Post-Quantum Migration Strategy for RSA Encryption

Udara Pathum[1][0009−0006−2194−6630] and Ashen De Silva[1,2][0009−0009−2256−4441]

[1] WSO2 LLC, USA hwupathum@gmail.com
[2] Northeastern University, USA desilva.ashen3@gmail.com

Abstract. Organizations relying on RSA-OAEP encryption in protocols such as JWE, SAML, and OIDC face a critical challenge: transitioning to post-quantum cryptography without disrupting operational continuity. This paper presents a phased migration strategy that uses RSA-KEM-ML-KEM composite Key Encapsulation Mechanisms as an intermediary step between current RSA-OAEP encryption and the target state of pure ML-KEM adoption. We formalize the RSA-KEM-ML-KEM construction, prove IND-CCA2 security via a Split-Key PRF combiner, and integrate it into the Hybrid Public-Key Encryption (HPKE) framework. Our implementation demonstrates that composite schemes enable quantum-resistant encryption while preserving existing RSA key infrastructure, though with measurable throughput trade-offs that inform migration timelines. We analyze protocol-specific integration for JWE, SAML, and OIDC encryption use cases, providing decision frameworks for transitioning from RSA-OAEP through composite approaches to pure post-quantum encryption. This work contributes a formally analyzed transition mechanism and practical migration guidance for organizations seeking to adopt quantum-resistant encryption in RSA-dependent systems, validated through application to identity and access management protocols.

Keywords: Post-quantum migration · HPKE · RSA encryption · ML-KEM · Identity protocols

1 Introduction

Modern digital infrastructures are highly dependent on RSA-based cryptography to secure widely deployed protocols such as OpenID Connect (OIDC), Security Assertion Markup Language (SAML), and JSON Web Encryption (JWE) [11]. Specifically, RSA-OAEP remains the dominant algorithm for encryption operations in these protocols, used to protect sensitive tokens, assertions, and encrypted payloads. However, the emergence of quantum computing and Shor's algorithm [19] threatens the security of these integer-factorization-based systems. Although NIST has standardized the Module-Lattice Key Encapsulation Mechanism (ML-KEM) [16] as a quantum-resistant alternative, many organizations cannot abruptly migrate from RSA-OAEP to pure ML-KEM without risking service interruption or requiring complete infrastructure replacement.

To address this challenge, we present RSA-KEM-ML-KEM, a composite Key Encapsulation Mechanism integrated via the Hybrid Public-Key Encryption (HPKE) framework, validated through the migration of RSA-OAEP encryption in JWE, SAML, and OIDC protocols. We position RSA-KEM-ML-KEM as an **intermediary migration step** that bridges legacy RSA-OAEP deployments (Phase 1) with the eventual goal of pure ML-KEM adoption (Phase 3).

Critically, this composite transition phase (Phase 2) does not require RSA key regeneration. Organizations maintain their existing RSA public keys and provision separate ML-KEM keys alongside them, with the composite operation occurring at encryption time. This key reuse capability significantly reduces migration friction, as existing infrastructure, key distribution mechanisms, and trust relationships remain intact while quantum resistance is added incrementally.

Our contributions are:

1. We formalize RSA-KEM-ML-KEM as a transition mechanism for encryption, proving IND-CCA2 security and integrating it into HPKE.
2. We provide comprehensive benchmarks demonstrating the 40× performance difference between composite and pure ML-KEM approaches, establishing quantitative criteria for migration planning.
3. We detail phased migration strategies for JWE, SAML, and OIDC encryption that use RSA-KEM-ML-KEM as a bridge.
4. We analyze when organizations should exit the composite phase based on infrastructure readiness and performance requirements.

2 Literature Review

2.1 Classical Public-Key Encryption

Classical public-key encryption [8] relies on mathematically hard problems such as integer factorization. The RSA cryptosystem [18] remains the most widely deployed scheme, with recent specifications defined in PKCS #1 v2.2 [14]. Variations include RSA-KEM [10, 20], which derives symmetric keys from a random integer and offers simplified security proofs, and RSA-OAEP, a padding-based scheme ubiquitous in the application protocols targeted by this work [11]. Limitations such as computational overhead and message size restrictions have historically motivated the shift toward hybrid encryption [20].

2.2 Post-Quantum Cryptography

Post-quantum cryptography (PQC) targets security against quantum adversaries. Following the threat posed by Shor's algorithm to RSA and Diffie-Hellman [19], NIST initiated a standardization process [17] that selected ML-KEM (Module-Lattice Key Encapsulation Mechanism) as the primary key establishment algorithm [15]. ML-KEM derives its security from the hardness of the Module Learning With Errors (MLWE) problem [16] and is now standardized as FIPS 203.

2.3 Composite KEMs and Secure Combiners

Composite cryptographic approaches combine multiple cryptographic primitives to provide security against both classical and quantum adversaries. The theory of generic KEM combiners has been extensively studied to ensure that the resulting scheme remains secure even if one or more of its ingredients are compromised [9, 6]. To address the lack of comparative benchmarks between different architectures, we evaluate three primary mechanisms along axes of security, key binding, and performance (Table 1).

Table 1. Comparison of KEM Combiner Architectures along Security and Performance Axes

Combiner Type	**Security Model**	**Key Binding**	**Computational Overhead**[a]
XOR Combiner	ROM / Full-CT	Weak	$\approx 1\mu s$
Generic CtKDF	Random Oracle	Moderate	$\approx 5\mu s$
Split-Key PRF	Standard Model	Strong	$\approx 8\mu s$

[a] Estimated latencies based on benchmarks from eBACS [5] and SHA-NI throughput on modern x86_64 processors (e.g., Intel Sapphire Rapids).

The security of the combined key depends on the chosen combiner mechanism:

- **XOR Combiners:** A simple construction that XORs individual shared secrets. While efficient, it only provides IND-CCA security if additional assumptions (like the Random Oracle Model) are made or if the secrets are combined with the full ciphertext [9].
- **Concatenation KDF (CtKDF):** This construction concatenates shared secrets and processes them through a KDF. It is widely used in practical standards [7, 4] but often relies on modeling the hash function as a Random Oracle for security proofs.
- **Split-Key PRF (Dual-PRF):** This construction, formally analyzed by Giacon et al. [9], uses a PRF to combine keys. It is proven to be IND-CCA secure in the standard model as long as at least one of the ingredient KEMs is IND-CCA secure, provided the PRF satisfies the Dual-PRF property.

