

Q-Shield: Automated Crypto-Agility Auditing and Quantum-Risk Prioritization for Legacy Infrastructure

Deepa P¹, Agalya S², Barath D³, Harisudhan K⁴, Nikidha M⁵, Reya Stebe S⁶
^{1,2,3,4,5,6}Department of CSE (IoT & Cybersecurity Including BCT), SNS College of Technology,
 Coimbatore, India

Abstract—Cryptographically relevant quantum computers (CRQCs) would break the public-key algorithms that secure most of today’s digital infrastructure, including RSA, Diffie–Hellman and elliptic-curve cryptography. Because adversaries can record encrypted traffic now and decrypt it later (Harvest Now, Decrypt Later, HNDL), migration cannot wait until a CRQC exists. The National Institute of Standards and Technology (NIST) has published its first post-quantum standards, yet organizations still struggle with two practical problems: they do not know where vulnerable cryptography is deployed, and they cannot tell which instances to migrate first. This paper presents Q-Shield, a web-based platform that (i) discovers cryptographic usage across source code, network services and certificates; (ii) consolidates findings into a cryptographic inventory; (iii) combines algorithm vulnerability with data sensitivity, confidentiality lifetime, asset criticality and exposure into a configurable quantum-risk score; and (iv) produces a dependency-aware migration roadmap. We describe the architecture, the scoring and prioritization models, a prototype implementation, and an evaluation methodology based on a synthetic enterprise dataset and expert-labelled ground truth. The main contribution of this paper is not the detection of RSA or ECC itself, but the transformation of fragmented cryptographic observations into a lifecycle-aware and dependency-aware migration priority.

Index Terms—crypto-agility, cryptographic inventory, Harvest Now Decrypt Later, migration planning, post-

quantum cryptography, quantum risk assessment, static analysis

I. INTRODUCTION

Cryptography protects financial transactions, healthcare records, authentication credentials, intellectual property, government communications and IoT telemetry. Much of this protection depends on public-key mechanisms (RSA, DH, DSA, ECDH, ECDSA) whose security rests on integer factorization or discrete logarithms. Shor’s algorithm [1] solves both in polynomial time on a sufficiently large fault-tolerant quantum computer. Published resource estimates for breaking RSA-2048 have fallen steadily [2], [3], although the timeline for a CRQC remains uncertain.

The urgency comes less from the date a CRQC appears than from data lifetime. Mosca’s inequality [4] captures this: if the time data must stay confidential (x) plus the time needed to migrate (y) exceeds the time until a CRQC exists (z), the data is already at risk (Fig. 1). Records protected today by quantum-vulnerable key exchange may be captured and stored by an adversary, and medical, financial and intellectual-property data may need confidentiality for decades.

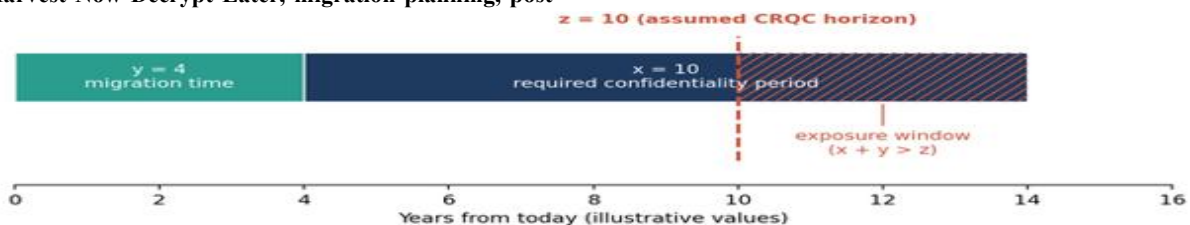


Fig. 1. Mosca’s inequality. When the confidentiality period (x) plus the migration time (y) exceeds the years until a CRQC (z), data protected today is already exposed. Values are illustrative, and z is a configurable scenario parameter in Q-Shield.

In August 2024 NIST published FIPS 203 (ML-KEM) for key establishment, FIPS 204 (ML-DSA) for digital signatures and FIPS 205 (SLH-DSA) for hash-based signatures [5]–[7]. NIST’s transition guidance [8] indicates that quantum-vulnerable public-key algorithms will be deprecated and later disallowed, and governments have issued migration directives and timelines [9], [10]. The standards therefore exist; the operational problem is now knowing what to migrate, and in what order.

Large organizations often have thousands of certificates, many codebases and long-lived legacy systems, with no central record of cryptographic use. Even when an inventory exists, a flat list of “RSA-2048 found” items does not say whether one RSA-2048 deployment should be migrated before another. Existing scanners (Section II-D) report what algorithm is used where; none of them combines that answer with what the protected data is, how long it must remain secret, how critical the asset is, or which other systems depend on it. Q-Shield is designed to fill this gap, and the novelty claimed here lies in that prioritization layer rather than in the detection of individual algorithms.

