



Quantum DevSecOps

Strategic Framework & Implementation Guide

Comprehensive Technical Report

April 2026

Table of Contents

Executive Summary	4
Quantum Computing Fundamentals	5
Qubits and Quantum States	5
Superposition and Entanglement	6
Quantum Security Threats	7
Cryptographically Relevant Quantum Computer	7
Shor's Algorithm	7
Grover's Algorithm	8
Post-Quantum Cryptography Standards	10
NIST PQC Standards	10
ML-KEM Key Encapsulation	10
ML-DSA Digital Signatures	11
Hash-Based Signatures	12
Quantum Key Distribution	14
BB84 Protocol	14
E91 Protocol	15
Quantum Random Number Generation	17
DevSecOps Integration Framework	19
Quantum-Safe SDLC	23
Mathematical Foundations	27
Implementation Roadmap	30
Conclusion	34

About this report

Sovereign Systems Labs is a technology consultancy and research lab in Sydney, Australia. We help organisations plan and build AI solutions, custom algorithms and secure systems, and we publish technical reports like this one so leaders and engineering teams can plan with clear information.

This report explains how quantum computing affects the cryptography in use today, the post-quantum standards that replace vulnerable algorithms, and a framework for building quantum-safe practice into the software development lifecycle.

It reflects published standards and guidance as at April 2026. It is general information, not legal, compliance or security advice for your organisation. Standards and dates change, so check the current NIST and Australian Signals Directorate guidance before you act. The roadmap, targets and key performance indicators are an illustrative starting point, not a promise of results.

Questions, or a quantum-safe readiness review for your organisation:
contact@sovereignsystemslabs.com

Executive Summary

The advent of quantum computing represents one of the most significant technological shifts of the 21st century, with profound implications for cybersecurity. This comprehensive Quantum DevSecOps Strategy report provides organizations with a complete framework for understanding, preparing for, and implementing quantum-safe security practices throughout their software development lifecycle.

Quantum computers leverage the principles of superposition and entanglement to perform calculations that would be practically impossible for classical computers. While this promises breakthroughs in fields like drug discovery, materials science, and artificial intelligence, it also poses an existential threat to current cryptographic systems that secure our digital infrastructure.

Key Findings

- Cryptographically Relevant Quantum Computers (CRQC) are projected to emerge within 10-15 years, with some estimates suggesting earlier arrival
- Current public-key cryptography (RSA, ECC, DSA) will be completely broken by Shor's algorithm running on a sufficiently powerful quantum computer
- Symmetric cryptography (AES) and hash functions (SHA-256) will be weakened but can be fortified with larger key sizes
- NIST has finalized three post-quantum cryptography standards: ML-KEM (FIPS 203), ML-DSA (FIPS 204), and SLH-DSA (FIPS 205)
- The "Harvest Now, Decrypt Later" (HNDL) threat is active today - adversaries are storing encrypted data for future decryption

Strategic Recommendations

Organizations must begin their quantum-safe transformation immediately. This report provides a comprehensive 3-month implementation roadmap covering cryptographic inventory, risk assessment, pilot implementations, and full-scale deployment strategies. The integration of quantum-safe practices into DevSecOps workflows ensures that security is built into every stage of the software development lifecycle.

Quantum Computing Fundamentals

Quantum computing represents a fundamental paradigm shift from classical computing. While classical computers process information using bits that represent either 0 or 1, quantum computers use quantum bits (qubits) that can exist in superpositions of states, enabling unprecedented computational capabilities.

Qubits and Quantum States

A qubit is the fundamental unit of quantum information. Unlike a classical bit, which can be either 0 or 1, a qubit can exist in a superposition of both states simultaneously. Mathematically, a qubit's state is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex probability amplitudes satisfying $|\alpha|^2 + |\beta|^2 = 1$. When measured, the qubit collapses to state $|0\rangle$ with probability $|\alpha|^2$ or state $|1\rangle$ with probability $|\beta|^2$.

Qubit Technologies

Several technologies are being developed to realize qubits, each with distinct advantages and challenges:

Technology	Coherence Time	Gate Fidelity	Scalability
Superconducting	100-500 us	99.5%+	High (1000+ qubits)
Trapped Ion	1-10 s	99.9%+	Medium (50+ qubits)
Photonic	Low (flying)	99%+	High
Neutral Atom	1-10 s	99.5%+	Medium
Topological	Theoretical	N/A	Unknown

Table 1: Comparison of Leading Qubit Technologies

Superposition and Entanglement

Superposition allows quantum systems to exist in multiple states simultaneously. This property enables quantum parallelism - the ability to evaluate multiple possibilities at once. For example, three qubits can represent eight states (000, 001, 010, 011, 100, 101, 110, 111) simultaneously, and quantum operations can manipulate all these states in parallel.

