Preparing IT Infrastructure for the Quantum Age: Post-Quantum SSH[⋆]

Pavlo Vorobets[1,†], Andii Horpenyuk[1,†] and Ivan Opirskyy[1,*,†]

[1] *Lviv Polytechnic National University, 12 Stepan Bandera str., 79000 Lviv, Ukraine*

Abstract

The paper explores the challenges and solutions related to preparing SSH for a post-quantum world, including the inherent risks posed by quantum-capable adversaries, the limitations of existing cryptographic algorithms, and the practical steps required to integrate post-quantum and hybrid mechanisms into real-world SSH deployments. The rapid development of quantum computing poses a significant threat to classical cryptographic algorithms that secure today's digital infrastructure. Among the protocols at risk is SSH (Secure Shell), a critical component of secure remote administration across countless systems. It assesses the current state of post-quantum cryptography (PQC) integration in the official OpenSSH project and its experimental forks, particularly emphasizing hybrid key exchange mechanisms that combine classical and quantum-safe algorithms. The role of supporting libraries, such as liboqs, is examined in enabling these cryptographic enhancements. In addition, the paper offers practical, actionable recommendations for infrastructure and security teams — including cryptographic asset inventory, deprecation of vulnerable RSA keys (especially those ≤2048 bits), enforcement of key rotation policies, and controlled testing of PQC-enabled SSH implementations in sandboxed or isolated environments.

Keywords

SSH, Post-quantum cryptography, quantum computers, standardization process, NIST

1. Introduction

The development of quantum computing profoundly challenges the cryptographic basis of the current IT infrastructure. Modern cryptography algorithms are built on complex mathematical functions and principles to provide strong security and protect sensitive information from unauthorized access and attacks [1]. Algorithms like RSA and ECDSA, which secure everything from online banking to remote server access, could be rendered obsolete by sufficiently powerful quantum computers. These algorithms rely on mathematical problems that are computationally hard for classical computers—specifically, integer factorization in the case of RSA and the elliptic curve discrete logarithm problem for ECDSA. Under current assumptions, these problems would take thousands of years to solve with traditional computing power, making them practically secure. However, the development of large-scale quantum computers fundamentally alters this landscape. As a result, the security community is shifting its focus toward post-quantum cryptography—a set of cryptographic algorithms designed to withstand attacks from classical and quantum computers [2–6]. These include lattice-based, code-based, hash-based, and multivariate polynomial cryptographic approaches. The U. S. National Institute of Standards and Technology (NIST) is in the final stages of standardizing post-quantum algorithms, such as CRYSTALS-Kyber for key encapsulation and CRYSTALS-Dilithium for digital signatures [7–9].

SSH (Secure Shell), a cornerstone protocol for secure remote administration, is among the most widely used tools affected by this shift. SSH is the de facto standard for accessing and managing servers, network devices, cloud workloads, and IoT systems—especially in Unix-like environments.

[⋆] *CQPC 2025: Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine*
[*] Corresponding author.
[†] These authors contributed equally.

pavlo.a.vorobets@lpnu.ua (P. Vorobets); andrii.y.horpeniuk@lpnu.ua (A. Horpenyuk); ivan.r.opirskyi@lpnu.ua (I. Opirskyy)
0009-0007-3870-829X (P. Vorobets); 0000-0001-5821-2186 (A. Horpenyuk); 0000-0002-8461-8996 (I. Opirskyy)

It provides encrypted channels over untrusted networks and relies heavily on public-key cryptography for authentication and secure key exchange.

The question of how to future-proof SSH and the larger IT infrastructure against the upcoming wave of quantum-enabled attacks emerges as organizations prepare for the post-quantum era. This is not merely a theoretical concern—it's a strategic imperative.

SSH, being a critical component in managing servers, cloud instances, and network appliances, is particularly at risk. A compromise in SSH authentication could lead to unauthorized access, lateral movement across systems, and full-scale infrastructure breaches. Therefore, rethinking and redesigning cryptographic mechanisms in SSH is necessary in building quantum-resilient IT systems.

SSH security is one of the first crucial steps to post-quantum resilience. Now is the time for organizations to assess post-quantum cryptography options, comprehend their trade-offs, and organize smooth transitions that don't interfere with current processes. If this transition starts earlier, systems will be more robust and flexible when the quantum tipping point occurs.