A. Contributions

- 1) An automated discovery pipeline that detects cryptographic usage in source code, TLS endpoints and certificates.
- 2) A centralized cryptographic inventory whose fields are compatible with Cryptographic Bill of Materials (CBOM) concepts [11].
- 3) A data-sensitivity and confidentiality-lifetime model grounded in Mosca’s inequality.
- 4) A configurable, quantum-specific risk score.
- 5) A dependency-aware prioritization method that produces an ordered migration roadmap.
- 6) An interactive web dashboard for security teams, and an evaluation methodology for the whole pipeline.

B. Research Questions

RQ1. Can automated static and dynamic analysis accurately identify quantum-vulnerable cryptographic implementations?

RQ2. Does incorporating data sensitivity and confidentiality lifetime improve prioritization over algorithm-only ranking?

RQ3. Does dependency-aware analysis yield more practical migration plans than algorithm-only reports?

RQ4. How much effort does Q-Shield save when building an organizational cryptographic inventory?

C. Organization of the Paper

Section II reviews the quantum threat, post-quantum standards and existing tooling. Section III formalizes the problem, and Section IV presents the system architecture and user interface. Section V describes the methodology (discovery, inventory, risk model and dependency-aware prioritization), Section VI the implementation, and Section VII the evaluation methodology. Section VIII walks through an illustrative case study, Section IX reports the results status, and Sections X–XIII discuss limitations, ethics and future work.

II. BACKGROUND AND RELATED WORK

A. Quantum Threat to Cryptography

Quantum computers affect cryptographic families unequally. Shor’s algorithm [1] breaks RSA, finite-field DH/DSA and elliptic-curve schemes, so these must be replaced. Grover’s algorithm [12] gives a quadratic speed-up on brute-force search, which roughly halves the effective security of symmetric keys and hash preimage resistance. AES-256 and SHA-256/SHA-384 are therefore generally considered adequate, while AES-128 offers a reduced margin. Q-Shield encodes this distinction explicitly rather than labelling all cryptography as “vulnerable”.

B. Post-Quantum Standards and Crypto-Agility

ML-KEM (FIPS 203) and ML-DSA (FIPS 204) derive from the CRYSTALS-Kyber and CRYSTALS-Dilithium submissions respectively, and SLH-DSA (FIPS 205) derives from SPHINCS+ [5]–[7]. Hybrid deployments, which combine a classical and a post-quantum algorithm, are widely discussed as a transition mechanism. Crypto-agility, the ability to replace algorithms without redesigning systems, is a prerequisite for any migration, and it depends on knowing where cryptography is used.

C. Cryptographic Inventory and CBOM

Software Bills of Materials (SBOMs) record software components. The CycloneDX specification extended this idea to cryptographic assets through the Cryptographic Bill of Materials (CBOM) [11]. Q-Shield produces an inventory compatible with CBOM concepts; exporting to a formal CBOM schema is listed as future work rather than a claimed feature.

D. Existing Tooling and Gap

Static analyzers such as Semgrep [13] detect cryptographic API calls through pattern rules. TLS scanners such as SSLyze [14] and testssl.sh [15] enumerate protocol versions, cipher suites and certificates. Network mappers such as Nmap [16] discover services. Table I compares these tools with Q-Shield along the capabilities needed for migration planning. The comparison shows that detection is well covered by existing tools, whereas lifecycle-aware and dependency-aware prioritization is not.

TABLE I CAPABILITY COMPARISON OF EXISTING TOOLS AND Q-SHIELD

Tool	Code detection	TLS / certificate analysis	Consolidated inventory	Data sensitivity & lifetime	Dependency-aware order	Migration roadmap
Semgrep [13]	Yes	No	No	No	No	No
SSLyze [14] / testssl.sh [15]	No	Yes	No	No	No	No
Nmap [16]	No	Partial	No	No	No	No
CycloneDX CBOM [11]	—	—	Format only	No	No	No
Q-Shield	Yes	Yes	Yes	Yes	Yes	Yes

Assessment reflects each tool's core documented function. "Format only" means the specification defines how to record cryptographic assets but does not discover or rank them. Commercial and research cryptography-discovery platforms are outside this comparison.

E. IoT and Embedded Deployments

Embedded and IoT systems form a growing part of the infrastructure that a cryptographic inventory must eventually cover. Recent application papers describe an AI-assisted dimensional-measurement and material-transportation system [17], an Arduino-based women's safety device that uses SMS alerts [18], and intelligent vibration-based fault diagnosis on microprocessor systems for roller bearings in turbine generators [19]. Such systems typically combine sensors, microcontrollers and wireless or cellular links, which are the settings where cryptographic choices are hardest to inspect and update. These works are cited as application context and not as baselines: none of them evaluates post-quantum migration, and IoT and firmware scanning in Q-Shield is left as future work (Section XIII).

III. PROBLEM FORMULATION

Let an organization consist of a set of assets $A = \{a_1, \dots, a_n\}$. Each asset a has:

- 1) a set of cryptographic usages $C(a)$, each with an algorithm, key size, protocol and location;
- 2) a data classification $s(a)$ and required confidentiality period $x(a)$, in years;
- 3) a criticality level $k(a)$ and exposure level $e(a)$;
- 4) an estimated migration time $y(a)$, in years;
- 5) a set of dependencies D , forming a directed graph $G = (A, D)$ in which an edge (u, v) means that v depends on cryptographic services provided by u (for example, an API gateway depends on an intermediate CA).

