IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

A Comparative Study of LWE-Based Post-Quantum Key Encapsulation and Traditional RSA

Rutuja Paraji Zaware[1], Miss. More N. N.[2]

Student, Department of MCA Engineering, Samarth College of Engineering and Management, Belhe, Pune, India[1]

Guide, Department of MCA Engineering, Samarth College of Engineering and Management, Belhe, Pune, India[2]

Abstract: Public-key key establishment protects almost all secure communication on the internet, and RSA remains one of its most widely deployed mechanisms. RSA's security rests on the difficulty of integer factorisation, a problem that Shor's quantum algorithm solves in polynomial time on a sufficiently large quantum computer. In response, NIST standardised the lattice-based ML-KEM (FIPS 203), whose security is derived from the Learning With Errors (LWE) problem. This paper presents a comparative study of LWE-based post-quantum key encapsulation and traditional RSA. The study compares the two families on security foundation, quantum resistance, key and ciphertext sizes, and computational cost. To support the comparison with measured data, a Module-LWE key encapsulation mechanism following the ML-KEM-512, ML-KEM-768 and ML-KEM-1024 parameter sets was implemented and benchmarked against RSA-2048, RSA-3072 and RSA-4096 with OAEP padding on the same machine; the study also measures decryption noise margins and end-to-end hybrid encryption. In the experiments, all 900 LWE round trips decapsulated correctly and tampered ciphertexts were rejected. RSA-2048 public keys and ciphertexts were about 2.7 and 3.0 times smaller than those of ML-KEM-512, and RSA encapsulation was far faster; however, RSA key generation was roughly 40 to 300 times slower than LWE key generation, even though the LWE code was an unoptimised Python prototype. The results indicate that LWE-based schemes trade larger messages for quantum resistance and fast key generation, and support a gradual, hybrid migration from RSA.

Keywords: Learning With Errors (LWE), post-quantum cryptography, ML-KEM, RSA, Module-LWE, key encapsulation, quantum resistance

I. INTRODUCTION

Key establishment allows two parties that have never shared a secret to agree on a symmetric key for bulk encryption. For more than four decades this task has been performed largely by RSA, whose security depends on the difficulty of factoring the product of two large primes [1]. Classical computers have no known efficient algorithm for this problem at current key sizes, so RSA-2048 and RSA-3072 are considered secure against classical attackers.

Quantum computing changes this assumption. Shor's algorithm factors integers and computes discrete logarithms in polynomial time on a quantum computer [2], which would break RSA at any key size. Although a cryptographically relevant quantum computer does not yet exist, resource estimates for breaking RSA-2048 have fallen steadily [7], [8]. Encrypted traffic recorded today can be stored and decrypted later, a threat often called "harvest now, decrypt later", so data that must remain confidential for many years is already exposed.

Post-quantum cryptography addresses this threat with problems that are not known to be easy for quantum computers. Among them, the Learning With Errors (LWE) problem introduced by Regev [3] has become the leading foundation, and the Module-LWE based scheme CRYSTALS-Kyber [5] was standardised by NIST as ML-KEM in FIPS 203 [6]. NIST's draft transition guidance proposes that RSA at the 112-bit security level be deprecated after 2030 and that quantum-vulnerable public-key algorithms be disallowed after 2035 [11].

This paper compares LWE-based key encapsulation with traditional RSA. Unlike a purely theoretical comparison, it also reports measurements from a working Module-LWE KEM prototype and from RSA-OAEP on the same machine. The study evaluates security basis, correctness, key and ciphertext sizes, and the time taken by key generation, encapsulation, and decapsulation, and then discusses which scheme is suitable in which setting.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510118

Contributions

1. A structured comparison of RSA and LWE-based key encapsulation across security basis, quantum exposure, message size, and computational cost.
2. A reproducible Module-LWE KEM prototype for the three ML-KEM parameter sets, validated against the standard's key and ciphertext sizes.
3. Measured data for RSA-2048, RSA-3072 and RSA-4096 and for ML-KEM-512, -768 and -1024, including key generation, encapsulation and decapsulation times.
4. An empirical examination of the decryption noise margin of Module-LWE, which shows directly why correct decryption is overwhelmingly likely.
5. An end-to-end hybrid encryption test (KEM plus AES-256-GCM) and a transfer-time analysis that relates message sizes to link speed.