Entanglement is a uniquely quantum phenomenon where two or more qubits become correlated such that the state of one instantly determines the state of the others, regardless of spatial separation. A maximally entangled Bell state is represented as:

$$|\Phi^{++}\rangle = (|00\rangle + |11\rangle) / \sqrt{2}$$

Entanglement is a critical resource for quantum communication protocols, including quantum key distribution, and underpins the exponential speedup of many quantum algorithms.

The Bloch Sphere

The state of a single qubit can be visualized on the Bloch sphere, a unit sphere where the north pole represents $|0\rangle$, the south pole represents $|1\rangle$, and any other point represents a superposition. The general qubit state can be written as:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$$

where θ ($0 \leq \theta \leq \pi$) is the polar angle and ϕ ($0 \leq \phi < 2\pi$) is the azimuthal angle. This geometric representation helps visualize quantum operations as rotations on the sphere.

Quantum Security Threats

The development of quantum computers poses significant threats to modern cryptographic systems. Understanding these threats is essential for developing effective countermeasures and migration strategies.

Cryptographically Relevant Quantum Computer (CRQC)

A Cryptographically Relevant Quantum Computer (CRQC) is a fault-tolerant quantum system capable of running algorithms like Shor's algorithm at a scale sufficient to break today's public-key cryptography within a useful operational timeframe. Key characteristics include:

- Sufficient logical qubits with high gate fidelity
- Error correction depth to sustain long and complex circuits
- Ability to factor large integers or solve discrete logarithms (e.g., RSA-2048, P-256)

While the exact arrival date of CRQCs remains uncertain, their future impact is inevitable. Organizations must act now to protect sensitive data that may still be valuable when quantum computers become available.

Shor's Algorithm

Developed by Peter Shor in 1994, Shor's algorithm efficiently factors large integers and computes discrete logarithms in polynomial time. This directly undermines the security assumptions of most public-key cryptographic systems.

Algorithm Overview

Shor's algorithm consists of classical pre-processing, quantum period-finding, and classical post-processing:

1. **Classical Reduction:** Reduce the factoring problem to finding the period of a modular exponentiation function
2. **Quantum Period-Finding:** Use Quantum Fourier Transform (QFT) to find the period in $O((\log N)^3)$ operations
3. **Classical Post-Processing:** Use the period to compute factors via greatest common divisor (GCD) calculations

Impact on Cryptography

Shor's algorithm renders the following cryptographic systems vulnerable:

Algorithm	Security Basis	Quantum Attack	Status
RSA	Integer factorization	Shor's algorithm	Broken
Diffie-Hellman	Discrete logarithm	Shor's algorithm	Broken
ECC (ECDH/ECDSA)	Elliptic curve DLP	Shor's algorithm	Broken
DSA	Discrete logarithm	Shor's algorithm	Broken
AES-128	Symmetric key	Grover's algorithm	Weakened (64-bit)
SHA-256	Hash function	Grover's algorithm	Weakened (128-bit)

Table 2: Cryptographic Systems Vulnerable to Shor's Algorithm

Grover's Algorithm

Grover's algorithm, developed by Lov Grover in 1996, provides a quadratic speedup for unstructured search problems. While less dramatic than Shor's exponential speedup, it significantly impacts symmetric cryptography.

Mathematical Foundation

For a search space of size N , classical algorithms require $O(N)$ operations in the worst case. Grover's algorithm reduces this to $O(\sqrt{N})$ quantum operations. The algorithm works through amplitude amplification:

Oracle: $O|x\rangle = (-1)^{f(x)} |x\rangle$

Diffusion: $D = 2|s\rangle\langle s| - I$

Grover Iterate: $G = D * O$

Iterations: $k = (\pi/4) * \sqrt{N}$

The oracle marks the solution by flipping its phase, and the diffusion operator amplifies the amplitude of the marked state through constructive interference.

Security Implications

Grover's algorithm effectively halves the security level of symmetric keys and hash outputs:

- AES-128 provides only 64-bit equivalent security against quantum adversaries
- AES-256 provides 128-bit equivalent security - still considered secure
- SHA-256 provides 128-bit collision resistance - requires upgrade to SHA-384 or SHA-512

Harvest Now, Decrypt Later (HNDL)

The HNDL threat is active today. Adversaries with long-term strategic interests are collecting and storing encrypted data, anticipating the future availability of quantum computers capable of decrypting it. This threat is particularly concerning for:

- National security communications and classified information
- Intellectual property and trade secrets
- Personal health records and financial data
- Long-term contractual agreements and legal documents

Data with a secrecy lifetime extending beyond 10-15 years should be considered at risk and prioritized for quantum-safe protection.