2. SSH as a high-risk protocol in a post-quantum world

SSH (Secure Shell) is a cryptographic protocol that connects remote computers securely to an insecure network. It's widely used by system administrators, developers, and DevOps engineers for remote administration, file transfers, and tunneling. SSH relies on asymmetric cryptography (public-key cryptography) for authentication and symmetric encryption for secure communication.

The Transport Layer in the SSH protocol establishes a secure and confidential communication channel between the client and the server. All data exchanged between the client and server is encrypted using symmetric encryption algorithms (e.g., AES, ChaCha20). This ensures the data remains unreadable to unauthorized parties even if the communication is intercepted. The transport layer uses Message Authentication Codes (MACs), such as HMAC with SHA-2, to verify the integrity of the transmitted data. This ensures that the data has not been tampered with during transit [10]. Before exchanging sensitive data, the client verifies the server's identity using the server's host key. This prevents man-in-the-middle (MITM) attacks by ensuring the client is talking to the intended server. The layer facilitates a secure key exchange (e.g., via Diffie-Hellman or Elliptic Curve Diffie–Hellman), allowing both parties to agree on a shared session key without transmitting it over the network.

The Authentication Layer in the SSH protocol is responsible for verifying the client's identity to ensure that only authorized users can access the server. Once the Transport Layer has established a secure, encrypted connection, the server must confirm that the connecting client is legitimate. This layer supports multiple authentication methods, offering flexibility and strong security: Public Key Authentication, Password Authentication, Keyboard-Interactive Authentication, Multifactor Authentication, and GSSAPI/Kerberos Authentication.

The most common and secure method is Public Key Authentication. The client proves ownership of a private key corresponding to a public key pre-approved and stored on the server. The server issues a challenge; only a client with the correct private key can decrypt and respond appropriately.

The Connection Layer in the SSH protocol operates on top of the secure and authenticated transport established by the lower layers, allowing multiple logical communication channels to coexist within a single SSH session. This layer provides a flexible and efficient framework for managing different activities over the same encrypted tunnel. The Connection Layer enables multiple independent channels (e.g., shell sessions, file transfers, port forwarding) to run simultaneously within one SSH connection. Each channel behaves like a virtual stream with its flow control and window size. It handles opening, closing, and terminating channels cleanly, along with data flow and error signaling between client and server. All these logical channels operate over a single encrypted SSH connection, eliminating the need to open multiple TCP connections for separate tasks.

SSH is deeply integrated into automation tools, scripts, and infrastructure-as-code deployments. A compromise of a single SSH private key could cascade into full access to production environments, allowing for lateral movement and privilege escalation. In post-quantum terms, one broken key may equal full systemic compromise.

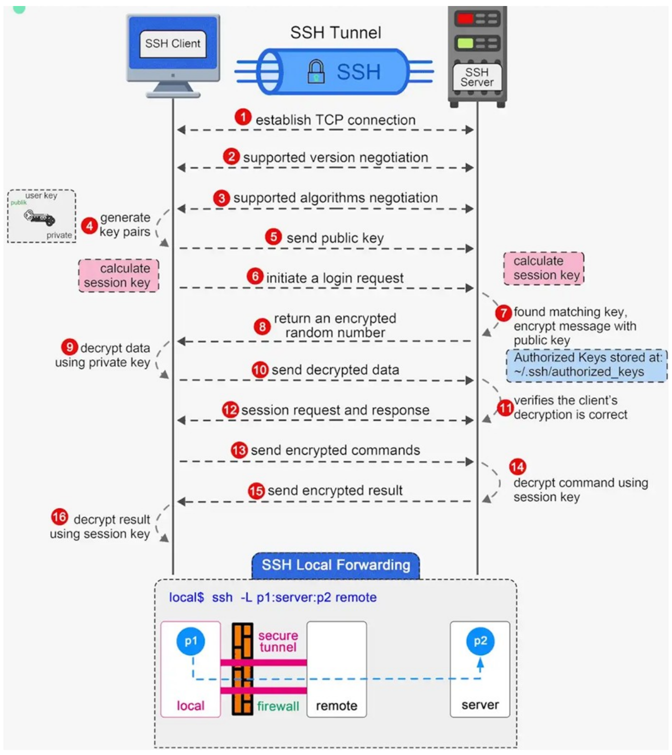
SSH Client
SSH Tunnel
SSH
SSH Server
1 establish TCP connection
2 supported version negotiation
3 supported algorithms negotiation
user key
publik
private
4 generate key pairs
5 send public key
calculate session key
6 initiate a login request
calculate session key
7 found matching key, encrypt message with public key
Authorized Keys stored at: ~/.ssh/authorized_keys
8 return an encrypted random number
9 decrypt data using private key
10 send decrypted data
11 verifies the client's decryption is correct
12 session request and response
13 send encrypted commands
14 decrypt command using session key
15 send encrypted result
16 decrypt result using session key
SSH Local Forwarding
local$ ssh -L p1:server:p2 remote
p1
secure tunnel
firewall
local
remote
p2
server