While these generic combiners provide a theoretical foundation, their application to the migration of RSA-OAEP infrastructure remains an open challenge. RSA-OAEP's specific performance profile (extreme asymmetry) and the strict operational constraints of identity protocols (e.g., URL limits) necessitate a protocol-specific analysis that goes beyond generic KEM theory.

2.4 Hybrid Public-Key Encryption (HPKE)

HPKE (RFC 9180) [4] is a modern framework combining a KEM, a KDF, and an AEAD scheme. It defines specific interfaces for these primitives.

- **Setup:** The recipient generates a key pair via *KeyGen*
- **EncapsulateAndSeal:** The sender encapsulates a key and encrypts the payload. We refer readers to the standard *Encaps* flow defined in RFC 9180 Section 5.1.
- **Open:** The recipient recovers the key and decrypts the ciphertext, as defined in RFC 9180 Section 5.2.

Extensions for post-quantum algorithms (PQ-HPKE) have been proposed [1], but current specifications primarily focus on pure ML-KEM or ECC-based composites [13], leaving the integration of RSA-based composites as an open implementation challenge.

2.5 Composite post-quantum KEMs

Recent efforts by the IETF have focused on standardizing hybrid post-quantum key encapsulation mechanisms (KEMs) to facilitate a secure transition. Notable constructions include X-Wing [3], which combines X25519 and ML-KEM-768, and various HPKE-based composite schemes. These efforts emphasize the importance of formal security proofs and interoperability in real-world deployments, such as X.509 certificate-based authentication and TLS 1.3. Our work aligns with these standardization trends by providing a rigorous security analysis of our composite construction.

3 Methodology

This section details the construction of the RSA-KEM-ML-KEM composite scheme and its integration into the HPKE framework

3.1 Construction of RSA-KEM-ML-KEM

Our proposed RSA-KEM-ML-KEM composite scheme is constructed by combining the classical RSA-KEM with the post-quantum ML-KEM in a parallel fashion. This design ensures that the resulting secret key is secure as long as at least one of the underlying KEMs remains unbroken. This approach provides a robust migration strategy, allowing organizations to gradually migrate to post-quantum cryptography while maintaining infrastructure compatibility and security against current threats.

Key Generation The recipient generates two distinct key pairs, one for RSA-KEM and one for ML-KEM. These are then combined into a single composite key pair.

Algorithm 1 RSA-KEM-ML-KEM Key Generation

Require: Security parameter 1^λ
Ensure: Composite key pair (pk_c, sk_c)
1: $(pk_{RSA}, sk_{RSA}) \leftarrow \mathsf{RSA\text{-}KEM.KeyGen}()$
2: $(pk_{ML}, sk_{ML}) \leftarrow \mathsf{ML\text{-}KEM.KeyGen}()$
3: $pk_c \leftarrow (pk_{RSA}, pk_{ML})$
4: $sk_c \leftarrow (sk_{RSA}, sk_{ML})$
5: **return** (pk_c, sk_c)

Encapsulation To establish a shared secret with the recipient, the sender uses the composite public-key pk_c to perform the encapsulation separately on each KEM component. The results are then combined using a KDF.

Algorithm 2 RSA-KEM-ML-KEM Encapsulation

Require: Composite public-key $pk_c = (pk_{RSA}, pk_{ML})$
Ensure: Composite ciphertext c_c and key k_c
1: $(c_{RSA}, k_{RSA}) \leftarrow \mathsf{RSA\text{-}KEM.Encaps}(pk_{RSA})$
2: $(c_{ML}, k_{ML}) \leftarrow \mathsf{ML\text{-}KEM.Encaps}(pk_{ML})$
3: $c_c \leftarrow (c_{RSA}, c_{ML})$
4: $k_c \leftarrow \mathsf{KeyCombine}(k_{RSA}, k_{ML}, c_{RSA}, c_{ML})$
5: **return** (c_c, k_c)

Decapsulation Upon receiving a composite ciphertext $c_c = (c_{RSA}, c_{ML})$, the recipient decapsulates each component separately using the corresponding secret keys. The two partial keys are then combined using the same KDF to recover the final composite key.

Algorithm 3 RSA-KEM-ML-KEM Decapsulation

Require: Composite secret key $sk_c = (sk_{RSA}, sk_{ML})$
Require: Composite ciphertext $c_c = (c_{RSA}, c_{ML})$
Ensure: Shared secret k_c
1: $k_{RSA} \leftarrow \mathsf{RSA\text{-}KEM.Decaps}(sk_{RSA}, c_{RSA})$
2: $k_{ML} \leftarrow \mathsf{ML\text{-}KEM.Decaps}(sk_{ML}, c_{ML})$
3: $k_c \leftarrow \mathsf{KeyCombine}(k_{RSA}, k_{ML}, c_{RSA}, c_{ML})$
4: **return** k_c

Combiner Selection and Construction The security of the composite KEM depends critically on the mechanism used to combine the keys derived from RSA-KEM and ML-KEM. Based on the trade-offs analyzed in Section 2.3, we selected the **Split-Key PRF** architecture. As shown in Table 1, while performance is not

a critical factor, the Split-Key PRF provides the necessary security robustness in the standard model for protecting legacy systems.

We represent the combiner using HKDF-Extract with SHA-256. While structurally similar to a CtKDF, our security argument relies on the Dual-PRF property of HMAC. Recent analysis by Backendal et al. [2] confirms that HMAC acts as a Dual-PRF, meaning it remains pseudo-random when keyed via the message input, provided at least one segment of that input contains high entropy.