II. LITERATURE REVIEW

RSA [1] is standardised in PKCS #1 [9], which specifies OAEP padding, and NIST SP 800-56B [10] describes RSA-based key establishment. Its advantages are maturity, a small public exponent that makes the public operation cheap, and decades of cryptanalysis. Gidney and Ekerå [7] estimated that RSA-2048 could be factored in about eight hours with roughly twenty million noisy qubits, and Gidney [8] later reported an estimate below one million. These are estimates, not demonstrations, but they explain why migration planning has begun early. Peikert's survey [4] reviews how LWE yields encryption and key-exchange schemes, and Bos et al. [5] proposed CRYSTALS-Kyber, standardised by NIST as ML-KEM in FIPS 203 [6] with parameter sets ML-KEM-512, -768 and -1024 for NIST categories 1, 3 and 5.

A. The RSA Cryptosystem and Its Known Attacks

In RSA, a user chooses two large primes p and q, publishes n = pq and a public exponent e (commonly 65537), and keeps the private exponent d with ed ≡ 1 modulo (p−1)(q−1). Encryption computes c = m^e mod n and decryption m = c^d mod n. Because e is small, the public operation is cheap, whereas the private operation needs a full-length exponentiation, usually accelerated with the Chinese Remainder Theorem. Textbook RSA is deterministic and malleable, so practical use requires randomised padding such as OAEP [14], specified in PKCS #1 [9].

Boneh's survey of twenty years of attacks on RSA [15] concludes that the algorithm has resisted attack with proper padding and large random primes; successful attacks exploited poor implementation, weak parameters, or side channels. This long record is a genuine strength of RSA.

B. The Quantum Threat

Shor's algorithm [2] breaks RSA by finding the period of a modular function, which reveals the factors of n. Its running time is polynomial in key length, so enlarging the RSA modulus does not restore security. Grover's algorithm [16] gives only a quadratic speed-up for unstructured search, which is why AES-256 is regarded as adequate against quantum attackers and why KEM shared secrets are normally 256 bits.

Mosca's argument [17] explains the urgency: if the time data must stay secret plus the migration time exceeds the time until a large quantum computer exists, the data is already at risk.

C. Lattice Cryptography and the LWE Problem

In the LWE problem, introduced by Regev [3], a secret vector s is chosen modulo a prime q, and an attacker sees pairs (a, b) with a uniformly random and b = ⟨a, s⟩ + e mod q, where e is a small error. Without the error, s follows by Gaussian elimination; with it, recovering s is believed to be hard even for quantum computers, and Regev proved it as hard as certain worst-case lattice problems. Concrete hardness estimates are given by Albrecht et al. [20].

Ring-LWE [18] replaces matrices by polynomial-ring multiplication to shrink keys, and Module-LWE [19] uses small matrices of ring elements. ML-KEM uses Module-LWE with degree-256 polynomials and modulus 3329; the security level is changed through the module rank k (2, 3 or 4).

D. Key Encapsulation and CCA Security

Basic LWE encryption is secure only against passive attackers. The Fujisaki–Okamoto transform [12], analysed modularly by Hofheinz et al. [21], yields a chosen-ciphertext secure KEM by re-encrypting the recovered message during decapsulation and comparing it with the received ciphertext. ML-KEM uses implicit rejection: on a mismatch a pseudorandom key is returned instead of an error, so failures reveal nothing.

The reviewed work shows that RSA is efficient and well understood but quantum-vulnerable, that LWE-based schemes are a plausible quantum-resistant alternative, and that standards bodies are setting migration timelines [11]. What is often missing is a side-by-side, reproducible comparison of sizes and timings measured the same way for both families; this paper addresses that gap at small scale.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

III. PROBLEM STATEMENT

Organisations relying on RSA for key establishment must decide when to move to post-quantum alternatives. LWE-based schemes differ from RSA in practice: keys and ciphertexts are larger, the arithmetic differs, and the schemes are newer. Premature migration wastes bandwidth, while delayed migration leaves long-lived data exposed.

Research Questions

1. **RQ1:** How do RSA and LWE-based KEMs differ in the underlying hard problem and in exposure to quantum attack?
2. **RQ2:** How do public key, private key, and ciphertext sizes compare at corresponding security levels?
3. **RQ3:** How do key generation, encapsulation, and decapsulation costs compare, and how do they grow with the security parameter?
4. **RQ4:** How large is the safety margin against decryption failure in Module-LWE, and what does it imply for reliability?
5. **RQ5:** What are the practical bandwidth and end-to-end costs of replacing RSA key transport with ML-KEM in a hybrid encryption setting?