Figure 1: SSH mechanism

Unlike ephemeral session keys, SSH keys often remain valid for years. Many organizations use static key-based authentication with poor or no key rotation practices. This makes SSH particularly vulnerable to “harvest now, decrypt later” attacks, where encrypted session data and public keys are collected today, intending to decrypt and exploit them once quantum decryption becomes viable.

From a formal security standpoint, SSH is analyzed under the Authenticated and Confidential Channel Establishment (ACCE) model—a framework that ensures secure key exchange and encrypted communication. In the ACCE model, the protocol is expected to provide strong authentication, confidentiality, and integrity guarantees. SSH achieves ACCE security through a combination of algorithms, including: Diffie-Hellman or elliptic-curve Diffie-Hellman (ECDH) for key exchange, RSA or ECDSA for authentication (typically host keys), Symmetric encryption (AES), and MACs for transport protection.

The security of Authenticated and Confidential Channel Establishment protocols relies on two main properties: a secure handshake, which ensures proper authentication of the communicating parties, and a secure channel, which guarantees confidentiality and integrity of the messages exchanged after the handshake.

However, these classical building blocks are the ones quantum computers are expected to break. Shor’s algorithm can efficiently solve both integer factorization and the elliptic curve discrete logarithm problem—rendering RSA, DSA, and ECDSA insecure. Likewise, the security of DH and ECDH collapses under quantum attack, compromising the confidentiality of the session key derived during the handshake. As a result, the ACCE security guarantees of SSH are fundamentally undermined in a post-quantum world. A quantum adversary could: recover private keys from public ones, forge valid signatures, decrypt previously captured SSH sessions, impersonate trusted hosts or clients [11].

To preserve the ACCE security model in the quantum era, SSH must evolve to incorporate post-quantum key exchange and authentication mechanisms, ideally through hybrid approaches that combine classical and quantum-safe primitives during the transition.

3. Post-quantum SSH: Current state

As the cryptographic community races to prepare for the quantum era, Secure Shell has become a focal point in the transition toward Post-Quantum Cryptography (PQC). Post-Quantum Cryptography is Quantum-Safe Cryptography (QSC) designed to be quantum-safe and operate on existing computers and networks. While SSH remains one of the most widely used and trusted protocols for secure remote access, its reliance on classical cryptographic primitives like RSA, ECDSA, and ECDH makes it fundamentally vulnerable to quantum attacks. Fortunately, efforts to future-proof SSH are already underway, though still in early stages of adoption. Post-Quantum Cryptography replaces Public Key algorithms already five decades old with ones that can resist the Quantum Threat [12].

The most significant development in post-quantum SSH has come from the OpenSSH project—the de facto standard SSH implementation in Unix-like systems. As of OpenSSH, support has been introduced for hybrid key exchange mechanisms, combining both classical and post-quantum algorithms. This is achieved through integration with Open Quantum Safe (OQS) libraries, particularly liboqs [13]. It is a project that develops and prototypes quantum-resistant cryptography algorithms. The OQS project provides an open-source library that implements several post-quantum cryptographic algorithms. It plays a crucial role in experimental integrations of Post-Quantum Cryptography into widely used protocols such as SSH. The OQS library aims to facilitate the research, development, and integration of quantum-resistant algorithms into various applications and systems [14].

