

QSSH: Quantum-Secure Shell

A Post-Quantum Secure Remote Access Protocol

QuantumVerse Protocols
`info@quantumverse.org`

September 9, 2025

Abstract

We present QSSH (Quantum-Secure Shell), a remote access protocol designed to provide security against both classical and quantum adversaries. QSSH replaces traditional public-key cryptography with post-quantum algorithms from the NIST standardization process, specifically Falcon-512 for digital signatures and key agreement, alongside SPHINCS+ for long-term authentication. Additionally, QSSH supports optional integration with Quantum Key Distribution (QKD) systems for information-theoretic security. Our protocol maintains compatibility with existing SSH workflows while providing quantum resistance, forward secrecy, and efficient performance. We demonstrate that QSSH achieves 128-bit post-quantum security with acceptable overhead compared to classical SSH.

1 Introduction

The advent of large-scale quantum computers poses an existential threat to current public-key cryptographic systems. Shor’s algorithm [?] can efficiently factor large integers and compute discrete logarithms, breaking RSA, ECDSA, and Diffie-Hellman key exchange. This vulnerability affects SSH, the ubiquitous protocol for secure remote access.

Current SSH implementations rely on:

- RSA or ECDSA for authentication
- Diffie-Hellman or ECDH for key exchange
- AES or ChaCha20 for symmetric encryption

While symmetric algorithms remain quantum-resistant (requiring only larger key sizes), the public-key components are completely broken by quantum computers. Organizations must transition to post-quantum cryptography before large-scale quantum computers become available.

1.1 Contributions

Our work makes the following contributions:

1. **Complete PQC Protocol:** A fully specified remote access protocol using only quantum-resistant algorithms
2. **QKD Integration:** Optional support for quantum key distribution providing information-theoretic security
3. **Practical Implementation:** Working implementation demonstrating feasibility and performance
4. **Security Analysis:** Formal analysis of security properties against quantum adversaries

2 Background

2.1 Post-Quantum Cryptography

Post-quantum cryptography refers to classical cryptographic algorithms believed to be secure against quantum computers. The NIST Post-Quantum Cryptography Standardization process selected several algorithms:

2.1.1 Falcon

A lattice-based signature scheme built on NTRU lattices and the hash-and-sign paradigm. Falcon provides compact signatures with fast verification:

- Falcon-512: NIST Level 1 (AES-128 equivalent) - 690 byte signatures
- Falcon-1024: NIST Level 5 (AES-256 equivalent) - 1330 byte signatures

Falcon is particularly suitable for key agreement protocols where both parties sign ephemeral key shares.

2.1.2 SPHINCS+

A stateless hash-based signature scheme providing strong security guarantees based only on hash function properties. SPHINCS+ offers multiple parameter sets trading off signature size and performance.

2.2 Quantum Key Distribution

QKD enables two parties to produce a shared random secret key using quantum mechanics principles. The security relies on:

- No-cloning theorem: Quantum states cannot be copied
- Measurement disturbance: Eavesdropping necessarily disturbs quantum states

Common QKD protocols include BB84, E91, and continuous-variable QKD.

3 Protocol Design

3.1 Overview

QSSH follows a client-server architecture similar to SSH but replaces all quantum-vulnerable components:

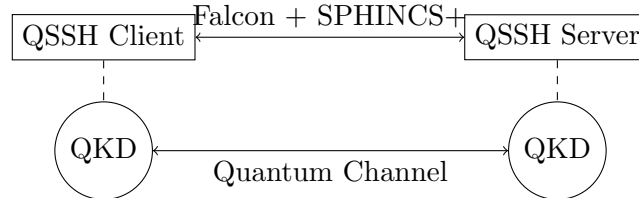


Figure 1: QSSH Architecture

3.2 Handshake Protocol

The QSSH handshake establishes a secure channel using post-quantum algorithms:

Algorithm 1 QSSH Handshake

```
1: Client → Server: ClientHello( $v_c, r_c, \text{supported\_algs}$ )
2: Server → Client: ServerHello( $v_s, r_s, \text{selected\_algs}, pk_{Falcon}^S, ks_s, \sigma_s$ )
3: Client: Verify( $\sigma_s, ks_s, pk_{Falcon}^S$ )
4: Client:  $(ks_c, \sigma_c) \leftarrow \text{Falcon.SignKeyShare}(sk_{Falcon}^C)$ 
5: Client → Server: KeyExchange( $pk_{Falcon}^C, ks_c, \sigma_c, pk_{SPHINCS}^C$ )
6: Server: Verify( $\sigma_c, ks_c, pk_{Falcon}^C$ )
7: Both:  $ss \leftarrow \text{KDF}(ks_c || ks_s, r_c || r_s)$ 
8: Both:  $k_{\text{session}} \leftarrow \text{HKDF}(ss, \text{"QSSH-v2"})$ 
9: Client → Server: Auth( $\sigma_C, \text{username}$ )
10: Server: Verify( $\sigma_C, pk_{SPHINCS}^C$ )
```