Goal. Produce (i) a risk score $R(a)$ in $[0, 100]$ for each asset, (ii) a risk class, and (iii) a migration order, consistent with G , that places high-risk and high-impact assets early.

IV. SYSTEM ARCHITECTURE

Q-Shield follows a layered architecture of a web front end, an API gateway, scanning workers, a processing pipeline and a persistent store (Fig. 2, Table II). The end-to-end workflow (Fig. 3) runs from user login and asset registration through scan

initiation, detection from code, network and certificate sources, inventory consolidation, data classification, lifetime estimation, risk calculation, dependency analysis, priority ranking and migration recommendation, and ends in the dashboard and report.

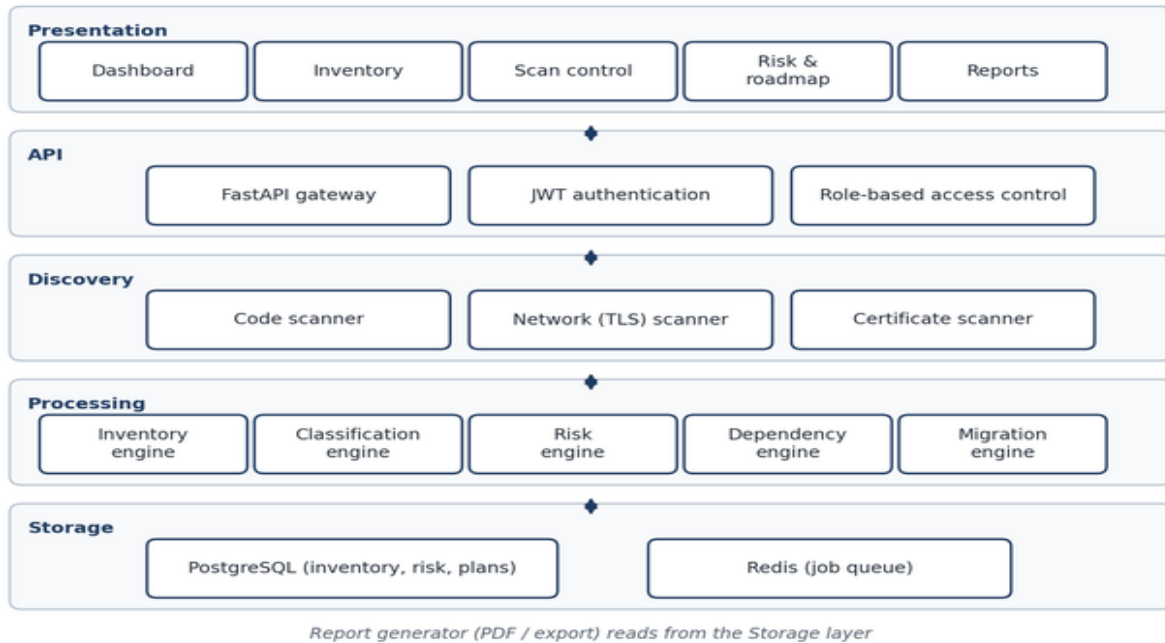


Fig. 2. Layered architecture of Q-Shield.

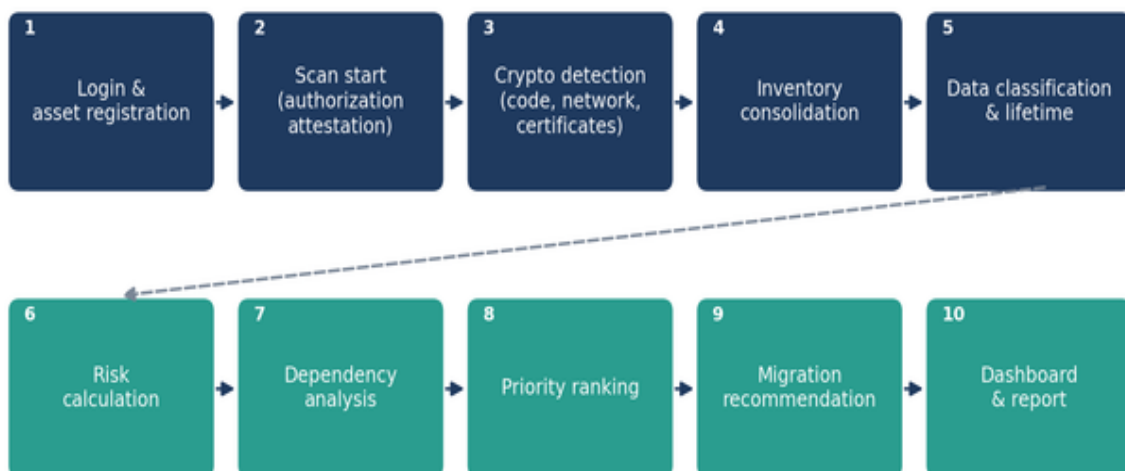


Fig. 3. End-to-end Q-Shield workflow, from login to dashboard and report.

