Secure Communications for Quantum and Beyond: Orchestrated Cryptography for Agility and Policy

Chris Cap, Scott Kawaguchi, Joey Lupo, Chris Trinidad

Abstract

Organizations across the public and private sectors depend on the ability to securely communicate data over computer networks. By Shor's algorithm, a quantum computer of sufficient size can efficiently break public-key algorithms such as RSA, ECDH, and ECDSA which secure these network communications today. As a result, organizations will need to replace all instances of vulnerable cryptography to protect their own and their customers' data-in-transit. In this paper, we describe how a decentralized cryptography deployment model complicates the migration to post-quantum and cryptography management more broadly. We then explain how QuSecure is developing a new paradigm of cryptographic orchestration with QuProtect™ to empower organizations to secure their networks and meet challenges of agility and policy at scale in a dynamic ecosystem of new algorithms, attacks, standards, and requirements.

1 Introduction

Since its advent in the seminal 1976 paper "New directions in cryptography" by Diffie and Hellman [1], public-key cryptography has enabled secure communications across untrusted networks. Alice can send her public key to Bob to establish a secure channel even if Eve is privy to all messages sent across the network. More advanced applications of public-key cryptography can establish a secure channel even if Mallory can arbitrarily insert, drop, or alter messages. Organizations within the public and private sectors already rely on hostile or untrusted networks such as the Internet to communicate sensitive data between services, devices, and people. Trends such as the Internet of Things (IoT), remote work, and bring your own device (BYOD) mean other untrusted networks such as mobile 4G/5G/LTE and public Wi-Fi increasingly carry sensitive traffic as well. Furthermore, zero-trust architectures which assume *all* networks are untrusted are gaining traction in government and industry.

A sophisticated adversary has a number of techniques to surreptitiously collect, block, inject, or otherwise tamper with data-in-transit. The principal tool in the security toolbox for protecting the confidentiality, integrity,

and authenticity of data sent over a network is a **secure channel protocol**. Examples include Transport Layer Security (TLS), Internet Protocol Security (IPsec), Secure Shell (SSH), and WireGuard [2]. An adversary equipped with a cryptographically-relevant quantum computer (CRQC), however, can break the public-key cryptography which undergirds these secure channel protocols. Furthermore, encrypted data-in-transit is vulnerable *today* to quantum threats via a "store now, decrypt later" attack wherein an adversary collects encrypted traffic in bulk for the purposes of later decryption with a quantum computer.

To address the quantum threat, the cryptography community is developing and testing algorithms which are resistant to quantum computing. So-called **post-quantum cryptography (PQC)** is the subject of a process by the National Institute of Standards and Technology (NIST) to select algorithms for standardization [3]. Meanwhile, organizations are faced with the monumental task of migrating all instances of vulnerable cryptography to PQC. QuSecure is developing a suite of products and protocols to upgrade applications and networks to post-quantum while requiring no code changes. At the same time, we are delivering on a vision of cryptographic orchestration to help organizations meet challenges of agility and policy at scale in a dynamic cryptographic ecosystem.

In Section 2, we define cryptographic agility and explain its critical importance in securely navigating the migration to PQC and in future migrations. We also outline the deprecation difficulties and downgrade attacks which complicate agility in practice. In Section 3, we look more broadly at cryptographic policy. In particular, we observe a misalignment between centralized, top-down cryptographic policy and a decentralized, bottom-up deployment model. In Section 4, we draw parallels between the current state of affairs in cryptography with the state of affairs in networking before the development of software-defined networking (SDN) and describe how SDN informs the QuProtect architecture. Section 5 presents QuProtect, which is QuSecure's post-quantum cryptography orchestration solution. Section 6 concludes the paper.

