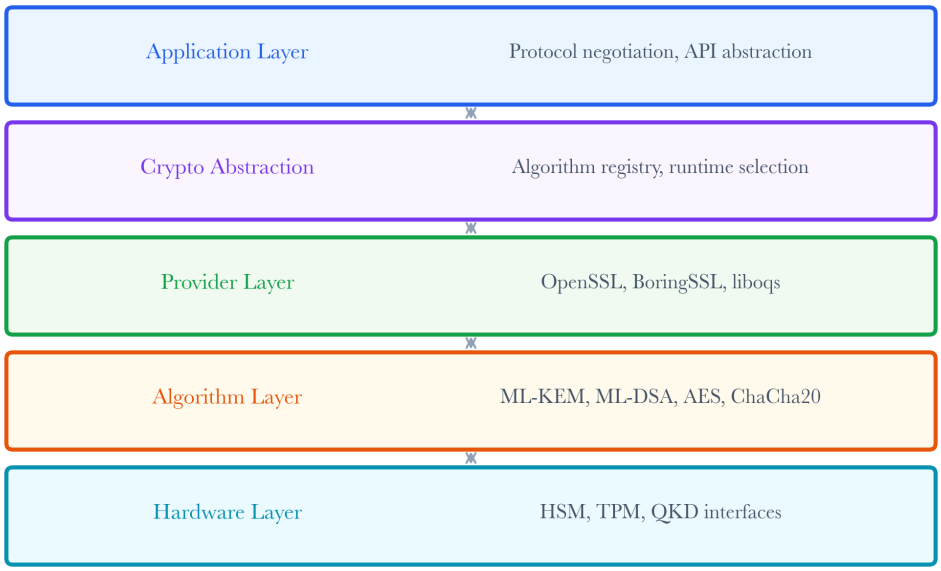

Figure 7.19: Cryptographic agility layers. Protocol support, firmware update paths, certificate management, and vendor commitments all determine whether migration is operationally feasible.

Example checklist for engaging vendors:

- Identify the designated technical and account contacts for each vendor and establish regular communication channels with them.
- Prepare specific questions, such as:
    - • What is your current timeline for supporting post-quantum cryptography (PQC) in firmware or product lines X, Y, Z?
    - • Do you have a roadmap or published update plan for PQC support?
    - • What are the minimum hardware or software requirements to receive PQC updates?
    - • Will older device models be eligible for firmware upgrades, replacement programs, or trade-ins targeting PQC readiness?
    - • Are existing side-channel protections evaluated for new PQ algorithms?
- Escalate through formal support tickets and, if progress stalls, request direct escalation to product management or security leads.
- Document all communications, requests, and vendor responses, maintaining a record for auditing and planning.
- Where possible, include specific PQC requirements as contractual language in new purchase agreements or renewals (for example: "Vendor shall provide PQC-capable firmware updates no later than 12 months after NIST standard publication").
- Monitor vendor progress and align organizational migration plans with announced vendor timelines, updating internal migration schedules as appropriate.

When devices cannot be updated, establishing strong vendor partnerships becomes essential. Teams should proactively engage vendors for post-quantum cryptography support, review firmware update roadmaps, and negotiate custom solutions as necessary. Maintaining regular communication can encourage vendors to prioritize organizational requirements, potentially providing early access to PQC-ready firmware or trade-in programs. It is important to document all communications and commitments to ensure migration timelines remain aligned. Where feasible, quantum-readiness requirements should be incorporated into new contracts to ensure future device deployments are adequately protected.

Certain devices encounter additional migration obstacles, such as regulatory restrictions or locked-down firmware that prevent upgrades. In such cases, teams should formally document accepted risks, implement compensating controls such as network segmentation or encrypted overlays, and escalate unresolved issues to compliance or governance for further review. Addressing these challenges systematically ensures that the quantum readiness plan remains comprehensive and audit-ready across all devices.

Where Q is normalized from the per-device quantum readiness score, devices with low quantum readiness and high HNDL exposure receive elevated BRS scores, surfacing them in the organization-wide risk posture dashboard.

