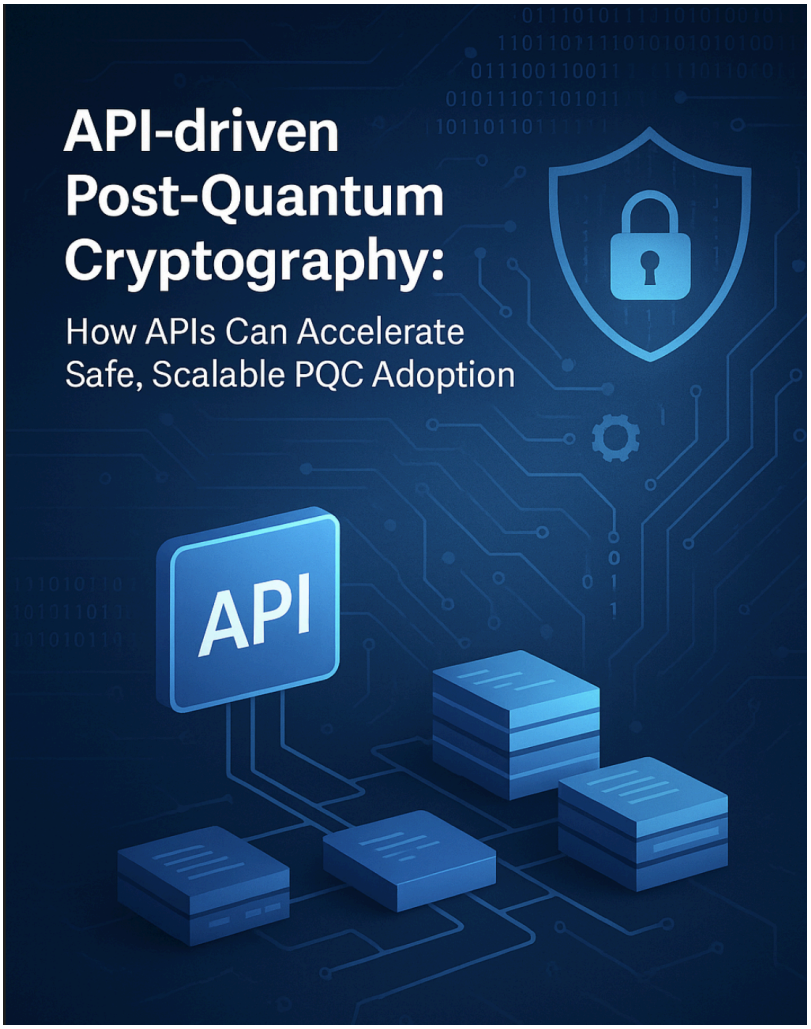


Authors:
Samuel Tseitkin, ExeQuantum, Australia.
Associate Professor Nalin Arachchilage, RMIT University, Australia.

1. Abstract
2. The Urgency for Post-Quantum Cryptography
    2.1 The Threat of Quantum Computing
    2.2 Industry Deadlines and Regulatory Pressure
    2.3 Implementation Bottlenecks and Risk
3. Why APIs? Making PQC Plug-and-Play
    3.1 Framework Agnosticism
    3.2 Plug-and-Play Integration
    3.3 Side-Channel Mitigation
    3.4 Simplified Upgrades
4. Standardization and Interoperability
    4.1 Avoiding Fragmentation
    4.2 Unified Cryptographic Behavior
    4.3 Version Control and Update Agility
5. Security Considerations for API-based PQC
    5.1 Key Exchange and Transport Protection
    5.2 PQ-TLS and Pre-shared Key Options
    5.3 On-premise Hosting for High Sensitivity Use
    5.4 Backend Transparency and Trust
6. Usability considerations for API-based PQC
    6.1 Tools and guidelines available for implementing PQC
    6.2 Challenges Developers Face When Using PQC Tools and Guidelines
    6.3 Solutions to Address Usability Challenges
    6.4 Towards Developer-Centric PQC Adoption
7. Performance and Latency Optimization
    7.1 Edge and Client-near Deployment
    7.2 Persistent Connections and Protocol Choices
    7.3 Caching, Queuing, and Load Management
8. Case Study: Quantum-Safe IoT Deployment
    8.1 Challenges in Constrained Environments
    8.2 API-based PQC Implementation
    8.3 Security and Performance Outcomes
9. Conclusion
    9.1 The Role of APIs in Post-Quantum Adoption

API-driven Post-Quantum Cryptography: How APIs Can Accelerate Safe, Scalable PQC Adoption

Abstract

Post-quantum cryptography (PQC) promises long-term protection against emerging quantum threats, but real-world adoption remains slow due to the complexity, cost, and risk of implementation. This white paper explores an infrastructure-centric solution: API-driven PQC. By abstracting quantum-safe primitives behind secure, standardized APIs, organizations can accelerate adoption while minimizing cryptographic integration errors, achieving language interoperability, and mitigating side-channel attack surfaces through cloud distribution.