Post-Quantum Cryptography Standards

Post-Quantum Cryptography (PQC) refers to cryptographic algorithms designed to be secure against both classical and quantum computers. In August 2024, NIST finalized the first three PQC standards, marking a significant milestone in the transition to quantum-safe security.

NIST PQC Standards Overview

The National Institute of Standards and Technology (NIST) initiated the PQC standardization process in 2016. After extensive evaluation, three algorithms were standardized:

Standard	Algorithm	Function	Status
FIPS 203	ML-KEM	Key Encapsulation	Final (Aug 2024)
FIPS 204	ML-DSA	Digital Signatures	Final (Aug 2024)
FIPS 205	SLH-DSA	Hash-based Signatures	Final (Aug 2024)
FIPS 206	FN-DSA (Falcon)	Compact Signatures	Draft
SP 800-208	LMS/XMSS	Stateful Signatures	Final (Oct 2020)

Table 3: NIST Post-Quantum Cryptography Standards

ML-KEM: Module-Lattice-Based Key Encapsulation

ML-KEM (FIPS 203), derived from CRYSTALS-Kyber, is a key encapsulation mechanism based on the hardness of the Module Learning With Errors (MLWE) problem. It enables two parties to establish a shared secret key over an insecure channel.

Mathematical Foundation

The security of ML-KEM is based on the MLWE problem. Given a matrix A and vector b , where:

```
A <- R_q^(k x k) (random matrix over polynomial ring)
s, e <- error distribution
b = A * s + e (mod q)
```

The problem asks to distinguish (A, b) from uniform random samples. The polynomial ring R_q is defined as:

$$R_q = \mathbb{Z}_q[X] / (X^n + 1)$$

where n is a power of 2 (typically 256), and q is a prime modulus. This ring structure enables efficient computation via the Number Theoretic Transform (NTT).

Parameter Sets

Parameter Set	NIST Level	n	k	η
ML-KEM-512	Level 1 (~AES-128)	256	2	3
ML-KEM-768	Level 3 (~AES-192)	256	3	2
ML-KEM-1024	Level 5 (~AES-256)	256	4	2

Table 4: ML-KEM Parameter Sets

Algorithm Operations

ML-KEM consists of three main algorithms:

1. **KeyGen()**: Generates public key (pk) and secret key (sk)
2. **Encaps(pk)**: Generates ciphertext (ct) and shared secret (ss)
3. **Decaps(sk, ct)**: Recovers shared secret from ciphertext

ML-DSA: Module-Lattice-Based Digital Signatures

ML-DSA (FIPS 204), derived from CRYSTALS-Dilithium, provides digital signature functionality based on the Module Short Integer Solution (MSIS) and MLWE problems.

Signature Scheme

ML-DSA uses the Fiat-Shamir with Aborts construction, which involves:

1. **Key Generation**: Generate signing key and verification key
2. **Sign**: Generate signature using rejection sampling
3. **Verify**: Verify signature against message and public key

Security Levels

Parameter Set	NIST Level	d	tau	gamma1
ML-DSA-44	Level 2	13	39	2 ¹⁷
ML-DSA-65	Level 3	13	49	2 ¹⁹
ML-DSA-87	Level 5	13	60	2 ¹⁹

Table 5: ML-DSA Parameter Sets

Hash-Based Signatures

Hash-based signatures derive their security solely from the properties of cryptographic hash functions, making them conservative choices with well-understood security assumptions.

Stateful Hash-Based Signatures

Stateful schemes (LMS, XMSS) require maintaining state to track used keys. They are specified in NIST SP 800-208 and are primarily used for firmware and software signing.

Scheme	Public Key	Signature	Stateful	Use Case
LMS	32-64 B	1.5-3 KB	Yes	Firmware signing
HSS	32-64 B	2-6 KB	Yes	Multi-message signing
XMSS	64-132 B	2-8 KB	Yes	Code signing
XMSS ^{MT}	64-132 B	4-12 KB	Yes	High-volume signing
SLH-DSA	32-64 B	8-49 KB	No	General purpose

Table 6: Stateful Hash-Based Signature Comparison

Stateless Hash-Based Signatures (SLH-DSA)

SLH-DSA (FIPS 205), derived from SPHINCS+, eliminates the state management requirement by using a hypertree structure where the message itself determines which key to use. This comes at the cost of larger signatures (8-49 KB depending on security level).

Global PQC Standards and Policies

Different countries and standards bodies have developed varying approaches to PQC adoption:

Country/Agency	KEM	Signatures	Timeline
US (CNSA 2.0)	ML-KEM-1024	ML-DSA-87, LMS/XMSS	2035 (NSS)
UK (NCSC)	ML-KEM-768	ML-DSA-65, SLH-DSA	2030 target
Germany (BSI)	ML-KEM-768/1024	ML-DSA, SLH-DSA	Phased
France (ANSSI)	ML-KEM-768/1024	ML-DSA, FN-DSA	Phased
Australia (ASD)	ML-KEM-1024	ML-DSA-87	2030

Table 7: Global PQC Policy Comparison

Quantum Key Distribution

Quantum Key Distribution (QKD) uses quantum mechanics principles to enable two parties to produce a shared random secret key known only to them. Unlike PQC, QKD provides information-theoretic security based on the laws of physics rather than computational assumptions.

