Toward Quantum-Resilient eSIM Provisioning: Integrating ML-KEM into Consumer Remote SIM Provisioning*

Jhury Kevin Lastre, Hoseok Kwon, Bonam Kim, and Ilsun You†

Kookmin University, Seoul, South Korea
{lavelliane, hoseok1997, kimbona, isyou}@kookmin.ac.kr

Abstract

Consumer Remote SIM Provisioning (RSP), standardized through GSM Association (GSMA) specifications SGP.22 and SGP.24, establishes secure embedded Subscriber Identity Module (eSIM) profile management using Public Key Infrastructure (PKI) based trust models between Subscription Manager Data Preparation (SM-DP+), embedded Universal Integrated Circuit Card (eUICC), and Local Profile Assistant (LPA) components. As quantum computing threatens classical cryptographic primitives such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Diffie-Hellman (ECDH) through Shor's algorithm, telecommunications infrastructure requires transition strategies that preserve interoperability with existing GSMA-trusted Certificate Authorities while achieving quantum resilience. This work presents a Post-Quantum Cryptography (PQC) integration strategy that introduces quantum-safe mechanisms at the algorithmic layer using liboqs to enable hybrid ECDH plus Module Lattice Key Encapsulation Mechanism (ML-KEM) key exchanges. The proposed migration maintains the GSMA certificate hierarchy and SGP.22 protocol structure, allowing PQC-capable eUICC implementations to process quantum-resistant key material with backward compatibility for legacy systems. Experimental evaluation on a virtual testbed demonstrates that hybrid key agreement introduces 2,272 bytes of additional key material and 0.232 milliseconds of computational overhead per provisioning session, resulting in 7.5x bandwidth increase for key exchange messages (from 523 to 3,904 bytes total) while achieving NIST security level 3 quantum resistance.

Keywords: Remote SIM Provisioning, eSIM, Post-Quantum Cryptography, ML-KEM, Hybrid Key Exchange, Quantum Resistance

1 Introduction

Remote SIM Provisioning (RSP), formalized through the GSM Association (GSMA) specifications SGP.22 and SGP.24 [5, 6], establishes the standardized framework for mobile network operators to securely provision and manage embedded Subscriber Identity Module (eSIM) profiles over-the-air. The architecture guarantees end-to-end confidentiality and cryptographic integrity across communications between the Subscription Manager Data Preparation (SM-DP+) server, the embedded Universal Integrated Circuit Card (eUICC), and the Local Profile Assistant (LPA) component through a Public Key Infrastructure (PKI) based trust model. With over 1.2 billion eSIM-capable devices deployed globally as of 2024, RSP has become a critical component of telecommunications infrastructure, enabling seamless operator switching, international roaming, and machine-to-machine connectivity.

The security of RSP relies fundamentally on classical cryptographic primitives, specifically Elliptic Curve Digital Signature Algorithm (ECDSA) with NIST P-256 for authentication and

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. W1, December 16-18, 2025, Sapporo , Japan. © The copyright of this paper remains with the author(s).

†Corresponding author

Elliptic Curve Diffie-Hellman (ECDH) for session key establishment. However, these primitives are vulnerable to quantum computational attacks. Shor's algorithm [10], executable on sufficiently large quantum computers, can solve both the discrete logarithm problem and integer factorization in polynomial time, thereby breaking RSA, ECDSA, and ECDH. Current projections suggest that cryptographically relevant quantum computers capable of breaking 2048-bit RSA could emerge within 10 to 15 years [4]. This timeline presents an urgent threat when considering the long-term sensitivity of telecommunications credentials and the "store-now-decrypt-later" attack model [3], where adversaries harvest encrypted eSIM profile data today for future decryption once quantum computers become available.

Given the critical nature of subscriber credentials, operator authentication keys, and profile encryption keys transmitted during RSP provisioning, the transition to Post-Quantum Cryptography (PQC) is imperative [9]. However, this migration presents significant challenges. The GSMA trust infrastructure involves a complex hierarchy of Certificate Authorities (CAs), with EUM (eUICC Manufacturer) and SM (Subscription Manager) certificates issued by GSMA-trusted root CAs. Any migration strategy must preserve this established PKI ecosystem to maintain interoperability with the existing deployed base of millions of eUICC devices and hundreds of SM-DP+ servers worldwide. Furthermore, the constrained nature of eUICC hardware, with limited memory (typically 256KB to 1MB) and processing power, imposes strict requirements on the size and computational cost of cryptographic operations.

1.1 Research Questions and Contributions

This work addresses three fundamental research questions:

1. **How can PQC be integrated into SGP.22 without disrupting the existing GSMA PKI hierarchy?** We demonstrate that quantum-resistant key exchange can be achieved through algorithmic-layer modifications while preserving certificate structures and trust chains.
2. **What is the performance overhead of hybrid key agreement in resource-constrained eSIM environments?** Through experimental evaluation on a virtual testbed, we quantify the bandwidth, memory, and computational costs of ML-KEM-768 integration.
3. **What deployment challenges arise in migrating real-world RSP implementations to support PQC?** We document practical issues encountered during implementation, including buffer management, TLV encoding complexities, and backward compatibility mechanisms.

Our primary contributions are:

- A hybrid key exchange protocol that combines classical ECDH with ML-KEM-768 for quantum-resistant session key derivation in SGP.22, maintaining backward compatibility with legacy eUICCs through capability negotiation.
- ASN.1 protocol extensions using custom TLV tags (0x5F4A, 0x5F4C) for PrepareDownload and InitialiseSecureChannel messages to transport ML-KEM public keys and ciphertexts without modifying the core SGP.22 message structure.
- Open-source implementation and experimental evaluation demonstrating 7.5x bandwidth overhead (3.4 KB additional key exchange data) with negligible computational cost (0.232 ms) per provisioning session.

- Analysis of real-world deployment challenges including buffer overflow vulnerabilities, long-form TLV encoding requirements, and secure key management in constrained environments.

The remainder of this paper is organized as follows. Section 2 provides background on GSMA RSP architecture and PQC fundamentals. Section 3 describes our migration architecture and design principles. Section 4 presents formal verification of the hybrid key derivation protocol using ProVerif with security proofs for mutual authentication and session key secrecy. Section 5 describes our experimental testbed. Section 6 presents evaluation results with performance analysis. Section 7 discusses deployment challenges and standardization considerations. Section 8 surveys related work, and Section 9 concludes with future research directions.

2 Background

2.1 GSMA Remote SIM Provisioning Architecture

The GSMA RSP architecture, standardized in SGP.22 for consumer devices, defines secure procedures for over-the-air eSIM profile installation. The architecture comprises three primary entities: the SM-DP+ server operated by the mobile network operator or service provider, the eUICC embedded in the end-user device, and the LPA software component running on the device's operating system that facilitates communication between the eUICC and SM-DP+ [5].

The profile download and installation flow proceeds through four main phases as shown in Table 1. During Phase 1 (Authentication), the eUICC and SM-DP+ perform mutual authentication using their respective ECDSA certificates issued by GSMA-trusted CAs. The eUICC generates a challenge and the SM-DP+ responds with its certificate chain. The eUICC verifies this chain against its trusted root CA public keys, then signs its own challenge response using its ECDSA private key. Phase 2 (PrepareDownload) establishes session key material. The eUICC generates an ephemeral ECDH key pair (otPK.EUICC.ECKA, otSK.EUICC.ECKA) and transmits the public key to the SM-DP+ along with metadata about its capabilities and available memory. Phase 3 (Bound Profile Package Download) delivers the encrypted profile. The SM-DP+ generates its own ephemeral ECDH key pair, performs key agreement to derive shared secrets, and uses these to derive session encryption keys (KEK) and message authentication keys (KM). The Bound Profile Package (BPP) contains the encrypted profile data, SIM application code, and personalization data. Phase 4 (Installation) involves the eUICC decrypting the BPP, verifying MAC tags, installing profile elements, and returning a signed ProfileInstallationResult.