Several experimental implementations and research prototypes have emerged as the cryptographic community explores how to bring post-quantum security to SSH. These efforts aim to test the feasibility, performance, and interoperability of post-quantum cryptographic primitives within the SSH protocol stack. While promising, these integrations are not yet ready for production use and come with various technical and practical limitations. By providing implementations of NIST-selected and other leading PQ algorithms, it helps developers test and integrate quantum-resistant primitives into real-world applications like SSH. While not yet ready for production, liboqs is an essential building block for the secure infrastructure of the quantum future.

OQS-OpenSSH is a fork of OpenSSH that adds quantum-safe cryptography to enable its use and evaluation in the SSH protocol. This fork is currently based on OpenSSH version 9.7. It is at an experimental stage and has not received the same auditing and analysis level as OpenSSH. liboqs is provided "as is," without warranty of any kind. As research advances, the supported algorithms may see rapid changes in their security and may even prove insecure against both classical and quantum computers. OQS-OpenSSH believes that the NIST Post-Quantum Cryptography standardization project is currently the best avenue to identifying potentially quantum-resistant algorithms, and strongly recommends that applications and protocols rely on the outcomes of the NIST standardization project when deploying quantum-safe cryptography [15].

Most early quantum-involved systems are anticipated to adopt a hybrid approach, utilizing both quantum and classical technologies. This hybrid model is designed to harness the strengths of both quantum and classical computing to create more robust and efficient solutions for various applications [16]. Quantum computers, while holding the potential for certain types of computations, are still in their nascent stages of development and are not yet ready to completely replace classical computers [17]. Thus, the practical implementation of quantum computing will likely involve integrating quantum capabilities into existing classical systems to address specific tasks where quantum advantages are prominent, such as cryptography, optimization, and simulations [18].

Table 1
Supported Algorithms on OQS-OpenSSH

Algorithm Name	Specification
	Key Exchange
BIKE	bike-l1-sha512*, bike-l3-sha512*, bike-l5-sha512*
ClassicMcEliece	classic-mceliece-348864-sha256*, classic-mceliece-348864f-sha256*, classic-mceliece-460896-sha512*, classic-mceliece-460896f-sha512*, classic-mceliece-6688128-sha512*, classic-mceliece-6688128f-sha512*, classic-mceliece-6960119-sha512*, classic-mceliece-6960119f-sha512*, classic-mceliece-8192128-sha512*, classic-mceliece-8192128f-sha512*
FrodoKEM	frodokem-640-aes-sha256*, frodokem-976-aes-sha384*, frodokem-1344-aes-sha512*, frodokem-640-shake-sha256*, frodokem-976-shake-sha384*, frodokem-1344-shake-sha512*
HQC	hqc-128-sha256, hqc-192-sha384, hqc-256-sha512
Kyber	kyber-512-sha256*, kyber-768-sha384*, kyber-1024-sha512*
ML-KEM	mlkem512-sha256*, mlkem768-sha256*, mlkem1024-sha384*
NTRU-Prime	ntruprime-sntrup761-sha512*
	Digital Signature
Dilithium	dilithium2, dilithium3, dilithium5
Falcon	falcon512*, falcon1024*, falconpadded512, falconpadded1024
MAYO	mayo1, mayo2*, mayo3*, mayo5*
ML-DSA	mldsa44*, mldsa65*, mldsa87*
SPHINCS	sphincssha2128fsimple*, sphincssha2128ssimple, sphincsshake128fsimple, sphincsshake128ssimple, sphincssha2192fsimple, sphincssha2192ssimple, sphincsshake192fsimple, sphincsshake192ssimple, sphincssha2256fsimple*, sphincssha2256ssimple, sphincsshake256fsimple, sphincsshake256ssimple

SSH was designed with algorithmic agility in mind—the ability to support and negotiate between multiple cryptographic algorithms within each functional category. This modular design allows both the client and the server to advertise and agree upon supported algorithms for key exchange, public key authentication, encryption (symmetric ciphers), and message authentication (MACs or hash functions) during the SSH handshake [19].