BB84 Protocol

The BB84 protocol, developed by Bennett and Brassard in 1984, was the first QKD protocol. It uses the polarization of photons to encode quantum bits.

Protocol Steps

1. **Preparation:** Alice randomly chooses a bit (0 or 1) and a basis (rectilinear + or diagonal x) for each photon
2. **Transmission:** Alice prepares and sends photons to Bob through a quantum channel
3. **Measurement:** Bob randomly chooses a basis for each photon and measures it
4. **Sifting:** Alice and Bob publicly compare bases and keep only matching measurements
5. **Error Estimation:** They estimate the Quantum Bit Error Rate (QBER) by comparing a subset of bits
6. **Error Correction:** Reconcile discrepancies using protocols like Cascade
7. **Privacy Amplification:** Apply hash functions to remove any partial information Eve may have

Security Foundation

BB84 security relies on fundamental quantum principles:

- No-Cloning Theorem: Unknown quantum states cannot be perfectly copied
- Measurement Collapse: Measuring a quantum state disturbs it irreversibly
- Non-orthogonal States: Cannot distinguish between non-orthogonal states with certainty

Any eavesdropping attempt introduces detectable errors. If QBER exceeds a threshold (typically 11% for BB84), the protocol is aborted.

E91 Protocol

The E91 protocol, proposed by Ekert in 1991, uses quantum entanglement and Bell's inequality violation to guarantee security. A trusted source generates entangled photon pairs and sends one photon to each party.

Entanglement-Based Security

The E91 protocol uses the Bell state $|\Phi^{++}\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. When Alice and Bob measure entangled photons in the same basis, they obtain perfectly correlated results. Security is verified by testing Bell's inequality - any eavesdropping reduces the correlation below the quantum bound.

Hybrid BB84-E91 Protocol

Modern QKD systems often combine BB84 and E91 advantages: BB84's efficiency for key generation with E91's entanglement-based security verification. The hybrid protocol provides enhanced security against both individual and collective attacks.

QKD Implementation Considerations

Channel Types

- Optical Fiber: Standard telecom fiber (0.2 dB/km loss), distances up to 400+ km with quantum repeaters
- Free Space: Satellite-based QKD enabling global coverage (Micius satellite achieved 1200 km)

Practical Limitations

- Distance Limitations: Photon loss in fiber limits practical distances without quantum repeaters
- Key Rate: Current systems achieve Mbps key rates over short distances, dropping with distance
- Authentication: QKD requires an authenticated classical channel, typically using PQC
- Side Channels: Implementation flaws can compromise theoretical security

QKD vs PQC

QKD and PQC serve different purposes and can be complementary:

Aspect	QKD	PQC
Security Basis	Physical laws	Computational hardness
Key Distribution	Direct generation	Mathematical derivation
Infrastructure	Specialized hardware	Software upgrade
Distance	Limited (~100km)	Unlimited
Authentication	Requires PQC	Self-authenticating
Deployment	Point-to-point links	Any network topology

Table 8: QKD vs PQC Comparison

Quantum Random Number Generation

Quantum Random Number Generators (QRNGs) leverage the inherent unpredictability of quantum mechanical processes to generate true randomness. Unlike pseudo-random number generators (PRNGs) that use deterministic algorithms, QRNGs produce entropy from fundamental physical processes.

Sources of Quantum Randomness

Quantum Vacuum Fluctuations

Even in a perfect vacuum, quantum field theory predicts zero-point energy fluctuations. These fluctuations can be measured using homodyne detection of the quadrature amplitude, producing truly random signals.

Photon Path Superposition

When a single photon encounters a beam splitter, quantum mechanics dictates it takes both paths simultaneously until measured. The detection at either output is fundamentally unpredictable, providing a source of true randomness.

Phase Noise in Lasers

Spontaneous emission in lasers creates quantum phase noise. By measuring the interference between two independent laser sources, this quantum noise can be extracted and digitized.

Tunneling Effects

Quantum tunneling of electrons through potential barriers is inherently probabilistic. The timing or occurrence of tunneling events can be converted to random bits.