We outline the practical benefits of API-based delivery, such as framework agnosticism, plug-and-play deployment, and ease of updates, while addressing key challenges such as key exchange security, backend assurance, and latency. Strategies including PQ-TLS, pre-shared keys, internal hosting, regional API distribution, and third-party certification are explored to ensure robust implementation even in high-sensitivity environments.

Co-authored by Samuel Tseitkin of ExeQuantum and Associate Professor Nail Arachchilage of RMIT University, this paper provides a roadmap for secure, scalable PQC integration across diverse digital ecosystems, from startups to critical infrastructure, without the burden of in-house cryptographic expertise.

The Urgency for Post-Quantum Cryptography

Modern digital infrastructure relies heavily on public-key cryptography, which underpins everything from secure messaging and financial transactions to identity verification and software updates. These systems were built on the assumption that certain mathematical problems, like integer factorization and discrete logarithms, were too hard to solve within a reasonable timeframe. That assumption no longer holds.

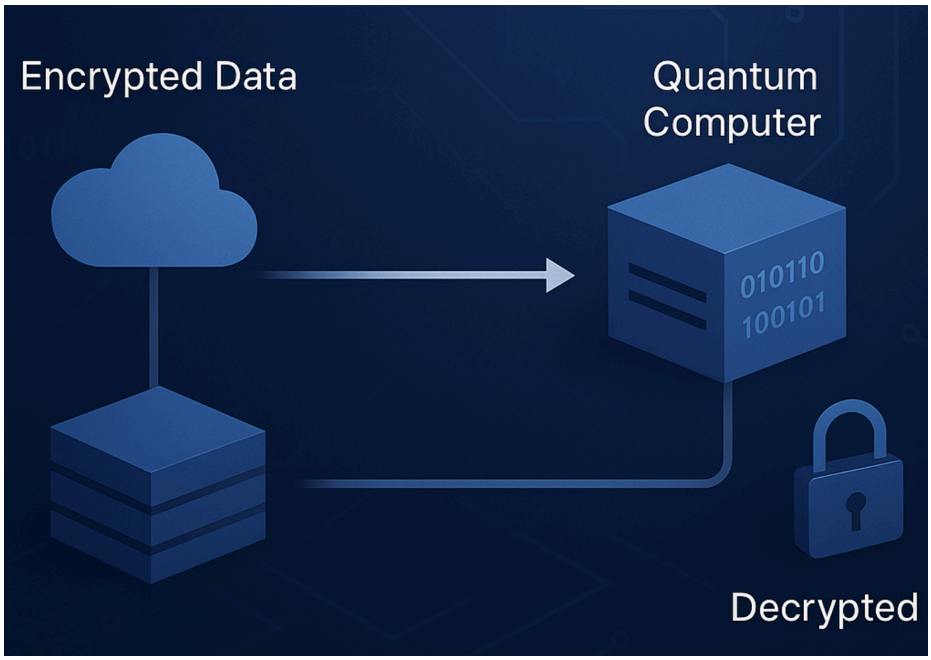
The emergence of quantum computing introduces a paradigm shift. Shor’s algorithm, in particular, demonstrates that once quantum hardware reaches sufficient scale, it will be capable

of breaking RSA and ECC, two of the most widely deployed cryptographic schemes today. Even symmetric algorithms like AES, while not directly threatened, rely on quantum-insecure asymmetric key exchange algorithms as Diffie-Heilman as physically pre-sharing keys at scale is not scalable.

This creates a ticking clock for institutions around the world. Regulatory bodies have begun to react. In Australia, the Information Security Manual outlines 2030 as a transition deadline for quantum-resistant algorithms. Globally, organizations like NIST and ETSI are formalizing post-quantum cryptographic standards, with the first round of NIST-approved algorithms already selected.

According to Gartner, by 2029, organizations that have not begun adopting post-quantum cryptography will face a significant risk of data compromise. This warning reflects a growing industry consensus that quantum threats are not theoretical, and that proactive mitigation is necessary to avoid operational, legal, and reputational fallout. Businesses that delay implementation until the final moment may find themselves scrambling to retrofit critical systems, often at greater cost and with higher risk of integration failure.

The timeline is further compressed by “harvest now, decrypt later” attacks, where encrypted data is stolen today with the intent to decrypt it in the future once quantum computers are capable. Sensitive data with long-term confidentiality requirements is already at risk.



Despite clear urgency, implementation remains difficult. Transitioning to PQC is not simply a drop-in upgrade. It involves:

- Navigating a new class of cryptographic algorithms, many of which are still maturing in terms of implementation quality and community understanding.

- Managing integration across multiple programming languages, libraries, and platforms, each with their own compatibility challenges and performance characteristics.

- Avoiding security pitfalls such as side-channel vulnerabilities, which can arise when algorithms are deployed in environments that were not designed for constant-time or fault-tolerant execution.

- Keeping systems agile enough to support cryptographic agility, allowing updates or rollbacks as standards evolve (“Crypto-Agility”).

For many organizations, especially those without a dedicated cryptography team, these barriers are too high. This creates a dangerous inertia where decision-makers know that change is necessary but feel unequipped to implement it safely. Security teams are caught between the urgency to act and the risk of acting improperly.