To prevent attacks where variable-length inputs could be shifted to produce identical concatenations (e.g., $A||BC = AB||C$), we enforce strict length prefixing using the I2OSP (Integer-to-Octet-String Primitive) function. The final key k_c is derived as:

$$k_c \leftarrow \mathsf{HKDF\text{-}Extract}(\text{salt}, \text{Input} \parallel \text{Context})$$

where:

- **Input**: $\text{I2OSP}(|k_{RSA}|, 4) \parallel k_{RSA} \parallel \text{I2OSP}(|k_{ML}|, 4) \parallel k_{ML}$
- **Context**: $\text{I2OSP}(|c_{RSA}|, 4) \parallel c_{RSA} \parallel \text{I2OSP}(|c_{ML}|, 4) \parallel c_{ML}$
- **Salt**: A non-secret, protocol-specific value used for domain separation
- **Hash function**: SHA-256 within HKDF as specified in RFC 5869 [12]

3.2 RSA-KEM-ML-KEM as an HPKE Scheme

To utilize the RSA-KEM-ML-KEM composite KEM within a HPKE framework, we integrate our composite KEM into the HPKE specification [4]. HPKE provides a standardized way to encrypt arbitrary-sized plaintexts to a recipient's public-key, leveraging a KEM for key establishment and a symmetric AEAD scheme for data protection. In our scheme, the KEM component of HPKE is instantiated with the composite KEM.

HPKE Setup with RSA-KEM-ML-KEM The HPKE setup involves the recipient generating a KEM key pair. In our case, this means the recipient generates the composite KEM key pair, as shown in Algorithm 4.

Algorithm 4 RSA-KEM-ML-KEM HPKE Setup

Require: Security parameter 1^λ
Ensure: Composite KEM key pair (pk_c, sk_c)
1: $(pk_c, sk_c) \leftarrow \mathsf{CKEM.KeyGen}()$
2: **return** (pk_c, sk_c)

HPKE Encapsulation and Sealing To send an encrypted message, the sender performs the HPKE EncapsulateAndSeal procedure, using the recipient's composite public-key pk_c.

Algorithm 5 RSA-KEM-ML-KEM HPKE Encapsulate and Seal

Require: Composite public-key pk_c, plaintext P, associated data AAD
Ensure: Composite ciphertext c_c and AEAD ciphertext C
1: $(c_c, k_c) \leftarrow \mathsf{CKEM.Encaps}(pk_c)$
2: $(K, N) \leftarrow \mathsf{KeySchedule}(k_c)$
3: $C \leftarrow \mathsf{AEAD.Seal}(K, N, P, AAD)$
4: **return** (c_c, C)

HPKE Opening The recipient uses the HPKE Open procedure to decapsulate the shared key and decrypt the ciphertext, as shown in Algorithm 6.

Algorithm 6 RSA-KEM-ML-KEM HPKE Open

Require: Composite secret key sk_c, composite ciphertext c_c, AEAD ciphertext C, associated data AAD
Ensure: Decrypted plaintext P
1: $k_c \leftarrow \mathsf{CKEM.Decaps}(sk_c, c_c)$
2: $(K, N) \leftarrow \mathsf{KeySchedule}(k_c)$
3: $P \leftarrow \mathsf{AEAD.Open}(K, N, C, AAD)$
4: **return** P

By instantiating the KEM component of HPKE with our RSA-KEM-ML-KEM composite KEM, we obtain a quantum-resistant key-establishment mechanism embedded within the HPKE framework, which then uses symmetric encryption to efficiently protect arbitrary-sized data. This modular approach also allows flexibility in choosing KDF and AEAD algorithms, enhancing cryptographic agility.

Key Schedule Function We adopt the standard HPKE key schedule as specified in RFC 9180 [4], which derives encryption keys and nonces from the shared secret established by our composite KEM. Note that k_c has been processed through the KEM combiner using HKDF-Extract.

For our implementation, we use HKDF-SHA256 for the key schedule and AES-256-GCM for AEAD, yielding:

- **Key size** (N_k): 32 bytes
- **Nonce size** (N_n): 12 bytes

3.3 Security Analysis

We analyze the RSA-KEM-ML-KEM composite construction in the standard model, relying on the Dual-PRF security of HKDF.

Algorithm 7 Key Schedule

Require: Shared secrets k_c
Require: salt (HKDF salt)
Require: AEAD key size N_k, nonce size N_n
Ensure: Encryption key K, nonce N
1: $K \leftarrow \mathsf{HKDF\text{-}Expand}(k_c, \text{"key"}, N_k)$
2: $N \leftarrow \mathsf{HKDF\text{-}Expand}(k_c, \text{"base_nonce"}, N_n)$
3: **return** (K, N)

Theorem 1 (Robustness of Split-Key Combiner). *Let Π_{RSA} and Π_{ML} be the RSA-KEM and ML-KEM schemes respectively. Let F be the HKDF-Extract function modeled as a Dual-PRF. For any IND-CCA2 adversary $\mathcal{A}$ against the composite scheme Π_c, there exist adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{D}$ such that:*

$$\mathbf{Adv}^{IND\text{-}CCA2}_{\Pi_c}(\mathcal{A}) \leq \min\left(\mathbf{Adv}^{IND\text{-}CCA2}_{\Pi_{RSA}}(\mathcal{B}_1), \mathbf{Adv}^{IND\text{-}CCA2}_{\Pi_{ML}}(\mathcal{B}_2)\right) + \mathbf{Adv}^{Dual\text{-}PRF}_{F}(\mathcal{D})$$

Proof. We prove the theorem by considering the case where at least one of the component KEMs is secure. Without loss of generality, we assume Π_{RSA} is secure. The symmetric case where Π_{ML} is secure follows an identical logic (swapping indices). We proceed via a sequence of games.

Game 0 (Real Experiment): This is the standard IND-CCA2 game for the composite KEM.