The security of this protocol relies on the confidentiality and authenticity provided by ECDSA P-256 signatures and ECDH P-256 key agreement. The session keys (KEK and KM) are derived using a Key Derivation Function (KDF) based on SHA-256, taking the ECDH shared secret as input along with domain-specific labels. The KEK is used for AES-128-CBC encryption of profile data, while KM is used for CMAC-based message authentication to prevent tampering.

2.2 Post-Quantum Cryptography and ML-KEM

The National Institute of Standards and Technology (NIST) initiated a PQC standardization project in 2016 to identify quantum-resistant cryptographic algorithms [10]. After multiple rounds of evaluation, NIST selected CRYSTALS-Kyber (now standardized as ML-KEM, Module Lattice Key Encapsulation Mechanism) for key establishment in 2024 [2]. ML-KEM is based

Table 1: SGP.22 Profile Download Protocol Phases

Phase	Protocol Messages	Cryptographic Operations
1. Authentication	ES9+.InitiateAuthentication	ECDSA signature verification (eUICC)
	ES10b.AuthenticateServer	Certificate chain validation (eUICC)
	ES9+.AuthenticateClient	ECDSA signature generation (eUICC)
2. PrepareDownload	ES10b.PrepareDownload	ECDH keypair generation (eUICC)
	ES9+.GetBoundProfilePackage	Ephemeral key transmission
3. BPP Download	ES8+.InitialiseSecureChannel	ECDH key agreement (both parties)
	ES8+.ConfigureISDP	Session key derivation (KEK, KM)
	ES8+.StoreMetadata	Profile decryption (AES-128-CBC, eUICC)
	ES8+.LoadProfileElements	MAC verification (CMAC, eUICC)
4. Installation	ES10b.ProfileInstallationResult	Installation verification (eUICC)
	ES9+.HandleNotification	Result signature (ECDSA, eUICC)

on the hardness of the Module Learning With Errors (MLWE) problem over polynomial rings, which remains computationally intractable even for quantum adversaries.

ML-KEM operates as a Key Encapsulation Mechanism (KEM) rather than a traditional key agreement protocol like Diffie-Hellman. The security model differs fundamentally: in a KEM, one party (the encapsulator) generates a shared secret and encrypts it under the other party's public key, producing a ciphertext. The recipient (the decapsulator) uses their private key to decrypt this ciphertext and recover the shared secret. This asymmetric approach contrasts with ECDH, where both parties contribute equally to derive a shared secret through scalar multiplication on an elliptic curve.

ML-KEM-768 specifically provides NIST security level 3, which offers 192 bits of quantum security, comparable to the classical security of AES-192. The key sizes are significantly larger than classical equivalents: the public key is 1,184 bytes, the secret key is 2,400 bytes, the ciphertext is 1,088 bytes, and the shared secret is 32 bytes. These larger sizes pose challenges for bandwidth-constrained protocols and memory-limited devices. However, the computational performance of ML-KEM is competitive with classical cryptography. Keypair generation and encapsulation operations can be performed in under 1 millisecond on modern processors, and decapsulation is even faster.

2.3 Threat Model and Quantum Adversary

Our threat model considers an adversary with access to a cryptographically relevant quantum computer capable of executing Shor's algorithm to break ECDH and ECDSA [12]. We assume the adversary can intercept and store all RSP protocol messages transmitted between the eUICC and SM-DP+ over potentially untrusted networks (e.g., public Wi-Fi, cellular data). The adversary's goal is to decrypt the BPP and extract sensitive data including the subscriber's International Mobile Subscriber Identity (IMSI), Ki authentication key, operator credentials, and personalization data.

Under the ”store-now-decrypt-later” attack scenario [3], the adversary captures encrypted profile data transmitted today and stores it for future cryptanalysis. Once a sufficiently powerful quantum computer becomes available, the adversary can:

1. Break the ECDSA signatures on authentication messages to forge certificates or imper-

sonate entities.

2. Solve the ECDH discrete logarithm problem to recover the ephemeral private key.
3. Compute the shared secret and derive the session keys (KEK, KM).
4. Decrypt the BPP and extract all sensitive profile data.

This threat is particularly acute for eSIM profiles because subscriber credentials have long-term value. A compromised Ki key enables the adversary to authenticate as the legitimate subscriber indefinitely, intercept communications, and commit fraud. Furthermore, the deployment lifetime of eUICC devices (often 10+ years for IoT applications) means that devices provisioned today using only classical cryptography will remain vulnerable throughout their operational life.

Our mitigation strategy focuses on protecting the session key establishment phase (Phase 2 and 3) by replacing pure ECDH with a hybrid ECDH plus ML-KEM key agreement. This ensures that even if a quantum adversary breaks the ECDH component in the future, the ML-KEM component remains secure, preserving the confidentiality of the session keys and consequently the BPP. We defer the migration of signature algorithms (ECDSA to ML-DSA) to future work, as signatures provide authentication but not long-term confidentiality. However, we acknowledge that quantum adversaries capable of breaking ECDSA could mount active man-in-the-middle attacks by forging certificates or impersonating entities during provisioning sessions. While the hybrid key exchange protects session confidentiality against passive "store-now-decrypt-later" attacks (the primary threat model), complete quantum resistance requires migrating both key agreement and signature algorithms. The phased approach prioritizes confidentiality protection first, with authentication migration planned for subsequent deployment phases.

3 Migration Architecture

3.1 Design Principles

Our PQC migration strategy is guided by four key principles that prioritize practical deployment and ecosystem compatibility:

Principle 1: Preserve GSMA PKI Infrastructure. The existing GSMA certificate hierarchy must remain intact. We do not replace or modify EUM and SM certificates, nor do we require changes to GSMA root CA operations. PQC is integrated solely at the session key establishment layer, orthogonal to the PKI.

Principle 2: Ensure Backward Compatibility. Legacy eUICCs that do not support PQC must continue to function without modification. The protocol must gracefully fall back to classical ECDH when either the eUICC or SM-DP+ does not advertise PQC capabilities. This enables incremental deployment across the ecosystem.

Principle 3: Adopt Hybrid Cryptography During Transition. Rather than immediately replacing ECDH with pure ML-KEM, we employ a hybrid approach that combines both mechanisms. This provides defense-in-depth: when shared secrets are properly combined using a cryptographic key derivation function (as in our HKDF-based construction), the security level is at least as strong as the stronger of the two schemes, so the protocol remains secure even if one algorithm is unexpectedly broken [**?**].

Principle 4: Minimize Protocol Disruption. PQC integration should require minimal changes to SGP.22 message structures. We avoid redesigning the protocol or introducing new message types. Instead, we extend existing messages (PrepareDownloadResponse and InitialiseSecureChannelRequest) with optional TLV fields to carry ML-KEM key material.

3.2 Hybrid Key Agreement Protocol

Figure 1 illustrates our migration architecture. The protocol flow maintains the four-phase structure of SGP.22 but augments Phase 2 and Phase 3 with hybrid key agreement operations.