What’s needed is a pathway that allows organizations to begin integrating quantum-safe cryptography in a manageable, scalable way that reduces the likelihood of introducing new vulnerabilities. That’s where API-driven infrastructure comes in.

Why APIs? Making PQC Plug-and-Play

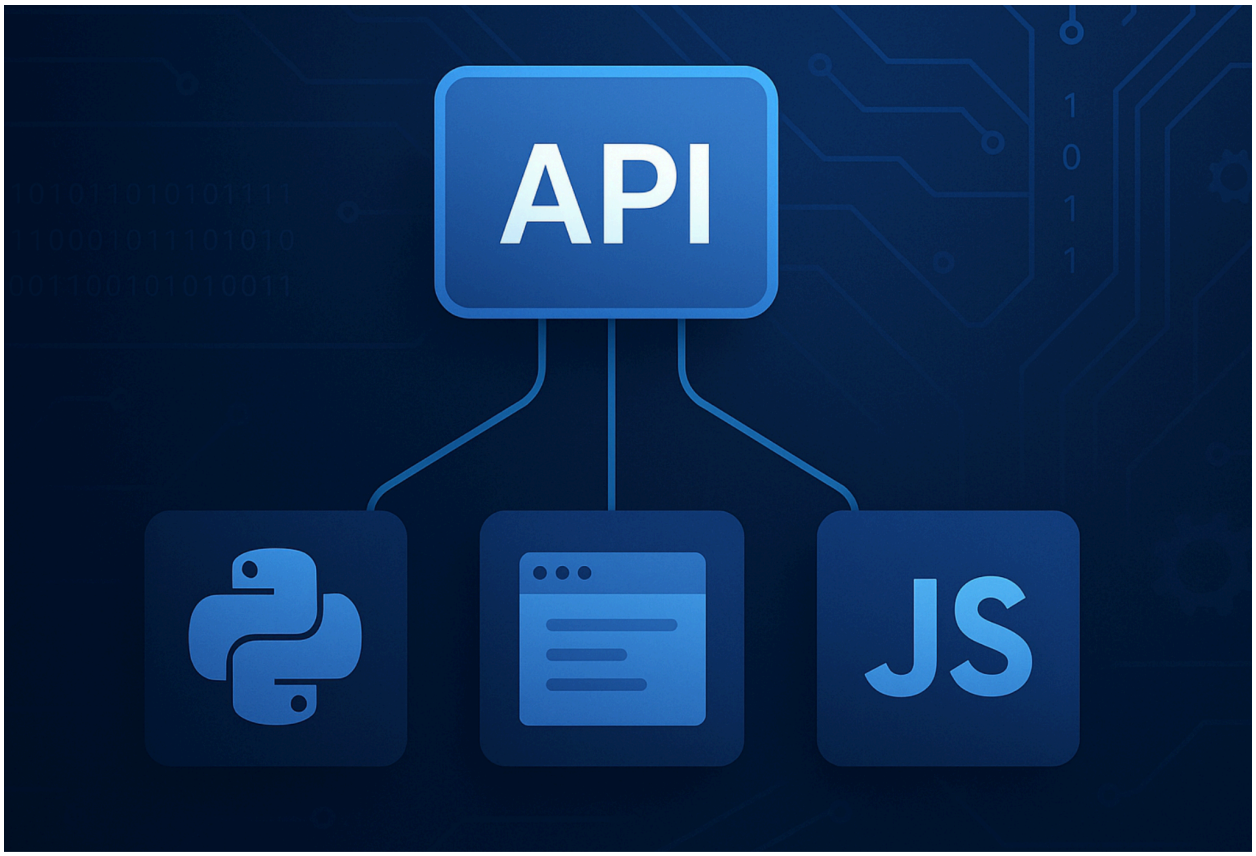
Most cryptographic failures aren’t due to broken algorithms. They’re due to broken implementations. Inconsistent libraries, incorrect parameter choices, and flawed usage patterns remain some of the most common causes of security breaches. This risk is even greater with post-quantum cryptography, where developers face unfamiliar primitives, evolving standards, and implementation gaps across languages.

API infrastructure offers a practical solution to this complexity. Rather than requiring teams to integrate raw cryptographic libraries into their systems, APIs provide a clean abstraction layer

that delivers quantum-safe cryptographic functionality in a consistent, testable, and language-agnostic format. This shift from code-level integration to service-level consumption dramatically reduces risk and accelerates deployment.

One of the most immediate advantages is framework agnosticism. APIs can be called from any environment, whether JavaScript, Python, Java, or embedded C, without requiring native PQC support. This allows organizations to begin using quantum-safe encryption across their full stack, even in legacy systems or environments where language bindings for PQC libraries do not yet exist.

APIs are also inherently plug-and-play. Developers don’t need to understand the underlying mathematics or memory management of lattice-based cryptography to start using ML-KEM or ML-DSA. They can simply call an endpoint, receive a response, and focus on their business logic. This model mirrors the adoption patterns of other complex technologies like AI, which transitioned from in-house model development to widespread use via cloud APIs.