2 Cryptographic agility

We can broadly define **cryptographic agility** to mean the ability to swap out vulnerable or otherwise non-optimal cryptography with minimal disruption to the operation of the consuming applications or systems. However, we find the more granular agility characterizations considered by McGrew in [4, Chapter 3] more useful. In particular, the author subdivides cryptographic agility into three distinct types, each of which corresponds to a different component of the cryptography stack which might be found vulnerable:

1. Algorithm agility
2. Protocol agility
3. Implementation agility

Algorithm agility refers to the ability to swap out a vulnerable algorithm within a system with minimal disruption. For example, a system which has algorithmic agility could easily swap out a weak symmetric cipher such as RC4 with AES, or an insecure hash function such as MD5 with SHA-2. **Protocol agility** means the ability to upgrade from a vulnerable protocol version, such as migrating from TLS 1.0 to TLS 1.3. Lastly, **implementation agility** means quickly patching vulnerable algorithm implementations. For example, a software library may implement an algorithm in a way which is vulnerable to a buffer overflow or a timing-based side-channel attack.

Agility allows us to future-proof systems against novel cryptanalysis, evolving adversaries, and implementation errors. In particular, agility implies a shorter **deprecation period**, which we understand as the time between (1) somebody demonstrates a vulnerability in an algorithm, protocol, or implementation, and (2) all applications or systems which use the vulnerable cryptography are patched or upgraded. Beyond security, agility also lets us take advantage of more efficient algorithms or implementations as they become available.

2.1 Agility during the PQC migration and beyond

The ongoing NIST PQC Standardization Process suggests a critical need for algorithmic and implementation agility. As a study by the European Union Agency for Cybersecurity (ENISA) report summarizes, "almost all of the post-quantum cryptosystems submitted to NIST have lower security against the best attacks known today than against the best attacks that were known in 2017" [5]. New cryptanalysis, side channel attacks, and software bugs have prompted submission teams to respond with algorithm tweaks, larger key sizes, and software patches. In several dramatic cases, researchers completely broke a candidate algorithm. In the third round of the NIST process, researchers found a catastrophic key recovery attack against the GeMSS digital signature scheme [6]. In the fourth round, a research team found a catastrophic key recovery attack against the KEM candidate SIKE which required only a consumer laptop to execute [7]. To react to this shifting landscape, any PQC solution must able to rapidly restore secure communications by supporting the ability to swap out a vulnerable algorithm, increase key sizes, or patch vulnerable implementations.

The migration to PQC, however, is not the first time we have had to upgrade vulnerable cryptography. Indeed, upgrades from MD5 to SHA-1, SHA-1 to SHA-2, and DES to AES are examples from years past. It is also not likely to be the last time. At a minimum, algorithms become weaker over time as computing capabilities improve. Furthermore, it is unlikely that Grover's and Shor's are the only cryptographically-relevant quantum algorithms. Experts may also develop more efficient post-quantum cryptography over time, just as was the case when elliptic curve cryptography overtook RSA as the dominant (pre-quantum) asymmetric primitive. Though we might not be able to anticipate what future changes will look like, we can be sure that cryptography will change over time. Therefore, we must build cryptographic systems which can adapt and evolve accordingly. In short, they must be agile.

2.2 Agility issues in practice

Despite the benefits, agility is difficult to attain. A fundamental obstacle is the decentralized, fragmented deployment model of cryptography widely used in practice. For example, eliminating all SHA-1 TLS certificates means tracking down all SHA-1 certificates on every single endpoint within your network, either manually or with one-off scripts or discovery tools. Furthermore, cryptography is usually tightly coupled to applications. Consider, for example, how configuring TLS for an Apache web server with mod_ssl differs from deploying Nginx with TLS, which differs from configuring TLS for Redis, which differs still from configuring TLS for MongoDB. Though the concepts are similar in each case, this fragmentation adds friction to upgrade processes for IT teams, especially at scale. The end result is decreased agility and longer deprecation periods during which data-in-transit is potentially vulnerable to compromise. As one conference report noted, "Algorithm elimination is a particularly notable failure in the industry as long-deprecated standards (e.g., RC4, MD5, DES) continue to be in use" [8]. In the case of TLS, the deprecation period for vulnerable algorithms and protocol versions has historically been on the order of *years* [9].

Another obstacle to agility is that it is often baked directly into a secure channel protocol by means of a negotiation phase. For example, both TLS and IPsec—or, more precisely, the Internet Key Exchange (IKE), which is a component of IPsec—include a cipher suite negotiation phase. This protocol-centric view of algorithmic agility is further described in [10]. The story for protocol agility is similar; the two parties typically settle on a mutually supported protocol version within the same negotiation phase.

