QRCS Corporation

Quantum Message Authentication Code Analysis

Title: Implementation Analysis of the Quantum Message Authentication Code (QMAC)
Author: John G. Underhill
Institution: Quantum Resistant Cryptographic Solutions Corporation (QRCS)
Date: November 2025
Document Type: Technical Cryptanalysis Report
Revision: 1.0

Chapter 1: Introduction

1.1 Scope and Provenance

This document provides a complete cryptanalytic examination of the Quantum Secure Messaging Protocol, referred to as QSMP. The analysis is based strictly on two authoritative sources: the QSMP protocol specification and the reference C implementation. All conclusions are derived from the protocol as it is actually defined and implemented. No hypothetical variants, conceptual generalizations, or inferred behaviors are assumed.

The objective of this analysis is to determine whether QSMP, in both SIMPLEX and DUPLEX modes, satisfies its intended security goals. These goals include confidentiality, integrity, authenticity, forward secrecy, and resistance to replay, truncation, and transcript manipulation. The analysis examines the handshake structure, the key establishment process, the construction of the two RCS channels, the session binding mechanism, and the behavior of the state machine encoded in the implementation.

This document is part of the QRCS Cryptanalysis Series and follows the same standards used in the analyses of DKTP, MPDC, PQS, RCS, and related components.

1.2 Construction Summary

QSMP is a post quantum authenticated key establishment protocol that produces two symmetric channels, one for each direction of communication. The design uses three cryptographic components:

1. A lattice based key encapsulation mechanism used to derive one or two shared secrets depending on the mode.
2. A post quantum digital signature scheme used to authenticate handshake values.
3. The RCS authenticated encryption construction used to protect all post handshake traffic.

QSMP supports two handshake modes with different KEM structures:

Simplex mode.
The server generates a fresh ephemeral KEM key pair and sends the public key to the client. The client does not generate a KEM key pair in this mode. The client encapsulates a shared secret to the server's public key and sends only the ciphertext. The server decapsulates, and both compute the same shared secret.

Duplex mode.
The server generates a fresh ephemeral KEM key pair as in Simplex. The client also generates its own KEM key pair. Both sides encapsulate to the other party's public key, and the resulting two shared secrets are combined with the session binding value in the key derivation function.

In both modes, the handshake includes configuration identifiers, key identities, ephemeral public keys where applicable, ciphertexts, and signatures. After all values are validated, the parties compute a session binding hash that incorporates configuration fields, key identifiers, and authenticated public keys. This binding value, combined with the shared secret or secrets, forms the input to the SHAKE based key derivation function.

The KDF produces four outputs: two symmetric keys and two nonces that initialize the transmit and receive RCS channels. These channels protect all encrypted QSMP packets, including their sequence numbers and timestamps, which are authenticated through the associated data mechanism.

The reference implementation follows this design precisely. It enforces strict state transitions for connect, exchange, establish, and data phases, and validates every field of every message before advancing the session.

1.3 Security Positioning

The security of QSMP is based on three assumptions. The KEM must satisfy indistinguishability under chosen ciphertext attack. The signature scheme must satisfy existential unforgeability under chosen message attack. The RCS construction must provide confidentiality and ciphertext integrity for messages under chosen associated data and ciphertext attack.

Under these assumptions, QSMP provides authenticated key establishment in both modes. In Simplex mode, the client authenticates the server by verifying the signature on the server's ephemeral public key and configuration. In Duplex mode, both sides authenticate each other through signatures on their respective handshake values.

QSMP incorporates explicit transcript binding through a session binding hash that covers configuration fields, key identifiers, and authenticated public keys. This prevents unknown key share attacks and ensures that derived keys are tied unambiguously to the negotiated identities. The separation into two independent RCS channels further isolates send and receive directions, limiting cross directional influence or error propagation.

The implementation augments the protocol design with strict packet validation. It enforces sequence monotonicity, checks timestamps for validity, applies complete parsing of packet lengths and identifiers, and passes the exact serialized header to the RCS instance as associated data. These measures provide additional protection against replay, truncation, and packet structure manipulation.

1.4 Organization of the Analysis

The chapters that follow examine the protocol from foundational principles through implementation specific details. The analysis includes the protocol model, the SIMPLEX and DUPLEX handshakes, the derivation of channel keys, the RCS transport layer, adversarial capabilities, reduction-based security arguments, attack surface evaluation, empirical validation, performance considerations, governance implications, and long term security posture. Each chapter is structured to support independent verification against both the specification and the reference implementation.

Chapter 2: Model and Assumptions

2.1 Cryptographic Model

QSMP is analyzed as an authenticated key establishment protocol that transitions into a pair of symmetric secure channels. The model used in this analysis follows standard approaches for protocol evaluation, similar to the Bellare and Rogaway model for authenticated key exchange and the Canetti style indistinguishability models for secure channels. The evaluation focuses on the properties guaranteed by the underlying primitives and on the explicit construction defined in the QSMP specification and the reference implementation.

The handshake produces a shared secret that is processed through a SHAKE based key derivation function, yielding two keys and two nonces. These outputs initialize two independent instances of the RCS authenticated encryption mechanism. All payload traffic is protected within these instances using sequence bound and timestamp bound associated data. The model therefore treats the post handshake channel as an AEAD protected transport layer with explicit packet structure and direction specific keys.

2.2 Adversarial Capabilities

The adversary considered in this analysis operates with full network control. The attacker may intercept, modify, reorder, drop, or replay packets. The adversary may initiate arbitrary handshake attempts with honest parties and may inject malformed or adversarial ciphertexts at any stage of the exchange. The attacker may also compromise long term keys or ephemeral keys, depending on the scenario under study. The objective is to evaluate whether these actions can produce a violation of confidentiality, integrity, or authenticity.

Quantum capabilities are assumed for the attacker in all forward-looking scenarios. This means the adversary may execute quantum algorithms against classical cryptographic assumptions, but does not have access to unbounded computation. The security of the key encapsulation mechanism and the signature scheme are considered with respect to the post quantum hardness claims made for their underlying lattice problems.

2.3 Trust Model

QSMP assumes that each party possesses an authentic copy of the other party's long term public key when such authentication is required. In the simplex mode, the server authenticates itself to the client through a signature on the handshake transcript, and the client does not provide its own authentication. In the duplex mode, both sides

authenticate each other through signatures over their respective ephemeral public keys and configuration fields.

The model assumes that the random number generator inside each endpoint is secure and provides sufficient entropy for generation of ephemeral key pairs. The model also assumes that the implementation correctly validates all received parameters, including key identifiers, algorithm identifiers, packet lengths, and signature values. The reference code enforces these checks explicitly, and the analysis treats these validation steps as part of the security boundary.

2.4 Channel and Environment Assumptions

QSMP is designed for operation over a reliable transport. The analysis therefore assumes that packet loss, corruption, or reordering is under adversarial control, but that the underlying channel does not create additional confidentiality or integrity guarantees. The protocol itself must provide all cryptographic protections.

The RCS based channel is assumed to provide standard AEAD security under chosen ciphertext attack. The associated data model, which includes the serialized packet header, ensures that sequence numbers, timestamps, and packet types are authenticated along with the payload. This binding is essential for preventing replay and truncation attacks.

The environment assumption includes the absence of side channel leakage from the implementation. This document does not evaluate timing, power, or microarchitectural leakage, although the reference implementation uses constant time operations for KEM and signature primitives provided by their respective libraries. A separate dedicated analysis would be required for side channel resistance.

2.5 Security Objectives

The protocol is expected to satisfy the following primary goals.

Confidentiality.
No adversary, classical or quantum, should be able to distinguish encrypted payloads or extract plaintext from intercepted messages.

Integrity.
The attacker should not be able to forge valid ciphertexts or modify transmitted packets without detection.

Authentication.
The adversary should not be able to impersonate a legitimate party in either simplex or duplex mode, nor should the attacker be able to induce an honest party to accept unauthenticated key material.

Forward Secrecy.
Compromise of long-term keys should not reveal past session keys. Compromise of transient per session secrets should not reveal future session keys.

Key Uniqueness.
Sessions established between honest parties must derive independent keys even when initiated concurrently or under repeated network manipulation.

These objectives form the framework within which all subsequent sections evaluate the correctness and security of QSMP.

Chapter 3: Related Work and Context

3.1 Position within the QRCS Architecture

QSMP occupies a central role in the QRCS protocol suite. It serves as the authenticated transport layer that provides confidentiality, integrity, and authenticity for application data after key establishment. QSMP depends on several other QRCS components. The key encapsulation mechanism and digital signature scheme are defined in the QSC library, and the underlying symmetric encryption and authentication are provided by RCS.

The specification of QSMP coordinates the capabilities of these components into a single handshake and channel construction. The reference implementation demonstrates that the design is compatible with constrained environments and that the RCS based channels can operate efficiently without external dependencies. This integration makes QSMP the practical deployment mechanism for the broader suite.

3.2 Relation to Classical Secure Messaging Protocols

QSMP can be viewed as a post quantum analog to classical secure messaging protocols that combine authenticated key exchange with symmetric channel protection. Well known examples include TLS, SSH, Noise based constructions, and hybrid key exchange

frameworks. These protocols share the general pattern of exchanging public keys, deriving a shared secret, authenticating the transcript, and then establishing forward secure symmetric channels.

However, QSMP differs in important ways. It uses only post quantum primitives for both key establishment and authentication. It also adopts a deterministic and compact packet format that binds sequence numbers, timestamps, and configuration fields directly into the associated data of the RCS authenticated encryption scheme. The two channel design, with separate transmit and receive keys and nonces, resembles duplex modes in modern designs but applies this separation from the start of the session.

QSMP therefore fits into a lineage of authenticated secure channels while representing a distinct construction tailored for quantum resistant operation.

3.3 Comparison with Post Quantum Protocol Proposals

Several research efforts have proposed post quantum secure messaging and key exchange protocols. Examples include post quantum variants of TLS that replace the key agreement layer with lattice based or code based KEMs while retaining classical authentication structures. Other proposals combine post quantum signatures and post quantum KEMs within frameworks similar to Noise or Signal.

QSMP differs from these proposals by adopting a fully integrated post quantum design rather than retrofitting classical frameworks. The handshake is constructed around explicit configuration fields, explicit binding of public keys and identifiers, and a session binding hash that captures all negotiated elements. This reduces structural ambiguity during negotiation and removes the legacy assumptions found in classical protocols.

The use of RCS as the symmetric layer also distinguishes QSMP. RCS does not follow the structure of AES GCM or ChaCha20 Poly1305 but instead uses SHAKE based keyed sponge operations to provide encryption and authentication. This alignment with the SHAKE family gives QSMP a consistent cryptographic foundation across asymmetric and symmetric operations.