A more subtle but equally important benefit is side-channel resistance. When cryptographic operations are performed on a cloud server rather than locally, attackers are less able to monitor power usage, timing patterns, or electromagnetic emissions. While no system is invulnerable, hosting sensitive cryptographic routines behind an API significantly increases the difficulty of mounting a successful side-channel attack. This is especially relevant in a post-quantum context, where performance variability and large key sizes can increase susceptibility if not carefully managed.

APIs also enforce standardization across platforms. When each team implements PQC independently, there is a risk that different versions of the same algorithm behave inconsistently. Even small differences, like how randomness is handled, how keys are serialized, or how parameters are interpreted, can lead to interoperability issues or security regressions. APIs abstract this away. The same logic, maintained in a central service, can be used everywhere, ensuring consistency and reducing room for error.

Finally, APIs allow for centralized updates and cryptographic agility. As standards evolve and new algorithms emerge, API providers can upgrade their internal implementations without requiring clients to redeploy software. This not only reduces long-term maintenance overhead, but also supports faster response to newly discovered vulnerabilities or performance optimizations.

By offering PQC as a service, APIs convert a high-risk integration project into a safe, modular upgrade path. Organizations can adopt quantum-safe capabilities in days rather than months, while maintaining confidence in the correctness, consistency, and security of their cryptographic foundation.

Standardization and Interoperability

One of the most overlooked challenges in adopting post-quantum cryptography is maintaining consistency across diverse systems and programming environments. PQC algorithms, particularly those recently standardized by NIST, are still new to most development teams. Native implementations vary widely in terms of language support, documentation quality, performance characteristics, and compatibility with existing infrastructure.

Without a central point of control, organizations run the risk of building fragmented systems where different teams integrate PQC in different ways. One team may use a Python binding of

an ML-KEM implementation. Another may work with a C++ version compiled from a different reference. Over time, these inconsistencies lead to serious problems. Keys may be encoded differently. Parameters may be misaligned. Updates may only propagate to parts of the system, leaving others exposed.

API-driven cryptography offers a reliable path to avoid this fragmentation. By centralizing the cryptographic logic within a service and exposing a standard interface, organizations can ensure that every system interacts with the same algorithms in the same way. This includes not just the algorithm itself but also the encoding of keys, the management of random number generation, the response formats, and the validation of inputs.

This approach provides two major benefits. First, it improves interoperability across teams and platforms. Systems written in JavaScript, Python, Go, and Rust can all access the same cryptographic backend using HTTP or gRPC, without the need to compile or audit different libraries for each environment. Second, it simplifies version control. When a new version of an algorithm is released or a vulnerability is discovered, the API provider can update the backend without forcing downstream teams to redeploy or refactor their code.

Additionally, APIs support easier testing and validation. Developers can run integration tests against a fixed endpoint, confident that the results will remain consistent over time. This is especially important in regulated environments where reproducibility and auditability are mandatory.

Finally, standardization through APIs aligns with long-term cryptographic agility. As the threat landscape changes and new quantum-safe algorithms are approved, organizations will need to pivot quickly. An API-based model ensures that these pivots can happen in one place, with minimal disruption to clients and without reintroducing implementation risk.

By placing the complexity of PQC behind a stable, language-neutral interface, APIs allow organizations to unify their security posture across applications, services, and devices.

Security Considerations for API-based PQC

While APIs offer a powerful solution to the complexity of post-quantum cryptography, they also introduce a new set of security considerations. Moving cryptographic operations to an external service changes the threat model. The API becomes part of the attack surface, and protecting the confidentiality, integrity, and authenticity of cryptographic transactions becomes paramount.

The first and most obvious concern is key transport security. Even if a secure post-quantum algorithm is used within the API backend, the communication channel between the client and server must also be protected. If this link is compromised, encrypted data or cryptographic material can be intercepted in transit. A classical TLS handshake is no longer sufficient. Instead, organizations must adopt PQ-TLS, a hybrid key exchange that combines post-quantum and classical mechanisms to safeguard against both current and future threats.

In environments where PQ-TLS is impractical or where additional assurance is required, alternative mechanisms can be used. Pre-shared keys (PSKs) offer a viable option, especially in constrained environments like IoT devices or air-gapped systems. Here, the key exchange step is removed entirely and replaced with a static shared secret. This may limit agility, but it ensures that even if the communication channel is intercepted, no key material is exposed.

Another solution is on-premise API hosting. For highly sensitive applications such as national security, defense, or critical infrastructure, the cryptographic API can be deployed inside the client's internal network. This eliminates reliance on external network security and places control entirely in the hands of the organization. The API model still applies, but all communication remains local, reducing exposure.

Latency is another key consideration. While cloud-hosted APIs are convenient, round-trip time between the client and the server can introduce delays, especially in real-time systems or edge devices. Solutions include edge deployment, where API endpoints are hosted close to the client using CDNs or regional infrastructure, or deploying the service directly inside the client's own environment. Load balancing and caching can also help reduce unnecessary cryptographic calls. It is crucial to remember not to overuse the key exchange, rather doing it in a session-based manner.