IV. PROPOSED COMPARATIVE FRAMEWORK

The proposed framework compares the two families as key encapsulation mechanisms (KEMs). A KEM consists of three algorithms: key generation produces a public key and a private key; encapsulation uses the public key to produce a ciphertext and a fresh shared secret; and decapsulation uses the private key to recover the same shared secret from the ciphertext. The shared secret then keys a symmetric cipher such as AES-256-GCM for the actual data.

For RSA, encapsulation is implemented as RSA-OAEP encryption (SHA-256, MGF1-SHA-256) of a random 32-byte secret, which is the usual key-transport form. For the LWE family, encapsulation follows the Module-LWE construction of ML-KEM. Both are evaluated through the same three operations and the same metrics.

Threat Model

Two attackers are considered. A classical attacker may observe, record and modify network messages. A quantum attacker additionally has a large fault-tolerant quantum computer, possibly years after traffic was recorded. The goal is to recover the shared secret from the public key and ciphertext. Side channels, denial of service, and public-key authentication are outside the scope.

V. SYSTEM ARCHITECTURE

Fig. 1 shows the generic KEM workflow used for both schemes. The receiver generates a key pair and publishes the public key. The sender encapsulates against that key, obtaining a ciphertext and a shared secret K, and returns the ciphertext. The receiver decapsulates to obtain the same K. Both sides then use K with an authenticated symmetric cipher to protect the data.

System Components

1. **Key Generation:** For RSA, two large primes are generated and multiplied to form the modulus. For ML-KEM, a public matrix is expanded from a seed, small secret and error vectors are sampled, and the public key is the noisy product plus the seed.
2. **Encapsulation:** For RSA, a random secret is padded with OAEP and raised to the public exponent. For ML-KEM, a random message is encrypted with fresh small noise and the secret is derived from it by hashing.
3. **Decapsulation:** For RSA, one private-key modular exponentiation recovers the secret. For ML-KEM, the message is recovered by rounding, re-encrypted, and compared with the received ciphertext; on mismatch a pseudorandom value is returned (implicit rejection).
4. **Key Derivation and Symmetric Encryption:** The shared secret is used with AES-256-GCM for bulk data.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

Working Principle

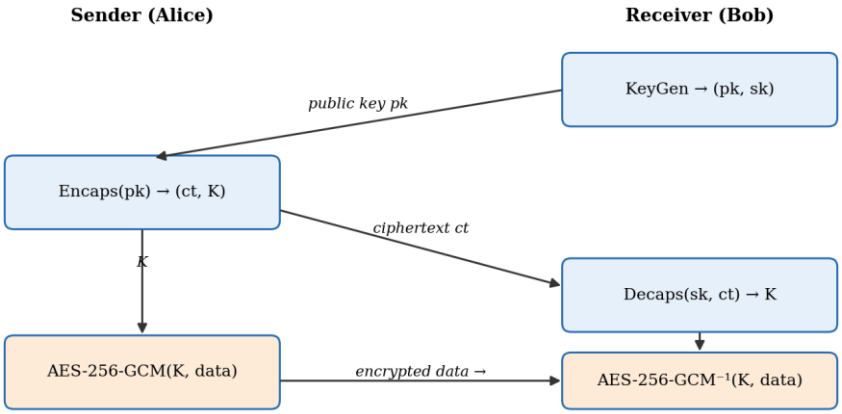


Fig. 1. Generic KEM workflow used for the comparison

VI. RESEARCH METHODOLOGY

1. **Algorithm Selection:** RSA-2048 and RSA-3072 (public exponent 65537, OAEP with SHA-256) represent the classical scheme, at approximately 112-bit and 128-bit classical security [13]. ML-KEM-512, ML-KEM-768 and ML-KEM-1024 represent the LWE family, corresponding to NIST categories 1, 3 and 5 [6].
2. **Correctness Testing:** For each ML-KEM parameter set, 300 independent key generation, encapsulation and decapsulation round trips were run, and the recovered secret was compared with the encapsulated one. A ciphertext with one flipped bit was also decapsulated to confirm that the true secret is not returned.
3. **Size Measurement:** Public key, private key and ciphertext lengths were taken from the byte strings produced by each implementation (DER encoding for RSA keys).
4. **Timing Measurement:** Each operation was timed with a monotonic high-resolution clock. The median is reported: 100 repetitions for each ML-KEM operation, 30 (RSA-2048) and 10 (RSA-3072) for RSA key generation, 500 for RSA encapsulation, and 200 for RSA decapsulation. Fewer RSA key generations were used because the cost per run is high and varies randomly with prime search.
5. **Analysis:** Results were compared as ratios between schemes and between parameter levels, with attention to what can and cannot be concluded given the difference in implementation maturity.