Figure 7.16: Post-Quantum Readiness Scores by Category

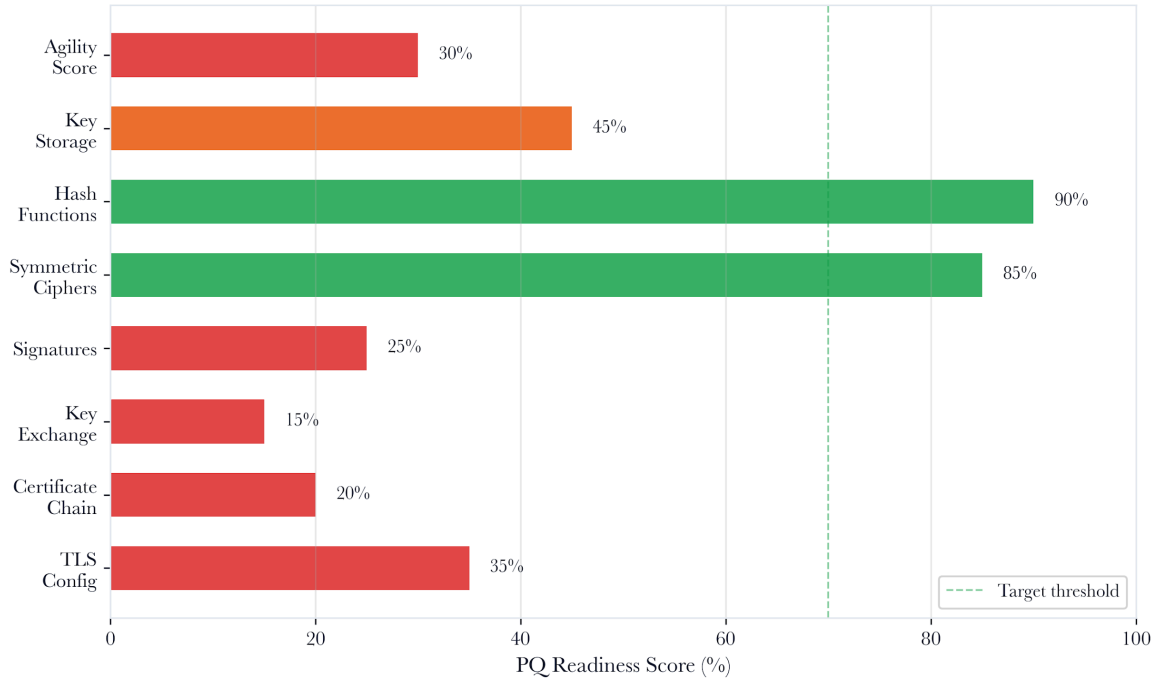

Figure 7.20: PQ readiness scores. The readiness queue combines current cryptographic posture, data sensitivity, migration feasibility, and implementation risk.

The quantum stream also feeds into the remediation planner (Phase 12/Chapter 12), which generates concrete PQ migration actions within the broader remediation plan.

7.11 Novel Research Extensions

7.11.1 Monte Carlo HNDL Simulation

Figure 7.18: Monte Carlo HNDL Risk Distribution (10k simulations)