Finally, backend assurance is crucial. Clients must trust that the algorithms, implementations, and entropy sources used by the API are secure and correctly maintained. Transparency mechanisms are essential. This includes independent security audits, formal code reviews, algorithm certifications (such as FIPS 140-3 or 140-4), and public documentation of update procedures. Without this transparency, clients may hesitate to delegate control of their cryptographic operations.

By addressing these challenges head-on, organizations can deploy PQC through APIs with a high degree of confidence. Strong transport security, deployment flexibility, performance

optimization, and verifiable assurance frameworks form the foundation of a secure API-based cryptographic strategy.

Usability Considerations for API-based PQC

As organizations move towards adopting API-driven post-quantum cryptography (PQC), usability becomes a critical factor influencing the success of implementation. Despite the availability of tools and guidelines to support PQC integration, developers, especially those without deep cryptographic expertise, face several usability challenges. Addressing these issues is essential to ensure safe, scalable, and efficient adoption.

Tools and Guidelines Available for Implementing PQC

A variety of tools, libraries, and guidelines have emerged to facilitate the integration of PQC into software development:

- **NIST PQC Standards and Draft Guidelines**: NIST has standardized four algorithms (ML-KEM, Falcon, Dilithium, and SLH-DSA) and released an initial public draft for PQC migration strategies. These documents (standards and guidelines) provide critical baseline recommendations for developers.
- **Open-Source PQC Libraries**: Implementations like Open Quantum Safe (OQS), liboqs, and pqcrypto offer open-source libraries that developers can integrate into their applications.
- **API Services for PQC**: Emerging platforms, such as ExeQuantum PQCaaS, offer APIs that abstract the complexity of PQC, providing developers with easy-to-use endpoints for key encapsulation, signatures, and encryption.
- **Migration Playbooks**: Organizations like ENISA and ANSSI have published transition guides outlining best practices for replacing classical cryptography with quantum-safe alternatives.
- **Vendor SDKs and PQC Modules**: Cloud providers and cryptography vendors increasingly provide SDKs with built-in support for PQC primitives, simplifying integration for application developers.

These resources aim to bridge the gap between cryptographic theory and practical implementation, especially when developers adopt API-driven post-quantum cryptography (PQC) into their software applications they build.

Challenges Developers Face When Using PQC Tools and Guidelines

Despite the availability of these resources, developers encounter several challenges when adopting PQC:

- **Complexity of New Algorithms**: Lattice-based and hash-based algorithms introduce concepts unfamiliar to most software engineers, making it difficult to correctly integrate or optimize them without cryptographic expertise.
- **Fragmentation and Inconsistent Implementations**: Different libraries may interpret PQC standards differently, leading to interoperability issues across platforms and applications.
- **Performance Overhead**: PQC algorithms typically require larger key sizes and heavier computational resources compared to classical counterparts, complicating their use in resource-constrained environments such as mobile devices or IoT hardware.
- **Lack of Practical Migration Guidelines**: While high-level transition plans exist, detailed, context-specific migration blueprints for real-world applications (e.g., web services, mobile apps, embedded systems) are often lacking.
- **Limited Testing and Debugging Tools**: Tools for testing, validating, and benchmarking PQC integrations remain immature compared to the mature ecosystem around traditional cryptography (e.g., for RSA or ECC).
- **Security Pitfalls with API Integration**: Developers may mishandle sensitive operations like key exchange or session management even when using PQC APIs, exposing applications to classical vulnerabilities such as man-in-the-middle attacks.

Solutions to Address Usability Challenges

Addressing the above challenges requires a multi-pronged approach involving both technical and educational strategies:

- **Development of Higher-Order Abstractions**: APIs should further abstract complexity by offering high-level constructs (e.g., "secureSession.start()") rather than exposing raw cryptographic primitives, making usage intuitive and error-resistant.
- **Standardized SDKs Across Languages**: PQC API providers should release SDKs with uniform behavior across major programming languages, ensuring consistent integration experiences and reducing fragmentation risks.
- **Integrated Migration Toolkits**: Tools that automatically scan existing applications for vulnerable cryptography and suggest or apply PQC upgrades (e.g., crypto-linter tools) can simplify the transition.
- **Increased Emphasis on Documentation and Training**: Vendor documentation must include easy-to-follow tutorials, threat models, and code samples for typical use cases. Industry-wide developer training initiatives (bootcamps, workshops) focusing on PQC fundamentals should also be expanded.
- **Testing and Certification Frameworks**: Establishing standardized test suites and certification frameworks (e.g., PQC Compliance Badges) would enable developers to verify their implementations more easily and gain confidence in deployment.
- **Secure Default Configurations**: APIs and SDKs should enforce secure defaults (e.g., PQ-TLS by default, ephemeral key usage, proper entropy management) to reduce reliance on developers making security-critical decisions.