Measurement Protocol

Noise Margin Experiment

To examine reliability directly, the decryption step was instrumented. For 300 random encryptions at each ML-KEM level, the value $w = v - s^{T}u$ was computed and the distance of every one of its 256 coefficients from the ideal value (0 or $\lfloor q/2 \rfloor$) was recorded. The largest distance in each trial was kept. Decryption fails only if some coefficient distance exceeds about $q/4 = 832$, so comparing the observed maxima with this threshold shows the margin of safety.

End-to-End Hybrid Encryption Experiment

In practice a KEM is used to key a symmetric cipher. The full workflow was therefore timed for payloads of 1 KB and 1 MB: encapsulate, encrypt with AES-256-GCM [22], decapsulate, and decrypt. The workflow was run once with RSA-2048 OAEP as the key transport and once with ML-KEM-768.

Transfer-Time Calculation

The time needed to send the public key and ciphertext was calculated from the measured sizes as bits divided by link rate, for rates of 250 kbit/s (low-power wireless), 1 Mbit/s, 10 Mbit/s, and 100 Mbit/s. This calculation ignores protocol headers, retransmission, and latency, and so shows only the serialisation delay.

VII. IMPLEMENTATION

The experiments were run on a single core of an Intel Xeon 2.80 GHz virtual machine under Linux with Python 3.12.3, NumPy 2.4.4 and the *cryptography* library 46.0.6 (OpenSSL backend).

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.1510118

1. **RSA Module:** Key generation, OAEP encryption and decryption use the *cryptography* library, which calls optimised C code in OpenSSL.
2. **Module-LWE Module:** A prototype was written in Python with NumPy. It uses polynomials of degree 256 with modulus q = 3329 and the ML-KEM parameters (k, η_1, η_2, du, dv) of (2, 3, 2, 10, 4), (3, 2, 2, 10, 4) and (4, 2, 2, 11, 5) for the three levels. Matrix expansion and noise sampling use SHAKE-256, and hashing uses SHA3. Polynomial products are computed by direct negacyclic convolution rather than the Number Theoretic Transform.
3. **CCA Transform:** A Fujisaki–Okamoto style transform with implicit rejection turns the underlying encryption scheme into a KEM, as in ML-KEM.
4. **Size Check:** The resulting public key, private key and ciphertext sizes (800/1632/768, 1184/2400/1088 and 1568/3168/1568 bytes) match those listed for ML-KEM in FIPS 203 [6].

Reproducibility

Benchmark scripts run all experiments from a fixed code base and write raw results to a data file from which the tables and figures were generated. Absolute timings depend on hardware, but relationships between the schemes should be similar.

The LWE prototype is intended for measurement and teaching. It is not bit-for-bit compatible with FIPS 203, does not use constant-time operations, and must not be used to protect real data. Production systems should use a validated library.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

A. Experimental Results

Correctness. All 300 round trips succeeded for each of ML-KEM-512, ML-KEM-768 and ML-KEM-1024 (900 of 900), and decapsulation of a tampered ciphertext did not return the encapsulated secret. For RSA-OAEP, the recovered secret matched in the verification step.

Sizes. Fig. 3 shows the measured sizes. RSA public keys and ciphertexts are the smallest: 294 and 256 bytes for RSA-2048, 422 and 384 bytes for RSA-3072, and 550 and 512 bytes for RSA-4096. ML-KEM-512 has a public key of 800 bytes and a ciphertext of 768 bytes, about 2.7 and 3.0 times those of RSA-2048; ML-KEM-768 uses 1184 and 1088 bytes, and ML-KEM-1024 uses 1568 and 1568 bytes. Private keys are 1632, 2400 and 3168 bytes for the three ML-KEM sets, against 1217 bytes (RSA-2048) and 1794 bytes (RSA-3072). The total exchanged per key establishment (public key plus ciphertext) is 550 bytes for RSA-2048 and 806 for RSA-3072, against 1568, 2272 and 3136 bytes for ML-KEM-512, -768 and -1024.

Timing. Table I and Fig. 4 give the median times. RSA key generation took about 60 ms for RSA-2048 and 148 ms for RSA-3072, compared with 0.5, 0.96 and 1.5 ms for the three ML-KEM sets. RSA encapsulation is extremely cheap (0.028 ms for RSA-2048, 0.048 ms for RSA-3072) because the public exponent is small. RSA decapsulation took 0.63 ms (RSA-2048) and 1.96 ms (RSA-3072), of the same order as the 1.1 to 2.5 ms measured for ML-KEM decapsulation; ML-KEM encapsulation took 0.84 to 2.0 ms.