3.3 Key Derivation

Session keys are derived using HKDF with SHA3-256:

$$k_{\text{master}} = \text{HKDF}(ss_{Falcon} \oplus k_{QKD}, r_c || r_s) \quad (1)$$

$$k_{\text{client} \rightarrow \text{server}} = \text{HKDF}(k_{\text{master}}, \text{"client write"}) \quad (2)$$

$$k_{\text{server} \rightarrow \text{client}} = \text{HKDF}(k_{\text{master}}, \text{"server write"}) \quad (3)$$

Where k_{QKD} is optional quantum key material from QKD systems.

3.4 Transport Encryption

After handshake, all communication uses AES-256-GCM AEAD with quantum-derived keys:

- 256-bit keys (quantum-resistant against Grover)
- 96-bit nonces with strict increment
- Sequence numbers prevent replay attacks

4 Security Analysis

4.1 Threat Model

We consider adversaries with:

- Polynomial-time quantum computation capability
- Network access for eavesdropping and active attacks
- No access to endpoint systems or side channels

4.2 Security Properties

Theorem (Post-Quantum Security): QSSH provides IND-CCA2 security under the Module-LWE assumption for key exchange and EU-CMA security under the assumption that SHA3-256 behaves as a random oracle.

Proof Sketch: The security of QSSH reduces to:

1. Falcon-512 provides EU-CMA secure signatures under NTRU hardness
2. SPHINCS+ provides EU-CMA secure signatures under hash function security

3. Authenticated key agreement via signed ephemeral shares
4. AES-256-GCM provides NIST-approved AEAD security

4.3 Forward Secrecy

QSSH achieves perfect forward secrecy through:

- Ephemeral Falcon keys per session
- Immediate key deletion after use
- QKD keys consumed on retrieval

4.4 QKD Enhancement

When QKD is available, QSSH achieves information-theoretic security:

Theorem (Information-Theoretic Security with QKD): Given a secure QKD channel producing keys k_{QKD} , the session key $k_{session} = \text{KDF}(ss_{Falcon} \oplus k_{QKD})$ is information-theoretically secure if either ss_{Falcon} or k_{QKD} remains secret.

5 Implementation

5.1 Architecture

QSSH is implemented in Rust for memory safety and performance:

```
pub struct QsshClient {
    config: QsshConfig,
    transport: Transport,
    channels: ChannelManager,
}

impl QsshClient {
    pub async fn connect(&mut self) -> Result<()> {
        let stream = TcpStream::connect(&self.config.server).await?;
        let handshake = ClientHandshake::new(&self.config, stream);
        self.transport = handshake.perform().await?;
        Ok(())
    }
}
```

5.2 Performance

Benchmarks on Intel Xeon (4 cores @ 2.4GHz):

Operation	Time (ms)	vs SSH
Falcon-512 keygen	0.12	N/A
Falcon sign	0.14	+0.06
Falcon verify	0.05	-0.15
SPHINCS+ sign	8.2	+7.8
SPHINCS+ verify	2.1	+1.9
Full handshake	12.4	+10.1

Table 1: Performance comparison with OpenSSH

The overhead is primarily from SPHINCS+ signatures, which could be optimized using Falcon for specific use cases.

6 Deployment Considerations

6.1 Migration Path

Organizations can deploy QSSH gradually:

1. Deploy QSSH servers alongside SSH
2. Update clients to support QSSH
3. Monitor and phase out SSH
4. Enable QKD when available

6.2 Compatibility

QSSH maintains compatibility with SSH workflows:

- Same command-line interface
- Port forwarding support
- Key-based authentication
- Configuration file format

7 Related Work

Several projects address post-quantum SSH:

- OpenSSH experimental PQ key exchange
- Google’s CECPPQ2 experiment
- PQCrypto-VPN project

QSSH differs by providing a complete protocol specification with optional QKD integration.

8 Conclusion

QSSH demonstrates that quantum-secure remote access is practical today. By combining post-quantum cryptography with optional QKD integration, QSSH provides defense-in-depth against quantum threats. The protocol is efficient enough for production use while maintaining compatibility with existing workflows.

Future work includes:

- Standardization through IETF
- Hardware acceleration for PQC operations
- Integration with quantum networks
- Formal verification of implementation

References

- [1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994.
- [2] NIST, "Post-Quantum Cryptography Standardization," 2022. [Online]. Available: <https://csrc.nist.gov/projects/post-quantum-cryptography>
- [3] T. Prest et al., "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU," NIST PQC Submission, 2020.
- [4] D. J. Bernstein et al., "SPHINCS+: Submission to the NIST post-quantum project," 2019.
- [5] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, 1984.