Towards Developer-Centric PQC Adoption

Ultimately, the success of API-driven PQC adoption depends not just on technical robustness but also on developer usability. The future of quantum-safe infrastructure hinges on:

- **Making Security the Path of Least Resistance**: By designing APIs, SDKs, and tools that are intuitive and hard to misuse, developers are naturally guided toward secure practices.
- **Prioritizing User Experience (UX) in Cryptography**: Just as front-end development evolved through UX best practices, cryptographic service providers must treat usability as a first-class design goal, simplifying complex workflows without compromising security.
- **Collaborating Across Communities**: Cryptographers, software engineers, standards bodies, and API providers must work together to close the knowledge gap, ensuring

that post-quantum readiness becomes achievable for all developers—not just a specialized few.

By embedding usability into the core of PQC infrastructure, the transition to a quantum-safe world can be accelerated, reducing risk while maintaining trust, performance, and developer productivity.

Performance and Latency Optimization

Security is only one side of the equation. For cryptographic APIs to be viable in production environments, especially those operating at scale or in real time, they must also be fast and reliable. Poor performance introduces usability issues, disrupts user experience, and creates operational bottlenecks. This is especially true for post-quantum cryptographic operations, which tend to be computationally heavier and require larger key sizes than classical alternatives.

The most effective way to manage performance is through geographic distribution. Hosting the API close to the client, whether through regional cloud infrastructure, CDNs, or dedicated edge compute nodes, can significantly reduce latency. For global applications, this means deploying API instances in multiple zones and using intelligent routing to direct traffic to the nearest node.

Another option is client-near deployment. In this model, the API is packaged and deployed on infrastructure controlled by the client, such as private cloud, local data centers, or high-security edge devices. This eliminates reliance on external networks and allows the client to maintain low-latency access even in bandwidth-constrained or high-throughput scenarios.

Connection persistence is also important. Repeatedly establishing secure connections adds overhead, especially when using hybrid key exchanges. Persistent connections using protocols like HTTP/2 or gRPC can help reduce this cost, allowing multiple cryptographic operations to occur over a single, long-lived connection.

Caching and rate limiting can further improve efficiency. Where appropriate, responses to repeat requests (such as public parameters or validation results) can be cached on the client side or at intermediary proxies. This reduces load on the API and speeds up response times for common

operations. Rate limiting ensures fair usage, protects against abuse, and helps maintain consistent performance under load.

Finally, asynchronous processing can be used in workflows that tolerate slight delays. In cases like document signing, secure archiving, or scheduled transmissions, requests can be queued and processed with a slight delay, reducing peak load pressure and enabling better resource allocation across the infrastructure.

Performance tuning is not a one-size-fits-all exercise. The right optimization strategy depends on the use case, regulatory environment, user base, and threat model. What matters most is that the API architecture is designed to scale, adapt, and deliver cryptographic performance without compromising security or usability.

Case Study: Quantum-Safe Encryption for an IoT Device Manufacturer

Context

A global IoT company designs and distributes smart environmental sensors used in agriculture, mining, and urban infrastructure. These devices operate in remote locations, often on low-power embedded hardware, with intermittent internet access. The company must ensure long-term confidentiality and integrity of the data collected and transmitted by its devices.

Challenge

Traditional public-key cryptography like RSA and ECC is becoming insufficient due to quantum threats. However, integrating post-quantum algorithms directly into the device firmware proved unfeasible due to:

- Limited RAM and CPU capacity
- Firmware written in C, with no maintained, product-ready PQC libraries available
- Exposure to side-channel attacks if cryptography is run locally
- Cost and complexity of re-certifying firmware with new cryptographic primitives