TABLE II ARCHITECTURE LAYERS AND RESPONSIBILITIES

Layer	Components	Responsibility
Presentation	React web UI	Dashboard, inventory, scan control, risk and roadmap views
API	FastAPI gateway, authentication, role-based access control	Request routing, access control
Discovery	Code scanner, network scanner, certificate scanner	Produce raw cryptographic findings
Processing	Inventory, classification, risk, dependency and migration engines	Normalize, score, order
Storage	PostgreSQL, Redis	Persistent data and job queue
Reporting	Report generator	PDF reports and exports

A. Roles

Three roles are supported. An Administrator manages users, organizations and scan configuration. An Analyst uploads code, starts scans and inspects findings. A Viewer, typically a CISO or manager, has read-only access to dashboards, priorities and reports.

B. Web Application

The interface is organized into the navigation modules visible in Fig. 4: Dashboard, Crypto Inventory, Vulnerable Assets, Risk Prioritization, Migration Roadmap, Security Scans, Reports, Settings and Profile. The dashboard summarizes four indicators (total discovered assets, quantum-vulnerable assets, critical-risk findings and a quantum readiness index), an organizational quantum-risk score, and a post-quantum posture alert. Fig. 4 shows the dashboard in its initial state, before any scan has been run.



Fig. 4. Q-Shield dashboard (initial state, before any scan has populated the inventory). The sidebar exposes the main modules, and the summary cards update as scans complete.

V. METHODOLOGY

A. Cryptographic Discovery

Source-code scanning. Users upload an archive or connect a repository. A rule-based scanner searches for cryptographic library imports, algorithm identifiers, key-generation calls, TLS configuration, certificate loading, cryptographic APIs and hard-coded keys. Example indicators include `crypto/rsa` (Go), `RSA.generate` (Python),

`Cipher.getInstance("RSA")` (Java), `EVP_PKEY_RSA` (OpenSSL) and `ECDSA_sign`. Initial language coverage is Java, Python, C, C++, JavaScript, Go and Rust. Each finding records algorithm, key size where determinable, file and line. Network scanning. For hosts that the organization owns or is explicitly authorized to test, the scanner records TLS version, cipher suites, key-exchange groups, certificate public-key algorithm and size, and signature algorithm. Scanning is restricted to

authorized targets, and the platform requires the user to attest to authorization before a scan starts.

Certificate scanning. Certificates are parsed for issuer, subject, public-key algorithm, key size, validity period and signature algorithm, and mapped to the chain that contains them.

B. Cryptographic Inventory

Findings are normalized into a single inventory with the fields asset, algorithm, key size, protocol, location, quantum status and source. Deduplication merges the same usage reported by several scanners. Each usage receives one of the quantum statuses in Table III. The policy is stored as configuration, so it can be updated as standards and guidance evolve.

TABLE III QUANTUM STATUS CLASSES
(ILLUSTRATIVE DEFAULT POLICY)

Class	Examples	Rationale
Quantum-vulnerable (public-key)	RSA, DH, DSA, ECDH, ECDSA, EdDSA	Broken by Shor's algorithm
Reduced margin (symmetric/hash)	AES-128, SHA-1, SHA-224	Grover-reduced security or classically weak
Adequate (symmetric/hash)	AES-256, ChaCha20, SHA-256/384/512	Considered adequate under known quantum attacks
Hybrid	Classical + ML-KEM key exchange	Protects against both threat types
Post-quantum	ML-KEM, ML-DSA, SLH-DSA	NIST-standardized PQC
Unknown	Unresolved or custom	Needs manual review

C. Data Classification and Confidentiality Lifetime

Each asset is assigned a sensitivity class (Low, Medium, High, Critical) and a required confidentiality period x . Assignment can be manual, policy-driven from asset type and data tags, or suggested by a classifier (Section V-F). Typical defaults are shown in Table IV and should be treated as configurable organizational policy, not universal constants.

TABLE IV EXAMPLE DEFAULT CONFIDENTIALITY PERIODS

Data type	Confidentiality period
Session tokens	Minutes
Temporary logs	Months

Data type	Confidentiality period
Financial records	7–10 years
Intellectual property	10–20 years
Medical records	Decades

D. Quantum Risk Model

Each factor is normalized to $[0, 100]$ (Table V). The confidentiality-lifetime factor follows Mosca's inequality: with confidentiality period x , migration time y and an assumed years-to-CRQC z (a configurable scenario parameter),

$$S_l = 100 \times \min(1, (x + y) / z) \quad (1)$$

so that any asset for which $x + y \geq z$ scores 100, meaning that it already satisfies Mosca's condition for risk. The overall risk is the weighted sum

$$R = w_c S_c + w_s S_s + w_l S_l + w_k S_k + w_e S_e \quad (2)$$

with default weights $w_c = 0.30$, $w_s = 0.25$, $w_l = 0.20$, $w_k = 0.15$ and $w_e = 0.10$ (they sum to 1). Migration difficulty is deliberately kept out of R , because it measures effort rather than risk, and is used later in scheduling. The weights are initial values; they are tuned and validated through the sensitivity analysis in Section VII-E.