A couple problems tend to surface with this approach to agility. For one, negotiation usually translates to additional round trips, adding latency and bandwidth to each handshake. Second, negotiation adds complexity to a protocol. Complexity, in turn, makes security harder to reason about and can introduce vulnerabilities, especially when we consider a network adversary that can freely drop, inject, or modify protocol messages. Indeed, the class of **downgrade attacks** refers to an active network adversary (also known as a man-in-the-middle (MITM)) attacking a protocol negotiation phase in a way which results in the subsequent secure channel or key exchange using weakened or no encryption. In [11], the authors develop a taxonomy for *fifteen* distinct downgrade attacks to vulnerable cipher suites or protocol versions in TLS alone. In [12], the authors study several downgrade attacks in TLS, IPsec, SSH, and ZRTP.

3 Cryptographic policy

A **cryptographic policy** can be defined as any organizational policy related to the use of cryptography. Such a policy might specify permissible algorithms, libraries, protocols, key sizes, key rotation strategies, key storage options, entropy sources, or random number generators for any given application of cryptography. A cryptographic policy is usually specified by the CIO, CTO and/or CISO

in a document and then implemented by an IT team.

An organization's cryptographic policy (if one exists) is typically informed by relevant regulations and compliance regimes. Examples include Federal Information Processing Standards (FIPS) 140-2/140-3, NIST Special Publication 800-53, the General Data Protection Regulation (GDPR), the Payment Card Industry Data Security Standard (PCI DSS), the Health Insurance Portability and Accountability Act (HIPAA), and System and Organization Controls (SOC). Each of these examples includes guidelines for protecting data-in-transit, with varying levels of specificity. For example, PCI DSS requires "strong cryptography," which it defines in terms of algorithms and key sizes [13]. The most comprehensive standard is FIPS 140-2/3, which specifies compliant algorithms and parameters for symmetric encryption, asymmetric encryption, key exchange, hash functions, random number generators, and entropy sources [14].

3.1 Policy during the PQC migration and beyond

Given that the current slate of widely used public-key algorithms are vulnerable to quantum threats, any standard or regulation which references vulnerable algorithms will need to updated. Indeed, any definition for security of data-in-transit which considers the quantum threat will require a compliant organization to have completed an upgrade to PQC. Interim requirements are also likely. Consider, for example, a recent directive by the Biden administration that instructed federal agencies to inventory all instances of vulnerable cryptography in their IT systems [15]. As a result of these evolving requirements, organizations will need to update their cryptographic policies accordingly. More importantly, those policies will need to be implemented by IT teams to satisfy auditors and meet compliance and certification targets.

As we discussed in Section 2.1, however, the migration to PQC is not the first or last such transformation of the cryptographic landscape. In the near-term, consider the likely push for zero-trust architectures, and how this will exponentially increase the compliance burden by considering communications within data centers or internal networks to be in-scope. More broadly, organizations need an ability for agile cryptographic deployments that can support cryptographic policies downstream from evolving requirements.

3.2 Policy issues in practice

A cryptographic policy is typically a single authoritative reference for an organization. In other words, a policy is a *centralized* and *top-down* construct originating from the C-suite or a relevant regulatory standard. By contrast, the actual deployment of cryptography is highly distributed and fragmented. In other words, managing and implementing cryptography is a *decentralized* and *bottom-up* endeavor for IT teams. In the case of TLS, you need to issue and distribute a certificate to each service, device, or user that needs one. And this can only happen after standing up a root certficate authority (CA) and intermediate CAs to support an internal public-key infrastructure (PKI). Many organizations

still manage their certificates with spreadsheets. Virtual private network (VPN) protocols such as IPsec, OpenVPN, or WireGuard require that someone individually configure each peer within the VPN. In SSH, manual configuration is needed to support public-key-based authentication. This fundamental mismatch between centralized policy and decentralized implementation is in large part what makes cryptography difficult to audit, manage, and migrate on an organizational level.