3.4 Prior Cryptanalysis and Evaluation

The components used by QSMP have received independent scrutiny. The post quantum KEM and signature schemes included in the PQS suite are based on lattice derived hardness assumptions and have been analyzed in the context of the NIST post quantum

standardization process. RCS has been examined within the QRCS project as an AEAD construction that provides confidentiality and integrity under adversarially chosen associated data and ciphertexts.

However, the combination of these elements into a unified secure messaging protocol has not previously undergone a complete end to end cryptanalysis. QSMP introduces design choices such as dual directional channels, explicit session binding, and a strict packet structure that have not been evaluated in prior standalone analyses. The purpose of this document is therefore to provide the first complete examination of QSMP as a finished protocol.

Chapter 4: Preliminaries and Formal Definitions

4.1 Notation and Conventions

This document uses standard cryptographic notation consistent with authenticated key establishment literature. All bit strings are written in lowercase, and all algorithms appear in uppercase when referenced as abstract primitives. Concatenation is written as $x \| y$. The length of a bit string x is written as $|x|$. Random sampling from a domain D is written as $x \leftarrow D$. Deterministic function evaluation is written as $y = F(x)$. All hash and extendable output functions follow the SHAKE based definitions provided in the specification.

All security definitions assume polynomial time adversaries unless stated otherwise. Quantum adversaries are permitted quantum computation on their internal state but do not receive quantum access to classical oracles unless required by the underlying primitive definitions.

4.2 Cryptographic Primitives Used in QSMP

The protocol relies on three classes of primitives. Each primitive is treated as an abstract idealized construction for the purpose of security evaluation, with properties defined below.

Key Encapsulation Mechanism. The KEM is required to provide indistinguishability under chosen ciphertext attack. Given a public key pk, the adversary should not be able to distinguish the shared secret encapsulated in a ciphertext from a uniformly random

value, even when allowed to obtain decapsulation results on arbitrary ciphertexts except the challenge ciphertext.

Digital Signature Scheme. The signature system is required to provide existential unforgeability under chosen message attack. The adversary should not be able to produce a valid signature on any message not previously submitted to the signing oracle, even after adaptive queries.

Authenticated Encryption with Associated Data. The secure channel uses the RCS construction, which follows a keyed sponge model for authentication (KMAC). The AEAD primitive must satisfy confidentiality and ciphertext integrity. It must be infeasible to recover plaintext from ciphertext without the key, and it must be infeasible to produce a new ciphertext or associated data pair that decrypts successfully under the key.

4.3 Formal Model of Sessions and Channels

A session is defined by the ordered transcript of all handshake messages exchanged between the two parties, the validated configuration fields, and the authenticated public keys. Both parties execute the handshake algorithm using their own private keys and the received messages. When both sides complete the handshake without error, they derive the same shared secret sec. This value, together with the binding hash sch, is processed by a SHAKE based key derivation function to produce the channel parameters.

A channel consists of the tuple (k, n), where k is the symmetric key and n is the initial nonce derived from the KDF. QSMP maintains two channels per session. The transmit channel uses (k_tx, n_tx) and the receive channel uses (k_rx, n_rx). Both channels operate independently and are initialized at the moment the handshake completes.

The serialization of packet headers, including sequence numbers, timestamps, configuration identifiers, and type fields, serves as the associated data for the AEAD operation. This ensures that packet ordering, packet structure, and context are authenticated with every message.

4.4 Correctness and Consistency Requirements

Correctness of the protocol requires that both parties derive identical session keys and nonces when they execute the handshake honestly. This also implies that the KEM decapsulation procedure must behave deterministically and that both sides must use the same configuration and identifier fields when computing the binding hash.

Consistency requires that the same transcript, when processed independently by both sides, leads to the same KDF input and therefore the same RCS channel parameters. Any deviation caused by mismatched configuration, truncated packets, malformed fields, or signature errors must cause immediate rejection and termination of the handshake.

The reference implementation enforces correctness through explicit checks on packet types, lengths, algorithm identifiers, signatures, and encapsulation values. Any failure leads to a transition to an error state and destruction of the connection context.

4.5 Security Properties of the Underlying Components

For clarity, this section provides the formal definitions underlying each property relied upon in later chapters.

IND CCA Security of the KEM. A KEM is IND CCA secure if no efficient adversary can distinguish the shared secret embedded in a challenge ciphertext from a random value, even with access to a decapsulation oracle for all other ciphertexts.

EUF CMA Security of the Signature Scheme. A signature scheme is EUF CMA secure if no efficient adversary can produce a valid signature on a new message after adaptive queries to a signing oracle.

AEAD Security of RCS. RCS is assumed to provide IND CPA confidentiality and INT CTXT integrity. Under these assumptions, the attacker cannot distinguish ciphertexts or forge valid ciphertext and associated data pairs.

These properties form the foundation for the security analysis of the QSMP handshake and secure channel.

Chapter 5: Protocol Overview and Handshake Flow

5.1 Overview of QSMP Handshake Modes

QSMP defines two operational modes for establishing authenticated and confidential communication: SIMPLEX mode and DUPLEX mode. Both modes follow the same structural sequence of Connect, Exchange, Establish, and Data stages, but they differ in the number of encapsulation operations and the authentication guarantees they provide.

SIMPLEX mode provides one way authentication. The server authenticates itself to the client using a long-term signature key. The server generates the only ephemeral KEM key pair used in the handshake. The client receives the server's public KEM key and performs a single encapsulation to derive the shared secret.

DUPLEX mode provides mutual authentication. Both the server and the client exchange signed values, and both parties contribute ephemeral KEM keys. Two shared secrets are established, one in each direction, and these secrets are combined in the key derivation function.

Both modes terminate with symmetric keys and nonces derived from a session binding hash and one or two KEM shared secrets. These values initialize the RCS encryption contexts for both communication directions.

5.2 SIMPLEX Handshake Flow

The SIMPLEX handshake uses a single encapsulation and a single shared secret. The steps are as follows.

1. **Client Connect Request**
   The client sends the configuration string and the key identity for the server's long term verification key. No KEM key is generated on the client side in this mode. The message contains only identification and configuration values.

2. **Server Connect Response**
   The server validates the client inputs, constructs the session binding value, and generates a fresh ephemeral KEM key pair. The server hashes the public KEM key together with the serialized response header and signs the hash using its long-term private signature key. The server returns the public KEM key and the signed hash to the client.

3. **Client Exchange Request**
   The client verifies the server's signature and recomputes the authenticated hash. If valid, the client encapsulates a shared secret using the server's public KEM key and sends only the ciphertext. The client then derives the final symmetric keys and nonces by applying the key derivation function to the shared secret and the session binding value.

4. **Server Exchange Processing**
   The server verifies the ciphertext length and decapsulates the shared secret using its ephemeral private KEM key. It computes the same symmetric keys and nonces as the client and confirms key synchronization in the establish stage.

After these steps, both parties enter the data phase. SIMPLEX mode uses one shared secret and one authenticated signature, ensuring that the client is communicating with the correct server.

5.3 DUPLEX Handshake Flow

The DUPLEX handshake uses two encapsulation operations and provides mutual authentication. The steps are as follows.

1. **Client Connect Request**
   Same as SIMPLEX. The client sends the configuration string and server key identity.

2. **Server Connect Response**
   The server validates the request, generates a fresh ephemeral KEM key pair, constructs the binding value, signs the hash of its public KEM key and header, and sends the public key and signature to the client.

3. **Client Exchange Request**
   The client verifies the server's signature and authenticated hash. The client encapsulates a shared secret to the server's public key. The client then generates its own ephemeral KEM key pair, computes a hash over its own KEM public key and the ciphertext, signs that hash using its long-term signature key, and sends the ciphertext, its ephemeral KEM public key, and the signature to the server.

4. **Server Exchange Response**
   The server verifies the client signature and authenticated hash. The server decapsulates the first shared secret and encapsulates a second shared secret using the client's ephemeral public KEM key. The server derives the final symmetric keys and nonces by combining both shared secrets with the binding hash. It initializes its encryption contexts and sends the second ciphertext and a signature over the response hash.

5. **Client Establish Request**
   The client verifies the server's signature. It decapsulates the second ciphertext using its private KEM key. The client derives the same symmetric keys and nonces and sends an encrypted confirmation message.
6. **Server Establish Response**
   The server decrypts and validates the confirmation message and transitions the connection to the established state.

DUPLEX mode provides stronger guarantees than SIMPLEX mode. Both parties authenticate each other, and the key material is derived from two independent shared secrets.

5.4 Summary of Handshake Differences

The differences between SIMPLEX and DUPLEX can be summarized as follows:

- SIMPLEX uses one shared secret. DUPLEX uses two shared secrets.
- SIMPLEX uses server only authentication. DUPLEX uses mutual authentication.
- SIMPLEX requires one encapsulation and one decapsulation. DUPLEX requires two encapsulations and two decapsulations.
- SIMPLEX does not involve a client KEM key pair. DUPLEX does.
- Both modes use the same key derivation and RCS initialization procedures.

Chapter 6: Formal Security Model and Proof Framework

This chapter defines the formal model used to analyze QSMP. It specifies the adversary interface, session and partnering notions, and the security goals for both the handshake and the encrypted channels. The focus is on game based definitions and reduction style reasoning that connect protocol level security to the underlying primitives: the KEM, the signature scheme, the SHAKE based key derivation function, and the RCS authenticated encryption construction.

The model treats QSMP as an authenticated key establishment protocol that derives two symmetric channels. SIMPLEX and DUPLEX are modeled within the same framework,

with SIMPLEX providing unilateral authentication and DUPLEX providing mutual authentication and additional key entropy.

6.1 Game Based Security Definition

Let Π denote the QSMP protocol executed between two participants, A and B. Each local instance of a participant is called a session and is denoted by Π_i^j, where i identifies the participant and j is a session index. A session Π_i^j maintains a local transcript τ_i^j, internal state σ_i^j, and, if it accepts, a session key K_i^j (consisting of the transmit and receive keys and nonces).

An adversary $\mathfrak{A}$ is a probabilistic polynomial time algorithm that has complete control over the network and interacts with sessions through the following oracle queries:

- Send(Π_i^j, m): delivers a message m to session Π_i^j and returns any response produced by that session. This models active control of network traffic.
- Reveal(Π_i^j): outputs the session key K_i^j for Π_i^j, modeling post establishment key compromise.
- Corrupt(i): reveals the long-term secret keys of participant i, including its signing key and any static secrets if present.
- Test(Π_i^j): used to define indistinguishability of session keys. For a fresh session Π_i^j, the oracle returns either the real session key K_i^j or a uniformly random key of the same length, depending on a hidden bit b.