QRNG Types

Type	Assumptions	Speed	Use Case
Trusted-device	Hardware functions correctly	Fast (Gbps)	General purpose
Source-independent	Detectors trusted	Medium	High assurance

Type	Assumptions	Speed	Use Case
Device-independent	No hardware trust needed	Slow (kbps)	Maximum security
Chip-based	Integrated verification	Fast	IoT, mobile devices

Table 9: QRNG Implementation Types

Entropy Assessment

Validating QRNG output requires comprehensive statistical testing:

- NIST SP 800-90B: Entropy assessment methodology
- NIST Statistical Test Suite: 15 tests for randomness
- Dieharder Tests: Extended battery of statistical tests
- Autocorrelation Analysis: Detects sequential dependencies

Applications in Cryptography

QRNGs are critical for cryptographic operations requiring high-quality entropy:

- Key Generation: Producing unpredictable encryption keys
- Nonce Generation: Ensuring unique values for each operation
- Initialization Vectors: Providing random starting states
- Seeding PRNGs: Providing initial entropy for deterministic generators

Modern systems increasingly integrate QRNGs at the chip level, with companies like ID Quantique, Quantinuum, and others offering certified quantum random number generation solutions.

DevSecOps Integration Framework

Integrating quantum-safe security into DevSecOps workflows requires a systematic approach that embeds cryptographic agility and PQC readiness throughout the software development lifecycle. This section provides a comprehensive framework for organizations to achieve quantum-resilient development practices.

Crypto-Agility Architecture

Crypto-agility is the ability to rapidly switch cryptographic algorithms and keys without significant changes to application code or infrastructure. This is essential for responding to new threats and migrating to PQC standards.

Design Principles

- **Algorithm Abstraction:** Cryptographic operations should use abstract interfaces, not hardcoded algorithms
- **Configuration-Driven:** Algorithm selection should be configurable at deployment or runtime
- **Version Management:** Cryptographic implementations should be versioned and updatable
- **Hybrid Support:** Systems should support both classical and PQC algorithms during transition

Implementation Pattern

```
// Crypto-Agile Implementation Pattern
interface CryptoProvider {
    KeyPair generateKeyPair(AlgorithmConfig config);
    byte[] encrypt(byte[] data, PublicKey key, AlgorithmConfig config);
    byte[] decrypt(byte[] data, PrivateKey key, AlgorithmConfig config);
    byte[] sign(byte[] data, PrivateKey key, AlgorithmConfig config);
    boolean verify(byte[] data, byte[] signature, PublicKey key, AlgorithmConfig config);
}
```

CI/CD Pipeline Integration

Quantum-safe security checks should be embedded at every stage of the CI/CD pipeline:

Source Code Analysis

- **Static Analysis:** Scan code for hardcoded cryptographic algorithms and vulnerable patterns

- **Dependency Scanning:** Identify cryptographic libraries and their quantum safety status
- **Crypto Bill of Materials (CBOM):** Generate and validate cryptographic inventory

Example: GitLab CI Integration

```
quantum-safe-scan:
  stage: security
  image: quantum-safe-scanner:latest
  script:
    - qs-explorer scan --format cbom --output crypto-inventory.json
    - qs-advisor analyze --input crypto-inventory.json --policy nist-2035
    - qs-validator check --cbom crypto-inventory.json --fail-on-high
```

Build and Test

- **PQC Algorithm Testing:** Validate PQC implementations against NIST test vectors
- **Performance Benchmarking:** Measure latency, throughput, and resource usage
- **Interoperability Testing:** Verify compatibility with other PQC implementations

Deployment

- **Gradual Rollout:** Deploy PQC in phases, starting with non-critical systems
- **Monitoring:** Track cryptographic operations and performance metrics
- **Rollback Capability:** Maintain ability to revert to classical algorithms if needed

Cryptographic Inventory Management

A comprehensive cryptographic inventory is the foundation of quantum readiness. IBM's CIO Office demonstrated this approach by scanning nearly 6,000 repositories and 47 million lines of code, identifying over 3,900 vulnerabilities requiring quantum-safe remediation.

Inventory Components

- **Algorithms:** All cryptographic algorithms in use (RSA, ECC, AES, SHA, etc.)
- **Key Sizes:** Current key lengths and their quantum safety levels
- **Libraries:** Cryptographic libraries and versions
- **Protocols:** TLS, SSH, VPN, and other cryptographic protocols

- Certificates: X.509 certificates and their algorithms
- Data Classification: Sensitivity and lifetime of protected data

Discovery Tools

- IBM Quantum Safe Explorer: Static code analysis for cryptographic discovery
- Fortinet Crypto Inventory: Network-based cryptographic asset discovery
- OpenSSL Scanner: TLS/SSL configuration analysis
- Custom Scripts: Organization-specific discovery automation

Hybrid Cryptographic Deployments

During the transition period, hybrid approaches combining classical and PQC algorithms provide both backward compatibility and forward security.

Hybrid Key Exchange

Combines a classical key exchange (e.g., ECDH) with a PQC key exchange (e.g., ML-KEM):

```
shared_secret = KDF(classical_shared_secret || pqc_shared_secret)
```

If either algorithm is compromised, the combined secret remains secure. This approach is specified in RFC 9794 and supported by major TLS implementations.