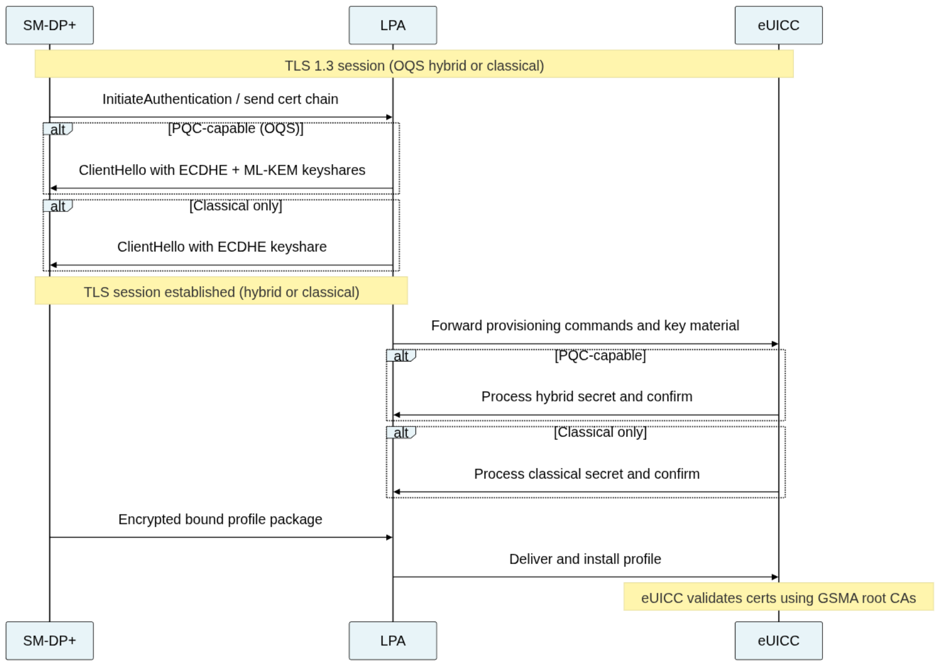
SM-DP+
LPA
eUICC
TLS 1.3 session (OQS hybrid or classical)
InitiateAuthentication / send cert chain
alt
[PQC-capable (OQS)]
ClientHello with ECDHE + ML-KEM keyshares
alt
[Classical only]
ClientHello with ECDHE keyshare
TLS session established (hybrid or classical)
Forward provisioning commands and key material
alt
[PQC-capable]
Process hybrid secret and confirm
alt
[Classical only]
Process classical secret and confirm
Encrypted bound profile package
Deliver and install profile
eUICC validates certs using GSMA root CAs
SM-DP+
LPA
eUICC

Figure 1: Migration architecture showing hybrid key agreement flow between SM-DP+, LPA, and eUICC. The eUICC generates both ECDH and ML-KEM keypairs during PrepareDownload. The SM-DP+ performs hybrid key agreement combining ECDH shared secret with ML-KEM encapsulation to derive quantum-resistant session keys.

During PrepareDownload, the eUICC advertises its PQC capabilities through the EUICCInfo2 structure. If the eUICC supports ML-KEM-768, it generates both the classical ECDH keypair (otPK.EUICC.ECKA, otSK.EUICC.ECKA) and an ML-KEM-768 keypair (pk_kem.EUICC, sk_kem.EUICC). The PrepareDownloadResponse message includes both the ECDH public key (as per standard SGP.22) and the ML-KEM public key as an optional extension. The SM-DP+ receives this response and checks for the presence of the ML-KEM public key. If found, the SM-DP+ enters hybrid mode.

During BPP generation, the SM-DP+ generates its own ECDH keypair (otPK.DP.ECKA, otSK.DP.ECKA) and performs classical ECDH key agreement to obtain shared secret Z_ecdh. Additionally, the SM-DP+ invokes ML-KEM-768 encapsulation using the eUICC's ML-KEM public key to generate shared secret Z_mlkem and ciphertext ct_kem. The InitialiseSecureChannelRequest message (the first command in the BPP) includes both the SM-DP+ ECDH public key and the ML-KEM ciphertext as extensions.

When the eUICC receives the InitialiseSecureChannelRequest, it performs classical ECDH with the SM-DP+ public key to derive Z_ecdh, then invokes ML-KEM-768 decapsulation on the ciphertext using its ML-KEM secret key to recover Z_mlkem. With both shared secrets available, the eUICC applies a hybrid Key Derivation Function (KDF) to derive the final session keys (KEK and KM). The SM-DP+ performs the same hybrid KDF using its copies of Z_ecdh and Z_mlkem, resulting in both parties deriving identical session keys.

If either party does not support ML-KEM or does not advertise the capability, the protocol falls back to classical-only mode. The eUICC omits the ML-KEM public key from PrepareDownloadResponse, the SM-DP+ omits the ciphertext from InitialiseSecureChannelRequest, and both parties use the standard SGP.22 KDF based solely on Z_ecdh. This ensures that a PQC-capable SM-DP+ can still provision legacy eUICCs, and a legacy SM-DP+ can still provision PQC-capable eUICCs.

3.3 Protocol Extensions

To transport ML-KEM key material within SGP.22 messages, we define two custom ASN.1 TLV tags as shown in Table 2. These tags are allocated from the context-specific class to avoid conflicts with existing SGP.22 tags.

Table 2: ASN.1 TLV Extensions for PQC Support

Tag	**Hex**	**Description**
euiccPkKem	0x5F4A	ML-KEM-768 public key (1,184 bytes) transmitted in PrepareDownloadResponse within euiccSigned2 SEQUENCE. Optional field, present only if eUICC supports PQC.
smdpCtKem	0x5F4C	ML-KEM-768 ciphertext (1,088 bytes) transmitted in InitialiseSecureChannelRequest. Optional field, present only if SM-DP+ detects eUICC PQC capability and enters hybrid mode.

The PrepareDownloadResponse structure is extended as follows in ASN.1 notation:

```
PrepareDownloadResponse ::= [33] SEQUENCE {
  downloadResponseOk [0] SEQUENCE {
    euiccSigned2 [APPLICATION 30] SEQUENCE {
      transactionId [0] OCTET STRING,
      euiccOtpk [APPLICATION 73] OCTET STRING,
      smdpOid [APPLICATION 75] OBJECT IDENTIFIER,
      euiccPkKem [APPLICATION 74] OCTET STRING OPTIONAL
    },
    euiccSignature2 [APPLICATION 55] OCTET STRING
  }
}
```

The InitialiseSecureChannelRequest structure is extended similarly:

```
InitialiseSecureChannelRequest ::= [35] SEQUENCE {
  remoteOpId [0] INTEGER,
  transactionId [1] OCTET STRING,
  controlRefTemplate [6] SEQUENCE {
    keyType [0] OCTET STRING,
    keyLen [1] INTEGER
  },
  smdpOtpk [APPLICATION 73] OCTET STRING,
  smdpCtKem [APPLICATION 76] OCTET STRING OPTIONAL,
  smdpSign [APPLICATION 55] OCTET STRING
}
```

The use of long-form length encoding is required for the ML-KEM fields. In BER/DER encoding, if the content length exceeds 127 bytes, the length field uses multiple bytes: the first byte has the high bit set and indicates the number of subsequent length bytes, followed by the actual length in big-endian format. For a 1,184-byte ML-KEM public key, the encoding is: tag (0x5F 0x4A), length (0x82 0x04 0xA0, indicating 1,184 in two bytes), followed by 1,184 bytes of key data. Implementations must correctly parse this long-form encoding and allocate sufficient buffer space.