TABLE V RISK FACTORS AND DEFAULT SCALES

Factor	Definition	Default scale
Crypto risk (S_c)	Lookup from quantum status and, where relevant, key size	100 for quantum-vulnerable public-key use; reduced for AES-128; low for adequate and post-quantum algorithms
Data sensitivity (S_s)	Classification of the protected data	Low 25, Medium 50, High 75, Critical 100
Confidentiality lifetime (S_l)	Mosca-based, (1)	0–100, capped at 100
Asset criticality (S_k)	Business impact of compromise or outage	Low 25, Medium 50, High 75, Critical 100
Exposure (S_e)	Reachability of the asset	Isolated 25, Internal 50, Partner-facing 75, Internet-facing 100

Worked example. For $S_c = 90$, $S_s = 95$, $S_l = 100$, $S_k = 90$ and $S_e = 80$: $R = 0.30(90) + 0.25(95) + 0.20(100) + 0.15(90) + 0.10(80) = 27 + 23.75 + 20 + 13.5 + 8 = 92.25$. Risk classes are Low (0–25), Medium (26–50), High (51–75) and Critical (76–100). These thresholds are part of the proposed model, and

Section VII evaluates whether they produce sensible prioritization.

E. Dependency-Aware Prioritization

Cryptographic services form chains: a root CA issues intermediate CAs, which issue certificates for gateways, which protect backends and databases. Migrating a leaf service before the trust anchor it depends on can be impossible or wasted effort. Q-Shield models this as the directed graph G . For each asset a , let $T(a)$ be the set of assets that transitively depend on it, and define the normalized dependent count and priority score as

$$C(a) = |T(a)| / (n - 1) \quad (3)$$

$$P(a) = R(a) + \lambda \cdot 100 \cdot C(a) \quad (4)$$

where λ is a configurable impact weight (Fig. 5). Assets are ordered by a topological ordering of G , with ties broken by descending $P(a)$ and, within equal P , ascending estimated migration effort so that quick wins come first. Cycles, which occur in real infrastructure, are collapsed into strongly connected components and scheduled as a single migration unit. Where the CA-first ordering conflicts with a very high-risk leaf asset, the roadmap flags the conflict and recommends interim mitigation, such as hybrid key exchange or shortening data retention, instead of silently reordering.

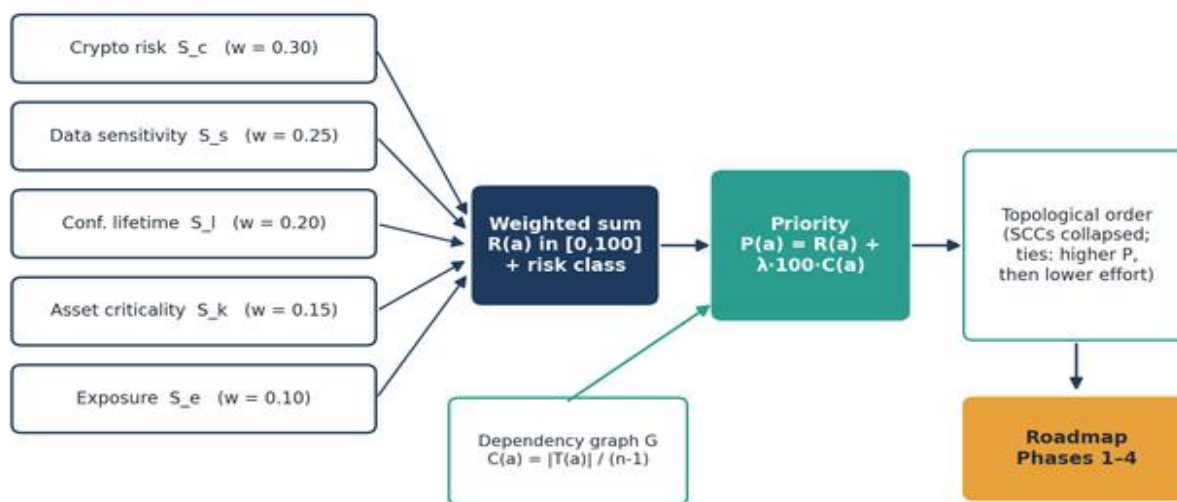


Fig. 5. Risk scoring and dependency-aware prioritization pipeline.

F. Hybrid Rule-Based and Machine-Learning Design
Deterministic facts such as “RSA is Shor-vulnerable” are handled by rules, which are auditable and need no training data. Machine learning is limited to tasks where policy is incomplete: suggesting data sensitivity and criticality from asset metadata, and predicting a risk class from the factor vector. Tree-based models (Random Forest, XGBoost) are used because they are interpretable through feature importance. The ML component is advisory, and the rule-based score remains the reference. We do not claim that a language model determines cybersecurity risk on its own.