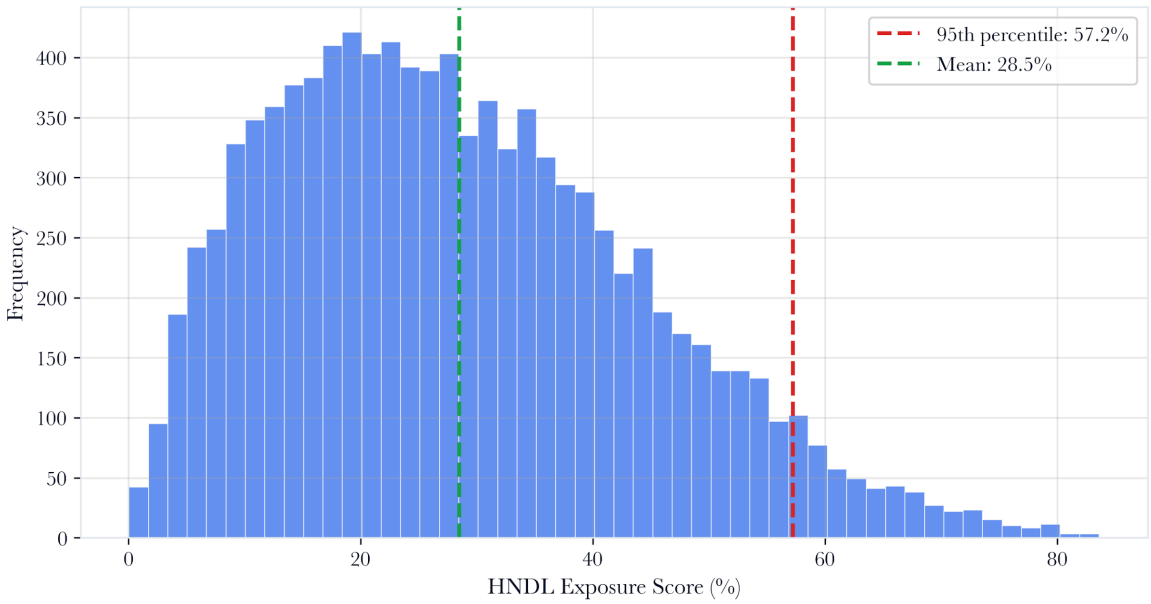

Figure 7.21: Monte Carlo HNDL risk. Scenario distributions make timeline uncertainty visible rather than hiding it behind a single speculative CRQC date.

Rather than using point estimates for the CRQC arrival date, the threat_simulator.py module runs Monte Carlo simulations with the quantum arrival modeled as a probability distribution. This produces confidence intervals for HNDL exposure, enabling risk-adjusted decision-making.

7.11.2 Cryptographic Agility Assessment

Figure 7.15: Hybrid TLS 1.3 Handshake with ML-KEM

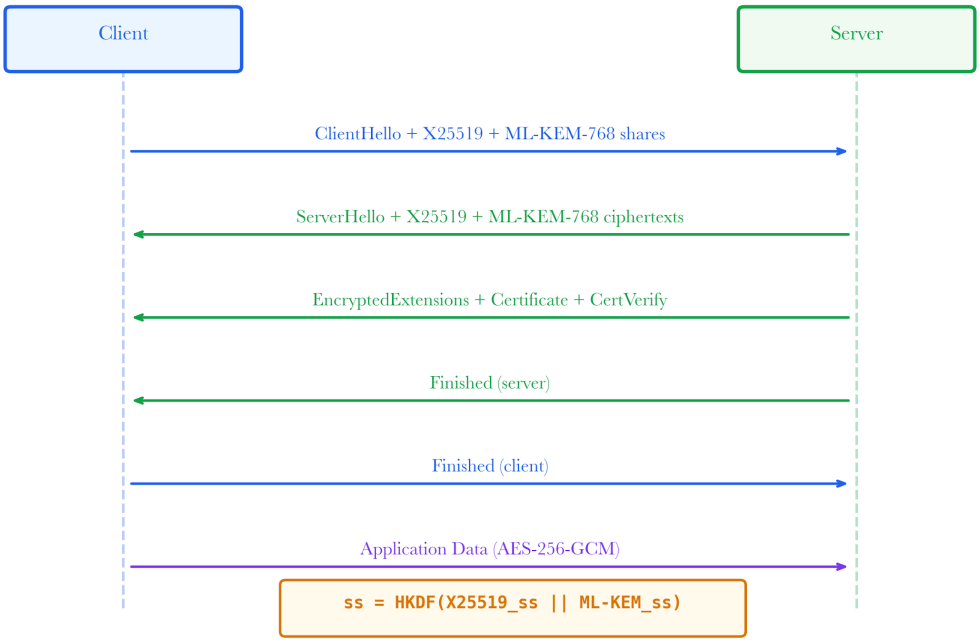

Figure 7.22: Hybrid handshake flow. Transitional deployments often combine classical and post-quantum key agreement, so a single broken family does not decide the session secret alone.