TABLE I. MEDIAN OPERATION TIME (MILLISECONDS)

Scheme	Key generation	Encapsulation	Decapsulation
RSA-2048	60.39	0.028	0.627
RSA-3072	147.90	0.048	1.959
ML-KEM-512	0.50	0.842	1.143
ML-KEM-768	0.96	1.778	1.785
ML-KEM-1024	1.52	2.003	2.506

The RSA timings come from optimised OpenSSL code, whereas the ML-KEM timings come from an unoptimised Python and NumPy prototype without the NTT. The ML-KEM figures are therefore pessimistic and the ratios must not be read as a speed ranking. Repeated runs gave ML-KEM-768 encapsulation medians between about 1.4 and 1.8 ms.

Scaling to RSA-4096. RSA-4096 was added to examine how cost grows with modulus length. Its public key is 550 bytes and its ciphertext is 512 bytes. The median key generation time was about 379 ms, encapsulation 0.076 ms, and decapsulation 4.22 ms. Relative to RSA-2048, key generation grew by a factor of about 6.3 and decapsulation by about

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

6.7. This steep growth reflects the cubic cost of modular exponentiation with respect to modulus length. RSA-4096 does not provide protection against a quantum attacker, since Shor's algorithm applies at any length.

B. Decryption Noise Margin

Fig. 2 shows the distribution of the largest decryption noise observed in each of the 300 trials per level. The average largest noise was about 216 for ML-KEM-512, 209 for ML-KEM-768 and 173 for ML-KEM-1024, and the largest single observation was 290, 279 and 247 respectively. The failure threshold is about 832, so the worst observed noise used roughly 30 to 35 percent of the available margin. The margin therefore comfortably exceeds what was seen in 76,800 coefficients per level. This experiment cannot verify the extremely small theoretical failure probabilities of 2^{-139} and lower, which are derived analytically [6] and cannot be observed by sampling, but it confirms that the practical behaviour is consistent with a wide margin.

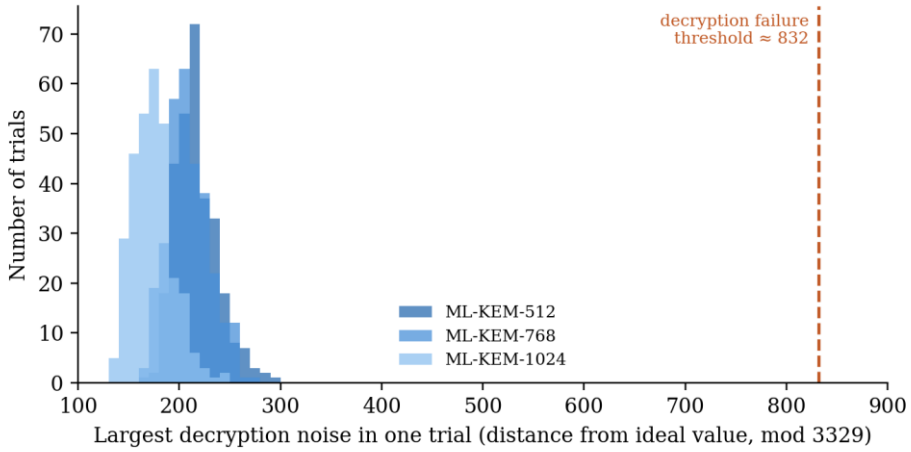


Fig. 2. Distribution of the largest decryption noise per trial

C. End-to-End Hybrid Encryption

The workflow includes encapsulation, symmetric encryption, decapsulation, and symmetric decryption. For a 1 KB payload the median time was 0.70 ms with RSA-2048 and 3.24 ms with ML-KEM-768 (AES-GCM alone: 0.004 ms), so key establishment dominates and the prototype is about 4.6 times slower. For 1 MB the times were 2.20 ms and 5.08 ms (AES-GCM alone: 0.36 ms), a difference of about 2.3 times, because the symmetric work is common to both. As payloads grow, the choice of KEM matters progressively less for computation time, though the comparison is again affected by implementation maturity.