G. Migration Recommendation and Roadmap

For each vulnerable usage, Q-Shield produces a recommendation: the current algorithm, the risk class, a candidate target (for example ML-KEM for key establishment, ML-DSA or SLH-DSA for signatures, potentially in a hybrid configuration), and a staged strategy: inventory dependencies, establish a test environment, test a PQC-capable implementation, evaluate hybrid deployment, validate interoperability, deploy gradually, and retire the legacy algorithm. Recommendations are advisory and must be validated against vendor support and system constraints. The roadmap groups assets into four phases (Table VI).

TABLE VI ROADMAP PHASES

Phase	Assets included
Phase 1	Critical assets that are internet-facing or hold long-lived sensitive data, together with the trust anchors they depend on
Phase 2	Internal high-value systems
Phase 3	Medium-risk legacy applications
Phase 4	Low-risk systems

VI. IMPLEMENTATION

Table VII summarizes the technology stack. Because the platform stores a map of an organization's cryptographic weaknesses, it is itself a sensitive asset. The prototype therefore uses salted password hashing, role-based access control, TLS for all traffic, audit logging of scans and encryption of the database at rest. The back end ships with a seed script that loads a demonstration organization and an automated test suite for the API, which are used for functional verification and are not evaluation results.

TABLE VII IMPLEMENTATION STACK

Component	Technology
Front end	React, Tailwind CSS, Recharts (risk charts), React Flow (dependency graph)
Back end	Python, FastAPI (chosen for its security-analysis ecosystem); long scans run as background jobs through Celery with Redis as broker
Database	PostgreSQL; core tables: users, assets, crypto_inventory, data_classification, risk_assessment, dependencies, migration_plan
Scanning	Semgrep with custom cryptographic rules (source code); SSLyze or testssl.sh (TLS); Nmap (service discovery in authorized ranges); Python cryptography library (certificate parsing)

VII. EVALUATION METHODOLOGY

This section specifies how the research questions will be answered. All reported figures will be obtained from these experiments and not from the illustrative values used in Section VIII.

A. Datasets

Synthetic enterprise dataset. Because real organizational data is rarely shareable, we generate

records with the fields `asset_id`, `asset_type`, `algorithm`, `key_size`, `data_type`, `sensitivity`, `confidentiality_years`, `internet_exposed`, `criticality`, `dependency_count`, `risk_score` and `priority`. Sampling distributions, random seeds and generation code will be published for reproducibility. A synthetic dependency graph with a realistic tree-and-mesh structure is generated alongside.

Code corpus. For RQ1, a labelled set of open-source repositories across the supported languages, plus purpose-built test projects with known cryptographic usage, so that ground truth is exact.

Test infrastructure. A laboratory environment of TLS servers with known configurations and certificates (RSA, ECDSA, mixed chains, hybrid-enabled) under our own control.

B. Ground Truth

For detection, ground truth is established by manual review of the corpus. For risk and priority, a panel of security practitioners independently ranks and classifies a sample of assets, and inter-rater agreement (for example Cohen's or Fleiss' kappa) is reported before the consensus is used as the reference.

C. Baselines

B1: algorithm-only listing (every quantum-vulnerable usage flagged equally).

B2: severity by algorithm and key size only.

B3: Q-Shield risk score without lifetime and without dependency analysis.

B4: full Q-Shield (R plus dependency-aware ordering).

Comparing B1 to B4 addresses RQ2 and RQ3 directly, and B3 isolates the contribution of each component.

D. Metrics

TABLE VIII RESEARCH QUESTIONS AND EVALUATION METRICS

Question	Metric
RQ1	Precision, recall and F1 of detected algorithms, per language and per scanner
RQ2	Accuracy, macro-F1 and Cohen's kappa of risk class against expert labels; rank correlation (Kendall's tau, Spearman's rho) and top-k overlap of priority order against expert ranking

Question	Metric
RQ3	Number of dependency-order violations in the roadmap; fraction of plans judged feasible by experts
RQ4	Time to build an inventory with and without Q-Shield on the same systems; scan time against codebase size (files, lines of code)

E. Sensitivity Analysis

We vary each weight by a fixed percentage, vary the CRQC horizon z across several scenarios, and measure the effect on class assignments and rank order, reporting how robust the roadmap is to these assumptions. We also compare expert-tuned weights with weights learned by the ML model.

VIII. ILLUSTRATIVE CASE STUDY

Consider a healthcare organization with a patient API, a billing system, an internal VPN and a mobile application, all relying on a shared internal CA. The scan finds RSA-2048 on the patient API and billing system, ECDH on the VPN and ECDSA in the mobile application. Table IX lists the assumed inputs, using $z = 15$ years and the default weights; Table X gives the resulting scores, which follow directly from (1) and (2).

These numbers are illustrative outputs of the model on assumed inputs, not experimental results; they should be replaced or confirmed by the lab study.