A session Π_i^j is considered fresh if it has accepted, if none of its partnered sessions have been revealed, if its own session key has not been revealed, and if corruption events do not trivially expose its key according to the rules of the particular security notion (for example, forward secrecy allows certain post establishment corruptions).

The adversary's advantage in distinguishing real keys from random is defined as

$$\mathrm{Adv}(\Pi, \mathfrak{A}) = |\Pr[b' = b] - 1/2|,$$

where b is the hidden bit used by the Test oracle and b′ is the adversary's guess. QSMP is considered AKE secure if this advantage is negligible in the security parameter.

6.2 Session Matching and Partnering

Session matching and partnering express when two local sessions correspond to the two sides of a single protocol run. Two sessions Π_i^j and Π_k^l are considered partnered if:

- Both have accepted.
- Their roles are opposite (client versus server).
- Their transcripts reflect a successful QSMP handshake for the same mode (SIMPLEX or DUPLEX).
- They derive identical session keys (transmit and receive keys and nonces).

The transcript of a session records, in order, all messages sent and received during the handshake, including configuration identifiers, key identities, ephemeral KEM public keys where applicable, ciphertexts, and signatures. For SIMPLEX, the transcript includes a client connect message, a server connect response with an authenticated public KEM key, and a client exchange message with a single ciphertext. For DUPLEX, the transcript includes the same initial messages plus the client KEM public key and signature, the server ciphertext and signature, and the encrypted establish confirmation.

Partnering is defined so that if Π_i^j and Π_k^l are partnered, then their transcripts match in all handshake relevant fields and they agree on the session mode and configuration. This partnering notion is used in the freshness conditions and in statements of authentication.

6.3 AKE Security Goals for SIMPLEX and DUPLEX

The primary goal of QSMP as an authenticated key establishment protocol is to provide session key indistinguishability and appropriate authentication properties.

For both SIMPLEX and DUPLEX, QSMP aims to provide:

- Correctness: honest sessions that complete a handshake without interference derive identical session keys.
- Key indistinguishability: the session key of a fresh session is indistinguishable from random to any efficient adversary with access to the allowed oracles.

Authentication goals differ by mode:

- SIMPLEX mode provides unilateral authentication. The client is assured of the server's identity (that it is communicating with the holder of the server's long-

term signing key). The server does not obtain cryptographic assurance of the client's identity beyond network and configuration context.

- DUPLEX mode provides mutual authentication. Both client and server are assured that they are communicating with the holders of the respective long-term signing keys.

Implicit authentication requires that if an honest session accepts with a given peer identity, then there is at most one partnered session of that peer with a matching transcript. Explicit authentication further requires that no honest session accepts unless its peer has also completed a matching session.

6.4 Channel Security Goals

After a successful handshake, QSMP establishes two symmetric channels using RCS, one in each direction. Channel security is defined with respect to an adversary that may continue to issue Send, Reveal, Corrupt, and Test queries.

The channel goals are:

- Confidentiality: ciphertexts produced and processed by RCS under the derived keys are indistinguishable from encryptions of random messages of the same length, even under chosen associated data and chosen ciphertext attacks, for all fresh sessions.
- Integrity: the adversary cannot, with non-negligible probability, produce a ciphertext and associated data pair that decrypts successfully under an honest session's receive key unless it was previously generated by the corresponding transmit key of a partnered session.
- Directional independence: compromise of the transmit key for a session does not give the adversary a practical advantage in attacking the receive key of that session, and vice versa, beyond what is implied by key exposure itself.

The associated data used in all RCS operations is the serialized QSMP header, which includes the packet flag, sequence number, payload size, and timestamp. This binds these metadata fields to the encrypted content.

6.5 Adversary Capabilities and Corruption Model

The adversary is assumed to control the network completely. It may reorder, drop, inject, or modify messages arbitrarily by issuing Send queries. The reference implementation enforces header validation, state machine checks, and cryptographic verification before advancing session states, but these checks are not assumed in the basic model; they are properties we verify within it.

Corrupt queries reveal the long-term signing keys of selected participants. For QSMP, this models compromise of the server's long term signature key and, in DUPLEX, compromise of the client's long term signature key if such a key is configured. We distinguish between pre handshake and post handshake corruptions:

- Pre handshake corruption allows the adversary to impersonate the corrupted party.
- Post handshake corruption is relevant to forward secrecy. In that case, the model assumes that ephemeral KEM keys and shared secrets have been erased as specified.

Reveal queries leak session keys after acceptance and are used to model later compromise of established channels. Test queries only apply to fresh sessions, for which neither the long-term key nor the session key has been compromised in a way that trivially determines the challenge bit.

6.6 Simplex and Duplex AKE Games

To capture the different authentication structures, we distinguish between SIMPLEX and DUPLEX AKE games.

In the SIMPLEX AKE game, the client plays the role of the entity that must authenticate the server. The adversary wins if it can cause an honest client session to accept with a server identity while there is no partnered server session with a matching transcript, or if it can distinguish the client's session key from random in a fresh session.

In the DUPLEX AKE game, both directions are enforced. The adversary wins if it can cause either party to accept with a peer identity for which there is no partnered session, or if it can distinguish the session key from random in a fresh session that satisfies the stricter mutual authentication freshness conditions.

These games encode both key indistinguishability and explicit authentication. They are used in later chapters to structure the reduction arguments.

6.7 Forward Secrecy and Replay Resistance

Forward secrecy is modeled by allowing Corrupt queries after sessions have accepted, under the restriction that ephemeral KEM keys and derived shared secrets are erased as prescribed. A session is still considered fresh for forward secrecy if long term keys are corrupted only after its acceptance, and no Reveal query has been made for its session key or any partnered session.

In SIMPLEX, forward secrecy depends on the erasure of the server's ephemeral KEM private key and the erasure of the client's local copy of the shared secret. In DUPLEX, forward secrecy depends on erasure of both parties' ephemeral KEM private keys and the corresponding shared secrets.

Replay resistance is captured by the combination of sequence numbers, timestamps, and AEAD integrity. While the game-based model does not explicitly track all low level checks, the effect is that replayed packets will not be accepted as fresh, and sequence or timestamp validation failures prevent the adversary from using old ciphertexts to influence new sessions.

6.8 Compositional Security Theorem

At a high level, security of QSMP can be expressed by a compositional theorem stating that if the underlying primitives satisfy their standard security properties, then QSMP satisfies the AKE and channel security goals defined above.

Assume that:

- The KEM used by QSMP is IND CCA secure.
- The signature scheme is existentially unforgeable under chosen message attack.
- The RCS construction is an AEAD scheme providing confidentiality and integrity under chosen associated data and chosen ciphertext attack.
- The SHAKE based KDF behaves as a pseudorandom function on its inputs.
- Randomness used for KEM and signature operations is uniform and unpredictable.

Under these assumptions, any adversary that achieves a non-negligible advantage in the SIMPLEX or DUPLEX AKE games, or in the channel confidentiality or integrity

experiments, can be used to construct an adversary that breaks at least one of the underlying primitives with non-negligible advantage. The reductions differ slightly between SIMPLEX and DUPLEX, but the overall structure is the same: successful attacks on QSMP imply attacks on the KEM, the signature scheme, the KDF, or the AEAD construction.

6.9 Conclusion of Formal Framework

This chapter defined the formal model used to analyze QSMP. It described sessions, transcripts, partnering, adversary oracles, and security goals for both handshake and data phases. The model distinguishes clearly between SIMPLEX (server authenticated, single shared secret) and DUPLEX (mutually authenticated, dual shared secret) while treating both within a unified AKE and AEAD security framework.

Subsequent chapters use this framework to perform detailed cryptanalytic evaluation, to structure reduction style proofs, and to interpret implementation level details in terms of these formal guarantees.

Chapter 7: Cryptanalytic Evaluation and Attack Surface Analysis

7.1 Overview

This chapter evaluates the attack surface of QSMP in both SIMPLEX and DUPLEX modes. The evaluation focuses on threats relevant to transcript manipulation, key encapsulation behavior, signature validation, session binding, state machine correctness, and the RCS data phase. All analysis is based strictly on the QSMP specification and the reference C implementation.

SIMPLEX and DUPLEX differ primarily in the number of encapsulation operations performed and the authentication guarantees provided. SIMPLEX uses a single encapsulation to the server’s ephemeral public KEM key, while DUPLEX uses two encapsulation operations, one in each direction, with mutual authentication through signatures. The remainder of the attack surface analysis applies to both modes unless stated otherwise. The SIMPLEX mode provides a bi-directional encrypted tunnel with 256-bit symmetric security, whereas the DUPLEX mode creates a 512-bit secure tunnel using RCS-512 operational mode and 64 bytes of initialization key for each party.

7.2 Transcript Manipulation Attacks

QSMP is designed to prevent transcript manipulation through strict message validation, deterministic transcript structure, and signature authentication.

7.2.1 Message Reordering

The reference implementation enforces a strict sequence of packet flags during the handshake. A session will only process a message if the packet flag matches the expected handshake stage. Any reordering of handshake packets results in immediate termination. This prevents attacks where an adversary replays or rearranges messages to force an inconsistent or ambiguous state.

7.2.2 Message Truncation and Omission

Handshake messages include mandatory fields that must be present before the implementation proceeds. For SIMPLEX, omitting the server's signed hash or the public KEM key causes signature verification to fail. For DUPLEX, omitting the client's public KEM key or associated signature causes immediate rejection. These structures are validated before any cryptographic computation occurs.

7.2.3 Injection of Forged Handshake Messages

The handshake relies on signatures over hashes of authenticated fields. In SIMPLEX, the client will reject any server connect response that contains a modified public KEM key, modified header fields, or any incorrect signature. In DUPLEX, both sides authenticate their ephemeral public keys and ciphertexts by signing the computed hash of their own handshake values. Any injection attempt would require forging a valid signature, which reduces to breaking the EUF CMA security of the PQS signature scheme.

7.2.4 Modification of KEM Ciphertexts

In SIMPLEX, modification of the ciphertext produced by the client causes decapsulation to fail on the server. In DUPLEX, modification of either ciphertext in the two encapsulation operations causes decapsulation failure on the receiving side. The implementation does not reveal partial information about decapsulation attempts and immediately terminates the handshake when decapsulation fails.

7.3 Attacks on KEM and Signature Layers

QSMP relies on a post quantum KEM and a post quantum signature scheme. Security claims about these layers reduce directly to the security properties of these primitives.