The agility_scanner.py module probes devices by sending TLS ClientHello messages with PQ key shares (ML-KEM, X25519Kyber768). Devices that accept these shares are treated as *likely agile* – a strong signal that migration may be possible without a firmware rewrite, subject to vendor documentation and interoperability testing.

7.11.3 Certificate Lifecycle Tracking

The cert_lifecycle.py module tracks certificate expiry dates and quantum vulnerability across the device fleet, identifying certificates that will still be valid when CRQCs arrive and therefore need early rotation.

7.12 Limitations

Figure 7.23: PQ Migration Compliance Checklist

#	Requirement	Standard	Status
1	Cryptographic inventory complete	NIST SP 800-208	Done
2	HNDL risk assessment performed	NSA CNSA 2.0	Done
3	Hybrid TLS deployed (testing)	RFC 9180	In Progress
4	ML-KEM integrated in key exchange	FIPS 203	Not Started
5	ML-DSA for code signing	FIPS 204	Not Started
6	Crypto-agility framework in place	NIST IR 8547	In Progress
7	Side-channel testing of PQ impls	NIST SP 800-185	Not Started
8	HSM firmware PQ-ready	FIPS 140-3	In Progress
9	Legacy algorithm sunset plan	NSA CNSA 2.0	In Progress
10	Continuous monitoring enabled	NIST CSF 2.0	Done

Figure 7.23: Compliance checklist. The final readiness posture records evidence, exceptions, vendor dependencies, and review cadence for audit and governance.

- The timeline for CRQC is still uncertain. Scenario analysis can help plan, but no one knows exactly when breakthroughs will happen. To stay prepared, organizations should regularly check their quantum risk status as new research and threats emerge. Plan formal reviews twice a year, or every quarter for high-risk systems, to keep migration plans up to date. Regular check-ins, flexible priorities, and phased rollouts help teams adjust quickly as things change.
- To help non-technical leaders understand quantum risk and uncertainty, use clear, standard reports that show where uncertainty exists. Dashboards can use "uncertainty flags" to mark timeline estimates, confidence intervals, or debated risks. Begin executive summaries by stating what is unknown, how different scenarios could change the plan, and what actions make sense even if details are missing. Major reports should include tables comparing optimistic, moderate, and pessimistic CRQC timelines, and should always state whether risk estimates are based on data or scenarios. Migration recommendations should indicate the forecast's confidence and highlight when ongoing monitoring or review is needed.
- Students should write down their assumptions and review migration schedules regularly. This helps ensure that new quantum developments are incorporated into ongoing risk assessments and updates.
- Mosca's inequality is an approximate tool. It highlights timing pressure but does not account for sudden cryptanalytic advances, undisclosed capabilities, or variations in organizational data storage.
- TLS inspection reveals the cryptographic setup on scanned interfaces, but does not guarantee that all device paths, firmware update channels, or management tunnels have been checked. The planner focuses on cost and technical feasibility in this chapter's model. It does not replace the need for interoperability testing, vendor agreements, or maintenance window planning.

- Side-channel estimates for constrained devices are an initial assessment. Actual leakage risks must still be reviewed during implementation and measured on the hardware. Best practice is to validate side-channel security using a combination of lab-based testing (such as power or electromagnetic analysis in a controlled environment) and review of vendor-provided side-channel evaluation results. Commonly used side-channel analysis toolkits include ChipWhisperer, Riscure Inspector, and the NewAE ChipSHOUTER platform. These tools provide practical frameworks for measuring and analyzing leakage from cryptographic implementations, supporting both power and electromagnetic attack vectors. It is recommended to require vendors to provide evidence of side-channel resistance and to conduct independent penetration testing before deployment in production. Regularly reassessing side-channel risk as firmware or cryptographic libraries are updated further reduces exposure and supports secure PQC deployment.