D. Transfer-Time Analysis

Sending the public key and ciphertext takes 17.6 ms (RSA-2048), 25.8 ms (RSA-3072) and 34.0 ms (RSA-4096) at 250 kbit/s, against 50.2, 72.7 and 100.4 ms for ML-KEM-512, -768 and -1024; at 100 Mbit/s all of these are below 0.3 ms. On a fast network the extra bytes of ML-KEM are negligible compared with typical round-trip latency. On a low-rate link such as 250 kbit/s, ML-KEM-768 adds about 55 ms of serialisation delay over RSA-2048 per key establishment, which can matter for battery-powered devices. In TLS, messages above the initial congestion window may also need an extra round trip, an effect not measured here.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

E. Graphical Analysis

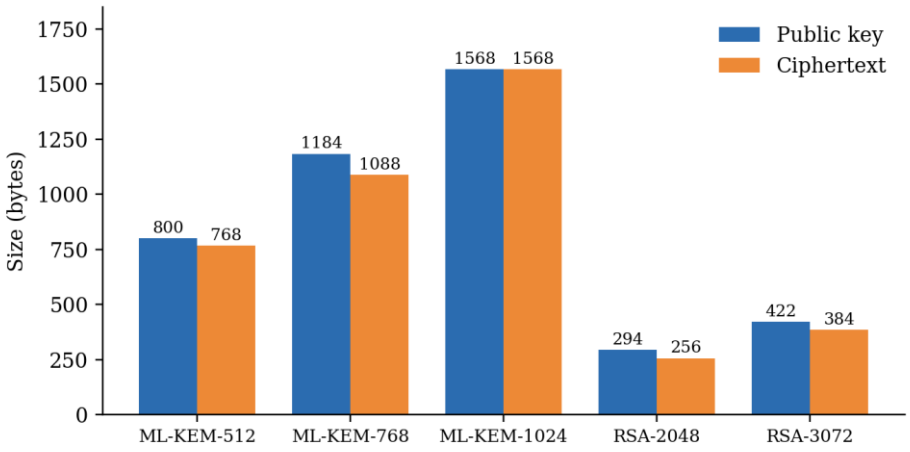


Fig. 3. Public key and ciphertext sizes (bytes)

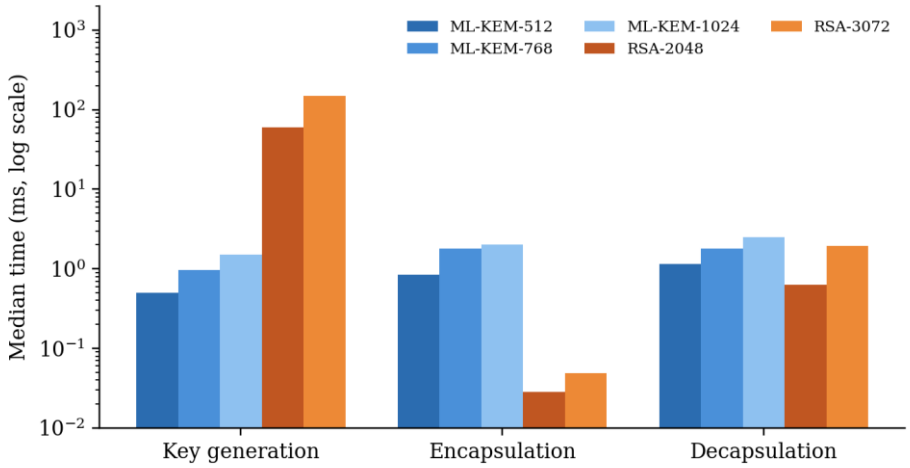


Fig. 4. Median operation time on a logarithmic scale

Fig. 3 shows that message size is the clearest cost of moving to LWE: sizes grow from a few hundred bytes to over a kilobyte per message. Fig. 4 uses a logarithmic axis because RSA key generation is more than three orders of magnitude slower than RSA encapsulation, while ML-KEM operations are all of similar magnitude.

F. Discussion

Three findings follow from the measurements. First, RSA is more compact: for the same number of operations it sends fewer bytes, which matters on constrained or high-latency links. Second, the cost profile differs. RSA's public operation is cheap and its private operation and key generation are expensive, whereas the three ML-KEM operations have similar costs. Even in an unoptimised prototype, ML-KEM key generation was about 40 to 120 times faster than RSA-2048 key generation, which is relevant to systems that create fresh keys often. Third, scaling differs: moving from RSA-2048 to RSA-3072 multiplied decapsulation time by about 3.1, while moving from ML-KEM-512 to ML-KEM-1024 multiplied it by about 2.2, although these steps do not correspond to identical increases in security.