A consequence of this decentralization is limited **visibility**. At a basic level, visibility means knowing which cryptography is in use, and which data it protects. More concretely, for any given channel which transmits data of a certain level of sensitivity, what does the cryptographic protection of that channel look like? Which cryptographic libraries, algorithms, or key sizes are actually used for any given secure connection? How often are keys rotated and where are they stored? How are cryptographically secure random numbers generated? The answers to these questions are dispersed in part through the various endpoints in configuration files or logs. For protocols with a negotiation phase, analysis of live protocol exchanges over the wire is often the only way to know which algorithms are actually used for any given connection. Once you have inventoried all instances of vulnerable cryptography, now you actually need to make changes to your system which ensure that those vulnerable instances are no longer used for data-in-transit. This requires agility, which is similarly affected by decentralization, as we covered in Section 2.2. In sum, decentralized cryptography deployment reduces visibility and agility, making it difficult to support an evolving cryptographic policy at scale.

4 Parallels with software-defined networking

Before discussing QuProtect, we take a brief detour to the world of computer networking. We find it useful to draw a comparison between the current cryptography ecosystem with the computer networking ecosystem before the development of software-defined networking (SDN). Managing a traditional IP network is a complex, time-intensive, error-prone process that involves manually configuring switches, routers, and middleboxes using vendor-specific command-line or network interfaces. As one data point, consider that the migration from IPv4 to IPv6 has been in progress for over a decade.

The same concepts apply to managing cryptography within an organization. As we covered Section 3.2, cryptography-related tasks such as setting up and managing an enterprise PKI or configuring IPsec peers are similarly complex, time-intensive, and error-prone and involve learning library-, application-, or vendor-specific configurations to properly deploy. This highly decentralized deployment makes the migration to PQC similar in some respects to the migration to IPv6.

The insight of SDN is to decouple various layers of the networking stack: "An important consequence of the software-defined networking principles is the *separation of concerns* introduced between the *definition* of network policies,

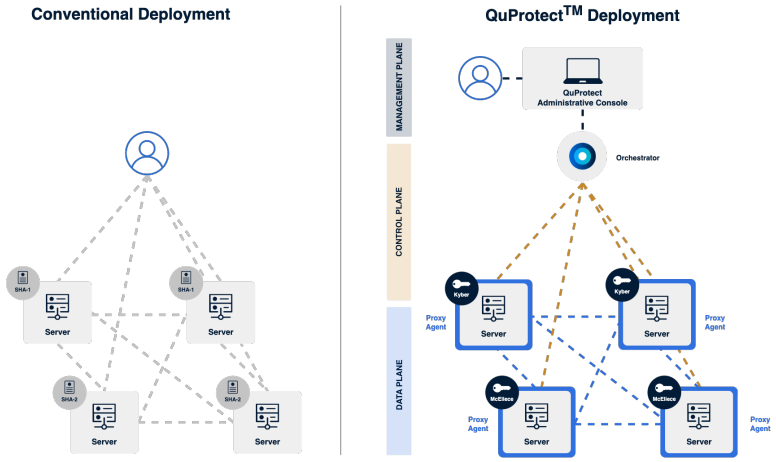

Figure 1: QuProtect versus traditional cryptography management. The Orchestrator exposes an abstract view of the cryptography securing your data to administrators.

their *implementation* in switching hardware, and the *forwarding* of traffic" [16]. These three layers map to an application plane, a control plane, and a data plane, respectively. The **data plane** consists of network devices (physical or virtualized) such as switches and routers which are responsible for the actual forwarding of data packets through a network. On the other hand, the **control plane** consists of a (logically) centralized SDN controller and application programming interfaces (APIs) to control the behavior of the data plane elements. Network applications in the **application plane** such as load balancers then can make use of the interface exposed by the SDN controller to define the routing policy without any knowledge of the underlying hardware devices or any vendor-specific management tool. In other words, network applications can work at the level of network *policy*.

The approach QuSecure takes to cryptography, post-quantum or otherwise, is similar. As we discuss in Section 5, we are developing a solution which decouples the definition of cryptographic policy, the implementation of that policy, and the actual encryption of traffic. These three layers correspond to our management plane, control plane, and data plane. Just as a SDN controller presents an abstracted view of the network, our Orchestrator presents an abstracted view of the cryptography which secures your network. The Administrative Console in the management plane interfaces with the Orchestrator to provide a single pane-of-glass to view and manage the cryptography deployment. In particular, the Administrative Console works at the level of cryptographic *policy*. Under the hood, the control plane ensures this policy is executed by configuring applications in the data plane that encrypt data and execute post-quantum secure channel protocols. See Figure 1.