Dual Signatures

Sign messages with both classical and PQC signature schemes. Verifiers must validate both signatures for acceptance. This provides immediate PQC protection while maintaining compatibility with legacy systems.

Strategy	Description	Use Case
Hybrid Key Exchange	Combine classical + PQC KEMs	TLS, VPN connections
Dual Signatures	Sign with both algorithms	Code signing, certificates
Algorithm Agility	Runtime algorithm selection	Long-term systems

Strategy	Description	Use Case
Crypto Proxy	Transparent PQC translation	Legacy integration

Table 10: Hybrid Deployment Strategies

Quantum-Safe Software Development Lifecycle

The Quantum-Safe SDLC (Q-SDLC) extends traditional secure development practices with quantum threat modeling, PQC integration, and crypto-agility requirements. This section outlines the enhanced phases and activities for quantum-resilient software development.

Phase 1: Requirements and Planning

Quantum Threat Modeling

Extend traditional threat modeling to include quantum-specific threats:

- Data Lifetime Analysis: Determine how long data must remain confidential
- HNDL Risk Assessment: Evaluate exposure to Harvest Now, Decrypt Later attacks
- Regulatory Compliance: Identify applicable PQC mandates (CNSA 2.0, etc.)
- Interdependencies: Map cryptographic dependencies across systems

Security Requirements

- PQC Algorithm Selection: Choose appropriate NIST-approved algorithms
- Crypto-Agility Requirements: Define algorithm replacement procedures
- Hybrid Mode Support: Specify when hybrid cryptography is required
- Performance Budgets: Establish acceptable overhead for PQC operations

Phase 2: Design

Cryptographic Architecture

Design systems with quantum-safe principles:

- Layered Security: Apply defense in depth with multiple cryptographic controls
- Algorithm Isolation: Encapsulate cryptographic operations in dedicated modules
- Key Hierarchy: Design key derivation and management for PQC key sizes
- Protocol Selection: Choose quantum-safe or hybrid protocols

Data Protection Strategy

Classify data and apply appropriate protection:

Classification	Lifetime	Protection Required
Critical/Secret	>15 years	PQC mandatory, QKD optional
Confidential	5-15 years	PQC recommended
Internal	<5 years	Hybrid acceptable
Public	N/A	Standard encryption

Table 11: Data Protection Strategy by Classification

Phase 3: Implementation

Secure Coding Practices

- Use Approved Libraries: Only use NIST-validated or well-vetted cryptographic libraries
- Avoid Hardcoded Algorithms: Use configuration for algorithm selection
- Proper Key Handling: Implement secure key generation, storage, and destruction
- Side-Channel Resistance: Consider timing, power, and cache side-channels

```
// Example: Crypto-Agile Implementation
public class QuantumSafeEncryption {
    private final CryptoConfig config;

    public byte[] encrypt(byte[] plaintext, PublicKey key) {
        Algorithm algorithm = config.getAlgorithm("key-encapsulation");
        switch (algorithm) {
            case ML_KEM_768:
                return encryptWithMLKEM(plaintext, key);
            case ML_KEM_1024:
                return encryptWithMLKEM1024(plaintext, key);
            default:
```

```
        throw new UnsupportedOperationException(algorithm);
    }
}
}
```

Phase 4: Testing

Cryptographic Testing

- Conformance Testing: Verify compliance with NIST specifications
- Interoperability Testing: Test with other PQC implementations
- Performance Testing: Measure and validate performance characteristics
- Negative Testing: Verify proper handling of invalid inputs and attacks

Security Testing

- Penetration Testing: Include quantum-specific attack scenarios
- Fuzzing: Test cryptographic implementations with malformed inputs
- Static Analysis: Scan for cryptographic anti-patterns

Phase 5: Deployment

Gradual Rollout

- Canary Deployments: Test PQC in production with limited traffic
- Feature Flags: Enable/disable PQC via configuration
- Monitoring: Track cryptographic operations and errors

Phase 6: Operations and Maintenance

Continuous Monitoring

- Crypto Posture: Monitor cryptographic configurations across environments
- Vulnerability Tracking: Stay informed of new quantum computing developments
- Compliance Auditing: Regular verification of PQC requirements

Incident Response

- Crypto-Agility Activation: Procedures for rapid algorithm replacement
- Key Revocation: Plans for large-scale key replacement if needed
- Communication: Stakeholder notification procedures

Mathematical Foundations

Understanding the mathematical foundations of quantum computing and post-quantum cryptography is essential for implementing quantum-safe systems. This section provides the key mathematical concepts and equations underlying the technologies discussed in this report.