1. **Setup:** The Challenger generates $(pk_{RSA}, sk_{RSA}) \leftarrow \Pi_{RSA}.\text{KeyGen}()$ and $(pk_{ML}, sk_{ML}) \leftarrow \Pi_{ML}.\text{KeyGen}()$.
2. **Phase 1:** The adversary $\mathcal{A}$ receives $pk_c = (pk_{RSA}, pk_{ML})$. $\mathcal{A}$ makes adaptive decapsulation queries for ciphertexts $c = (c_{RSA}, c_{ML})$. The Challenger computes $k_{RSA} \leftarrow \text{Decaps}(sk_{RSA}, c_{RSA})$ and $k_{ML} \leftarrow \text{Decaps}(sk_{ML}, c_{ML})$, then returns $k_c \leftarrow F(salt, \text{Input}||\text{Context})$
3. **Challenge:** $\mathcal{A}$ requests a challenge. The Challenger generates:
   - $(c^*_{RSA}, k^*_{RSA}) \leftarrow \text{Encaps}(pk_{RSA})$
   - $(c^*_{ML}, k^*_{ML}) \leftarrow \text{Encaps}(pk_{ML})$
   - $k_{real} = F(salt, \text{Input}^*||\text{Context}^*)$
   - $k_{rand} \leftarrow \{0,1\}^{|k_{real}|}$

   The Challenger flips a bit $b \in \{0,1\}$ and returns $(c^* = (c^*_{RSA}, c^*_{ML}), K_b)$ where $K_0 = k_{real}, K_1 = k_{rand}$.
4. **Phase 2:** $\mathcal{A}$ makes additional adaptive decapsulation queries for ciphertexts $c \neq c^*$. The Challenger responds as in Phase 1.
5. **Guess:** $\mathcal{A}$ outputs a guess b'.

The advantage of $\mathcal{A}$ in Game 0 is defined as $|\Pr[b' = b] - 1/2|$.

Game 1 (Replace k^*_{RSA}): Same as Game 0, but in the challenge phase, the Challenger replaces the actual RSA secret k^*_{RSA} with a truly random string $\tilde{k}_{RSA}$ of the same length *before* feeding it into the Combiner function F.

- *Justification:* We construct an adversary $\mathcal{B}_1$ against Π_{RSA}. $\mathcal{B}_1$ embeds the challenge RSA public key into pk_c. When $\mathcal{A}$ requests the challenge, $\mathcal{B}_1$ requests a challenge from its own RSA challenger, receiving $(c^*_{RSA}, \text{key}^*)$ (where key^* is either real or random). $\mathcal{B}_1$ generates the ML-KEM components locally.
- *Decapsulation Oracle:* Since $\mathcal{B}_1$ does not hold sk_{RSA}, for any query $c \neq c^*$, $\mathcal{B}_1$ queries its own decryption oracle to recover k_{RSA} and uses its locally generated sk_{ML} to recover k_{ML}, effectively simulating the full oracle.
- *Analysis:* If $\mathcal{A}$ distinguishes Game 1 from Game 0, $\mathcal{B}_1$ distinguishes the RSA-KEM key, breaking IND-CCA2 security. Thus: $|\Pr[S_0] - \Pr[S_1]| \leq \mathbf{Adv}^{\text{IND-CCA2}}_{\Pi_{RSA}}(\mathcal{B}_1)$.

Game 2 (Replace Combiner Output): Same as Game 1, but we now replace the derived key k_{real} with a truly random string K_{random}.

- At this stage, the input to F is $(\tilde{k}_{RSA}||k^*_{ML})$. The first component $\tilde{k}_{RSA}$ is truly random (from Game 1).
- We rely on the Dual-PRF security of F (specifically, the property that $F(k, \cdot)$ is pseudo-random if k is random).
- Even though the adversary may know k^*_{ML} (since we do not assume ML-KEM security in this specific reduction trace), the randomness of $\tilde{k}_{RSA}$ ensures the output of F is indistinguishable from random.
- *Context Binding:* The Context field includes (c^*_{RSA}, c^*_{ML}). If $\mathcal{A}$ queries a modified ciphertext (e.g., (c^*_{RSA}, c'_{ML})), the KDF input context changes, ensuring the oracle output remains independent of the challenge key.
- Thus: $|\Pr[S_1] - \Pr[S_2]| \leq \mathbf{Adv}^{\text{Dual-PRF}}_F(\mathcal{D})$.

Conclusion: In Game 2, the challenge key K_b is completely independent of the ciphertexts (it is chosen randomly in both the $b = 0$ and $b = 1$ cases). Therefore, the adversary's advantage in Game 2 is exactly 0. Combining the bounds, we obtain:

$$\mathbf{Adv}_{\Pi_c}(\mathcal{A}) \leq \mathbf{Adv}_{\Pi_{RSA}}(\mathcal{B}_1) + \mathbf{Adv}^{\text{Dual-PRF}}_F(\mathcal{D})$$

By symmetry, if we assumed ML-KEM was secure instead, we would obtain the bound based on $\mathbf{Adv}_{\Pi_{ML}}$. Since the scheme is secure if *either* holds, the advantage is bounded by the minimum of the two KEM advantages plus the PRF advantage. □

4 Implementation and Performance Evaluation

4.1 Implementation Sketch

We developed a comprehensive implementation of the RSA-KEM-ML-KEM composite scheme in Java to validate its practicality and performance characteristics. The implementation leverages the Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) frameworks, with BouncyCastle (version 1.81) providing ML-KEM support. The implementation provides:

- Complete RSA-KEM following RFC 9690 [10] with 2048, 3072, and 4096-bit key sizes
- ML-KEM implementations for all NIST-standardized parameter sets (ML-KEM-512, ML-KEM-768, ML-KEM-1024) conforming to FIPS 203 [16]
- Four composite RSA-KEM-ML-KEM combinations using HKDF-SHA256 for secure key derivation
- RSA-OAEP implementations for baseline comparison
- Integration with AES-256-GCM for practical end-to-end encryption scenarios

4.2 Experimental Methodology

Performance measurements were conducted on an AWS EC2 c7i-flex.large instance (2 vCPU, 4GB RAM) running Amazon Linux 2023, using Java 21.0.7. Each experiment was repeated 8,000 times; the first 3,000 iterations were eliminated to avoid the JVM "warm-up" phase. The 95th percentile score was calculated for each metric.

We evaluated two primary metrics:

- Pure cryptographic operation performance (encapsulation and decapsulation times)
- End-to-end encryption performance with varying payload sizes (1KB, 10KB, 100KB, 1MB)

All algorithms uniformly used SHA-256 or HKDF-SHA256 for key derivation to ensure consistency across benchmarks and maintain fair comparisons.

4.3 Performance Results

Cryptographic Operation Performance Figure 1 and Table 2 present the benchmark results for KEM-related cryptographic operations. The results reveal fundamentally different performance profiles between post-quantum and RSA-based algorithms.