This flexibility has historically enabled SSH to transition from weaker or deprecated algorithms (like DSA or 3DES) to stronger ones (such as Ed25519, ChaCha20-Poly1305, and SHA-2) without rewriting the protocol from scratch. Each party maintains a prioritized list of supported algorithms, and during the connection setup, the two sides negotiate the best standard option. If a particular algorithm is deprecated or compromised, administrators can update configurations to disallow its use without upgrading every client and server simultaneously.

In the context of post-quantum cryptography, this architectural feature becomes especially important. It allows for the gradual introduction of quantum-safe algorithms—even in hybrid deployments where classical and post-quantum mechanisms are used together for compatibility and risk mitigation. For example, recent versions of OpenSSH have begun to support hybrid key exchange schemes, which combine classical elliptic-curve Diffie-Hellman (ECDH) with post-quantum algorithms like Kyber, leveraging SSH's negotiation mechanism [20, 21].

SSH's algorithm agility makes it well-positioned for cryptographic evolution, including the upcoming migration to NIST-standardized post-quantum algorithms. However, agility alone isn't enough—secure defaults, consistent key rotation, and robust deployment practices must accompany this flexibility to ensure strong protection in the quantum era.

The hybrid approach offers organizations the opportunity to harness the emerging potential of quantum computing while maintaining compatibility with their established classical infrastructure.

It also provides a gradual transition as quantum technologies advance and become more applicable for broader usage. As the quantum computing field progresses and matures, the integration of quantum and classical technologies is expected to become more seamless and sophisticated, realizing more capable and efficient quantum-involved systems.

4. Recommendations for IT infrastructure

As the quantum threat becomes increasingly real, IT and security teams must begin preparing their infrastructure — especially cryptographic components like SSH — for a secure transition.

Begin by creating a detailed inventory of where and how cryptographic algorithms are used across your infrastructure. Review how SSH keys are provisioned, distributed, and managed. Organizations must begin by auditing existing SSH key usage and implementing regular rotation policies to reduce long-term exposure and prepare for a smooth migration toward post-quantum cryptography. The first step is understanding what keys exist in the environment, their use, and whether they comply with current cryptographic standards. A detailed understanding of the SSH key landscape enables infrastructure teams to minimize unnecessary risk and prioritize the most vulnerable components for replacement [22].

The following step is for organizations to move beyond passive auditing and take active steps to reduce exposure to vulnerable cryptographic primitives. One of the most immediate and impactful actions is to disallow RSA keys, particularly those with key lengths of 2048 bits or less, which are among the most widely used but soon-to-be-cryptographically-insecure key types. For OpenSSH versions ≥8.8, this enforcement aligns with the protocol's move to deprecate ssh-rsa by default.

In traditional SSH deployments, it is common for keys to remain valid for years, often without lifecycle management. This practice, while convenient, is risky—especially in the context of evolving threats such as credential leakage, insider misuse, and the looming quantum decryption threat. Defining a finite key lifespan and enforcing regular rotation is fundamental in improving operational security and reducing cryptographic exposure. Organizations should establish a standard maximum validity period for all SSH keys based on the system's sensitivity. Organizations should combine policy, tooling, and user education to enforce SSH key rotation effectively.

Finally, hybrid mechanisms—currently being evaluated and prototyped in experimental forks of OpenSSH—should be thoroughly tested and gradually implemented as part of a forward-looking cryptographic strategy. These mechanisms combine classical and post-quantum algorithms and offer a practical transitional path toward quantum-resistant secure communication. Organizations can maintain compatibility with existing systems by adopting a hybrid approach while gaining protection against future quantum adversaries. Testing in isolated environments allows security teams to evaluate interoperability, performance implications, and key management challenges without disrupting production [23]. Over time, as standards solidify and official OpenSSH releases begin incorporating post-quantum support, hybrid key exchanges can be a critical foundation for a robust and future-proof SSH infrastructure.

5. SSH with PQ algorithms implementation

The official OpenSSH project has taken a more conservative path, prioritizing stability and standards alignment. While it has begun introducing limited hybrid key exchange support (e.g., sntrup761x25519-sha512@openssh.com), full PQ integration will likely follow the finalization of NIST's standardization process and broader ecosystem readiness.