Phase 7 transforms cryptographic risk from a general concern regarding quantum computing into a quantifiable assessment with actionable migration plans. The primary components are:

1. **TLS Deep Inspector** – extracts cipher suite, key exchange, and certificate details from every TLS-enabled device.
2. **Crypto Classifier** – scores each device on a 0-100 quantum readiness scale using five weighted factors.
3. **HNDL Risk Modeler** – applies Mosca's inequality with three timeline models to quantify harvest-now-decrypt-later exposure.
4. **Grover Oracle Estimator** – computes precise qubit requirements per symmetric cipher using published circuit analyses.
5. **Side-Channel Estimator** – evaluates implementation risks of deploying PQ algorithms on constrained IoT hardware.
6. **Migration Planner** – generates prioritized, costed migration plans with vendor-grouped deployment schedules.

Review Questions

1. Explain why ECDHE key exchange is vulnerable to quantum attack, but AES-256 is not. What is the fundamental difference between Shor's and Grover's algorithms that makes this so?
2. A hospital network has 200 IP cameras running TLS 1.2 with TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256. The video feeds contain patient movement data with a 25-year retention requirement. Using Mosca's inequality with the moderate CRQC timeline (2033), calculate whether this data is at risk today. Assume migration will take 3 years.
3. Why does the pq_classifier.py weight key exchange at 40% but TLS protocol version at only 10%? Under what circumstances would you adjust these weights?
4. An ARM Cortex-M4 IoT sensor needs to be migrated to ML-KEM. The side_channel_estimator.py assigns a leakage risk of 0.70. What specific side-channel attacks is this device vulnerable to, and what mitigation strategies would you recommend?
5. Compare the three migration strategies (firmware update, VPN overlay, gateway proxy) for an IP camera that cannot be firmware-updated. What are the trade-offs in cost, latency, and operational complexity?

6. Critically evaluate the Monte Carlo HNDL simulation approach. What are the limitations of modeling CRQC arrival as a probability distribution? How would you improve the model?
7. Design an agentic migration planner for a fleet with mixed TLS stacks, long data-retention obligations, and hardware constraints. What state should the planner retain, what verifier should challenge overconfident recommendations, and when should a human override the priority order?

References

1. Shor, P. W. (1994). "Algorithms for quantum computation: discrete logarithms and factoring." *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 124–134.
2. Grover, L. K. (1996). "A fast quantum mechanical algorithm for database search." *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212–219.
3. Grassl, M., Langenberg, B., Roetteler, M., & Steinwandt, R. (2016). "Applying Grover's algorithm to AES: quantum resource estimates." *Post-Quantum Cryptography (PQCrypto, 2016)*, 29-43.
4. Beauregard, S. (2003). "Circuit for Shor's algorithm using 2n+3 qubits." *Quantum Information & Computation*, 3(2), 17–– -185.
5. Roetteler, M., Naehrig, M., Svore, K. M., & Lauter, K. (2017). "Quantum resource estimates for computing elliptic curve discrete logarithms." *ASIACRYPT 2017*, 24–-270.
6. Mosca, M., & Mulholland, J. (2017). "A methodology for quantum risk assessment." *Global Risk Institute*.
7. NIST FIPS 203: ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism Standard), August 2024.
8. NIST FIPS 204: ML-DSA (Module-Lattice-Based Digital Signature Standard), August 2024.
9. NIST FIPS 205: SLH-DSA (Stateless Hash-Based Digital Signature Standard), August 2024.
10. Ravi, P., et al.(2023). "Exploiting CRYSTALS-DILITHIUM on Cortex-M4." *CHES 2023*.
11. Primas, R., et al.(2024). "Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption." *EUROCRYPT 2024*.
12. NIST IR 8547: "Transition to Post-Quantum Cryptography Standards." 2024.

Cross-References

- **Chapter 1** (Phase 1) – TLS certificate extraction in Phase 2 enrichment provides the raw data for quantum assessment.
- **Chapter 3** (Phase 3) – Protocol transcripts record cipher suites negotiated in TLS handshakes.
- **Chapter 6** (Phase 6) – Digital twin validates that PQ migration does not break device connectivity.
- **Chapter 8** (Phase 8) – Federated learning aggregates quantum posture intelligence across sites.
- **Chapter 12** (Phase 12) – Autonomous remediation executes PQ migration plans with safety verification.
- **HYDRA** – Quantum exposure feeds into the composite BRS as stream Q (weight w6 = 0.05).