For post-quantum schemes, all ML-KEM variants achieve symmetric performance, with encapsulation and decapsulation times remaining tightly coupled. For example, ML-KEM-512 achieves 0.045 ms encapsulation and 0.050 ms decapsulation. This symmetry enables very high throughput, ranging from $9,709$ *ops/sec* (ML-KEM-1024) to $20,000$ *ops/sec* (ML-KEM-512).

RSA-based algorithms, in contrast, exhibit severe asymmetry. RSA-KEM-4096 reaches an extreme $143.8\times$ ratio, with decapsulation taking 18.99 ms. Consequently, throughput is constrained to just 53 *ops/sec* (RSA-KEM-4096).

Composite RSA-KEM-ML-KEM schemes lie between these extremes. They incur higher encapsulation costs due to the sequential execution of both KEMs (e.g., 0.129 ms for RSA-2048-ML-768) but add only minimal overhead to the RSA decapsulation time. As a result, composites inherit RSA's bottlenecks. RSA-2048-ML-768 achieves 338 *ops/sec*, while the RSA-4096-ML-1024 variant matches the pure RSA-KEM-4096 throughput of roughly 52 *ops/sec*.

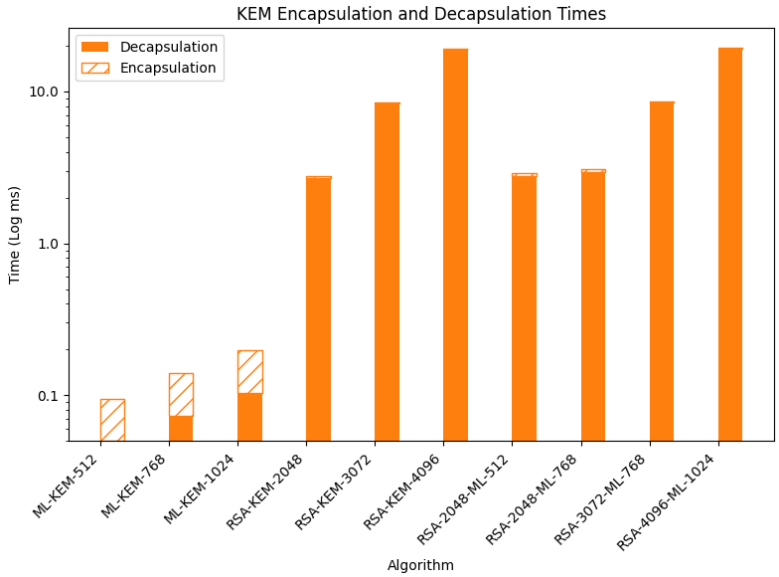


Fig. 1. Encapsulation vs. decapsulation times (log scale). ML-KEM exhibits symmetric performance, whereas RSA-based schemes show extreme asymmetry.

End-to-End Encryption Performance Table 3 extends this analysis to full encryption and decryption workloads across payloads from 1KB to 1MB.

Encryption costs tend to converge at larger payloads as the AES-256-GCM overhead amortizes the differences in key establishment. However, decryption remains dominated by RSA decapsulation regardless of payload size. For instance, RSA-4096-ML-1024 requires approximately 19.1–19.6 ms for decryption across all payload sizes, while ML-KEM variants consistently remain below 0.14 ms. This confirms that the RSA bottleneck is structural and independent of data volume.

4.4 Performance Analysis

Algorithm Performance Characteristics Our measurements reveal three distinct performance tiers with critical asymmetries:

Tier 1: ML-KEM (Symmetric Excellence) ML-KEM variants deliver exceptional throughput (9,700 – 20,000 ops/sec) with minimal asymmetry (ratio ≈1.1). Decapsulation remains consistently fast (<0.11 ms), making ML-KEM suitable for real-time, high-volume applications.

Tier 2: RSA-OAEP and RSA-KEM-2048 These algorithms offer reasonable encapsulation times (<0.09 ms) but suffer noticeable decapsulation penalties. RSA-KEM-2048 exhibits a 49.0× asymmetry (2.695 ms decapsulation), limiting throughput to roughly 371 ops/sec.

Table 2. Pure cryptographic operation times (ms). ML-KEM demonstrates symmetric performance, whereas RSA shows severe decapsulation asymmetry with significantly reduced throughput.

Algorithm	**Encap. (ms)**	**Decap. (ms)**	**Asym. Ratio**	**Encap. (ops/sec)**	**Decap. (ops/sec)**
Post-Quantum Algorithms					
ML-KEM-512	0.045	0.05	1.1	22222	20000
ML-KEM-768	0.066	0.073	1.1	15152	13699
ML-KEM-1024	0.094	0.103	1.1	10638	9709
Classical RSA Algorithms					
RSA-KEM-2048	0.055	2.695	49.0	18182	371
RSA-KEM-3072	0.09	8.34	92.7	11111	120
RSA-KEM-4096	0.132	18.987	143.8	7576	53
Composite RSA-KEM-ML-KEM Algorithms					
RSA-2048-ML-512[a]	0.105	2.783	26.5	9524	359
RSA-2048-ML-768	0.129	2.961	23.0	7752	338
RSA-3072-ML-768	0.162	8.474	52.3	6173	118
RSA-4096-ML-1024	0.236	19.112	81.0	4237	52

[a] Abbreviated from RSA-KEM-X-ML-KEM-Y for space

Tier 3: Large RSA Keys (Critical Performance Degradation) RSA-3072 and RSA-4096 are severely constrained. RSA-KEM-4096 requires nearly 19 ms per decapsulation, limiting throughput to roughly 53 ops/sec with asymmetry ratios exceeding 140×. This penalty persists across all tested payloads.

Composite Scheme Analysis Composite RSA-KEM-ML-KEM schemes combine the quantum resistance of ML-KEM with RSA's established trust infrastructure but inherit RSA's performance weaknesses. While encapsulation overhead increases (e.g., RSA-KEM-2048 rises from 0.055 ms to 0.105 ms when combined with ML-KEM-512), the decapsulation bottleneck remains the defining factor. Server-side throughput for the "Balanced Default" profile (RSA-3072-ML-768) is constrained to 118 ops/sec, a significant reduction compared to pure post-quantum alternatives.