Organizations exploring PQ-SSH deployments should use hybrid algorithms to bridge the gap between classical and quantum-safe security. Hybrid algorithms maintain compatibility with existing infrastructure while introducing quantum resilience. If the post-quantum component turns out to be flawed or poorly implemented, the classical component still provides a known level of security [24].

Before deploying post-quantum or hybrid SSH algorithms into production, it's essential to test them in controlled, isolated environments rigorously. These testbeds allow infrastructure teams to assess how the new cryptographic primitives affect real-world performance — including connection latency, CPU and memory usage, and handshake durations. Testing in isolation also helps identify compatibility issues with existing clients, servers, libraries, and automation workflows. Isolated testing enables a safe and iterative transition, offering insights that guide configuration tuning, user training, and fallback planning. It also helps organizations stay aligned with compliance requirements and anticipate potential interoperability gaps before quantum-safe algorithms are adopted more broadly in production.

It is also crucial to track developments from NIST, OpenSSH, and OQS to ensure alignment in the future. These organizations are at the forefront of shaping the post-quantum security landscape. By staying informed about evolving standards, algorithm selections, and implementation best practices, infrastructure teams can proactively adapt their systems and avoid costly retrofits when post-quantum cryptography becomes a baseline requirement [25].

To better understand the practical implications of post-quantum SSH, let's walk through deploying and configuring an SSH host that supports post-quantum algorithms using the OQS-OpenSSH fork. This hands-on setup enables security teams to evaluate the real-world performance, compatibility, and operational considerations of hybrid and quantum-safe key exchange mechanisms in a controlled environment.

To begin testing post-quantum SSH in a realistic environment, let's deploy two EC2 instances running Ubuntu 24.04: one will act as the SSH server, and the other as the client. This setup will allow us to test key exchange, authentication, and session stability using hybrid post-quantum algorithms.

AWS provides a quick and isolated environment where you can safely build and experiment without interfering with production systems.

Before installing any dependencies or building post-quantum libraries, ensuring that both EC2 instances are fully updated is essential. This minimizes compatibility issues and ensures you work with the latest security patches.

Once we've confirmed that the system is up to date, the next step is to install all the necessary dependencies to build our quantum-safe SSH stack. These packages include compilers, development libraries, and tools needed to create the liboqs library and the OQS-enhanced OpenSSH fork from the source.

Now that the system is updated and all necessary dependencies are in place, we're ready to begin building. The first step is to clone the source code for both the Open Quantum Safe (liboqs) library and the OQS-OpenSSH fork,

Full list of commands:

```
sudo apt update
sudo apt -y upgrade
sudo apt -y install autoconf \
            automake \
            cmake \
            gcc \
            git \
            libtool \
            libssl-dev \
            make \
            ninja-build \
            zlib1g-dev
git clone --depth 1 --branch OQS-v9 https://github.com/open-quantum-safe/openssh.git
```

The following instructions will install liboqs in a subdirectory inside the OpenSSH source. Building liboqs requires your system to have OpenSSL 1.1.1 or higher already installed. It will automatically be detected if it is under /usr or another standard location. To simplify the process of building both liboqs and the custom OQS-OpenSSH fork, the Open Quantum Safe team provides a set of automation scripts. These scripts handle your full setup—from cloning dependencies to compiling the SSH binaries.

```
cd openssh
./oqs-scripts/clone_liboqs.sh
./oqs-scripts/build_liboqs.sh
./oqs-scripts/build_openssh.sh
```

After the build is complete, the SSH binaries will be located in the openssh/oqs-bin directory. You're ready to configure and test post-quantum SSH key exchanges using any of the supported algorithms.

Basic interoperability and algorithm verification can be performed using the provided script. This script runs predefined connection and key exchange tests across supported hybrid and pure PQC algorithms, helping teams quickly assess readiness and identify integration issues before deploying in production.