7.3.1 Attacks on the KEM in SIMPLEX Mode

SIMPLEX uses a single encapsulation performed by the client using the server's public KEM key. An adversary observing the ciphertext cannot recover the shared secret unless it breaks IND CCA security. The server performs a single decapsulation using its private KEM key. Because the client does not generate a KEM key pair in SIMPLEX mode, there is no second encapsulation to attack and no structural symmetry to exploit.

7.3.2 Attacks on the KEM in DUPLEX Mode

DUPLEX uses two encapsulation operations and therefore relies on two independent instances of KEM security. The adversary must break at least one of the two IND CCA assumptions to recover any part of the session key. Any attack on the second ciphertext requires breaking the client's ephemeral KEM key pair, which is generated during the exchange stage and erased after use.

7.3.3 Signature Forgery Attempts

Both modes require signatures to authenticate public KEM keys and ciphertext dependent hashes. In SIMPLEX, the server signs the hash of its KEM public key and the response header. In DUPLEX, both sides sign a hash of their handshake values. Any attacker attempting to impersonate either party must forge signatures, which reduces to breaking the PQ signature scheme.

7.4 Channel Level Attacks

After the handshake, QSMP transitions to the data phase protected by RCS. RCS is a sponge based authenticated encryption construction. All data packets include the serialized header as associated data, ensuring that changes to critical metadata fields are detected.

7.4.1 Ciphertext Forgery

Any ciphertext that does not have a valid authentication tag is rejected. Because the header is authenticated as associated data, tampering with any field in the header invalidates the tag. An attacker cannot create a new valid ciphertext without breaking the integrity guarantees of RCS.

7.4.2 Replay Attacks

QSMP uses monotonically increasing sequence numbers. The implementation checks that each received sequence number matches the expected next value. Replayed packets therefore fail state machine validation even before AEAD authentication.

7.4.3 Header Manipulation

Headers contain the packet flag, sequence number, payload size, and timestamp. Any modification to these fields is detected by the RCS integrity check. Because the header is authenticated separately for each packet, the attacker cannot manipulate metadata without detection.

7.4.4 Nonce Misuse

QSMP uses fixed nonces derived from the KDF, with one nonce for each direction. Because each session uses fresh shared secrets, nonce reuse across sessions does not occur. RCS is designed to operate correctly under this fixed nonce pattern, provided that each nonce is associated with exactly one key.

7.5 Cross Protocol and Downgrade Attacks

QSMP mitigates downgrade and cross protocol attacks through authenticated configuration and explicit algorithm identifiers.

7.5.1 Algorithm Downgrade Attempts

Both SIMPLEX and DUPLEX hash the configuration string and the key identity into the session binding value. Because the server signs the connect response and the client signs the DUPLEX exchange request, any attempt to downgrade configurations or algorithm identifiers results in signature mismatch.

7.5.2 Cross Protocol Injection

QSMP messages have fixed packet type values and a strict state machine. Messages from other protocols cannot be misinterpreted as QSMP messages. Any packet with an unexpected flag is rejected immediately.

7.6 State Machine and Parsing Attacks

The reference implementation enforces strict validation:

- All header fields are checked for bounds and correctness.
- Payload sizes must match message structures.

- Message ordering is strictly enforced.
- No partial processing occurs after errors.
- Sensitive memory is cleared on termination.

This mitigates parsing and state machine manipulation attacks. The handshake transitions only under correct packet flags and verified fields, preventing adversaries from causing silent downgrades or inconsistent state transitions.

7.7 Summary

QSMP provides strong protection against transcript manipulation, KEM based attacks, signature forgery, and attacks on the encrypted data channel. SIMPLEX and DUPLEX differ in authentication guarantees and entropy sources, but both modes rely on validated signatures, strict state machine transitions, and the AEAD security of RCS. The protocol structure and implementation together minimize the attack surface and reduce cryptanalytic weaknesses to the hardness of the underlying cryptographic primitives.

Chapter 8: Verification, Proof Reproduction, and Empirical Validation

8.1 Purpose of Verification and Reproduction

The goal of this chapter is to demonstrate that the security analysis of QSMP is reproducible, verifiable, and aligned precisely with the protocol as implemented. Verification includes reconstructing handshake transcripts directly from the specification, confirming the correctness of key derivation through direct evaluation of the C implementation, examining the consistency of signature validation logic, reproducing the decapsulation behavior, and validating the RCS encryption layer using controlled empirical tests.

The analysis must be reproducible using only the publicly documented specification, the uploaded C code, and well-known cryptographic assumptions. No hidden behaviors, undocumented features, or unstated requirements are used anywhere in the evaluation. QSMP's handshake structures, state transitions, binding mechanisms, and KDF computations are fully deterministic given valid inputs. This deterministic behavior is essential, because any reproducible security analysis must be able to examine the internal logic of the key exchange and confirm consistency between the specification and the code.

8.2 Reconstruction of SIMPLEX Handshake Behavior

Reproducing SIMPLEX handshake behavior involves tracing the control flow through the Connect, Exchange, and Establish stages and comparing these flows against the specification.

8.2.1 Connect Stage Reproduction

An independent verifier can reconstruct the Connect stage by observing the following:

1. The client sends a packet containing only:
   - the configuration string,
   - the key identity that corresponds to the server's verification key,
   - a QSMP header specifying ConnectRequest as the packet flag,
   - a monotonic sequence number of zero.
2. The server performs:
   - header validation,
   - signature key lookup using the key identity,
   - construction of the session binding value using the configuration string, key identity, and the server's long term verification key.
3. The server generates an ephemeral KEM key pair. A verifier can confirm this by checking the code path where the KEM key pair is produced during ConnectResponse processing.
4. The server signs the hash of:
   - the KEM public key,
   - the serialized response header.

The client later verifies this signature. This replicable sequence can be observed in the C code by inspecting the connect response building functions.

8.2.2 Exchange Stage Reproduction

The verifier can then inspect the Exchange stage as follows:

1. The client verifies:
   - the signature on the hash of the KEM public key,
   - the correctness of the response header,
   - time validity and sequence number.

2. The client encapsulates a shared secret to the server's KEM public key. This uses the encapsulation function exposed through the PQS library, which is called through the QSMP KEX code.
3. The client constructs an ExchangeRequest packet containing:
   - the ciphertext,
   - the correct packet flag,
   - sequence number one.
4. The server receives the ciphertext and performs decapsulation using its private KEM key. A verifier can confirm the exact decapsulation behavior in the server-side Exchange parsing code.
5. Both sides compute:
   - the session binding value (which the client recomputes),
   - the same SHAKE based key derivation inputs,
   - identical symmetric keys and nonces.
6. The server then moves into Establish stage.

An independent verifier can reconstruct every step by feeding identical inputs into the same hashing, signature, and KEM operations.

8.2.3 Establish Stage Reproduction

The Establish stage provides confirmation that both sides derived identical key material. The verifier can reproduce this by:

1. Observing that the server sends an EstablishResponse encrypted under the newly initialized RCS transmit key.
2. Observing that the client decrypts this packet under its RCS receive key.
3. Confirming that the decrypted value matches the session binding cookie generated by the server.
4. Verifying that the client returns an encrypted EstablishConfirmation containing the same cookie.

All encrypted packets can be decrypted correctly if and only if the verifier uses the same symmetric keys and nonces derived from the KDF.
This provides a direct method for verifying that the derivation process is correct.

8.3 Reconstruction of DUPLEX Handshake Behavior

DUPLEX introduces more complex authenticated structures, and its verification requires observing both directions of encapsulation and both signature checks.

8.3.1 Authenticating the Server's Contribution

Independent reproduction begins with verifying the server's connect response exactly as in SIMPLEX:

1. Generation of the server's ephemeral KEM public key.
2. Construction of the hash over the public key and the header.
3. Signature over the hash with the server's long-term signing key.

A verifier confirms that changes in any of the authenticated fields cause signature verification to fail, ensuring consistency with the specification.

8.3.2 Verifying the Client's Signed Exchange Request

In DUPLEX, the client performs three authenticated actions:

1. It encapsulates the first shared secret to the server's public key.
2. It generates its own ephemeral KEM key pair.
3. It signs the hash of:
   - its public KEM key,
   - the first ciphertext,
   - the serialized exchange request header.

A verifier checks that the signature precisely matches the structure defined in the specification and the code. Any deviation in key order, field order, or header serialization invalidates the signature.

8.3.3 Server Decapsulation and Secondary Encapsulation

Verification continues by observing that the server:

1. Verifies the client's signature.
2. Decapsulates the first shared secret from the first ciphertext.
3. Encapsulates a second shared secret using the client's ephemeral public KEM key.
4. Signs the hash of:
   - the second ciphertext,
   - the serialized exchange response header.

This completes the second authenticated contribution. The verifier can examine the server code to confirm that the two secrets are placed in the KDF in the correct order and combined with the session binding value.

8.3.4 Client Decapsulation and Final Establishment

The client then:

1. Verifies the server's signature.
2. Decapsulates the second shared secret using its private KEM key.
3. Computes the same combined KDF input.
4. Decrypts the server's EstablishResponse.
5. Sends an encrypted EstablishConfirmation.

A verifier confirms that incorrect shared secret values lead to decryption failure in the final establish exchange, which demonstrates that both shared secrets must match exactly.

8.4 Verification of the Session Binding Value

A central feature of QSMP is the deterministic session binding value. A verifier can recompute the binding hash using:

- the configuration string,
- the server's key identity,
- the server's long term verification key,
- and in DUPLEX, the client's long term verification key.

Inspecting the code shows that the binding hash appears identically in the KEX state on both sides. Any mismatch yields inconsistent KDF output, resulting in decryption failure of the Establish messages.
The binding hash is therefore reproduced exactly by hashing the prescribed fields in the required order. This enables formal verification that transcript elements are correctly incorporated.

8.5 Reproducing the Key Derivation Function

Both SIMPLEX and DUPLEX use a SHAKE based KDF that absorbs:

- SIMPLEX: one shared secret and the binding hash,

- DUPLEX: two shared secrets and the binding hash.

A verifier computes the SHAKE output stream and partitions it into:

- transmit key,
- receive key,
- transmit nonce,
- receive nonce.

Independent evaluation confirms:

1. Both parties derive identical values.
2. The mapping from KDF inputs to outputs is deterministic and unambiguous.
3. Keys change when the binding hash or shared secret changes.

This reconstruction verifies the correctness and integrity of the KDF process.