Payload Size Impact Payload scaling affects encryption performance more significantly than decryption. For small payloads (1KB), ML-KEM-512 is significantly faster than composite schemes. However, at 1MB, the relative difference narrows as AES-256-GCM operations dominate the total execution time. Conversely, decryption performance is dictated entirely by the KEM decapsulation time, which remains constant regardless of payload size, reinforcing that the RSA decryption penalty is absolute.

Table 3. End-to-end encryption/decryption times (ms) for various payload sizes. RSA decapsulation dominates performance regardless of payload size.

	1 KB		10 KB		100 KB		1 MB	
Algorithm	**Enc.**	**Dec.**	**Enc.**	**Dec.**	**Enc.**	**Dec.**	**Enc.**	**Dec.**
Post-Quantum Algorithms								
ML-KEM-512	0.063	0.061	0.063	0.061	0.084	0.082	1.187	0.362
ML-KEM-768	0.084	0.083	0.085	0.084	0.107	0.105	1.204	0.371
ML-KEM-1024	0.115	0.116	0.118	0.118	0.136	0.136	0.326	0.317
Classical RSA Algorithms								
RSA-OAEP-2048	0.068	0.877	0.069	0.866	0.092	0.891	0.336	1.101
RSA-OAEP-3072	0.101	2.873	0.103	2.87	0.126	2.921	0.381	3.374
RSA-OAEP-4096	0.149	6.037	0.151	6.042	0.171	6.054	0.385	6.803
RSA-KEM-2048	0.071	2.705	0.071	2.672	0.091	2.717	1.189	3.016
RSA-KEM-3072	0.106	8.399	0.108	8.343	0.128	8.456	1.224	8.931
RSA-KEM-4096	0.152	19.245	0.153	19.237	0.172	19.096	0.352	19.492
Composite RSA-KEM-ML-KEM Algorithms								
RSA-2048-ML-512[a]	0.131	2.789	0.128	2.744	0.151	2.791	1.261	3.133
RSA-2048-ML-768	0.143	2.783	0.142	2.738	0.165	2.881	0.428	3.171
RSA-3072-ML-768	0.193	8.496	0.193	8.442	0.217	8.549	1.317	9.076
RSA-4096-ML-1024	0.276	19.402	0.28	19.45	0.296	19.244	0.571	19.632

[a] Abbreviated from RSA-KEM-X-ML-KEM-Y for space

Table 4. Key sizes for RSA, ML-KEM, and composite variants (in bytes). Composite schemes combine both key types, resulting in significant size overhead.

Algorithm	pk_{RSA}	sk_{RSA}	pk_{ML}	sk_{ML}	pk_c	sk_c
RSA-KEM-2048-ML-KEM-512	256	256	800	1632	1056	1888
RSA-KEM-2048-ML-KEM-768	256	256	1184	2400	1440	2656
RSA-KEM-3072-ML-KEM-768	384	384	1184	2400	1568	2784
RSA-KEM-4096-ML-KEM-1024	512	512	1568	3168	2080	3680

Key Size Comparison Table 4 highlights the substantial differences in key sizes across RSA, ML-KEM, and composite constructions. RSA public and private keys scale linearly with modulus size, growing from 256 bytes to 512 bytes. ML-KEM keys are comparatively compact, with public keys between 800 and 1568 bytes and secret keys between 1632 and 3168 bytes.

In composite constructions, key sizes are additive: for example, RSA-KEM-2048-ML-KEM-768 yields a composite public key of 1440 bytes and secret key of 2656 bytes, while RSA-KEM-4096-ML-KEM-1024 produces a 2880-byte public key and 3680-byte secret key.

From a performance perspective, larger key sizes have two primary implications. First, they increase storage and network transmission costs, particularly in bandwidth-constrained environments. Second, they elevate encapsulation and decapsulation latency by scaling RSA's computational cost with key length, as

reflected in Table 2. Together, these effects mean that while composite schemes add quantum resistance with moderate time overhead, they impose non-trivial size overheads that must be considered for protocol integration (e.g., SAML assertions, JWE headers, or OIDC tokens).

Implementation Validation The Java implementation successfully demonstrates the feasibility and performance characteristics of the RSA-KEM-ML-KEM composite construction. ML-KEM variants maintain symmetric performance profiles with minimal variation between encapsulation and decapsulation operations. RSA-based algorithms exhibit the expected computational asymmetry, with decapsulation times growing polynomially with key size. The composite constructions successfully combine both algorithms while preserving their individual performance characteristics, validating the compositional approach to quantum-resistant cryptography. These results provide a clear performance baseline for system architects evaluating post-quantum migration strategies and demonstrate that composite schemes can be efficiently implemented using existing cryptographic libraries.

5 Migration Strategy and Protocol Validation

This section presents a phased migration strategy for transitioning RSA-OAEP encryption in Identity and Access Management (IAM) protocols to quantum-resistant alternatives. We focus on three widely deployed protocols, JSON Web Encryption (JWE), SAML 2.0, and OpenID Connect (OIDC), where RSA-OAEP is currently used for key encryption and data protection. We demonstrate how RSA-KEM-ML-KEM serves as a practical intermediary step toward pure ML-KEM adoption.

5.1 Three-Phase Migration Framework

Our migration framework recognizes that organizations cannot immediately replace RSA-OAEP encryption infrastructure with pure ML-KEM due to operational constraints. Instead, we propose a staged approach:

1. **Phase 1 - Legacy Operation:** Current RSA-OAEP encryption deployments in JWE, SAML, and OIDC
2. **Phase 2 - Composite Transition:** HPKE with RSA-KEM-ML-KEM for encryption as intermediate step
3. **Phase 3 - Target Architecture:** Pure ML-KEM encryption with HPKE

The composite phase (Phase 2) provides quantum resistance for encryption operations while allowing gradual infrastructure updates, making it a critical stepping stone rather than a permanent solution.