Solution

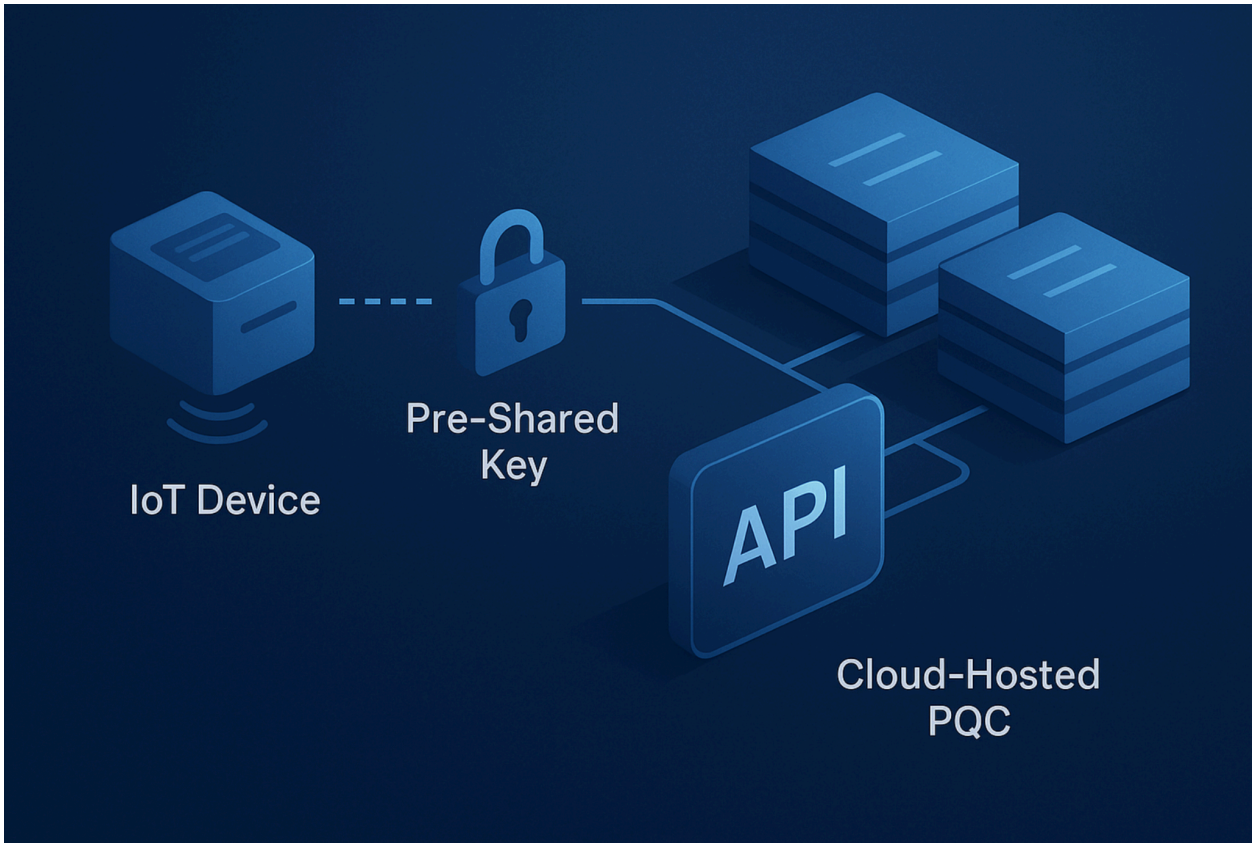
The company partnered with ExeQuantum to implement an API-based PQC architecture:

- Devices used lightweight HTTP calls to connect with a cloud-hosted PQC API to perform key encapsulation using ML-KEM (Kyber-based) hybridized with classical ECC for fallback compatibility.

- A pre-shared key model was used between the device and the cloud for encrypting the communication channel. This avoided the need for TLS handshakes or local certificate validation.

- Instead of performing PQC operations locally, the devices offloaded key exchanges and encryption to the API, sending only non-sensitive metadata and telemetry.

- Data was encrypted at rest and in transit using symmetric AES keys derived from API-generated PQC key material.



Performance Strategy

- The API was deployed across regional edge locations to minimize latency, particularly in rural and remote deployments.

- Non-urgent data was queued and transmitted during scheduled intervals, allowing time for retries and reducing bandwidth spikes.

Result

- Devices remained lightweight, power-efficient, and quantum-resistant without changing the firmware.

- The company reduced its cryptographic update cycle from 4 months of R&D to a 2-day API configuration update.

Case Study: Quantum-Safe Digital Identity Linking in Australia

Context

An Australian startup in the digital identity space developed a platform that enables users to link their verified banking identities to their social media accounts. The goal was to simplify account verification, reduce fraud, and create a persistent cross-platform digital identity backed by a trusted institution.

The platform integrated with banks via Open Banking APIs and allowed users to verify their social presence through OAuth. Given the sensitive nature of linking financial credentials to public platforms, the company needed to guarantee long-term confidentiality, strong authentication, and immunity from quantum-era attacks.

Challenge

The company needed a cryptographic solution that could:

- Ensure future-proof encryption of user identity linkages and authentication tokens