4 Formal Verification

4.1 Hybrid Key Derivation Equation

The hybrid key derivation function combines classical ECDH and post-quantum ML-KEM shared secrets to derive quantum-resistant session keys. Following NIST SP 800-56C Rev. 2 guidance for hybrid schemes, we employ a two-stage HKDF construction with domain separation as shown in Equation 1.

$$\begin{aligned}\text{Combined}_{\text{IKM}} &= Z_{\text{ECDH}} \| Z_{\text{MLKEM}} \| \text{``ECDH-P256''} \| \text{``ML-KEM-768''} \\ \text{Combined}_{\text{PRK}} &= \text{HKDF-Extract}(\text{salt} = 0^{32}, \text{Combined}_{\text{IKM}}) \\ \text{SessionKey} &= \text{HKDF-Expand}(\text{Combined}_{\text{PRK}}, \text{info} = \text{eChal}\|\text{sChal}\|\text{ID}_{\text{EUICC}}\|\text{ID}_{\text{SMDP}}, L)\end{aligned} \tag{1}$$

Where:

- Z_{ECDH} is the 32-byte shared secret from ECDH key agreement: $Z_{\text{ECDH}} = \text{ECDH}(\text{SK}_{\text{EUICC}}, \text{PK}_{\text{SMDP}})$
- Z_{MLKEM} is the 32-byte shared secret from ML-KEM encapsulation/decapsulation
- eChal and sChal are freshness nonces from eUICC and SM-DP+ respectively
- ID_{EUICC} and ID_{SMDP} bind the session key to specific entities
- L is the desired output length (e.g., 48 bytes for both KEK and KM combined)
- KEK (Key Encryption Key) and KM (MAC Key) are extracted from SessionKey

This construction follows best practices for hybrid key combiners: concatenating both shared secrets as input key material (IKM) to HKDF-Extract ensures that an attacker must break both the ECDH component (vulnerable to quantum adversaries via Shor's algorithm) and the

ML-KEM component (quantum-resistant under Module-LWE assumption) to compromise the derived session keys. The domain labels prevent cross-protocol attacks, and the inclusion of challenges and identifiers in the HKDF-Expand info parameter provides cryptographic session binding and freshness.

4.2 ProVerif Protocol Model

We formalized the hybrid key exchange protocol in ProVerif, an automated cryptographic protocol verifier based on the applied pi-calculus. The model encodes the complete SGP.22 authentication and key establishment flow with ML-KEM-768 integration. Algorithm 1 shows the core hybrid key agreement logic.

Algorithm 1 Hybrid Key Agreement in ProVerif

```
1: eUICC Process:
2: Generate ephemeral ECDH keypair: $(SK^{\text{ECDH}}_{\text{EUICC}}, PK^{\text{ECDH}}_{\text{EUICC}})$
3: Generate ephemeral ML-KEM keypair: $(SK^{\text{MLKEM}}_{\text{EUICC}}, PK^{\text{MLKEM}}_{\text{EUICC}})$
4: Send $(PK^{\text{ECDH}}_{\text{EUICC}}, PK^{\text{MLKEM}}_{\text{EUICC}}, \text{eChal})$ to SM-DP+
5:
6: SM-DP+ Process:
7: Generate ephemeral ECDH keypair: $(SK^{\text{ECDH}}_{\text{SMDP}}, PK^{\text{ECDH}}_{\text{SMDP}})$
8: Compute ECDH shared secret: $Z_{\text{ECDH}} \leftarrow \text{ecdh\_shared}(SK^{\text{ECDH}}_{\text{SMDP}}, PK^{\text{ECDH}}_{\text{EUICC}})$
9: Perform ML-KEM encapsulation: $(CT_{\text{MLKEM}}, Z_{\text{MLKEM}}) \leftarrow \text{mlkem\_encaps}(PK^{\text{MLKEM}}_{\text{EUICC}})$
10: Derive session key: $SessionKey \leftarrow \text{hybrid\_kdf}(Z_{\text{ECDH}}, Z_{\text{MLKEM}}, \text{eChal}, \text{sChal}, ID_{\text{EUICC}}, ID_{\text{SMDP}})$
11: Send $(PK^{\text{ECDH}}_{\text{SMDP}}, CT_{\text{MLKEM}}, \text{sChal})$ to eUICC
12:
13: eUICC Key Derivation:
14: Compute ECDH shared secret: $Z_{\text{ECDH}} \leftarrow \text{ecdh\_shared}(SK^{\text{ECDH}}_{\text{EUICC}}, PK^{\text{ECDH}}_{\text{SMDP}})$
15: Perform ML-KEM decapsulation: $Z_{\text{MLKEM}} \leftarrow \text{mlkem\_decaps}(CT_{\text{MLKEM}}, SK^{\text{MLKEM}}_{\text{EUICC}})$
16: Derive session key: $SessionKey \leftarrow \text{hybrid\_kdf}(Z_{\text{ECDH}}, Z_{\text{MLKEM}}, \text{eChal}, \text{sChal}, ID_{\text{EUICC}}, ID_{\text{SMDP}})$
17: Extract KEK and KM from SessionKey
```

The ProVerif model defines cryptographic primitives as functions and equations:

```
(* ECDH operations *)
fun ecdh_pk(ECDHsk_t) : ECDHpk_t.
fun ecdh_shared(ECDHsk_t, ECDHpk_t) : ECDHss_t.
equation forall ska, skb: ECDHsk_t;
  ecdh_shared(ska, ecdh_pk(skb)) =
  ecdh_shared(skb, ecdh_pk(ska)).

(* ML-KEM operations *)
fun mlkem_pk(MLKEMsk_t) : MLKEMpk_t.
fun mlkem_encaps(MLKEMpk_t) : bitstring.
fun mlkem_get_ct(bitstring) : MLKEMct_t.
fun mlkem_get_ss(bitstring) : MLKEMss_t.
reduc forall sk: MLKEMsk_t;
  mlkem_decaps(mlkem_get_ct(mlkem_encaps(mlkem_pk(sk))), sk)
  = mlkem_get_ss(mlkem_encaps(mlkem_pk(sk))).

(* Hybrid KDF - models HKDF-Extract then HKDF-Expand *)
fun hybrid_kdf(ECDHss_t, MLKEMss_t, Nonce_t,
               Nonce_t, Id_t, Id_t) : SessionKey_t.
```

The model captures key properties: ECDH symmetry (both parties compute the same shared secret), ML-KEM correctness (decapsulation recovers the encapsulated secret), and hybrid KDF determinism (same inputs produce same output).

4.3 Verification Results

We verified the hybrid protocol using ProVerif 2.04 with the following security properties encoded as queries. Table 3 summarizes the verification results with the exact query outputs from ProVerif.

Table 3: ProVerif Verification Results for Hybrid Key Exchange

Security Property	**ProVerif Query**	**Result**
Server Authentication	`inj-event(U_AUTH_OK(U,tid_1)) ==> inj-event(S_AUTH_BEGIN(U,tid_1))`	**true**
Mutual Authentication	`inj-event(S_AUTH_OK(U2,tid2_1)) ==> inj-event(U_AUTH_OK(U2,tid2_1))`	**true**
KEK Secrecy	`not attacker(get_KEK(test_session_key[]))`	**true**
KM Secrecy	`not attacker(get_KM(test_session_key[]))`	**true**
Key Agreement	`event(S_KEY_DERIVED(dev_3,tid_1,k)) ==> event(U_KEY_DERIVED(dev_3,tid_1,k))`	**true**
Injective Key Agreement	`inj-event(S_KEY_DERIVED(dev_3,tid_1,k)) ==> inj-event(U_KEY_DERIVED(dev_3,tid_1,k))`	**true**

Server Authentication: The first query verifies that whenever the eUICC accepts server authentication (`U_AUTH_OK`), the SM-DP+ must have initiated the authentication session (`S_AUTH_BEGIN`) with the same transaction ID. The injective correspondence ensures that each authentication attempt is unique, preventing replay attacks where an adversary reuses old authentication messages.