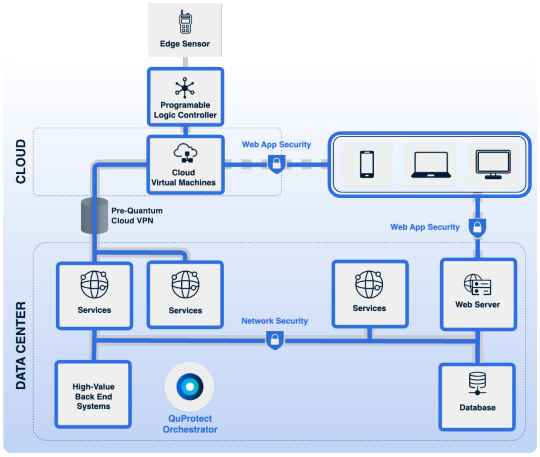

Figure 2: QuProtect lifts your network to post-quantum with no code changes and provides an easy interface for managing cryptographic policy at scale. An on-premises deployment is illustrated above. Cloud deployment is also available for the QuProtect Orchestrator.

5 QuProtect cryptographic orchestration

QuSecure applies a comprehensive approach to the problems of cryptographic agility and policy in the migration to post-quantum cryptography and beyond. Our basic observation is that the existing enterprise cryptography ecosystem is too decentralized and fragmented to support evolving algorithms, attacks, regulations, and policies at scale. This solution is encapsulated within **QuProtect**. In the following sections, we describe the management, control, and data planes which comprise QuProtect. See Figure 2 for an example deployment.

5.1 Cryptographic management plane

Within the management plane is a user-facing **Administrative Console**, in which administrators can register new endpoints and manage the algorithms used by those endpoints. In particular, we support total algorithmic agility via this console. For example, if new cryptanalysis degrades the security of Kyber512 to an unacceptable level, then an administrator can upgrade all channels currently using Kyber512 to Kyber768 or Kyber1024 with the click of a button. Similarly, in the unlikely event that Kyber is shown totally broken, then administrators can immediately switch all channels using Kyber to BIKE, HQC, or Classic-McEliece. In short, we reduce the deprecation period for vulnerable algorithms from months or years to minutes.

The Administrative Console also maintains visibility into the cryptography used to secure each communication channel within a QuProtected network. This unprecedented visibility greatly simplifies auditing and compliance tasks for

IT teams. Furthermore, failed connections can be detected and pinpointed, providing insight for IT and security teams. A visual dashboard provides a birds-eye view of successful and failed connections within the network.

The Administrative Console is intended to work at the level of cryptographic *policy*, without exposing any of the low-level implementation details to administrators. We are continuing to develop this vision to support policies relating to key rotation times, libraries, and random number generation for all communications within a QuProtected network.

5.2 Cryptographic control plane

Given a cryptographic policy from the management layer, it is the responsibility of the control plane to ensure that policy is enforced. Or, viewed from the other direction, the control plane exposes an interface to the management plane to manage the cryptography used to secure communications between endpoints. The primary component of the control plane is a high-availability, (logically) centralized **Orchestrator**.

In the case of algorithmic agility, the Orchestrator is responsible for configuring algorithms and key sizes for the data plane elements which actually encrypt data and execute the secure channel protocols. We use a configuration protocol for QuProtected endpoints to accomplish this task. This configuration protocol executes over an independent post-quantum secure channel, which we describe in further detail in Section 5.3.1. Since we move agility to the control plane, any downgrade attack from an active network adversary requires compromising this secure channel, which requires an attack beyond the downgrade attacker threat model. In any case, if this configuration channel *is* compromised, a malicious reconfiguration is an auditable event that can be detected and investigated. We are continuing to expand the control plane to support a richer cryptographic policy interface at the management layer, including: pushing updates to data plane elements for implementation agility, implementing key rotation and management policies, and setting the source of entropy for keys.