8.6 Empirical Validation of RCS Encryption Behavior

The data phase uses two RCS instances. Empirical validation consists of:

- encrypting packets using the transmit key and nonce from the KDF,
- validating that the authenticated associated data (the serialized header) must match,
- confirming that modified headers cause authentication failure,
- confirming that ciphertext forgery attempts fail,
- ensuring that sequence enforcement prevents replayed packets from reaching the decryption stage.
-

Empirical tests can demonstrate that:

- All ciphertext bits depend on both the key and nonce.
- Tampering with the payload or header results in immediate rejection.
- Replaying packets results in sequence errors before decryption.

The verifier can directly call the RCS transform functions in the implementation to perform these tests.

8.7 Validation of Forward Secrecy Behavior

Forward secrecy can be validated empirically by confirming:

- The server’s ephemeral KEM private key in SIMPLEX is generated fresh and erased after the handshake.
- Both parties’ ephemeral KEM private keys in DUPLEX are erased after the handshake.
- The shared secrets never persist after KDF computation.

Validation involves instrumenting the code or manually inspecting memory after session termination to confirm that:

- ephemeral private keys no longer exist,
- shared secrets have been overwritten,
- RCS state initialization uses only the final symmetric keys and nonces.

8.8 Verification by Independent Reimplementation

QSMP can be reimplemented by any evaluator using only:

- the public specification,
- the reference C implementation,
- the documented behavior of the cryptographic library.

An evaluator can:

1. Reconstruct handshake messages from documented formats.
2. Recompute signatures and verify them.
3. Perform encapsulation and decapsulation using the KEM.
4. Recompute the session binding hash.
5. Run the KDF to reproduce the symmetric keys.
6. Decrypt and validate encrypted packets.

Any mismatch indicates an implementation or specification deviation. Independent reproduction validates the internal consistency of QSMP's design.

8.9 Summary

The verification process demonstrates that QSMP’s security claims are reproducible directly from the specification and independently confirmable using the reference implementation. All handshake computations, signature validations, decapsulation operations, session binding values, and key derivations can be reconstructed. The RCS

data phase behavior is empirically validated through controlled encryption and decryption tests.
DUPLEX introduces additional structure, but remains fully reproducible through the same methodology. The deterministic nature of QSMP, combined with explicit authenticated transcript elements and strict state machine logic, enables a high degree of auditability and confirms that its security properties arise directly from its cryptographic components.

Chapter 9: Performance Evaluation and Deployment Considerations

9.1 Computational Cost of Core Operations

The computational cost of QSMP depends on the mode of operation and the number of KEM operations performed during the handshake. SIMPLEX mode uses a single encapsulation by the client and a single decapsulation by the server. DUPLEX mode uses two encapsulation operations and two decapsulation operations, one in each direction. This difference leads to slightly higher cost for DUPLEX during the handshake.

The memory requirements of QSMP follow directly from the handshake state structures in the reference implementation. The server stores an ephemeral KEM private key in both modes, while the client stores only a shared secret in SIMPLEX. In DUPLEX, both parties store their own ephemeral KEM private keys until the establish stage completes. All ephemeral secrets are cleared once the session transitions into the established state.

Symmetric operations in the data phase use RCS, which requires minimal memory for its state and performs efficiently on both low power devices and server hardware. The overall computational footprint of QSMP is dominated by the KEM operations during handshake and is consistent with modern post quantum cryptographic workloads.

9.2 Memory and State Requirements

The handshake latency of QSMP depends on the number of round trips and the cost of KEM and signature operations. SIMPLEX mode completes the handshake using a single encapsulation and requires fewer computations than DUPLEX. DUPLEX mode introduces an additional encapsulation and decapsulation operation, increasing handshake cost but providing higher authentication and security guarantees.

Once the handshake completes, data transmission uses authenticated RCS encryption. The throughput of the data phase is similar across both modes because symmetric encryption dominates the cost, and this component is identical in SIMPLEX and DUPLEX. The reference implementation updates sequence numbers and timestamps efficiently and performs header serialization in constant time relative to message size.

Session lifetime considerations depend on the security requirements of the deployment. Because the KDF derives fresh transmit and receive keys for each session, long lived connections remain secure as long as keys are not reused across sessions. Implementations should limit session duration in environments where long term key compromise risk is high, although QSMP has strong forward secrecy properties when ephemeral secrets are erased correctly after the handshake.

9.3 Throughput and Latency Characteristics

QSMP is adaptable to a wide range of deployment scenarios. The computational differences between SIMPLEX and DUPLEX affect deployment decisions. SIMPLEX may be preferred in environments where low computational cost is required and unilateral server authentication is acceptable. DUPLEX is well suited to mutual authentication scenarios, such as peer-to-peer links or high assurance communications, where the additional KEM and signature cost is justified by increased security.

Resource constrained devices benefit from the simplicity of the SIMPLEX handshake, which uses only one encapsulation and minimal memory for ephemeral key material. The DUPLEX handshake requires more memory to store the additional ephemeral KEM private key on the client and slightly increases processing time, but remains practical on embedded devices due to the efficiency of the underlying KEM and signature algorithms.

Both modes maintain the same packet structure and data encryption scheme, allowing mixed deployment environments to use the same data plane regardless of handshake mode. This uniformity simplifies interoperability and integration with existing systems.

9.4 Deployment in Constrained Environments

QSMP is designed with compatibility for constrained environments, including embedded devices and low power systems.

Limited Code Footprint. The protocol logic is compact and does not depend on large external libraries. Most complexity resides in the cryptographic primitives, which are already optimized in standard PQS and RCS implementations.

Bounded Packet Sizes. The protocol avoids excessively large message structures. Public keys and signatures dominate handshake sizes, but once the channel is established, packet sizes remain small and predictable.

Low Memory Persistence. Since temporary handshake values are discarded after use and channel keys are small, QSMP suits devices with tight memory budgets.

The protocol can therefore operate on systems with limited CPU, RAM, and storage capacity.

9.5 Interoperability and Integration Considerations

QSMP is designed to integrate into layered communication stacks.

Transport Independence. The protocol operates over any reliable transport, including TCP, QUIC style reliable streams, or custom application transports. It does not assume properties such as ordering or reliability beyond what the underlying transport provides.

API Simplicity. The reference implementation offers clear entry points for handshake initiation, handshake processing, packet encryption, and packet decryption. Integration into existing systems requires only mapping these functions to send and receive events.

Compatibility with Existing Cryptographic Infrastructure. Since QSMP uses well defined post quantum primitives, deployments can rely on existing implementations of KEMs and signature schemes. This compatibility reduces the need for bespoke cryptographic code.

The protocol can be integrated into systems with mixed classical and post quantum components as long as the handshake logic and channel construction follow the specification.

9.6 Operational Behaviors and Failure Handling

Correct deployment of QSMP requires consistent handling of error conditions and network anomalies.

Handshake Failures. The reference implementation terminates the session upon detection of any malformed or invalid handshake element. Deployment must preserve this behavior and avoid retry logic that could cause inconsistent transcript reuse.

Channel Failures. Encrypted packet failures, such as authentication errors or sequence violations, result in immediate session termination. Reuse of the channel after failure is forbidden.

Timeout Policies. Implementations should impose reasonable timeouts on handshake progression to prevent partial state accumulation due to stalled peers. The specification does not mandate exact timeout values, leaving this to deployment policy.

These considerations ensure that error handling remains consistent with the design goals of the protocol.

9.7 Summary of Deployment Considerations

QSMP exhibits computational efficiency consistent with modern post quantum secure messaging protocols. Its compact memory footprint, predictable packet structure, and efficient RCS based channels make it suitable for both high performance systems and constrained embedded devices.

Deployment requires careful adherence to the handshake rules, error handling behavior, and state management expectations defined in the specification and demonstrated in the reference implementation. When these conditions are met, QSMP provides a secure and performant foundation for quantum resistant communication.

Chapter 10: Security Robustness and Long-Term Resilience

10.1 Long Horizon Threat Considerations

QSMP relies on a small set of well-defined cryptographic primitives whose long-term security can be evaluated independently. The handshake uses a post quantum signature scheme for authentication and a post quantum KEM for key establishment. The data phase uses the RCS authenticated encryption construction built on SHAKE based sponge primitives.

In SIMPLEX mode, long term stability depends on the security of a single encapsulated shared secret and the long-term signature key used by the server. In DUPLEX mode,

stability depends on two independent shared secrets and on the long-term signature keys of both parties. The use of ephemeral KEM key pairs ensures that the security of each session depends on fresh post quantum hardness assumptions rather than on long term static secrets.

RCS inherits its robustness from SHAKE, and its behavior does not rely on any mode specific assumptions. As long as KEM and signature algorithms remain secure against quantum and classical adversaries, the overall protocol maintains long term cryptographic strength.

10.2 Resistance to Protocol Evolution Attacks

Forward secrecy in QSMP depends on the correct generation and erasure of ephemeral KEM private keys and shared secrets.

In SIMPLEX mode, the server generates the only ephemeral KEM key pair. Once the handshake completes and the symmetric session keys have been derived, the server erases its ephemeral private key. The client performs no decapsulation in SIMPLEX mode and therefore does not retain any ephemeral KEM key material. Both sides erase the shared secret immediately after computing the output of the KDF.

In DUPLEX mode, both parties generate ephemeral KEM key pairs. The client decapsulates the second shared secret during the establish stage, and the server decapsulates the first. Both ephemeral KEM private keys must be erased once the handshake transitions into the established state. Erasure of the shared secrets is performed at the same time.

Under these conditions, an attacker who compromises the long-term signing keys after the handshake cannot recover past session keys. Forward secrecy holds unless ephemeral private keys or shared secrets are improperly retained.

10.3 Durability of Cryptographic Primitives

QSMP separates long term authentication keys from ephemeral key establishment material. This separation limits the effect of key compromise.

Compromise of the server’s long-term signing key before the handshake allows impersonation in both SIMPLEX and DUPLEX. After the handshake, compromise of the server’s signing key does not reveal the shared secret or symmetric session keys because they depend only on ephemeral KEM key pairs and the session binding hash.

Compromise of the client's long-term signing key affects only DUPLEX mode, since SIMPLEX does not authenticate the client cryptographically. In DUPLEX mode, post handshake compromise of the client's signature key does not reveal the shared secrets or symmetric keys, provided that ephemeral KEM keys and shared secrets were erased.

Compromise of a session's symmetric key through the Reveal oracle or side channels enables decryption of messages in that session only. Because new symmetric keys are derived for each handshake, compromise does not spread across sessions.

10.4 Persistence of Forward Secrecy and Key Isolation

QSMP's robustness under hostile network conditions is largely derived from:

- authenticated KEM public keys,
- strict state machine enforcement,
- signature validation of handshake components,
- authenticated encryption of all data,
- sequence and timestamp validation,
- and deterministic construction of the session binding value.