5.2 Protocol-Specific Migration Pathways

JSON Web Encryption (JWE)

Current State (Phase 1): JWE currently relies on RSA-OAEP for key encryption, specified through the "alg" header parameter (e.g., `RSA-OAEP-256`). Organizations have established JWKS endpoints distributing RSA public keys.

Phase 2 Integration: Organizations maintain their existing RSA public keys and add separate ML-KEM public keys:

- Publish **separate** RSA and ML-KEM public keys in JWKS (no composite key pair generation required)
- Define new algorithm identifiers (e.g., `HPKE-RSA-2048-ML-768`)
- At encryption time, senders use both keys to perform composite encapsulation
- Encode HPKE encapsulated ciphertext in JWE's encrypted key field
- Enable content negotiation: HPKE-capable clients use composite encryption; legacy clients fall back to RSA-OAEP

Challenges:

- **Header Size:** Composite ciphertexts increase from 256 bytes to 1,344 bytes (RSA-2048-ML-768)
- **Compact Serialization:** Base64url encoding may impact HTTP header size limits
- **Algorithm Negotiation:** JWKS must clearly indicate HPKE support for client capability detection

Phase 3 Target: Pure ML-KEM-768 encryption via HPKE reduces ciphertext to 1,088 bytes and achieves 13,699 ops/sec.

SAML 2.0

Current State (Phase 1): SAML assertions use XML Encryption with RSA-OAEP for key transport. Identity Providers (IdPs) publish RSA certificates in metadata.

Phase 2 Integration:

- Extend `<KeyDescriptor>` elements in SAML metadata to include ML-KEM public keys alongside existing RSA keys
- IdPs provision ML-KEM keys—no RSA key regeneration required
- Service Providers (SPs) query metadata to determine HPKE encryption support
- Assertions encrypted with HPKE indicate this via `<EncryptionMethod>` algorithm URI
- Maintain backward compatibility with legacy RSA-OAEP for SPs that have not upgraded

Challenges:

- **Synchronous Flow Latency:** RSA-KEM-2048-ML-KEM-768 adds 2.961 ms decapsulation time versus 0.877 ms for RSA-OAEP-2048, impacting authentication experience
- **Federation Coordination:** Multi-party federations require coordinated upgrades; Phase 2 enables gradual rollout
- **Assertion Size:** Encrypted assertions grow due to larger composite ciphertexts
- **Infrastructure Scaling:** Federations processing 1000 assertions/second require 3× computational capacity in Phase 2

Phase 3 Target: Pure ML-KEM reduces decryption latency to 0.073 ms, enabling infrastructure consolidation.

OpenID Connect (OIDC)

Current State (Phase 1): OIDC uses JWE (with RSA-OAEP) for encrypting ID tokens and authorization responses. Relying Parties discover encryption keys via JWKS endpoints.

Phase 2 Integration:

- Authorization Servers (AS) publish RSA and ML-KEM public keys separately in JWKS
- Use `alg` parameter for capability negotiation
- Encrypt sensitive tokens with HPKE composite encryption
- Maintain RSA-OAEP encryption for backward compatibility
- Existing RSA keys continue to function—only ML-KEM keys need to be added

Challenges:

- **Implicit Flow URL Limits:** RSA-KEM-2048-ML-KEM-768 produces 1,344-byte ciphertexts versus 256 bytes for RSA-OAEP-2048 (5.25× expansion)
- **Browser Constraints:** Some browsers limit URL lengths to 2048 bytes, which may be exceeded
- **Recommended Approach:** Deploy Phase 2 HPKE first for server-to-server flows (token introspection, UserInfo endpoints); migrate user-facing flows to authorization code flow

Phase 3 Target: Pure ML-KEM reduces ciphertext to 1,088 bytes, eliminating URL limit concerns.

5.3 Comparison with Direct ML-KEM Migration

Organizations might consider skipping Phase 2 entirely for encryption. We analyze this alternative:

Direct Migration (Phase 1 → Phase 3):

Advantages:

- Avoids Phase 2 performance penalty for encryption (338 vs. 13,699 ops/sec)
- Simpler long-term encryption architecture
- Lower ciphertext overhead

Disadvantages:

- Requires replacing entire key provisioning infrastructure simultaneously
- No fallback to classical security for encryption during transition
- Higher operational risk (no gradual rollout)
- Forces legacy encryption endpoints offline immediately

Composite Transition (Phase 1 → Phase 2 → Phase 3):

Advantages:

- Preserves RSA key material during encryption transition
- Maintains both classical and quantum security for encryption
- Enables gradual endpoint migration
- Lower operational risk

Disadvantages:

- Requires managing three cryptographic profiles for encryption over migration period
- Performance penalty in Phase 2 may require infrastructure scaling
- Increased ciphertext size in Phase 2

Decision Guidance: Organizations should choose direct encryption migration if they:

- Have minimal RSA encryption infrastructure investment
- Can coordinate simultaneous endpoint upgrades
- Prioritize performance over migration risk

Organizations should use phased composite encryption migration if they:

- Have extensive RSA-OAEP encryption deployments across federated systems
- Require gradual rollout with fallback capability
- Need to maintain both classical and quantum security for encryption during transition

5.4 Migration Timeline Recommendations

Based on our performance analysis, we recommend for encryption migration:

1. **Immediate (0-6 months):** Begin Phase 2 planning; deploy test HPKE encryption endpoints
2. **Short-term (6-18 months):** Full Phase 2 deployment for quantum-sensitive encrypted data; accept performance penalty; scale infrastructure accordingly
3. **Medium-term (18-36 months):** Monitor client adoption; prepare Phase 3 infrastructure (pure ML-KEM key management systems)
4. **Long-term (36+ months):** Transition to Phase 3 when adoption criteria met; deprecate Phase 2 and Phase 1 encryption endpoints

The composite phase (Phase 2) should be viewed as a **temporary but necessary** stepping stone for encryption, not a permanent solution. While the underlying RSA-KEM-ML-KEM construction instantiates a Split-Key PRF combiner, our contribution lies in the formalization and quantitative analysis of this construction as a bridge for legacy systems. By identifying and solving protocol-specific constraints—such as URL limits in OIDC and throughput bottlenecks in SAML—we provide a practical roadmap that generic KEM theory alone does not address.