5.3 Quantum-Secure Layer, cryptographic data plane

Recall that in SDN the data plane elements are network devices such as switches and routers which are responsible for the actual forwarding of packets. In other words, the data plane is the "dumbest" layer. Applying the analogy to cryptography, we consider the data plane components as the secure channel protocols and the entities executing those protocols. Within QuProtect, we call the data plane the **Quantum-Secure Layer (QSL)**. We have developed a number of data plane elements in-house, which we detail next.

5.3.1 PQNoise

To upgrade a network to post-quantum, you have to actually develop or leverage a post-quantum secure channel protocol. To this end, we have implemented the

PQNoise Protocol Framework [17] as the basis for establishing post-quantum secure channels. PQNoise is a post-quantum adaptation of the Noise Protocol Framework [18], a framework which has seen use in popular applications such as WhatsApp end-to-end encrypted messaging and the WireGuard VPN protocol. Whereas Noise uses Diffie-Hellman key exchanges as the only asymmetric primitive, PQNoise uses post-quantum key encapsulation mechanisms (KEMs).

We follow in the spirit of Noise and PQNoise by omitting any in-band negotiation of cipher suite or protocol version within a QSL protocol handshake. Instead, each party executes the protocol with respect to a control-plane-configured cipher suite consisting of a static KEM algorithm, an ephemeral KEM algorithm, an authenticated encryption with associated data (AEAD) (symmetric encryption) algorithm, and a cryptographic hash function. At a high-level, the static KEM provides a longer term key pair for purposes of authentication. The use of a KEM for authentication has practical advantages over a post-quantum digital signature scheme [5, Section 3.1]. On the other hand, the ephemeral KEM provides forward secrecy to all connections. This property means that a future compromise of the static KEM private key has no implications on the security of past communications.

Executing the protocol with respect to a fixed configuration avoids negotiation overhead and means security is easy to reason about: message formats are unambiguous, message sizes are deterministic, and the protocol state machine is a straight line. Furthermore, we avoid the types of downgrade attacks covered in Section 2.2 and can apply the proofs in [17] to understand the security guarantees of any given PQNoise instantiation, limiting the need for protocol agility.

Note that the PQNoise specification permits the use of two different KEMs to fill the role of static KEM and ephemeral KEM. Importantly, we can exploit this fact by using two KEMs which depend on substantially different security assumptions. This configuration provides redundancy for the security of past data in the case that future cryptanalysis breaks or weakens one of the two security assumptions. In other words, we avoid a single point of failure. In release notes, we outline our recommended algorithm profile of the code-based Classic-McEliece as static KEM and the lattice-based Kyber as ephemeral KEM. This configuration mirrors PQWireGuard [19], except with the NIST-standardized Kyber in place of Saber as a lattice-based KEM.

5.3.2 Key distribution center

We employ a high-availability and scalable key distribution center (KDC) to support centralized generation of high-entropy keys. In particular, the KDC can integrate with a quantum random number generator (QRNG) or bring-your-own-entropy via an API. This use case is particularly valuable for IoT devices that might not have a reliable source of quality entropy.

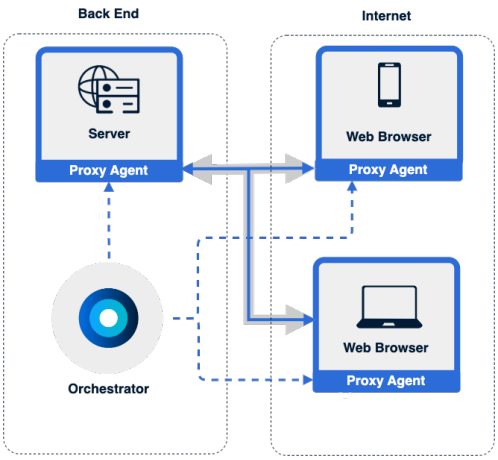

Figure 3: QuProtect Web App Security lifts a web application to post-quantum with no code changes and no client-side installs.