These properties together prevent transcript truncation, replay, reordering, and cross stage substitution. An adversary cannot force acceptance of invalid handshake values because all critical elements are authenticated either through signatures or decapsulation correctness. Data phase robustness follows from RCS, which ensures that any attempt to forge or manipulate ciphertext results in immediate rejection.

10.5 Mitigation of Post Compromise Recovery Limitations

QSMP does not grant automatic post compromise recovery. If an endpoint is compromised and its channel keys are exposed during an active session, future confidentiality is lost until a new handshake is performed. This limitation is common to non-ratcheted protocols.

QSMP deployments that require long lived sessions and automatic recovery from compromise can employ explicit ratcheting mechanisms or periodic re keying. These techniques are compatible with the structure of the protocol because the KDF and RCS instances can be replaced through an additional authenticated handshake stage.

10.6 Anticipation of Structural Attacks

Structural attacks typically exploit protocol level design flaws rather than weaknesses in isolated primitives. These include cross protocol interactions, rollback attacks, transcript mismatches, and identity confusion.

Transcript Binding.
QSMP binds all critical handshake material into signatures and the binding hash. This eliminates the possibility of accepting mismatched transcripts or silently diverging configuration fields over time.

Identity Anchoring.
Public keys and key identifiers are authenticated and incorporated into the binding hash, ensuring that identity substitution attacks cannot emerge as the protocol evolves.

Strict State Progression.
The handshake enforces unidirectional state transitions. Future extensions that add optional stages must retain strict control flow to avoid introducing ambiguous or overlapping states.

Together, these structural properties help ensure that QSMP remains robust as the broader ecosystem of post quantum protocols evolves.

10.7 Long Term Operational Considerations

Operational resilience requires careful management of key material, session duration, and cryptographic updates.

Key Lifecycle Management.
Long term private keys must be rotated periodically to reduce the impact of potential future cryptanalytic advances. Session keys, derived exclusively from ephemeral material, require no long-term storage.

Algorithm Agility.
QSMP supports algorithm identifiers that allow controlled migration to new primitives if required. Migration must maintain transcript binding so that downgrade or cross protocol confusion is not possible. Asymmetric parameter sets can be changed at the QSC library level, and asymmetric ciphers can be either Kyber or McEliece, while signature scheme choices are Dilithium or SPHINCS+, to account for performance or long-term security profiles.

Session Duration.
Very long-lived sessions should incorporate periodic re keying. Although RCS does not lose security when used with long streams, operational policy may still require re-establishment of channels after defined intervals.

These operational factors contribute significantly to the continued resilience of QSMP deployments.

10.8 Summary of Long-Term Resilience

QSMP maintains strong long-term resilience due to its reliance on post quantum secure primitives, explicit identity binding, strict transcript validation, and separation of directional keys. Ephemeral key exchanges provide forward secrecy, and authenticated configuration identifiers prevent unintended changes or silent downgrades. Provided that deployments implement regular key management and remain aligned with the specification's structural constraints, QSMP remains robust against long horizon adversaries and the evolving cryptographic landscape.

Chapter 11: Implementation Guidance and Compliance Considerations

11.1 Implementation Objectives

Correct and secure implementation of QSMP requires strict adherence to the state machine described in the specification. The reference C implementation enforces a sequence of expected packet types through a combination of header validation and explicit state transitions. Each stage of the handshake accepts only the packet flag that corresponds to that stage. All other flags cause immediate rejection.

This strict enforcement prevents out of order messages, replayed handshake packets, and cross stage substitution. Since SIMPLEX and DUPLEX use different sets of authenticated fields in the exchange stage, it is important that the implementation verifies the mode before processing the incoming message. Although the underlying packet structures are similar, the authenticated elements differ. SIMPLEX validates only the server signed hash, while DUPLEX validates both client and server signatures over their respective handshake values.

All message processing functions must validate payload sizes, sequence numbers, timestamps, and configuration identifiers before processing the data. Packet headers are

serialized exactly as defined, and this serialization must remain consistent across implementations because the RCS layer authenticates the serialized header as associated data.

The correctness and security of QSMP depend on enforcing these validation steps without exception.

11.2 Correct Handling of Handshake Logic

Ephemeral key material is central to the forward secrecy guarantees of QSMP. Correct implementation requires clear distinction between SIMPLEX and DUPLEX regarding how ephemeral KEM key pairs are generated, stored, and erased.

In SIMPLEX mode, only the server generates an ephemeral KEM key pair. The client does not generate or store any ephemeral KEM private key in this mode. The server must erase this private key after the handshake completes and once the symmetric keys have been derived. The shared secret produced by the KEM encapsulation must also be erased on both sides immediately after the KDF output is computed.

In DUPLEX mode, both parties generate ephemeral KEM key pairs. The client generates its KEM key pair when constructing the exchange request, and the server generates its key pair during the connect response stage. After the handshake transitions into the established state, both parties must erase their private KEM keys and any intermediate shared secret values. The implementation must not retain these ephemeral values in persistent memory or diagnostic logs.

The correct erasure of ephemeral data is essential for maintaining forward secrecy and for preventing recovery of past session keys in the event of long-term key compromise.

11.3 Secure Use of Cryptographic Primitives

QSMP requires cryptographically strong randomness for key generation and signature operations. The implementation must use a secure random number generator to generate the ephemeral KEM keys and to perform any randomness required by the signature scheme. Poor entropy sources weaken both SIMPLEX and DUPLEX and undermine authentication and confidentiality.

The implementation must also avoid exposing timing or memory access patterns that leak information about decapsulation success or shared secret values. Decapsulation

failures must produce identical timing and output behavior to prevent adversaries from learning information about ciphertext structure or key correctness.

While the reference implementation does not include constant time protections for every primitive, the underlying KEM and signature algorithms are designed to operate in constant time with respect to secret dependent operations. Implementers must ensure that these properties remain intact in future revisions or in embedded deployments.

11.4 Channel Initialization and Packet Processing

The reference implementation performs strict input validation at every stage of the handshake and data path. All parsed fields are bounds checked, and the code ensures that buffer lengths match expected values before passing data to cryptographic functions.

Safe implementation requires:

1. Validating header fields before accepting packet contents.
2. Checking payload lengths against message type requirements.
3. Ensuring ciphertext lengths match KEM and RCS requirements.
4. Avoiding out of bounds memory access during parsing.
5. Zeroizing sensitive memory after failed handshake attempts.
6. Ensuring error codes do not disclose secret dependent information.

In SIMPLEX, decapsulation errors must not reveal partial information about the KEM. In DUPLEX, signature verification failures must terminate the handshake before any further processing occurs. The same principle applies to attempts to process malformed exchange requests or connect responses.

Memory safety is essential because compromised memory access may reveal long term verification keys, ephemeral KEM private keys, shared secrets, or symmetric channel keys.

11.5 Failure Handling and Session Management

QSMP implementations must adhere strictly to the ordering, structure, and cryptographic requirements defined in the specification. The following items require special attention:

- The session binding hash must use exactly the fields described in the specification and must be computed identically across all implementations.
- In SIMPLEX mode, the client must not generate a KEM key pair. Any deviation introduces incorrect entropy assumptions and violates the security proof model.
- In DUPLEX mode, the client must generate a KEM key pair and must sign the correct hash that includes its public KEM key and the ciphertext.
- All signature validation steps must occur before processing any value that depends on the authenticated field.
- RCS encryption and decryption must use the serialized header as associated data exactly as defined.
- Timestamps and sequence numbers must be validated consistently to prevent replay and time shifting attacks.

Implementers must also ensure that configuration identifiers, key identity values, and associated public verification keys correspond accurately to the long-term signature keys used by the protocol. Any mismatch may cause silent degradation or unintended negotiation behavior.

11.6 Compliance with Cryptographic and Organizational Standards

QSMP deployments may be subject to organizational, regulatory, or industry standards.

Algorithm Identifier Consistency. Implementations must track algorithm identifiers carefully and ensure that no downgrade or mismatch occurs. All identifiers must match the specification exactly.

Documentation of Security Parameters. Deployments must document the chosen security level, including key sizes, signature parameters, and KEM variants. This documentation ensures reproducibility and auditability.

Interoperability Profiles. Systems using QSMP must document supported modes, configuration fields, and transport assumptions to ensure consistent behavior across different implementations.

Compliance with Security Policies. Deployments must align with organizational policies on key lifetimes, entropy sources, code review, and lifecycle management. The protocol itself does not mandate these policies but requires them for secure operation.

11.7 Summary of Implementation Guidance

QSMP implementations must maintain strict adherence to the specification, enforce complete validation of all handshake elements, protect all cryptographic operations with side channel aware techniques, and apply robust error handling. The security of QSMP depends on correct behavior at every stage of the handshake and channel operation. When these requirements are satisfied, QSMP can be implemented reliably and deployed securely across diverse environments.

Chapter 12: Privacy, Governance, and Ethical Considerations

12.1 Privacy Objectives of the Protocol

QSMP provides confidentiality and integrity for all application data through the RCS authenticated encryption construction. Privacy at the protocol level depends on the handshake structure and on the correct use of ephemeral key material. SIMPLEX and DUPLEX differ in how they construct this material, but both modes ensure that long term identifiers and authentication keys are never exposed through the encrypted channel.

In SIMPLEX mode, only the server generates an ephemeral KEM key pair. The client derives all session keys from a single shared secret encapsulated to the server’s public KEM key. This preserves privacy of client origin because the client does not reveal any cryptographic keying material beyond what is required to validate the server. The server’s signature authenticates the public KEM key, but the client contributes no additional identifying cryptographic structure.

In DUPLEX mode, both sides generate ephemeral KEM key pairs and authenticate their handshake values through signatures. The client’s ephemeral public KEM key is therefore visible to the server and to any adversary that can observe handshake messages. This is expected behavior and does not compromise user privacy, because ephemeral public keys do not identify the user across sessions. They are generated fresh for every handshake and erased after use.

Across both modes, QSMP does not expose unique identifiers beyond the long-term verification keys already distributed out of band. QSMP does not reveal session keys, shared secrets, or internal state outside the encrypted RCS channel.

12.2 Metadata Exposure and Minimization

Governance of QSMP deployments requires careful management of long-term verification keys, configuration identifiers, and algorithm selections. The protocol assumes that these values are assigned and controlled by a trusted authority. Incorrect governance may result in clients selecting expired or incorrect signatures keys or accepting deprecated configurations.

In SIMPLEX mode, governance focuses on the server side because the client does not generate or authenticate its own key material. The server must publish its verification key and maintain correct configuration identifiers. The server is also responsible for ensuring that ephemeral KEM key pairs are generated using high quality randomness and are erased immediately after the handshake.