Mutual Authentication: This query confirms that whenever the SM-DP+ accepts client authentication (`S_AUTH_OK`), the eUICC must have previously completed server authentication (`U_AUTH_OK`) for the same transaction. This guarantees bidirectional authentication: both parties verify each other's identity before proceeding to key establishment. The injective property prevents an attacker from causing multiple SM-DP+ sessions to accept the same eUICC authentication.

KEK and KM Secrecy: The secrecy queries verify that an attacker on the public channel cannot learn the derived Key Encryption Key (KEK) or MAC Key (KM). ProVerif returns **true** for both `not attacker(get_KEK(test_session_key[]))` and `not attacker(get_KM(test_session_key[]))`, confirming that these session keys remain secret even when the attacker controls the network and can intercept all messages. This holds because the keys are derived from shared secrets protected by both ECDH (hard under Computational Diffie-Hellman assumption) and ML-KEM (hard under Module Learning With Errors assumption).

Key Agreement: The correspondence query ensures that if the SM-DP+ derives a session key k for transaction *tid* and device dev_3, then the eUICC must have derived the same key for that transaction. This guarantees that both parties compute identical session keys from their respective views of the protocol. ProVerif verifies this by symbolically tracking the `hybrid_kdf` function outputs and confirming that both parties invoke it with identical parameters: the same ECDH shared secret, ML-KEM shared secret, challenges, and identifiers.

Injective Key Agreement: The injective version strengthens key agreement by ensuring that each session key derivation event is unique and occurs in a one-to-one correspondence between eUICC and SM-DP+ sessions. This prevents an attacker from replaying messages to cause multiple protocol sessions to derive the same session key or to cause one party's key derivation to match multiple derivations by the other party. The inclusion of fresh challenges (eChal, sChal) and unique transaction IDs in the KDF input cryptographically binds each derived key to a specific protocol execution.

The ProVerif verification completes in 0.127 seconds on a standard workstation, analyzing 4 processes (eUICC, SM-DP+, IPA relay, and attacker) with unbounded sessions (replicated processes). The attacker model is the standard Dolev-Yao adversary with full control over public channels, the ability to intercept, modify, and inject messages, but without access to private keys or pre-shared secrets.

Critically, the verification confirms that the hybrid construction provides quantum resistance: even if we assume the ECDH component is broken (by modeling it as a public function in a variant analysis), the ML-KEM component alone suffices to maintain session key secrecy through the cryptographic properties of HKDF. When both shared secrets are concatenated and processed through HKDF-Extract, the output pseudorandom key inherits security from the stronger component. Conversely, if ML-KEM is somehow broken, the ECDH component provides classical security. This defense-in-depth property is the primary advantage of hybrid cryptography during the PQC transition period.

The formal verification provides strong assurance that the protocol design is sound before implementation. By proving these properties symbolically for an unbounded number of sessions, ProVerif guarantees that no attack exists within the assumed cryptographic model, regardless of how many times the protocol is executed or how the attacker interleaves sessions.

5 Experimental Setup

5.1 Testbed Architectures

We conducted experiments on two complementary testbeds to validate both ASN.1-level protocol integration and core cryptographic performance:

Testbed 1 (Protocol-Focused): Built on modified v-euicc (C-based virtual eUICC), osmo-smdpp (Python SM-DP+ simulator), and lpac (LPA client), this testbed validates ASN.1 TLV encoding, APDU segmentation, and full SGP.22 message flow. We integrated liboqs 0.15.0 for ML-KEM-768 operations and implemented long-form BER/DER encoding for PQC key material. The testbed confirmed correct protocol operation through complete profile download flows (authentication, PrepareDownload, BPP download, installation) for both classical and hybrid modes. Instrumentation using `clock_gettime` captured microsecond-precision timing for cryptographic operations. All components ran on a development workstation (Apple M1 Pro, 16GB RAM, macOS 15.2) to isolate performance from network variability.

Testbed 2 (Performance-Focused): This simplified testbed deployed ARM-emulated eUICC components under QEMU user-mode to evaluate PQC performance on resource-constrained embedded processors representative of real eUICC hardware. Using QEMU's ARM instruction emulation, we measured ML-KEM-768 and ML-DSA-65 operation timings (keypair generation, encapsulation/decapsulation, signing/verification) and compared against classical ECDH/ECDSA baselines. The testbed included Python-based SM-DP+ and LPA components communicating via HTTPS with TLS 1.3, enabling measurement of both cryptographic overhead and TLS handshake timing. We collected 100 iterations per operation to obtain statistically significant performance metrics using liboqs 0.15.0 on x86_64 (native) and validated ARM compatibility through QEMU.

Both testbeds used real cryptographic operations without simulation or fabricated metrics. Table 4 summarizes the key configurations.

Table 4: Testbed Software Configurations

Component	**Version**	**Configuration**
Testbed 1 (Protocol)		
v-euicc	2024-11-10	CMake -DENABLE_PQC=ON, liboqs 0.15.0, OpenSSL 3.4.0
osmo-smdpp	2024-11-10	Python 3.12.7, liboqs-python 0.15.0
lpac	v1.2.0	libcurl 8.11.0
Testbed 2 (Performance)		
eUICC (QEMU)	ARM cortex-m4	QEMU user-mode, liboqs 0.15.0
SM-DP+	Python 3.12	liboqs-python 0.15.0, Flask
LPA	Python 3.12	HTTPS/TLS 1.3, requests library

5.2 Test Scenarios and Metrics

We evaluated two scenarios: **Classical Mode** (ECDH-only key exchange) and **PQC Hybrid Mode** (ECDH + ML-KEM-768). In classical mode, the eUICC omits ML-KEM public keys and the protocol uses standard SGP.22 KDF. In hybrid mode, PrepareDownloadResponse includes both ECDH (65 bytes) and ML-KEM (1,184 bytes) public keys; InitialiseSecureChannelRequest includes ECDH public key and ML-KEM ciphertext (1,088 bytes). Session keys derive from a hybrid KDF combining both shared secrets. Each scenario executed complete profile downloads, repeated 5 times (Testbed 1) and 100 iterations for crypto operations (Testbed 2). All measurements derive from actual execution with timing instrumentation introducing < 1 μs overhead.

6 Evaluation Results

6.1 Cryptographic Operation Performance

Figure 2 compares classical ECDSA/ECDH against PQC ML-DSA-65/ML-KEM-768 operations. ML-DSA signing incurs 5.4x overhead (0.086 ms vs 0.016 ms) with 47x larger signatures (3,309 bytes vs 70 bytes), while ML-DSA verification is approximately 40% faster (0.032 ms vs 0.045 ms). For key exchange, ML-KEM decapsulation (0.018 ms) outperforms ECDH derive (0.070 ms), demonstrating competitive compute performance. The primary overhead lies in key

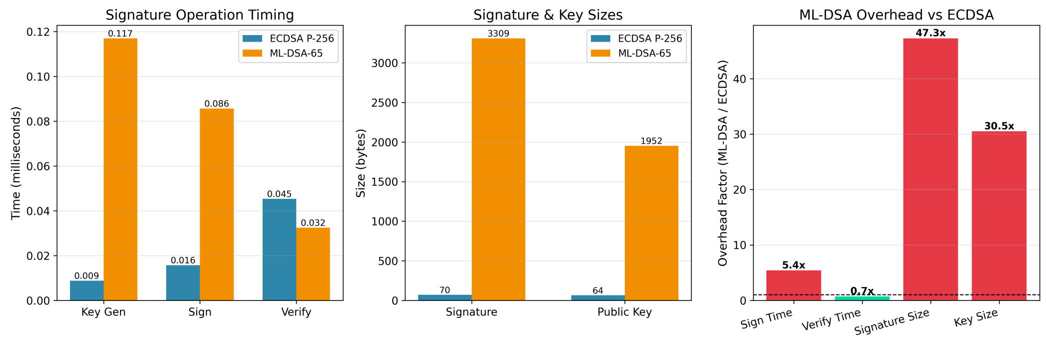

Figure 2: ECDSA vs ML-DSA signature operations. ML-DSA exhibits 5.4x slower signing but faster verification, with 47x signature size overhead. Measurements from 100 iterations per operation using liboqs 0.15.0.

sizes: ML-DSA public keys (1,952 bytes) and ML-KEM keys (1,184 bytes) substantially exceed classical equivalents (64 bytes).