Quantum Mechanics Fundamentals

Dirac Notation

Quantum states are represented using Dirac (bra-ket) notation. A quantum state $|\psi\rangle$ (ket) is a column vector in a complex Hilbert space. The conjugate transpose $\langle\psi|$ (bra) is a row vector.

Inner product: $\langle\phi|\psi\rangle = \sum_i \phi_i^* \psi_i$

Outer product: $|\psi\rangle\langle\phi| = \text{matrix with elements } \psi_i \phi_j^*$

Norm: $\| |\psi\rangle \| = \sqrt{\langle\psi|\psi\rangle}$

Quantum Gates

Quantum computations are performed using unitary gates. Common single-qubit gates include:

Hadamard: $H = (1/\sqrt{2}) * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Pauli-X: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Pauli-Y: $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$

Pauli-Z: $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Phase: $S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

PI/8: $T = \begin{bmatrix} 1 & 0 \\ 0 & e^{(i\pi/4)} \end{bmatrix}$

Two-qubit gates enable entanglement. The CNOT (Controlled-NOT) gate is fundamental:

$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Lattice-Based Cryptography Mathematics

Lattice Definitions

A lattice is a discrete additive subgroup of $\mathbb{R}^n$. Given n linearly independent basis vectors $b_1, b_2, \dots, b_n$ in $\mathbb{R}^n$, the lattice L is:

$$L = \{ \sum_{i=1}^n (x_i * b_i) : x_i \in \mathbb{Z} \}$$

The same lattice can have many different bases. Some bases are "good" (short, nearly orthogonal vectors) while others are "bad" (long, nearly parallel vectors).

Hard Lattice Problems

The security of lattice-based cryptography relies on the hardness of certain problems:

Shortest Vector Problem (SVP):

Given a lattice L , find the shortest non-zero vector v in L

minimize $||v||$ subject to v in L , $v \neq 0$

Closest Vector Problem (CVP):

Given a lattice L and target vector t , find v in L closest to t

minimize $||t - v||$ subject to v in L

Shortest Independent Vectors Problem (SIVP):

Given a lattice L , find n linearly independent short vectors

Learning With Errors (LWE)

The LWE problem, introduced by Regev in 2005, is the foundation of many PQC schemes:

Parameters: n (dimension), q (modulus), error distribution χ

Secret: $s \leftarrow \mathbb{Z}_q^n$

Samples: $(a, b = \langle a, s \rangle + e \bmod q)$ where $a \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi$

The decisional LWE problem asks to distinguish LWE samples from uniform random pairs. Regev proved that solving LWE is at least as hard as solving worst-case lattice problems.

Ring-LWE and Module-LWE

Ring-LWE works over polynomial rings for efficiency:

$R = \mathbb{Z}[X] / (X^n + 1)$ where n is power of 2

$R_q = \mathbb{Z}_q[X] / (X^n + 1)$

Module-LWE generalizes to modules over these rings, providing a balance between security and efficiency. ML-KEM uses Module-LWE with module dimension k (typically 2, 3, or 4).

Hash-Based Signature Mathematics

Merkle Trees

Hash-based signatures use Merkle trees to authenticate multiple one-time signatures:

Tree height: h

Number of leaves: 2^h

Leaf i contains: $H(\text{public_key}_i)$

Internal node: $H(\text{left_child} || \text{right_child})$

Root is the public key

Winternitz One-Time Signatures (WOTS)

WOTS signs messages using hash chains:

```
Parameters: n (security parameter), w (Winternitz parameter)
Chain length: w - 1
Number of chains: l = ceil(n / log2(w)) + ceil(log2(l*(w-1)) / log2(w))
Private key: random x_1, ..., x_l
Public key: f^(w-1)(x_1), ..., f^(w-1)(x_l)
```

Quantum Algorithm Complexity

Shor's Algorithm Complexity

```
Classical factoring: O(exp((64/9)^(1/3) * (ln N)^(1/3) * (ln ln N)^(2/3)))
Shor's algorithm: O((log N)^3)
Qubits required: ~2*log N for factoring N-bit integer
```

Grover's Algorithm Complexity

```
Classical search: O(N)
Grover's algorithm: O(sqrt(N))
Qubits required: log2(N)
Iterations: (pi/4) * sqrt(N)
```

Security Level Calculations

NIST Security Categories

NIST defines five security categories based on the hardness of breaking AES at various key lengths:

```
Level 1: At least as hard as AES-128 (exhaustive key search)
Level 2: At least as hard as SHA-256 (collision search)
Level 3: At least as hard as AES-192 (exhaustive key search)
Level 4: At least as hard as SHA-384 (collision search)
Level 5: At least as hard as AES-256 (exhaustive key search)
```

In the quantum setting, Grover's algorithm reduces the security level by half for symmetric cryptography. Therefore, AES-256 provides 128-bit quantum security (Level 3 equivalent).