The next step is to create a set of quantum-safe SSH keys to use when connecting to the VM. With the growing range of supported post-quantum algorithms. Organizations can experiment with cryptographic approaches to evaluate their performance, compatibility, and operational impact. By generating keys for each supported algorithm using the command, administrators can build a comprehensive test set, enabling side-by-side comparisons of key sizes, handshake speeds, and connection stability, and ensuring their infrastructure is ready for the post-quantum era.

```
./oqs-test/run_tests.sh
make tests -e LTESTS=""
```

To make an SSH connection using the quantum-safe tools we just built, we need to run a separate instance of the OQS-enhanced SSH daemon, instead of replacing the system-wide SSH service. This allows for safe, side-by-side testing without disrupting existing SSH access.

```
/usr/openssh/sshd -D -e \
     -f /usr/openssh/regress/sshd_config \
    -o KexAlgorithms=kyber-512-sha256 \
    -o HostKeyAlgorithms=ssh-ecdsa-nistp521-falcon1024 \
    -o PubkeyAcceptedKeyTypes=ssh-ecdsa-nistp521-falcon1024
```

On the client side, connect to the quantum-safe SSH server using the matching hybrid cryptographic options and specifying the quantum-safe private key.

```
/usr/openssh/ssh -o KexAlgorithms=kyber-512-sha256 \
   -o HostKeyAlgorithms=ssh-ecdsa-nistp521-falcon1024 \
   -o PubkeyAcceptedKeyTypes=ssh-ecdsa-nistp521-falcon1024 \
   -o PasswordAuthentication=no \
   -o LogLevel=DEBUG3 \
   -i /usr/openssh/regress/ssh-ecdsa-nistp521-falcon1024 \
   -p 4242 \
   ububntu@server-ip
```

In a production environment, the OQS-enhanced SSH daemon can be configured to run alongside or eventually replace the standard SSH service, allowing organizations to phase in post-quantum security gradually. Access policies can be set to restrict or limit connections via the legacy SSH daemon while encouraging or enforcing the use of quantum-safe protocols. This staged approach minimizes operational risks and provides flexibility during the migration to a fully quantum-resistant remote access infrastructure. This hands-on setup demonstrates that integrating post-quantum cryptography into existing infrastructure—particularly for critical protocols like SSH—is no longer theoretical. With tools like liboqs and OQS-OpenSSH, organizations can start experimenting today, laying the foundation for a smooth and secure transition into the post-quantum era.

Conclusions

The arrival of quantum computing presents a fundamental challenge to classical cryptographic protocols—and SSH, as a foundational tool for secure remote administration, is no exception. Algorithms like RSA and ECDSA, which currently secure countless SSH connections worldwide, are particularly vulnerable to quantum attacks and will inevitably need to be retired once sufficiently powerful quantum machines become available. This looming shift underscores the urgency of transitioning toward post-quantum cryptographic (PQC) solutions.

Organizations have a critical opportunity to prepare as standards evolve and NIST finalizes its post-quantum cryptography selections. This preparation should not be postponed until a quantum threat is imminent—by then, migration timelines and operational risks could be unmanageable. Instead, the transition should begin by testing hybrid algorithms combining classical and quantum-safe techniques, enforcing cryptographic agility across systems, and gradually phasing out vulnerable key types.

Proactive steps such as maintaining a comprehensive cryptographic inventory, enforcing SSH key rotation policies, and closely tracking developments from NIST, OpenSSH, and the Open Quantum Safe (OQS) project will help ensure a smooth migration. By aligning with upstream development and participating in the early adoption of PQC-enabled SSH implementations, infrastructure teams can safeguard remote access in the current classical computing era and the coming quantum age.

In short, the post-quantum transition for SSH is not a matter of if, but when—and organizations that act now will be better positioned to maintain secure, trusted communications well into the future.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] S. Yevseiev, et al., Development of niederreiter hybrid crypto-code structure on flawed codes, East.-Eur. J. enterprise Technol. Inf. Control. Syst. 9(97) (2019) 27–38. doi:10.15587/1729-4061.2019.156620

[2] A Bessalov, et al., Computing of odd degree isogenies on supersingular twisted Edwards curves, in: Cybersecurity Information and Telecommunication Systems, 2923, 2021, 1–11.

[3] A. Bessalov, et al., Multifunctional CRS encryption scheme on isogenies of nonsupersingular Edwards curves, in: Classic, Quantum, and Post-Quantum Cryptography, 3504, 2023, 12–25.

[4] A. Bessalov, et al., CSIKE-ENC combined encryption scheme with optimized degrees of isogeny distribution, in: Cybersecurity Providing in Information and Telecommunication Systems, 3421, 2023, 36–45.