5.3.3 Proxy Agent

Another important QSL component is the **Proxy Agent**, which runs on or near endpoints and serves a couple of roles. The first is to execute PQNoise handshakes with the KDC. In particular, the end result of a PQNoise handshake is a post-quantum secure channel between the Proxy Agent and the KDC. Subsequent handshakes establish forward-secure shared keys. At this point, we can use this channel to securely distribute keys from the KDC to the Proxy Agent.

The Proxy Agent additionally includes the capabilities of a reverse proxy. A reverse proxy is the basis for the cloud-native Istio service mesh [21], and we have developed the Proxy Agent with a similar deployment model in mind. In general, the Proxy Agent can upgrade any proxied application to post-quantum. In the following two sections, we describe how we leverage the Proxy Agent to upgrade web applications and internal service communications to post-quantum with no code changes.

5.3.4 Web App Security

Web App Security is a QSL solution that can transparently upgrade a web application to post-quantum with no code changes or client-side installs. From a cryptography perspective, we execute an ephemeral KEM exchange over HTTPS with a browser-based service worker, which proxies web application requests. See Figure 3. As part of the protocol, the KDC generates and distributes a high-entropy session secret to both the browser agent and the Proxy Agent. Each party then derives AEAD (symmetric) session keys from the session secret for encrypting subsequent communications. Note that "post-quantum" here should be understood to mean an active classical adversary that will have access

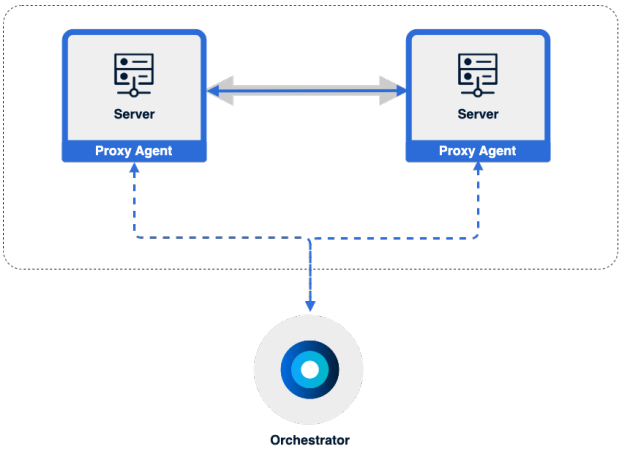

Figure 4: QuProtect Network Security lifts service-to-service communications to post-quantum and zero-trust with no code changes.

to a CRQC in the future. In particular, this adversary models the "store now, decrypt later" attack which poses a threat to data *today*.

5.3.5 Network Security

Network Security is a QSL solution designed to transparently upgrade service-to-service communications at layer 4 or layer 7 to post-quantum with no code changes. We achieve this goal by placing a Proxy Agent in front of each network application to apply a uniform layer of encryption, authentication and authorization. See Figure 4. We use the KDC to establish mutual authentication between the two Proxy Agents and securely distribute a high-entropy session secret. Each party then derives AEAD session keys from the session secret for encrypting subsequent communications.

This deployment model of fronting applications with a reverse proxy is known as a **service mesh** in the parlance of cloud-native applications. In effect, the service mesh extracts security-related functionalities from applications, so developers can focus on developing applications and not worry about encrypting communications, configuring certificates, or managing keys. In particular, the service mesh model avoids many of the difficulties noted in Sections 2.2 and 3.2. A service mesh architecture can support complex deployments of hundreds or thousands of different services as in a cloud-native or microservices architecture, in addition to legacy or monolithic applications.

6 Conclusion

Organizations across the public and private sectors rely on the ability to securely communicate data over untrusted networks. Though "untrusted network" historically referred to a public network such as the Internet, a mobile network, or a public Wi-Fi, the increasingly prevalent zero-trust architecture treats *all* networks as untrusted. The quantum threat in this setting is easy to grasp: a cryptographically-relevant quantum computer can efficiently break the public-key algorithms which secure data-in-transit. As a result, organizations will need to upgrade all instances of vulnerable cryptography to support post-quantum cryptography. Just as there were previous migrations such as MD5 to SHA and DES to AES, there will be future cryptographic migrations beyond PQC. Therefore, it is critical that organizations be able to quickly and effectively respond to new attacks or regulations with cryptographic agility and policy.