In DUPLEX mode, governance must also account for the client's long term verification key when client authentication is required. The client is responsible for maintaining its own signing key, for generating ephemeral KEM key pairs securely, and for ensuring proper erasure. The server must know the client's verification key or must receive it through a trusted provisioning mechanism.

Implementers must ensure that configuration identifiers correspond accurately to the algorithms in use. Session binding depends on hashing the configuration string and key identities. Any misalignment in these values breaks compatibility or undermines security guarantees.

12.3 Governance of Cryptographic Parameters

QSMP provides strong cryptographic properties that remain secure against quantum capable adversaries. This strength must be balanced against ethical and regulatory requirements for responsible use of encryption.

The differences between SIMPLEX and DUPLEX modes influence compliance considerations. SIMPLEX provides unilateral server authentication and is suitable for environments where identity assurances apply only to one side. DUPLEX provides mutual authentication and dual sided key contribution. This may impact regulatory

requirements for identity validation and key management, especially in systems that mandate mutual proof of identity.

Privacy regulations may require that ephemeral keys be erased promptly after use. In SIMPLEX, only the server handles an ephemeral private KEM key. In DUPLEX, both parties must erase ephemeral KEM private keys and shared secrets, and implementers should ensure that erasure behavior is consistent with data retention policies.

QSMP's reliance on authenticated encryption and strict header validation helps organizations comply with security requirements that mandate protection of both data payloads and metadata integrity. The protocol's explicit design minimizes ambiguity and simplifies analysis, which reduces the risk of unintentional misuse or misconfiguration.

12.4 Ethical Use of Encryption and Secure Messaging

QSMP provides strong confidentiality that is expected to withstand quantum equipped adversaries. This capability entails ethical responsibilities for developers, operators, and organizations adopting the protocol.

Responsible Use. Deployments must ensure that strong encryption is not misapplied in ways that undermine regulatory obligations, public safety requirements, or organizational policies. QSMP itself does not provide mechanisms for access control beyond cryptographic authentication, so governance frameworks must handle lawful access considerations without weakening the protocol.

Protection of Legitimate Users. Secure messaging protocols play a critical role in protecting users from surveillance, exploitation, or unauthorized interception. QSMP's design supports privacy preserving communication and should be implemented in a way that protects legitimate use while preventing misuse through uncontrolled distribution of private keys or bypass mechanisms.

Transparency and Accountability. Implementations should document their cryptographic behavior, update schedules, and security assumptions clearly. Transparency facilitates auditability and ensures that deployments remain compliant with industry and organizational standards.

12.5 Compliance with Regulatory and Industry Frameworks

Organizations adopting QSMP may be subject to legal or industry driven requirements regarding cryptographic controls, privacy protections, and security operations.

Data Protection Regulations. QSMP's inherent confidentiality aligns with common data protection requirements that mandate secure handling of user information. Deployments must ensure that private keys, session keys, and binding values are managed according to applicable data protection policies.

Audit and Certification. Systems using QSMP may need to demonstrate compliance with formal security certifications or organizational security frameworks. Although QSMP is a protocol rather than a certification standard, its deterministic structure and strict validation rules aid in demonstrating compliance.

Long Term Data Governance. Since QSMP aims to resist future cryptanalytic advancements, organizations must account for long term retention policies and ensure that archived data is protected appropriately. Secure deletion policies for ephemeral keys and derived secrets are essential to limiting long term exposure.

12.6 Ethical Considerations for Future Extensions

Future extensions to QSMP, such as ratcheting, multi-party operation, or integration with identity systems, must preserve privacy and ethical design principles. Extensions must avoid adding metadata that could weaken privacy or reveal sensitive information about user behavior. All added fields must undergo the same rigor of transcript binding and validation.

Moreover, new cryptographic primitives introduced in protocol extensions must follow transparent review, governance, and auditing processes to ensure that they do not introduce structural vulnerabilities or compromise user privacy.

12.7 Summary of Privacy and Ethical Analysis

QSMP provides strong privacy protections by design, pairing authenticated handshake procedures with confidential and integrity protected transport channels. These protections remain robust against evolving computational capabilities when supported by sound governance, responsible deployment policies, and adherence to ethical guidelines. Organizations implementing QSMP must maintain disciplined oversight of cryptographic parameters and operation, ensuring that privacy and security guarantees are preserved throughout the protocol's lifecycle.

Chapter 13: Reproducibility, Limitations, and Future Work

The analysis presented in this document is fully reproducible using only the QSMP specification and the reference C implementation. Every handshake step, including configuration selection, session binding computation, ephemeral key generation, ciphertext construction, signature verification, and key derivation, can be reconstructed using these two sources. The internal structure of both SIMPLEX and DUPLEX is explicit and deterministic, allowing external evaluators to replicate all results.

In SIMPLEX mode, reproducibility requires observing the server generated ephemeral KEM public key, verifying the server signature, and performing a single encapsulation using the client-side implementation of the KEM. Because the client does not generate any ephemeral KEM key pair in SIMPLEX, evaluators need only replicate the server's ephemeral key generation and the decapsulation behavior. The shared secret produced by encapsulation, combined with the session binding value, yields all session keys through the documented SHAKE based KDF.

In DUPLEX mode, reproducibility extends to verifying the client's signed handshake values and confirming the two independent KEM operations. Evaluators can recompute the hash of the client exchange request, verify the client signature, decapsulate the first shared secret, encapsulate the second shared secret using the client's public KEM key, and validate the final signature and establish messages. The two shared secrets and the binding hash produce a deterministic KDF output that matches the implementation exactly.

Because all authenticated transcript elements and KDF inputs are explicitly defined and validated, third party evaluators can reconstruct the protocol flow without accessing private keys or implementation specific behavior. This transparency supports independent cryptographic audits and long-term maintainability.

13.2 Limitations of the Present Analysis

The cryptanalysis presented in this document focuses on the protocol mechanics, cryptographic correctness, state machine enforcement, transcript integrity, and channel security derived from the KDF and RCS layers. Several areas remain outside the scope of this analysis and are noted here as limitations rather than omissions.

First, the analysis does not evaluate side channel resistance of the underlying KEM and signature algorithms. While the specification assumes that these primitives operate in constant time with respect to secret dependent values, the reference C implementation

relies on the underlying cryptographic library to provide these guarantees. A full side channel analysis would require specialized measurement techniques and is not included here.

Second, the analysis does not assess implementation behavior on hardware with constrained randomness sources. Correct operation of QSMP requires high quality randomness for ephemeral KEM key generation and signature operations. The analysis assumes that a secure random number generator is available, but this assumption should be validated for embedded or hardware limited deployments.

Third, the analysis does not include formal machine checked proofs. The reductions described in earlier chapters follow established patterns but are not mechanized in a proof assistant. Formal verification would require lifting the C implementation into a verifiable model or producing an executable symbolic model that captures all handshake transitions.

Fourth, the analysis is limited to the documented behavior of the reference implementation. Alternative implementations must adhere strictly to the specification, and deviations in header processing, signature validation, or binding hash computation may introduce vulnerabilities that the present analysis does not account for.

Finally, although the analysis considers replay, truncation, and transcript manipulation attacks, it does not examine multi session or multi-protocol interactions beyond those explicitly covered in the specification. A full compositional analysis under concurrent execution conditions remains an open area for further evaluation.

13.3 Directions for Further Formalization

QSMP offers a well-structured basis for formal verification due to its deterministic transcript structure, explicit authentication of handshake elements, and limited reliance on ambient negotiation. Several avenues exist for deeper formal development.

One direction is the construction of a symbolic model for use with automated protocol verifiers such as ProVerif or Tamarin. Because SIMPLEX and DUPLEX both have fixed sequences and authenticated public keys, such a model can express the core security properties without relying on implicit contextual assumptions. The use of explicit transcript hashing simplifies correspondence proofs and reduces ambiguity in state transitions.

A second direction is the development of a mechanized proof in a proof assistant such as Coq, Isabelle, or F*. This would require formalizing the KEM, signature, KDF, and RCS components and defining the session state machine precisely. The key derivation function in particular is suitable for machine checked proofs because it is defined entirely through the SHAKE sponge construction and does not rely on secret dependent branching.

A third direction is the creation of an executable test harness that automatically reproduces handshake flows, validates signatures, and verifies that independent implementations produce identical KDF outputs. Such a harness could support interoperability testing and provide a standard reference for verifying correctness of implementations written in different languages or running on different platforms.

A final direction concerns the long-term evolution of QSMP. As post quantum cryptographic standards evolve, future versions of the protocol may incorporate updated KEMs or signature schemes. Formalizing the interface between the protocol and the underlying primitives will support modular updates and maintain analytical continuity.

13.4 Paths for Protocol Evolution

Several enhancements could be developed while preserving compatibility with the existing design.

Optional Ratcheting Enhancements. QSMP already supports limited ratcheting via explicit state transitions. More advanced symmetric or asymmetric ratchet stages could be integrated, provided they remain bound to authenticated transcripts.

Multi Party Extensions. Future versions may extend QSMP to group or multi recipient scenarios. Such designs must preserve the transcript binding guarantees and avoid ambiguity in identity association.

Lightweight Variants. For constrained devices, reduced handshake structures or streamlined configuration fields may be introduced, provided they do not weaken the session binding hash or authentication logic.

These evolutions must be introduced cautiously and governed by the same structural constraints that ensure security in the current version.

13.5 Open Questions and Future Work

Several areas merit continued research and development.

Long Term Cryptanalytic Monitoring. Post quantum primitives remain under active review. Continued monitoring of lattice-based assumptions is necessary to ensure long term resilience.

Advanced Side Channel Protections. Although the reference implementation follows constant time practices, a comprehensive side channel evaluation would increase assurance for hardware deployments.

Interoperability Testing. As additional implementations of QSMP emerge, formal interoperability testing frameworks should be developed to ensure consistent behavior across platforms.

Formal Proof Completion. Completing a full formal proof of security, including machine verified models of the handshake and channels, remains an important future goal.

13.6 Summary

This chapter aligns the analysis with principles of reproducibility and openness. It also identifies the boundaries of the present work and highlights areas for further formalization and development. The QSMP protocol presents a strong design rooted in post quantum secure primitives and rigorous transcript binding, yet continued research and formal verification are essential for maintaining assurance over time.

Chapter 14: Editorial Evaluation and Conclusion

14.1 Editorial Evaluation

This document presents a complete cryptanalytic evaluation of the Quantum Secure Messaging Protocol. The analysis draws directly from the protocol specification and the reference C implementation, and maintains exact alignment between described behavior and observed implementation logic. All handshake stages, authenticated transcript elements, key derivation inputs, message formats, and state transitions have been examined in detail to ensure that no unstated assumptions influence the security assessment.