Implementation Roadmap

This 3-month implementation roadmap provides a structured approach to achieving quantum-safe security posture. The roadmap is divided into three phases: Discovery and Assessment, Pilot Implementation, and Production Deployment.

Month 1: Discovery and Assessment

Week 1-2: Cryptographic Inventory

- Deploy discovery tools across all environments
- Scan all code repositories for cryptographic usage
- Inventory network cryptographic configurations
- Generate initial Crypto Bill of Materials (CBOM)

Week 3: Risk Assessment

- Classify data by sensitivity and lifetime
- Assess HNDL exposure for each data class
- Identify critical systems and dependencies
- Map regulatory compliance requirements

Week 4: Gap Analysis and Planning

- Compare current state against target state
- Prioritize systems for migration
- Develop detailed migration plans
- Establish metrics and success criteria

Week	Activities	Deliverables
1-2	Cryptographic inventory	Complete CBOM for all systems
3	Risk assessment	Risk matrix and HNDL analysis

Week	Activities	Deliverables
4	Gap analysis	Migration priority list and plan

Table 12: Month 1 Activities and Deliverables

Month 2: Pilot Implementation

Week 5-6: Test Environment Setup

- Deploy PQC libraries in test environment
- Configure hybrid cryptographic modes
- Establish performance baselines
- Implement monitoring and logging

Week 7: Application Integration

- Select pilot applications (low-risk, high-value)
- Implement crypto-agile interfaces
- Integrate PQC key exchange and signatures
- Update certificate management

Week 8: Testing and Validation

- Execute functional test suites
- Perform performance benchmarking
- Conduct interoperability testing
- Validate against NIST test vectors

Week	Activities	Deliverables
5-6	Test environment setup	Configured PQC testbed

Week	Activities	Deliverables
7	Pilot integration	Crypto-agile interfaces
8	Testing and validation	Test reports and benchmarks

Table 13: Month 2 Activities and Deliverables

Month 3: Production Deployment

Week 9-10: Staged Rollout

- Deploy to non-production environments
- Enable canary deployments for critical services
- Monitor performance and error rates
- Establish rollback procedures

Week 11: Production Deployment

- Deploy to production in phases
- Monitor cryptographic operations
- Track key performance indicators
- Document lessons learned

Week 12: Optimization and Handoff

- Optimize performance based on production data
- Finalize documentation and runbooks
- Train operations teams
- Establish ongoing monitoring

Week	Activities	Deliverables
9-10	Staged rollout	Non-prod PQC deployment
11	Production deployment	Production PQC enabled
12	Optimization	Tuned configurations, documentation

Table 14: Month 3 Activities and Deliverables

Key Performance Indicators

KPI	Target	Measurement
Cryptographic coverage	>95%	CBOM completeness
PQC migration progress	>80% critical systems	Systems using PQC
Performance overhead	<20% latency increase	Benchmark comparison
Error rate	<0.1%	Cryptographic failures operation
Compliance	100%	Audit findings

Table 15: Key Performance Indicators

Risk Mitigation

- Performance Degradation: Implement caching, hardware acceleration, and algorithm tuning
- Compatibility Issues: Maintain hybrid modes and gradual transition paths
- Implementation Bugs: Use well-vetted libraries and comprehensive testing
- Operational Complexity: Automate key management and monitoring

Conclusion

The quantum threat to modern cryptography is real and imminent. While cryptographically relevant quantum computers may still be years away, the "Harvest Now, Decrypt Later" threat demands immediate action. Organizations that delay their quantum-safe transition risk exposing sensitive data to future decryption.

Key Takeaways

- **Start Now:** Begin cryptographic inventory and risk assessment immediately
- **Crypto-Agility:** Design systems that can rapidly switch cryptographic algorithms
- **Hybrid Approach:** Use hybrid cryptography during the transition period
- **Standards Compliance:** Follow NIST PQC standards (ML-KEM, ML-DSA, SLH-DSA)
- **DevSecOps Integration:** Embed quantum-safe practices throughout the SDLC

Future Considerations

The field of post-quantum cryptography continues to evolve. Organizations should:

- Monitor NIST and other standards bodies for algorithm updates
- Stay informed about quantum computing advances
- Participate in industry working groups and information sharing
- Maintain flexibility to adapt to new developments

Call to Action

The quantum-safe transition is not a one-time project but an ongoing journey. By following the framework and roadmap outlined in this report, organizations can build resilient, future-proof security architectures that protect against both classical and quantum threats.

The time to act is now. Every day of delay increases the risk of future compromise. Begin your quantum-safe transformation today and secure your organization's digital future.



Quantum DevSecOps Strategy

Comprehensive Technical Report

April 2026

© 2026 Sovereign Systems Labs. All rights reserved.