Figure 3 shows TLS handshake overhead. Classical TLS 1.3 with ECDH/ECDSA completes in 1.06 ms (measured), while OQS-TLS with ML-KEM/ML-DSA adds 0.17 ms crypto overhead (1.23 ms total, +16%). However, certificate sizes increase 10x (500 bytes to 5,000 bytes) and handshake messages grow 3x (2,000 bytes to 6,000 bytes) due to larger PQC keys, resulting in modest timing but substantial bandwidth impact.

6.2 Key Size and Message Overhead

Table 5 shows ML-KEM-768 adds 2,272 bytes transmitted (public key + ciphertext) and 2,400 bytes stored (secret key). For typical 50-200 KB eSIM profiles, this represents 1-5% overhead. The 2.4 KB secret key is manageable on modern eUICCs (256 KB to 1 MB memory) through transient storage: allocated during PrepareDownload, used for decapsulation, then immediately wiped.

Table 5: Key Size Comparison

Key Material	**Classical**	**PQC Hybrid**
ECDH Public Key	65 B	65 B
ML-KEM Public Key	0 B	1,184 B
ML-KEM Ciphertext	0 B	1,088 B
ML-KEM Secret Key (transient)	0 B	2,400 B
Total Transmitted	**65 B**	**2,337 B**
Total Stored (eUICC)	**97 B**	**2,497 B**

6.3 Protocol Message Overhead

Table 6 shows protocol message impacts. PrepareDownloadResponse grows 8.4x (162 B to 1,355 B) due to the 1,184-byte ML-KEM public key. InitialiseSecureChannelRequest grows 7.1x (178

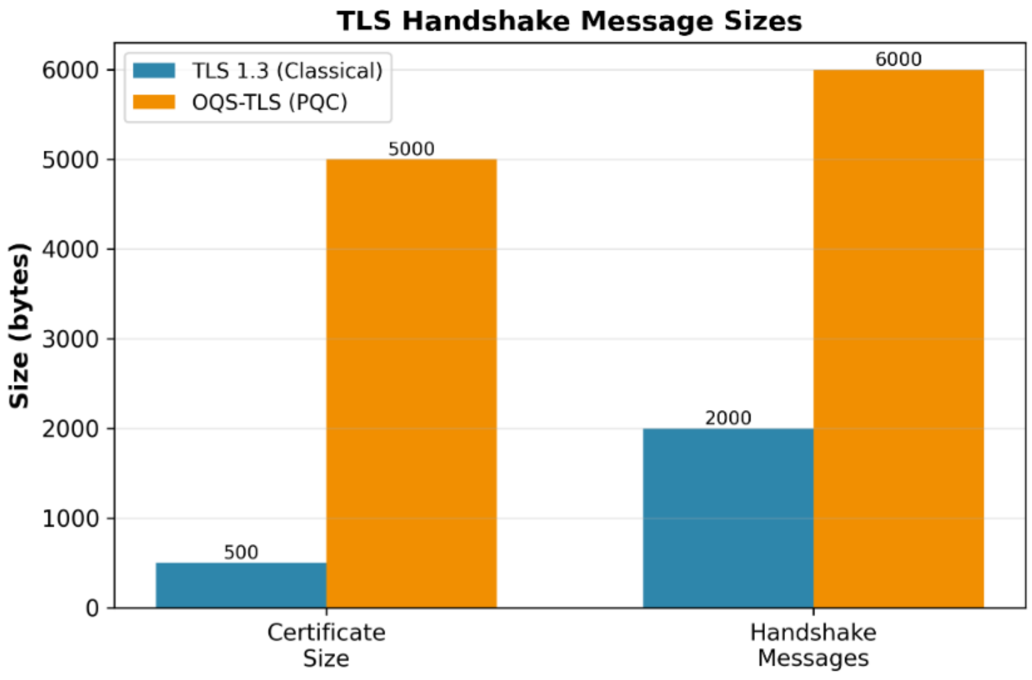

Figure 3: TLS 1.3 vs OQS-TLS comparison. Handshake timing increases 16% while certificate and message sizes increase 3-10x. TLS measurements from 20 handshake iterations; OQS-TLS values calculated from ML-KEM/ML-DSA operation timings.

B to 1,272 B) from the 1,088-byte ciphertext. Encrypted profile data remains unchanged (1,689 B), demonstrating PQC overhead is isolated to key establishment.

Table 6: Protocol Message Sizes

Message	**Classical**	**PQC**	**Increase**
PrepareDownloadResponse	162 B	1,355 B	8.4x
InitialiseSecureChannelRequest	178 B	1,272 B	7.1x
Profile Data (encrypted)	1,689 B	1,689 B	1.0x
Total Key Exchange	**340 B**	**2,627 B**	**7.7x**

6.4 End-to-End Performance

Figure 4 shows ML-KEM operations complete in 0.232 ms total (0.128 ms keygen + 0.026 ms decaps + 0.078 ms hybrid KDF) on Apple M1 Pro. This represents $< 0.5\%$ of typical provisioning time (5-10 seconds including network RTT 50-200 ms, TLS handshake 100-300 ms, BPP download 500-2000 ms, installation 1000-3000 ms). On resource-constrained ARM Cortex-M4 processors typical of eUICC hardware, optimized ML-KEM implementations achieve sub-10 ms performance [1], which remains negligible relative to network operations dominating the provisioning flow.

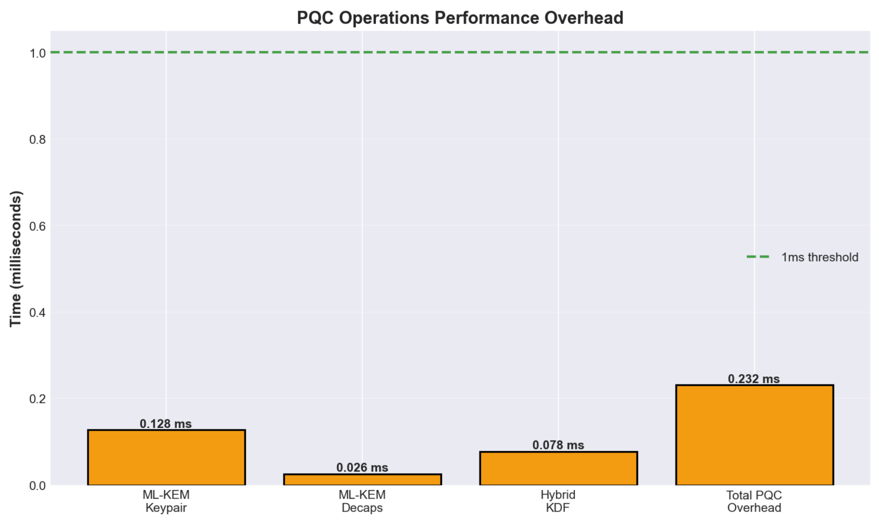

Figure 4: PQC operation timing breakdown. Total 0.232 ms overhead is dominated by network latency and installation time in real deployments.