[5] A. Bessalov, V. Sokolov, S. Abramov, Efficient commutative PQC algorithms on isogenies of Edwards curves, Cryptogr. 8(3(38)) (2024) 1–17. doi:10.3390/cryptography8030038

[6] S. Abramov, A. Bessalov, V. Sokolov, Properties of isogeny graph of non-cyclic Edwards curves, in: Cybersecurity Providing in Information and Telecommunication Systems, 3550 (2023) 234–239.

[7] L. Chen, et al., Report on post-quantum cryptography, NIST Publications, 2016. doi:10.6028/NIST.IR.8105

[8] G. Alagic, et al., Status report on the second round of the NIST post-quantum cryptography standardization process, NIST Publications, 2020. doi:10.6028/NIST.IR.8309

[9] G. Alagic, et al., Status report on the third round of the NIST post-quantum cryptography standardization process, NIST Publications, 2022. doi:10.6028/NIST.IR.8413

[10] M. Kumar. Post-quantum cryptography algorithm's standardization and performance analysis, Array 15, 2022, art. 100242. doi:10.1016/j.array.2022.100242

[11] V Pastushenko, D. Kronberg, Improving the performance of quantum cryptography by using the encryption of the error correction data, Entropy 25, 2023, art. 956. doi:10.3390/e25060956

[12] P. Wallden, E. Kashefi, Cyber security in the quantum era, Communications of the ACM, 62(4) (2019) 120–120. doi:10.1145/3241037

[13] AWS, AWS-LC Library, 2022. https://github.com/awslabs/aws-lc

[14] OpenSSH, OpenSSH 10.0 introduces default post-quantum key exchange algorithm, 2025. https://quantumcomputingreport.com/openssh-10-0-introduces-default-post-quantum-key-exchange-algorithm/

[15] B. Zhou, H. Jiang, Y. Zhao, CPA-secure KEMs are also sufficient for post-quantum TLS 1.3 2024, to appear at ASIACRYPT, 2024. https://eprint.iacr.org/2024/1360

[16] W. Barker, W. Polk, M. Souppaya, Getting ready for post-quantum cryptography: Exploring challenges associated with adopting and using post-quantum cryptographic algorithms, NIST Cybersecurity White Paper, 2021. doi:10.6028/NIST.CSWP.04282021

[17] Cloudflare, Encrypting your WAF payloads with hybrid public key encryption (HPKE), 2022. https://blog.cloudflare.com/encrypt-waf-payloads-hpke/

[18] M. Friedl, J. Mojzis, S. Josefsson, Secure shell (SSH) key exchange method using hybrid Streamlined NTRU Prime sntrup761 and X25519 with SHA-512: sntrup761x25519-sha512,ver. 3, 2024. https://datatracker.ietf.org/doc/draft-josefsson-ntruprime-ssh/

[19] F. Bergsma, et al., Multi-ciphersuite security of the Secure Shell (SSH) protocol, 2019. https://eprint.iacr.org/2013/813.pdf

[20] B. Bencina, B. Dowling, V. Maram, K. Xagawa, Post-quantum cryptographic analysis of SSH. IEEE Symposium on Security and Privacy (SP), 2025, 595–613, doi:10.1109/SP61157.2025.00126

[21] P. Vorobets, O. Vakhula, A. Horpenuk, N. Korshun. (2024) Implementing post-quantum KEMs: practical challenges and solutions, in: Cybersecurity Providing in Information and Telecommunication Systems II, 3826, 2024, 212–219.

[22] P. Vorobets, A. Horpenyuk, I. Opirskyy, Analysis of problems and prospects of implementation of post-quantum cryptographic algorithms, in: Classic, Quantum, and Post-Quantum Cryptography, 3504, 2023, 39–49.

[23] C. Portmann, R. Renner, Security in quantum cryptography, Rev. Mod. Phys. 94 (2022) art. 025008. doi:10.1103/RevModPhys.94.025008

[24] V. Pastushenko, D. Kronberg, Improving the performance of quantum cryptography by using the encryption of the error correction data, Entropy 25 (2023) art. 956. doi:10.1103/RevModPhys.94.025008