Performance is not the main reason to migrate. RSA-based key establishment is vulnerable to Shor's algorithm at any key size [2], while no efficient quantum algorithm against Module-LWE is known. This statement concerns current knowledge: LWE-based schemes are younger than RSA, and their security rests on continuing cryptanalysis. For this reason NIST's draft guidance permits hybrid constructions that combine a classical and a post-quantum algorithm [11], so that security holds if either component remains unbroken.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.1510118

G. Security Level Comparison

NIST defines categories 1, 3 and 5 for post-quantum schemes by comparison with the cost of key search against AES-128, AES-192 and AES-256 [6], and ML-KEM-512, -768 and -1024 correspond to these categories. RSA-2048 and RSA-3072 are assigned classical strengths of 112 and 128 bits [13], and RSA-4096 somewhat more. These strengths describe resistance to conventional factoring only: against a quantum attacker, Shor's algorithm breaks RSA at every size, whereas no efficient quantum attack on ML-KEM is known.

"No known attack" describes the state of cryptanalysis, not a proof; ML-KEM's security rests on the assumed hardness of Module-LWE and on correct implementation.

H. Hybrid Key Establishment

Because lattice schemes are comparatively new, many deployment plans combine a classical key exchange with ML-KEM and derive the key from both secrets, so security holds unless both fail. The added cost of such a hybrid is dominated by the larger ML-KEM messages (roughly one to three kilobytes) rather than by computation. NIST's draft guidance allows this approach [11].

I. Qualitative Comparison

TABLE II. SUMMARY COMPARISON OF RSA AND LWE-BASED KEM

Aspect	RSA-OAEP	ML-KEM (Module-LWE)
Hard problem	Integer factorisation	Module Learning With Errors
Quantum attack	Broken by Shor's algorithm at any size	No efficient quantum attack known
Public key size	294–550 bytes	800–1568 bytes
Ciphertext size	256–512 bytes	768–1568 bytes
Key generation	Slow, random prime search	Fast, fixed amount of work
Encapsulation	Very fast (small exponent)	Moderate
Decapsulation	Slow, grows rapidly with key size	Moderate, similar to encapsulation
Decryption failure	None (if padding valid)	Negligible but non-zero probability
Maturity	Over 45 years of analysis	Standardised 2024; younger analysis
Standard status	Quantum-vulnerable; deprecation proposed [11]	FIPS 203 [6]

Table II summarises the comparison: there is no single winner. RSA has smaller keys and ciphertexts, very fast encapsulation, no decryption failure, and over 45 years of analysis, but key generation and decapsulation are slow and grow rapidly with key size, and it is broken by Shor's algorithm. ML-KEM (FIPS 203) has larger messages, a negligible but non-zero decryption failure probability, and younger analysis, but it resists known quantum attacks and has a balanced cost profile. The choice is driven mainly by the required protection horizon and by link constraints.

IX. APPLICATIONS

1. **Secure Web and Network Protocols:** Key establishment in TLS, VPN and SSH can adopt ML-KEM, ideally in hybrid mode with a classical exchange, with the larger handshake messages taken into account.
2. **Protection of Long-Lived Data:** Medical, legal, government and financial records that must stay confidential for decades benefit most from early migration because of harvest-now-decrypt-later attacks.
3. **Constrained and IoT Devices:** The low key generation cost and the absence of large-integer arithmetic suit some embedded settings, while message size and memory use need checking against device limits.
4. **Messaging and Cloud Services:** Services that create many short-lived keys can benefit from fast LWE key generation.
5. **Migration Planning:** Cryptographic inventories can use size and timing data like that in this study to estimate bandwidth and latency effects before deployment.
6. **Education and Training:** The framework and prototype give students a reproducible way to study how lattice-based and factoring-based schemes differ.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510118

Migration Roadmap

The measurements suggest the following practical sequence for an organisation moving from RSA key establishment:

1. **Inventory:** Identify every use of RSA for key establishment, including embedded devices and third-party services, and classify the data by required confidentiality lifetime.
2. **Prioritise:** Start with data that must remain secret for ten years or more, since these are exposed to harvest-now-decrypt-later attacks.
3. **Pilot hybrid mode:** Combine a classical exchange with ML-KEM so that security holds if either is broken, while testing message-size effects on real networks.
4. **Measure:** Use optimised libraries to confirm handshake size and latency, especially on constrained links.
5. **Deploy and monitor:** Roll out gradually, keep cryptographic agility so that algorithms can be changed again, and track updates to standards and cryptanalysis.

X. LIMITATIONS AND FUTURE WORK

Limitations

1. **Unequal Implementations:** RSA used optimised OpenSSL code and ML-KEM used a Python prototype, so timing ratios between the families are indicative only.
2. **Prototype Status:** The LWE code is not FIPS 203 bit-exact, not constant-time, and not validated, so side-channel behaviour was not evaluated.
3. **Single Platform:** All results come from one virtual machine and one core; other processors, compilers and embedded targets will give different numbers.