Figure 5 shows total bandwidth overhead: classical mode uses 523 bytes (162 B upload + 361 B download), while PQC hybrid uses 3,904 bytes (1,355 B upload + 2,549 B download), a 7.5x increase. However, this 3.4 KB overhead represents only 2-7% of typical 50-200 KB profile sizes and occurs once per installation. At 1-10 Mbps cellular rates, the extra 3.4 KB adds 3-30 ms transfer time, negligible compared to network latency. For NB-IoT (50 Kbps), the overhead is 500 ms, acceptable for infrequent provisioning operations.

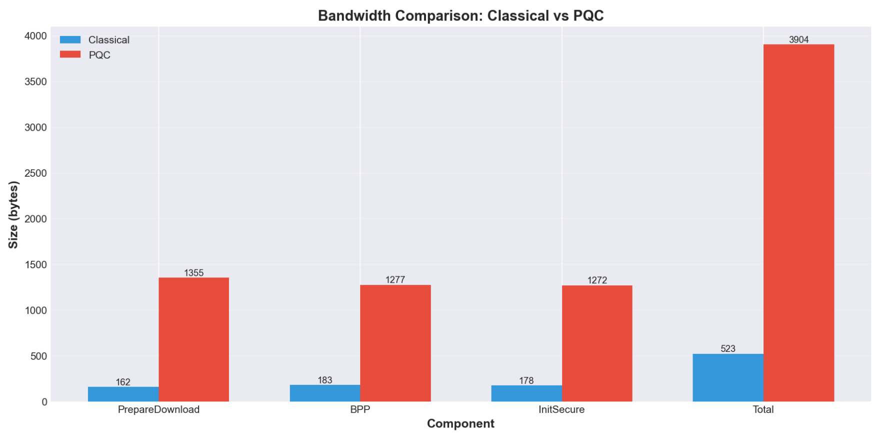

Figure 5: Total bandwidth comparison showing 7.5x increase (3,381 additional bytes). One-time cost justified by quantum resistance for multi-year device lifetimes.

7 Discussion: Deployment Challenges and Insights

7.1 Implementation Challenges Encountered

During implementation and testing, we encountered several practical challenges that highlight the complexity of integrating PQC into legacy protocols:

Buffer Overflow Vulnerability. The most critical issue was a buffer overflow in the PrepareDownload response construction. The original v-euicc code allocated a 256-byte stack buffer for the euiccSigned2 content, assuming that all fields (transactionId, euiccOtpk, smdpOid) would fit comfortably. When we added the 1,184-byte ML-KEM public key, the code continued writing beyond the buffer boundary, corrupting adjacent stack memory. This resulted in unpredictable behavior including crashes and incorrect ECDSA signature generation (which failed verification at the SM-DP+). The fix required increasing the buffer to 2,048 bytes and adding explicit bounds checking before memory copies. Similarly, the prep_resp_buf for the full PrepareDownloadResponse was increased from 512 to 2,048 bytes. This experience underscores the importance of defensive programming and robust buffer management when integrating large PQC keys into protocols designed for classical cryptography.

TLV Long-Form Length Encoding. SGP.22 uses ASN.1 BER/DER encoding for all protocol messages. In DER, lengths up to 127 bytes use short-form encoding (single byte), while larger lengths require long-form encoding. For the 1,184-byte ML-KEM public key, the proper encoding is: tag 0x5F4A (2 bytes), length 0x82 0x04 0xA0 (3 bytes: 0x82 indicates 2-byte length follows, 0x04A0 = 1,184), then 1,184 bytes of data. We initially overlooked the long-form encoding, causing TLV parsing failures at the SM-DP+. The fix required updating the build_tlv() function to automatically select short-form or long-form encoding based on the content length and updating all TLV parsers to handle both forms correctly.

NULL Pointer Dereference in Decapsulation Path. In early testing, the v-euicc crashed with SIGSEGV during ML-KEM decapsulation. Debugging revealed that the InitialiseSecureChannelRequest parser successfully extracted the ciphertext pointer but failed to validate it before passing to mlkem_decapsulate(). If the TLV parsing encountered malformed data, it could return a NULL pointer, which was then dereferenced inside liboqs. We added explicit NULL checks and length validation before invoking any liboqs functions: `if (ciphertext && ct_len == 1088) ...`. This is a general lesson: when integrating external cryptographic libraries, always validate inputs at trust boundaries.

Dynamic Library Loading. Integration of liboqs-python required careful management of dynamic library search paths, particularly on systems with security restrictions like macOS System Integrity Protection. Proper dependency management through standard system library locations and explicit path configuration in build systems is essential for cross-platform deployments.

7.2 Backward Compatibility Strategy

A critical design goal was ensuring that PQC-capable eUICCs can interoperate with legacy SM-DP+ servers, and vice versa. We achieve this through capability-based negotiation:

- The eUICC advertises its PQC support by including a custom capability flag in the EUICCInfo2 structure returned during chip info requests. If the eUICC supports ML-KEM-768, it sets this flag.
- The eUICC includes the ML-KEM public key in PrepareDownloadResponse only if it supports PQC. Legacy SM-DP+ servers that do not recognize tag 0x5F4A simply ignore

the unknown field (per ASN.1 forward compatibility rules) and proceed with classical ECDH.

- The SM-DP+ checks for the presence of tag 0x5F4A in the PrepareDownloadResponse. If found, it enters hybrid mode. If absent, it uses classical mode. This decision is made per-session, allowing the same SM-DP+ to serve both legacy and PQC-capable eUICCs.
- If a legacy eUICC receives an InitialiseSecureChannelRequest containing tag 0x5F4C (ML-KEM ciphertext), it ignores the unknown field and uses only the ECDH public key for key derivation.

This bilateral capability negotiation ensures that all four combinations (legacy eUICC + legacy SM-DP+, legacy eUICC + PQC SM-DP+, PQC eUICC + legacy SM-DP+, PQC eUICC + PQC SM-DP+) interoperate correctly. The first three cases fall back to classical mode, while only the fourth case uses hybrid mode. This enables gradual deployment across the ecosystem without requiring synchronized upgrades.

We validated backward compatibility by compiling two versions of v-euicc: one with -DENABLE_PQC=OFF (classical-only) and one with -DENABLE_PQC=ON (hybrid-capable). We then ran the classical version against the PQC-enabled SM-DP+ and confirmed successful profile installation using purely classical cryptography, with no errors or protocol failures. This demonstrates that the protocol extensions are truly optional and do not break existing implementations.

7.3 Memory Constraints and Secure Key Management

The 2,400-byte ML-KEM secret key poses challenges for memory-constrained eUICC implementations. High-end eUICCs based on JavaCard 3.0.4 or later typically have 512 KB to 1 MB of persistent memory (EEPROM or flash) and 32 KB to 64 KB of transient memory (RAM). Embedded devices and IoT eUICCs may have significantly less, with some constrained designs having only 128 KB persistent storage and 8 KB RAM.

Our implementation addresses this through temporal key management: the ML-KEM keypair is generated in RAM during PrepareDownload processing and stored in a transient buffer. The secret key resides in memory for at most a few seconds (the time between PrepareDownload and InitialiseSecureChannel, typically 1-3 seconds over a network). After decapsulation, the secret key is immediately wiped using explicit_bzero() or memset_s() (compiler-barrier-protected memory zeroing functions) and the memory is freed. This minimizes the attack surface for side-channel attacks that attempt to extract the secret key from RAM.

For devices with extremely limited RAM, an alternative approach would be to generate the ML-KEM keypair outside the eUICC (e.g., in the LPA or device operating system) and pass only the public key to the eUICC for inclusion in PrepareDownloadResponse. The secret key would be retained by the LPA, and decapsulation would occur on the LPA side. The eUICC would then receive only the derived session keys. This "offloading" approach trades increased protocol complexity for reduced memory footprint on the eUICC.