TABLE IX ASSUMED INPUTS FOR THE CASE STUDY ($z = 15$ YEARS)

Asset	Algorithm	S_c	S_s	x (yr)	y (yr)	S_k	S_e
Patient API	RSA-2048	100	100 (Critical)	20	2	75	100 (Internet)
Billing system	RSA-2048	100	75 (High)	10	3	75	50 (Internal)
VPN	ECDH	100	75 (High)	5	2	75	75 (Partner)
Mobile app	ECDSA	100	25 (Low)	1	1	50	100 (Internet)

TABLE X CASE-STUDY OUTPUTS

Asset	S_i	R	Class	Roadmap order
Shared CA (trust anchor)	—	—	—	1
Patient API	100.0	96.25	Critical	2
Billing system	86.7	82.33	Critical	3
VPN	46.7	76.83	Critical	4
Mobile app	13.3	56.42	High	5

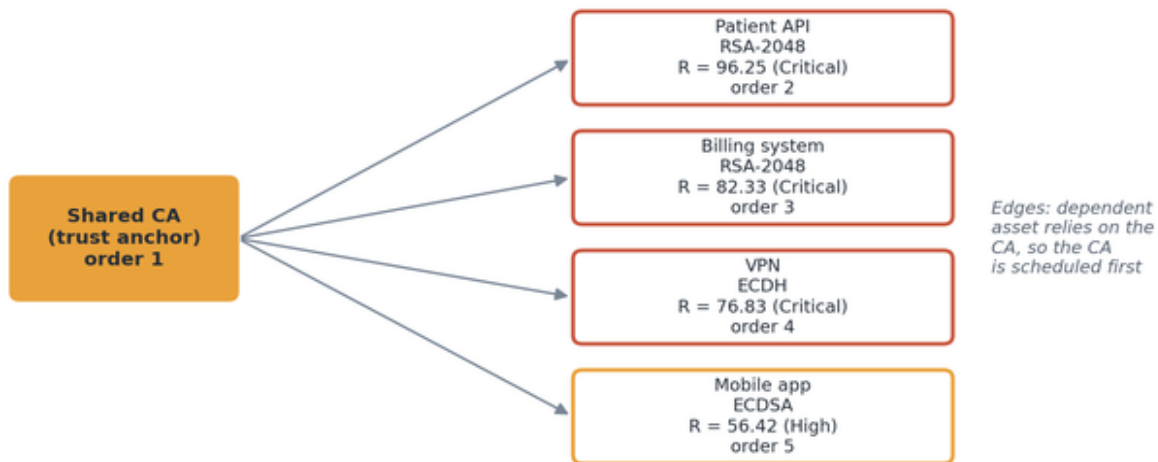


Fig. 6. Case-study dependency graph with risk scores and resulting migration order. The shared CA is scheduled first because every other asset depends on it.

The patient API and billing system use the same algorithm yet are separated by 14 points of risk, driven by data lifetime and criticality, which an algorithm-only list cannot do. The mobile application uses a quantum-vulnerable signature scheme but protects low-sensitivity data with a short lifetime, so it is correctly deferred.

IX. RESULTS

Experimental results will be reported in the form shown in Table XI, together with (a) detection precision, recall and F1 per language; (b) risk-classification confusion matrices for B1–B4; (c) scan time against codebase size; and (d) weight and horizon sensitivity plots. The hypothesis is that Q-Shield (B4) will concentrate attention on a smaller, better-justified set of assets than algorithm-only listing. For example, instead of reporting, say, 87 vulnerable assets as equal, it should separate them into Critical, High, Medium and Low classes with a ranked top-10 for immediate assessment. This hypothesis will be accepted or rejected on the data.

TABLE XI RESULTS TEMPLATE (TO BE COMPLETED FROM EXPERIMENTS)

Metric	B1	B2	B3	B4
Macro-F1 of risk class vs expert labels	TBD	TBD	TBD	TBD
Kendall's tau vs expert ranking	TBD	TBD	TBD	TBD
Top-10 overlap with expert ranking	TBD	TBD	TBD	TBD
Dependency-order violations	TBD	TBD	TBD	TBD

X. DISCUSSION

The main finding to examine is whether lifecycle awareness changes the ordering, since assets using identical algorithms differ greatly in HNDL exposure. A second point is the value of explicit dependency modelling: migrating at the wrong layer wastes effort, and the CA-first rule gives security teams a defensible sequence. Third, the platform is designed so that policy (statuses, periods, weights, thresholds, horizon) is configuration. This matters because both the standards landscape and quantum

timelines will change, and the tool should be re-run rather than rewritten.

XI. LIMITATIONS AND THREATS TO VALIDITY

Static analysis coverage. Dynamically generated or obfuscated cryptographic calls may be missed, and reflective loading or configuration-driven algorithm choice reduces recall.

Opaque protocols. Proprietary encrypted protocols and embedded firmware may prevent complete inspection.

Input quality. Risk scores depend on the quality of organizational data classification and criticality ratings.

Timeline uncertainty. The CRQC horizon z is an assumption; results are therefore reported across scenarios.

Weight validity. Default weights are heuristic until validated; synthetic data may not reflect real enterprise distributions, which limits external validity.

Compatibility. Migration recommendations cannot guarantee compatibility with every legacy system or vendor product.

Scope. The prototype supports a limited set of languages and infrastructure types.