XI. CONCLUSION

This paper compared LWE-based post-quantum key encapsulation with traditional RSA using both analysis and measurement. A Module-LWE KEM prototype following the ML-KEM parameter sets reproduced the standard's key and ciphertext sizes, decapsulated correctly in all 900 trials, and rejected tampered ciphertexts. RSA offered smaller keys and ciphertexts and very cheap encapsulation, while ML-KEM offered fast key generation, similar costs across operations, and resistance to known quantum attacks.

The key contribution is a reproducible, like-for-like view of the trade-off: LWE-based schemes cost more bandwidth but remove the dependence on factoring that Shor's algorithm breaks. The timing comparison is limited by the unequal maturity of the two implementations, and its practical value must be confirmed with optimised libraries, protocol-level tests and real hardware. Given the migration timelines proposed by standards bodies, a staged, hybrid transition from RSA to ML-KEM appears to be a sensible direction for systems that protect long-lived data.

The noise experiment showed that the worst observed decryption noise used about a third of the available margin, and the end-to-end test showed that the difference between schemes shrinks as payloads grow, since the symmetric cipher dominates. For research, the next steps are comparison with optimised libraries, protocol-level handshake tests, and hybrid designs. For practitioners, the main message is that the cost of post-quantum key encapsulation is mostly paid in bytes rather than in computation, and that this cost is small on modern networks but deserves attention on constrained links.

ACKNOWLEDGMENT

The author thanks her guide, Prof. More N.N., and the Department of Master of Computer Application, Samarth College of Engineering and Management, Belhe, for their guidance and support.

REFERENCES

[1]. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978. https://doi.org/10.1145/359340.359342

[2]. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Journal on Computing, vol. 26, no. 5, pp. 1484–1509, 1997. https://doi.org/10.1137/S0097539795293172

[3]. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," Journal of the ACM, vol. 56, no. 6, Art. 34, 2009. https://doi.org/10.1145/1568318.1568324

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.1510118

[4]. C. Peikert, "A decade of lattice cryptography," Foundations and Trends in Theoretical Computer Science, vol. 10, no. 4, pp. 283–424, 2016. https://doi.org/10.1561/0400000074
[5]. J. Bos et al., "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in Proc. IEEE European Symposium on Security and Privacy (EuroS&P), 2018, pp. 353–367. https://doi.org/10.1109/EuroSP.2018.00032
[6]. National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, 2024. https://doi.org/10.6028/NIST.FIPS.203
[7]. C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," Quantum, vol. 5, p. 433, 2021. https://doi.org/10.22331/q-2021-04-15-433
[8]. C. Gidney, "How to factor 2048 bit RSA integers with less than a million noisy qubits," arXiv:2505.15917, 2025.
[9]. K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2," RFC 8017, IETF, 2016.
[10]. E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, "Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography," NIST SP 800-56B Rev. 2, 2019. https://doi.org/10.6028/NIST.SP.800-56Br2
[11]. D. Moody et al., "Transition to Post-Quantum Cryptography Standards," NIST IR 8547 (initial public draft), Nov. 2024.
[12]. E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," Journal of Cryptology, vol. 26, no. 1, pp. 80–101, 2013. https://doi.org/10.1007/s00145-011-9114-1
[13]. E. Barker, "Recommendation for Key Management: Part 1 – General," NIST SP 800-57 Part 1 Rev. 5, 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5
[14]. M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in Advances in Cryptology – EUROCRYPT '94, LNCS vol. 950, Springer, 1995, pp. 92–111.
[15]. D. Boneh, "Twenty years of attacks on the RSA cryptosystem," Notices of the AMS, vol. 46, no. 2, pp. 203–213, 1999.
[16]. L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proc. 28th ACM Symposium on Theory of Computing (STOC), 1996, pp. 212–219.
[17]. M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?," IEEE Security & Privacy, vol. 16, no. 5, pp. 38–41, 2018.
[18]. V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in EUROCRYPT 2010, LNCS vol. 6110, Springer, pp. 1–23.
[19]. A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," Designs, Codes and Cryptography, vol. 75, no. 3, pp. 565–599, 2015.
[20]. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," Journal of Mathematical Cryptology, vol. 9, no. 3, pp. 169–203, 2015.
[21]. D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki–Okamoto transformation," in TCC 2017, LNCS vol. 10677, Springer, pp. 341–371.
[22]. M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST SP 800-38D, 2007.