6 Conclusion

This paper has presented a comprehensive migration strategy for transitioning RSA-OAEP encryption to post-quantum cryptography. The phased migration framework we present, moving from RSA-OAEP encryption (Phase 1) through the RSA-KEM-ML-KEM composite transition (Phase 2) to pure ML-KEM (Phase 3), balances theoretical security with operational reality.

Our performance evaluation reveals the fundamental trade-off organizations face during encryption migration. The composite RSA-KEM-2048-ML-KEM-768 construction achieves only 338 operations per second compared to 13,699 operations per second for pure ML-KEM-768—a 40-fold difference that establishes clear quantitative criteria for migration planning. While this performance penalty is significant, the composite phase enables organizations to achieve quantum resistant encryption immediately while maintaining RSA key material and infrastructure.

The security analysis demonstrates that our construction achieves IND-CCA2 security through a Split-Key PRF combiner, ensuring protection against both classical and quantum adversaries in the standard model. This robustness is essential for enterprises that cannot immediately abandon classical threat models while preparing for quantum computing advances.

Our analysis of JWE, SAML, and OIDC encryption integration demonstrates that protocol-specific considerations significantly impact migration timelines. SAML's latency sensitivity, OIDC's ciphertext size constraints, and JWE's negotiation capabilities each require tailored approaches. We have shown that the

migration of RSA-OAEP is not merely a matter of choosing a generic combiner, but of managing the interaction between cryptographic expansion and protocol-level limits.

RSA-KEM-ML-KEM composite schemes should be adopted as a transition mechanism for encryption, not a destination. Organizations should:

1. Begin Phase 2 immediately to establish quantum-resistant encryption, leveraging the ability to reuse existing RSA keys.
2. Provision ML-KEM keys separately and publish them alongside existing RSA keys in JWKS/metadata.
3. Scale infrastructure to accommodate the throughput reduction for encryption operations.
4. Plan Phase 3 transition as client adoption reaches 90% and browser/protocol limits permit pure PQC deployment.

As quantum computing capabilities advance, maintaining a clear migration trajectory with quantifiable milestones is essential. This work provides the cryptographic construction, empirical performance data, and protocol-level insights necessary for organizations to execute responsible post-quantum encryption migration while preserving operational continuity. The composite phase serves as an indispensable bridge, enabling quantum-resistant encryption today while preparing for the pure post-quantum future.

Acknowledgments. Portions of this paper's text were rephrased with the help of generative AI tools (Claude 4.0, Gemini 1.5). Specifically, they were used to generate the paper structure and initial paragraph drafts. All algorithms, pseudocode, figures, benchmarks, and performance data were created and verified by the authors, who accept full responsibility for the correctness and integrity of the work.

References

1. Anastasova, M., Kampanakis, P., Massimo, J.: PQ-HPKE: post-quantum hybrid public key encryption. Cryptology ePrint Archive (2022), https://eprint.iacr.org/2022/414
2. Backendal, M., Bellare, M., Günther, F., Scarlata, M.: When Messages are Keys: Is HMAC a dual-PRF? In: Advances in Cryptology - CRYPTO 2023. pp. 661–693 (2023). https://doi.org/10.1007/978-3-031-38548-3_22
3. Barbosa, M., Connolly, D., Duarte, J.D., Kaiser, A., Schwabe, P., Varner, K., Westerbaan, B.: X-wing: The hybrid KEM you've been looking for. IACR Communications in Cryptology (2024). https://doi.org/10.62056/a3qj89n4e
4. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.A.: Hybrid Public Key Encryption. RFC 9180 (Feb 2022). https://doi.org/10.17487/RFC9180
5. Bernstein, D.J., Lange, T.: eBACS: ECRYPT Benchmarking of Cryptographic Systems. https://bench.cr.yp.to (2026), accessed: 2026-05-31
6. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. In: Post-Quantum Cryptography. pp. 206–226. Springer (2019). https://doi.org/10.1007/978-3-030-25510-7_12

7. Campagna, M., Petcher, A.: Security of hybrid key encapsulation. Cryptology ePrint Archive, Paper 2020/1364 (2020), https://eprint.iacr.org/2020/1364
8. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1145/3549993.3550007
9. Giacon, F., Heuer, F., Poettering, B.: KEM combiners. In: Public-Key Cryptography – PKC 2018. vol. 10769, pp. 190–218. Springer (2018). https://doi.org/10.1007/978-3-319-76578-5_7
10. Housley, R., Turner, S.: Use of the RSA-KEM Algorithm in the Cryptographic Message Syntax (CMS). RFC 9690 (Feb 2025). https://doi.org/10.17487/RFC9690, https://www.rfc-editor.org/info/rfc9690
11. Jones, M.B., Hildebrand, J.: JSON Web Encryption (JWE). RFC 7516 (05 2015). https://doi.org/10.17487/RFC7516, https://www.rfc-editor.org/info/rfc7516
12. Krawczyk, H., Eronen, P.: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869 (May 2010). https://doi.org/10.17487/RFC5869, https://www.rfc-editor.org/info/rfc5869
13. Kwiatkowski, K., Kampanakis, P.: Post-quantum hybrid ECDHE-Kyber Key Agreement for TLSv1.3. Internet-Draft draft-kwiatkowski-tls-ecdhe-kyber-01, Internet Engineering Task Force (05 2023), https://datatracker.ietf.org/doc/draft-kwiatkowski-tls-ecdhe-kyber/01/, work in Progress
14. Moriarty, K., Kaliski, B., Jonsson, J., Rusch, A.: PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017 (Nov 2016). https://doi.org/10.17487/RFC8017
15. National Institute of Standards and Technology: NIST Announces First Four Quantum-Resistant Cryptographic Algorithms (07 2022), https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms
16. National Institute of Standards and Technology: Module-Lattice-Based Key-Encapsulation Mechanism Standard (2024). https://doi.org/10.6028/NIST.FIPS.203
17. National Institute of Standards and Technology: Post-Quantum Cryptography Standardization (2025), https://csrc.nist.gov/pqc-standardization
18. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342
19. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review **41**(2), 303–332 (1999). https://doi.org/10.1137/S0036144598347011
20. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive (2001), https://eprint.iacr.org/2001/112