XII. ETHICAL CONSIDERATIONS

Network scanning is limited to systems owned or explicitly authorized for testing, with an authorization attestation in the workflow. The inventory and risk data that Q-Shield produces are sensitive, since they identify an organization's weakest cryptography; access control, audit logging and encryption at rest are therefore treated as requirements, not options.

XIII. CONCLUSIONS AND FUTURE WORK

Q-Shield addresses the gap between discovering quantum-vulnerable cryptography and deciding what to migrate first. By combining discovery, a cryptographic inventory, data sensitivity, confidentiality lifetime, asset criticality, exposure and dependency analysis, it turns a flat vulnerability list into a justified, ordered migration roadmap. Future work includes:

- 1) export to a formal CBOM schema and SBOM integration;
- 2) cloud integrations (AWS, Azure, GCP);
- 3) IoT and firmware scanning, since constrained devices are where cryptographic use is hardest to inspect [17]–[19];
- 4) continuous monitoring that recalculates risk when new weaknesses or new guidance appear;
- 5) tracking of migration progress and verification of PQC or hybrid deployment;
- 6) an assistant that explains priorities in natural language from the inventory and risk model.

ABBREVIATIONS

CA, certificate authority; CBOM, cryptographic bill of materials; CISO, chief information security officer; CRQC, cryptographically relevant quantum computer; HNDL, harvest now, decrypt later; ML-DSA, module-lattice digital signature; ML-KEM, module-lattice key encapsulation; NIST, National Institute of Standards and Technology; PQC, post-quantum cryptography; RBAC, role-based access control; SBOM, software bill of materials; SLH-DSA, stateless hash-based digital signature.

DECLARATION

A generative AI tool was used solely for grammar checking of the manuscript text. The authors reviewed all content and take full responsibility for the manuscript.

REFERENCES

- [1]. P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in Proc. 35th IEEE Symp. Foundations of Computer Science, 1994, doi: 10.1109/SFCS.1994.365700.
- [2]. C. Gidney and M. Ekerå, “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits,” Quantum, vol. 5, p. 433, 2021, doi: 10.22331/q-2021-04-15-433.
- [3]. C. Gidney, “How to factor 2048 bit RSA integers with less than a million noisy qubits,” arXiv:2505.15917, 2025.
- [4]. M. Mosca, “Cybersecurity in an era with quantum computers: Will we be ready?,” IEEE Security & Privacy, vol. 16, no. 5, pp. 38–41, 2018, doi: 10.1109/MSP.2018.3761723.
- [5]. NIST, “Module-lattice-based key-encapsulation mechanism standard,” FIPS 203, Aug. 2024, doi: 10.6028/NIST.FIPS.203.
- [6]. NIST, “Module-lattice-based digital signature standard,” FIPS 204, Aug. 2024, doi: 10.6028/NIST.FIPS.204.
- [7]. NIST, “Stateless hash-based digital signature standard,” FIPS 205, Aug. 2024, doi: 10.6028/NIST.FIPS.205.
- [8]. NIST, “Transition to post-quantum cryptography standards (initial public draft),” NIST IR 8547, Nov. 2024, doi: 10.6028/NIST.IR.8547.ipd.
- [9]. U.S. Office of Management and Budget, “Migrating to post-quantum cryptography,” Memorandum M-23-02, 2022.
- [10]. National Security Agency, “Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) advisory,” 2022.
- [11]. CycloneDX, “Cryptographic Bill of Materials (CBOM) specification,” OWASP CycloneDX Project.
- [12]. L. K. Grover, “A fast quantum mechanical algorithm for database search,” in Proc. 28th Annu. ACM Symp. Theory of Computing (STOC), 1996, pp. 212–219, doi: 10.1145/237814.237866.
- [13]. Semgrep, “Static analysis tool documentation.”
- [14]. SSLyze, “TLS scanning library documentation.”
- [15]. testssl.sh, “TLS/SSL testing tool documentation.”
- [16]. G. Lyon, Nmap Network Scanning. Nmap Project, 2009.
- [17]. R. Senthil Kumar, C. Mohan Raj, W. Rajan Babu, K. Uday, V. Sudha, and V. Anandhakumar, “Smart dimensional measurement and material transportation (SDMMT) system using artificial intelligence,” in Proc. Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT), IEEE, 2023, doi: 10.1109/IDCIoT56793.2023.10053429. [Online]. Available: <https://ieeexplore.ieee.org/document/10053429>
- [18]. W. RajanBabu, N. Pushpalatha, R. Senthil Kumar, V. Gowsalya Devi, A. Mohamed Harrish, S. Supriya, and M. Vinothkumar, “Arduino based women safety device: An

experimental study,” Premier Journal of Science,
doi: 10.70389/PJS.100241. [Online]. Available:
<https://premierscience.com/pjs-25-1256/>

- [19]. L. Mubaraali, N. Kuppuswamy, and R. Muthukumar, “Intelligent fault diagnosis in microprocessor systems for vibration analysis in roller bearings in whirlpool turbine generators real time processor applications,” *Microprocess. Microsyst.*, vol. 76, Art. no. 103079, 2020. [Online]. Available: <https://scholar.google.com/citations?user=Z7o62tIAAAAJ&hl=en>