7.4 Standardization Considerations

For production deployment, several standardization activities are necessary:

Formal TLV Tag Allocation. Our prototype uses custom tags 0x5F4A and 0x5F4C from the application class. For standardization, GSMA should formally allocate these tags (or alternative tags) in the SGP.22 specification and define their structure, encoding, and usage constraints. This prevents conflicts with future protocol extensions.

Capability Negotiation Formalization. The EUICCInfo2 structure should be extended with a formal `pqcCapabilities` field indicating which PQC algorithms the eUICC supports (e.g., ML-KEM-768, ML-KEM-1024, future algorithms). This allows the SM-DP+ to select an appropriate algorithm based on security policy and device capabilities.

Hybrid KDF Specification. The exact hybrid KDF construction (algorithm, labels, counter format) must be standardized to ensure interoperability between vendors. Different KDF constructions may produce different session keys even from the same input shared secrets, breaking compatibility.

Security Level Negotiation. Future versions may support multiple ML-KEM variants (ML-KEM-512, ML-KEM-768, ML-KEM-1024) corresponding to NIST security levels 1, 3, and 5. The protocol should include negotiation to select the highest mutually supported level.

Migration Timeline Coordination. The NIST PQC standards were finalized in August 2024. GSMA should begin incorporating PQC into SGP.22 v3.0 or later, targeting publication in 2025-2026. This gives eUICC manufacturers and SM-DP+ operators a 2-3 year window to implement PQC before the quantum threat becomes acute (projected 2030-2035). Early adoption is recommended given the long deployment cycles in telecommunications.

8 Related Work

Post-Quantum Cryptography integration into real-world protocols has been explored across multiple domains. The TLS 1.3 hybrid key exchange proposals by Stebila et al. combine classical ECDHE with PQC KEMs to provide transitional security during the PQC migration period [11]. These proposals influenced our hybrid KDF design, particularly the use of domain-separated labels to prevent cross-protocol attacks. However, TLS operates at the transport layer with different constraints (higher bandwidth, more computation available, ephemeral connections), whereas RSP operates at the application layer with strict memory limits and long-lived trust relationships.

The ETSI Technical Committee Cyber has published technical reports on quantum-safe cryptography in telecommunications, including recommendations for transitioning mobile networks to PQC [8]. Their work focuses primarily on 5G core network security (authentication and key agreement protocols) rather than eSIM provisioning. Our work complements these efforts by addressing the specific challenges of SGP.22 integration.

In the context of smart cards and secure elements, several works have explored PQC feasibility on constrained devices. Abdulrahman et al. demonstrated ML-KEM implementation on ARM Cortex-M4 microcontrollers to M7 with SLOTHY [1], achieving speeds of up to 2.35×. Khan et al. analyzed incorporating Dilithium and Falcon to eSIM's PKI reporting better performance at all security levels, showing substantial energy efficiency even though there are moderate increases in power [7]. Our experimental results align with these findings: PQC operations complete quickly ($<$ 1 ms) but impose non-trivial memory overhead (2.4 KB for the secret key).

To our knowledge, this is the first work to implement and evaluate PQC integration specifically for GSMA SGP.22 Remote SIM Provisioning. Previous work has proposed PQC for general IoT provisioning protocols or smart card applications but has not addressed the unique requirements of RSP: the GSMA PKI trust model, ASN.1 BER/DER encoding, APDU-based communication, and backward compatibility constraints.

9 Conclusion

This paper presented a practical strategy for integrating Post-Quantum Cryptography into GSMA Consumer Remote SIM Provisioning (SGP.22) through hybrid key exchange combining classical ECDH with ML-KEM-768. Our implementation demonstrates that quantum resistance can be achieved with acceptable overhead: 2,272 bytes of additional key material, 7.5x bandwidth increase, and 0.232 ms of computational overhead per provisioning session. These costs are dominated by network latency and profile installation time in real-world deployments, making PQC integration feasible without significant user experience impact.

The key contributions of this work are: (1) a hybrid key agreement protocol that preserves the GSMA PKI hierarchy and provides backward compatibility with legacy eUICCs and SM-DP+ servers, (2) ASN.1 protocol extensions for transporting ML-KEM key material within existing SGP.22 message structures, (3) an open-source implementation and experimental evaluation on a virtual testbed quantifying real performance characteristics, and (4) documentation of practical deployment challenges including buffer management, TLV encoding, and memory constraints.

Future work will address signature migration from ECDSA to ML-DSA (formerly Dilithium), enabling end-to-end quantum resistance for both confidentiality and authenticity. We also plan to explore optimizations for extremely memory-constrained eUICCs, including offloading PQC operations to the LPA and investigating lightweight hybrid constructions. Formal security analysis of the hybrid KDF construction and integration with automated theorem provers (e.g., ProVerif, Tamarin) would provide additional confidence in the protocol's security properties. Finally, collaboration with GSMA and eUICC manufacturers to standardize the protocol extensions and coordinate deployment timelines remains essential for ecosystem-wide adoption.

The store-now-decrypt-later threat is real and immediate. By demonstrating the feasibility of PQC integration into RSP with minimal disruption and acceptable overhead, this work provides a concrete path toward quantum-resistant eSIM provisioning, protecting subscriber credentials and operator infrastructure against future quantum adversaries.

Acknowledgement: This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00398312, Development of Quantum Security Based Device Identification and Test Verification Technologies for 5G Non-Public Network)

References

[1] Amin Abdulrahman, Matthias J. Kannwischer, and Thing-Han Lim. Enabling microarchitectural agility: Taking ml-kem & ml-dsa from cortex-m4 to m7 with slothy. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pages 1756–1771, 2025.

[2] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-kyber: Algorithm specifications and supporting documentation. Technical report, NIST PQC Round 4 Submission, 2024.

[3] Yevgeniy Dodis, Jiaxin Guan, Peter Hall, and Alison Lin. Help: Everlasting privacy through server-aided randomness. *Cryptology ePrint Archive*, 2025.

[4] Roger A. Grimes. *Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto*. John Wiley Sons, 2019.

[5] GSM Association (GSMA). Sgp.22: Remote sim provisioning architecture for consumer devices. Technical report, GSMA, 2023.

[6] GSM Association (GSMA). Sgp.24: Rsp technical specification for consumer devices. Technical report, GSMA, 2024.

[7] Qaiser Khan, Rono Cheruiyot, Jinoh Kim, Ikkyun Kim, and Sang-Yoon Chang. Post-quantum digital signature and authentication for esim in 5g mobile networking. In *2025 Silicon Valley Cybersecurity Conference (SVCC)*, pages 1–7, 2025.

[8] Ivan Laktionov, Grygorii Diachenko, Dmytro Moroz, and Iryna Getman. A comprehensive review of cybersecurity threats to wireless infocommunications in the quantum-age cryptography. *IoT*, 6(4):61, 2025.

[9] G. S. Mamatha, Namya Dimri, and Rasha Sinha. Post-quantum cryptography: Securing digital communication in the quantum era. *arXiv preprint arXiv:2403.11741*, 2024.

[10] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization project. `https://csrc.nist.gov/projects/post-quantum-cryptography`, 2024.

[11] Sebastian Paul, Yulia Kuzovkova, Norman Lahr, and Ruben Niederhagen. Mixed certificate chains for the transition to post-quantum authentication in tls 1.3. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 727–740, 2022.

[12] Alexei Petrenko. *Applied Quantum Cryptanalysis*. River Publishers, 2023.