The earlier drafts of the analysis introduced inaccuracies regarding the role of ephemeral KEM key pairs in SIMPLEX mode. These inaccuracies have been fully

corrected. In SIMPLEX, only the server generates an ephemeral KEM key pair, and the client performs a single encapsulation using the server's public KEM key. The client does not generate any ephemeral private key in SIMPLEX. In DUPLEX, both parties generate ephemeral KEM key pairs and authenticate their contributions through signatures. The final form of this document now reflects these facts precisely.

The editorial goal of this chapter is to confirm that the corrected analysis is consistent, technically exact, and reproducible. All sections have been aligned with the specification and the code. Terminology is used consistently throughout, and the security claims correspond directly to the underlying cryptographic assumptions. The text avoids speculative interpretations and adheres strictly to verifiable protocol behavior.

The style maintains clarity and neutrality, presenting the protocol without exaggeration or omission. All conclusions are supported either by direct analysis of the implementation or by reductions to standard cryptographic assumptions. The document now offers a reliable reference for implementers, auditors, and researchers examining the security properties of QSMP.

14.2 Conclusion

QSMP provides a well-structured, post quantum secure messaging protocol with clear boundaries between handshake authentication and channel encryption. Its design uses only established cryptographic components and enforces strict validation of all handshake inputs. The protocol avoids ambiguity by using authenticated public keys, deterministic transcript hashing, and a KDF that binds session specific and configuration specific values into all final symmetric keys.

In SIMPLEX mode, QSMP delivers server authenticated key establishment using a single shared secret produced by a single encapsulation. The client verifies the server's signature over the ephemeral KEM public key and header, derives keys using the shared secret and binding hash, and transitions to the encrypted data phase. The security of SIMPLEX relies on the signature scheme, the KEM, and correct erasure of ephemeral secrets on the server side.

In DUPLEX mode, QSMP offers mutual authentication and uses two shared secrets derived from two KEM operations. Each party generates and authenticates its ephemeral KEM public key and signs the transcript elements that depend on its respective

contribution. The final symmetric keys are derived from both shared secrets and the binding hash, providing stronger security assurances and higher entropy for the session.

Across both modes, RCS ensures confidentiality and integrity of data packets. The serialized header is authenticated as associated data, preventing manipulation of sequence numbers, timestamps, or packet flags. Replay resistance, truncation protection, and strict state machine enforcement strengthen the protocol against practical attacks on message ordering and packet legitimacy.

The reference implementation adheres closely to the specification. It validates all incoming messages, enforces correct packet sequencing, and erases ephemeral material according to the requirements of forward secrecy. Both handshake correctness and data encryption behavior have been shown to be fully reproducible by independent evaluators using only the public specification and the implementation.

QSMP demonstrates a coherent and secure design grounded in post quantum cryptography. Through explicit authentication, clearly defined handshake stages, deterministic transcript construction, and provable key derivation behavior, the protocol provides a dependable foundation for secure communication in the presence of classical and quantum adversaries. The corrected analysis confirms that its security goals are achieved under the standard assumptions of KEM security, signature unforgeability, and AEAD confidentiality and integrity.

14.3 Completeness of the Cryptanalytic Coverage

The analysis addresses all major components of QSMP. This includes:

1. The handshake structure and transcript consistency.
2. The correctness and security of the key encapsulation and signature primitives.
3. The construction and operation of the RCS based secure channels.
4. The attack surface introduced by message structure, state machines, and implementation choices.
5. Performance and deployment considerations that influence practical security.
6. Governance, ethical, and operational aspects relevant to long term use.
7. Limitations and opportunities for further formalization.

The scope is complete relative to the intent of a protocol level cryptanalysis. Any remaining areas, such as hardware side channel evaluation or machine verified proofs, are explicitly identified as future work rather than implicit assumptions.

14.4 Final Security Assessment

Based on the combined evidence from specification analysis, implementation review, reduction based reasoning, and empirical evaluation, QSMP appears to meet its stated security objectives. The protocol provides confidentiality, integrity, authenticity, and forward secrecy within the assumptions of its post quantum cryptographic primitives. The explicit session binding and strict state progression mitigate common protocol level vulnerabilities such as unknown key share attacks, downgrade attempts, and transcript manipulation.

The secure channels constructed from RCS exhibit predictable AEAD behavior under adversarial testing. No structural weaknesses were identified in the transformation from handshake outputs to symmetric keys and nonces. Sequence number enforcement and associated data binding preserve the end-to-end integrity of post handshake communication.

14.5 Concluding Remarks

QSMP is a robust and carefully constructed protocol designed for a post quantum security environment. Its reliance on modern lattice-based primitives, strict validation of handshake elements, deterministic key derivation, and compact authenticated encryption channels reflect a mature and security focused design philosophy. When deployed with adherence to implementation guidance and governance requirements, QSMP provides a strong foundation for secure communication in systems expected to endure evolving adversarial capabilities.

Although continued analysis and formal verification remain valuable future steps, the protocol as examined here stands on solid ground both cryptographically and operationally. This document provides a complete and reproducible foundation for evaluating the security of QSMP and serves as a reference for implementers, auditors, and researchers working within the QRCS ecosystem.

References

1. **Bernstein, D. J., Buchmann, J., & Dahmen, E. (Eds.).** *Post-Quantum Cryptography*. Springer, 2009.
   Comprehensive reference text introducing the principal hardness assumptions underlying lattice-, code-, and multivariate-based schemes.
2. **Bos, J. W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D.** "CRYSTALS–Kyber: Algorithm Specifications and Supporting Documentation." *NIST PQC Project Submission (Finalist Round 3)*, 2022.
   Defines the IND-CCA security model and implementation parameters of Kyber used in QSMP.
3. **Bernstein, D. J., Lange, T., & Peters, C.** "Classic McEliece: Post-Quantum Code-Based Cryptography." *NIST PQC Submission Document*, 2022.
   Establishes the code-based assumptions and decryption-failure probability limits relevant to QSMP's optional KEM.
4. **Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D.** "CRYSTALS–Dilithium: Algorithm Specifications and Supporting Documentation." *NIST PQC Project*, 2022.
   Primary reference for the EUF-CMA signature primitive used in the DUPLEX authentication mode.
5. **Hülsing, A., Kölbl, S., Rijneveld, J., Song, F., & Zhang, D.** "SPHINCS+: Submission to the NIST Post-Quantum Cryptography Standardization Project." 2022.
   Defines the hash-based signature system used as an alternative to lattice signatures in QSMP.
6. **Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G.** "Keccak." *NIST SHA-3 Standardization Documentation*.
   Primary source for the permutation-based sponge construction underlying SHA-3, SHAKE, and RCS.
7. **National Institute of Standards and Technology (NIST).** *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. NIST, 2015.
   Authoritative specification of SHA3-256, SHA3-512, SHAKE-256, and SHAKE-512.
8. **Bellare, M., & Rogaway, P.** "Entity Authentication and Key Distribution." *CRYPTO '93 Proceedings*, pp. 232-249.
   Introduces the formal AKE framework used for QSMP's security definitions.
9. **Canetti, R., & Krawczyk, H.** "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels." *EUROCRYPT 2001 Proceedings*.
   Defines compositional security proofs connecting AKE and AEAD properties.

10. **Krawczyk, H.** "The Composition of Authenticated Encryption and Key Exchange." *CRYPTO 2001 Proceedings*.
    Provides the reduction methodology for KEM-plus-AEAD constructions applied in QSMP.
11. **Bellare, M., & Namprempre, C.** "Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm." *Journal of Cryptology*, 21(4), 469-491 (2008).
    Framework used to validate the confidentiality and integrity properties of the RCS AEAD cipher.
12. **Paterson, K. G., Ristenpart, T., & Shrimpton, T.** "Tag-Size Does Matter: Attacks and Proofs for the TLS Record Protocol." *CRYPTO 2011 Proceedings*.
    Referenced in analysis of tag length and forgery bounds under the RCS AEAD model.
13. **Bellare, M., Desai, A., Jokipii, E., & Rogaway, P. "**A Concrete Security Treatment of Symmetric Encryption." *FOCS 1997 Proceedings*.
    Supporting framework for probabilistic AEAD security proofs.
14. **National Institute of Standards and Technology (NIST).** *FIPS 140-3: Security Requirements for Cryptographic Modules*. 2019.
    Defines implementation assurance criteria relevant to QSMP compliance analysis.
15. **ISO/IEC 19790:2012.** *Information Technology – Security Requirements for Cryptographic Modules*.
    International standard referenced in implementation and certification guidance.
16. **ISO/IEC 18031:2011.** *Random Bit Generation for Security Purposes*.
    Establishes the randomness and entropy sourcing requirements cited in Chapter 11.
17. **ISO/IEC 17825:2016.** *Testing Methods for the Mitigation of Non-Invasive Attack Classes Against Cryptographic Modules*.
    Referenced for side-channel and implementation assurance recommendations.
18. **Daemen, J., & Van Assche, G.** *The Design of Keccak*. Springer, 2017.
    Formal description of the sponge and duplex constructions foundational to RCS.
19. **Bernstein, D. J., & Lange, T.** "Post-Quantum Cryptography." *Nature*, 549, 188–194 (2017).
    Overview of the post-quantum security landscape informing the long-term analysis in Chapter 10.
20. **Lyubashevsky, V., Micciancio, D., Peikert, C., & Regev, O.** "On Ideal Lattices and Learning With Errors over Rings." *EUROCRYPT 2010 Proceedings*.

Primary theoretical underpinning for the hardness assumption of Kyber's MLWE foundation.

21. **Bernstein, D. J., Chou, T., Schwabe, P., & Stoffelen, K.** "Implementation Aspects of Post-Quantum Cryptography." *NIST PQC Workshop Reports*, 2019.
    Cited in the performance and side-channel sections regarding constant-time implementations.
22. **Rogaway, P.** "The Moral Character of Cryptographic Work." *Proceedings of Asiacrypt 2015, Invited Lecture*.
    Referenced in ethical and governance discussions (Chapter 12) concerning responsible cryptographic deployment.
23. **European Union.** *General Data Protection Regulation (GDPR)*. Official Journal of the European Union, 2016.
    Used as baseline privacy framework in Chapter 12's policy analysis.
24. **Government of Canada.** *Personal Information Protection and Electronic Documents Act (PIPEDA)*. 2000.
    Referenced for privacy compliance considerations in Chapter 12.
25. **ISO/IEC 27001:2022.** *Information Security, Cybersecurity and Privacy Protection – Information Security Management Systems*.
    Framework informing institutional governance recommendations for QSMP deployment.