- Maintain compliance with Australia’s 2030 ISM post-quantum mandate

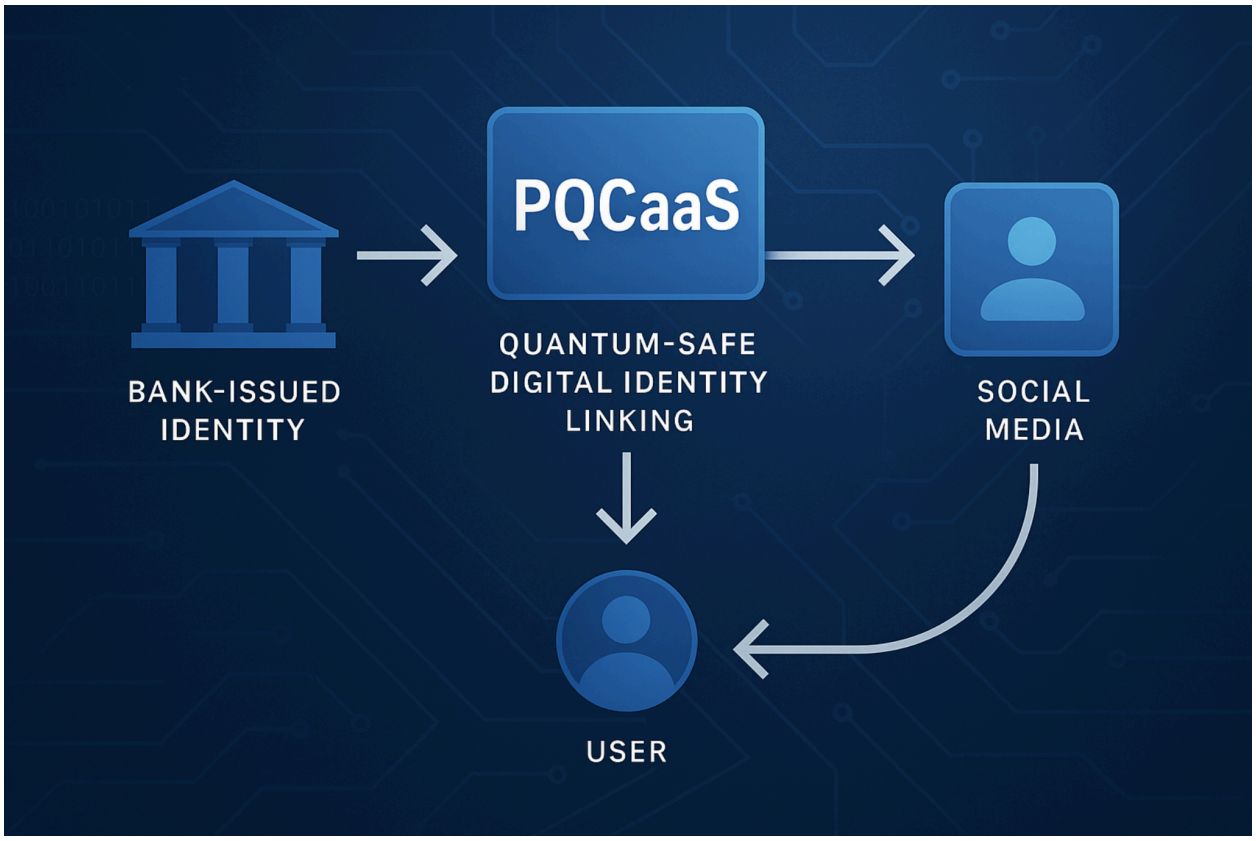
- Avoid storing sensitive key material on client devices or in their own infrastructure
- Generate high-entropy, non-predictable digital identity tokens for social media linking
- Deploy with zero disruption to the frontend or Open Banking flow

The in-house team had limited cryptographic expertise and was focused on product delivery, making local PQC integration too risky and slow.

Solution

The company adopted ExeQuantum's Post-Quantum Cryptography as a Service (PQCaaS) with Quantum Random Number Generator (QRNG) support:

- Identity linking tokens were generated using ML-KEM for key encapsulation and ML-DSA for digital signatures, all handled server-side via API.
- Quantum-generated entropy from ExeQuantum's QRNG was used for session keys and nonce generation, preventing entropy reuse and bias.
- All token and key lifecycle operations were handled through ExeQuantum's cloud API, accessed via a secure SDK with built-in PQ-TLS.



Social media OAuth verifications and bank-issued identity attestations were securely bridged using the PQCaaS backend, with no cryptographic operations exposed to client devices.

Security and Performance Strategy

- PQC endpoints were hosted regionally in Sydney and backed by an edge network to minimize latency during token generation and verification.
- All inter-service communications used hybrid PQ-TLS with automatic key lifecycle auditing.
- Each identity linkage was cryptographically signed with Dilithium and backed by tamper-evident logs for verifiability.

Result

- The company launched the platform with full post-quantum cryptographic support in under 6 weeks.

- They avoided all cryptographic maintenance overhead by outsourcing cryptographic operations to the PQCaaS layer.

- The solution was adopted by two Australian fintechs and a challenger bank seeking post-quantum compliant identity flows.

- Regulatory reviews highlighted the platform's quantum-resilient design and cryptographic transparency as key differentiators.

Conclusion

Post-quantum cryptography is no longer optional. It is a foundational requirement for securing digital infrastructure in the face of rapidly advancing quantum capabilities. Yet despite the urgency, the path to adoption remains steep. Complex algorithms, evolving standards, and inconsistent implementation environments continue to slow progress, leaving many organizations exposed to future decryption risks.

API-driven infrastructure offers a practical and effective way forward. By abstracting quantum-safe cryptographic functions behind clean, standardized interfaces, APIs eliminate much of the risk and complexity traditionally associated with cryptographic transitions. They provide a plug-and-play foundation that works across languages, platforms, and industries, enabling teams to deploy post-quantum protections without needing deep cryptographic expertise.

This approach does not eliminate all challenges. Secure key exchange, latency, and backend assurance must still be addressed. But these are solvable problems, and the solutions, such as PQ-TLS, edge hosting, pre-shared key options, and verifiable audit frameworks, are already available. What remains is the will to act and the infrastructure to support it.

By building quantum-safe capabilities into API infrastructure today, organizations gain a future-proof foundation. They move faster, integrate more securely, and reduce the long-term

cost of compliance and cryptographic agility. Most importantly, they take an active role in shaping a resilient and trustworthy digital future.