In this paper, we argued that the fundamental misalignment between centralized, top-down cryptographic policy and decentralized, bottom-up cryptography deployment in practice complicates attaining agility and policy goals. Next, we described how QuSecure addresses these problems with the QuProtect orchestrated cryptography platform. We detailed different components of QuProtect within the cryptographic management, control, and data planes, and described future directions for expanding these components to expose a rich policy language to administrators and security teams. In sum, we present QuProtect to meet the quantum threat today, and unforeseen threats tomorrow.

References

[1] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[2] Jason A Donenfeld. WireGuard: Next Generation Kernel Network Tunnel. In *NDSS*, pages 1–12, 2017.

[3] National Institute of Standards and Technology (NIST). Post-Quantum Cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography`.

[4] Engineering National Academies of Sciences and Medicine. *Cryptographic Agility and Interoperability: Proceedings of a Workshop*. The National Academies Press, Washington, DC, 2017.

[5] Daniel J. Bernstein, Andreas T. Hülsing, and Tanja Lange. *Post-Quantum Cryptography - Integration study*. ENISA, October 2022.

[6] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Improved Key Recovery of the HFEv- Signature Scheme. *IACR Cryptol. ePrint Arch.*, page 1424, 2020.

[7] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.

[8] David Ott, Christopher Peikert, and et al. Identifying research challenges in post quantum cryptography migration and cryptographic agility. *CoRR*, abs/1909.07353, 2019.

[9] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. Coming of age: A longitudinal study of TLS deployment. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*, pages 415–428. ACM, 2018.

[10] Russ Housley. Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms. RFC 7696, November 2015.

[11] Eman Salem Alashwali and Kasper Rasmussen. What's in a downgrade? A taxonomy of downgrade attacks in the TLS protocol and application protocols using TLS. In *Security and Privacy in Communication Networks: 14th International Conference, SecureComm 2018, Singapore, Singapore, August 8-10, 2018, Proceedings, Part II*, pages 468–487. Springer, 2018.

[12] Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella-Béguelin. Downgrade resilience in key-exchange protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 506–525, 2016.

[13] The Payment Card Industry Security Standards Council. Glossary of Terms, Abbreviations, and Acronyms. `https://www.pcisecuritystandards.org/wp-content/uploads/2022/04/PCI_DSS_Glossary_v3-2.pdf`, 2016.

[14] National Institute of Standards and Technology (NIST). Security requirements for cryptographic modules: Federal Information Processing Standards publication 140–142. `https://csrc.nist.gov/publications/detail/fips/140/2/final`, 2001.

[15] Justin Doubleday. White house tells agencies to participate in post-quantum cryptography tests, November 2022. Available at https://federalnewsnetwork.com/cybersecurity/2022/11/white-house-tells-agencies-to-participate-in-post-quantum-cryptography-tests/.

[16] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[17] Yawning Angel, Benjamin Dowling, Andreas Hülsing, Peter Schwabe, and Florian Weber. Post Quantum Noise. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 97–109, 2022.

[18] Trevor Perrin and Moxie Marlinspike. Noise Protocol Framework. `https://noiseprotocol.org/noise.html`, 2018.

[19] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R Zimmermann. Post-quantum WireGuard. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 304–321. IEEE, 2021.

[20] Envoy proxy. `https://www.envoyproxy.io/`.

[21] Istio service mesh. `https://istio.io/latest/about/service-mesh/`.

A Acronyms

AEAD authenticated encryption with associated data 10

API application programming interface 7

CA certficate authority 5

CRQC cryptographically-relevant quantum computer 2

FIPS Federal Information Processing Standards 5

IKE Internet Key Exchange 4

IoT Internet of Things 1

IPsec Internet Protocol Security 2

KDC key distribution center 10

KEM key encapsulation mechanism 10

MITM man-in-the-middle 4

NIST National Institute of Standards and Technology 2

PKI public-key infrastructure 5

PQC post-quantum cryptography 2

QRNG quantum random number generator 10

QSL Quantum-Secure Layer 9

SDN software-defined networking 2

SSH Secure Shell 2

TLS Transport Layer Security 2

VPN virtual private network 6