© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718108

Bridging the Quantum Threat and Post-Quantum Defense: An Empirical Study of Shor's Algorithm and ML-KEM-768 Performance

UTKARSH CHATURVEDI[1], UTKARSH SINGH GUNJIYAL[2], VANSH KUKREJA[3]
[1, 2, 3]*Delhi Technological University*

Abstract*—The advent of quantum computing poses a funda- mental threat to public-key cryptography based on integer factoring (RSA) and discrete logarithms (ECC). This paper presents an empirical study bridging both sides of this challenge. On the threat side, we implement Shor's algorithm from first principles using Qiskit and execute it on quantum simulators, successfully factoring N = 15 (50.1% success rate) and N = 21 (32.8%). Resource extrapolation based on Gidney and Ekera˚ (2021) places RSA-2048 at approximately 20 million noisy qubits—roughly four orders of magnitude beyond current hardware. On the defense side, we benchmark ML-KEM-768 (NIST FIPS 203) against X25519 over 10,000 iterations per operation, finding that ML- KEM-768 is 7–18× faster than X25519 on Apple Silicon (keygen: 9.6 µs vs. 66.3 µs median), with the primary cost being a 35× increase in key and ciphertext sizes (2272 B vs. 64 B total wire cost). Our integrated analysis demonstrates that the quantum threat is approaching but distant, while the post-quantum defense is ready and computationally inexpensive, supporting the case for immediate migration under the harvest-now-decrypt-later threat model.

***Index Terms*—Shor's Algorithm, Post-Quantum Cryptography, ML-KEM, Quantum Computing, Lattice-Based Cryptography, RSA, NIST FIPS 203, Benchmarking**

I. INTRODUCTION

Public-key cryptography underpins the security of modern digital infrastructure. RSA [1], whose security rests on the hardness of integer factorization, and Elliptic Curve Cryptography (ECC), based on the discrete logarithm problem, collectively secure the vast majority of internet communications, financial transactions, and digital signatures in use today. However, Shor's algorithm [2], running on a sufficiently large quantum computer, solves both problems in polynomial time, rendering these foundational schemes insecure.

While cryptographically relevant quantum computers (CRQCs) do not yet exist, the threat has motivated a decade- long effort by the National Institute of Standards and Technology (NIST) to standardize post-quantum cryptographic (PQC) algorithms. The competition, launched in 2016 with 69 submissions, culminated in August 2024 with the finalization of three standards: FIPS 203 (ML-KEM) [3] for key encapsulation, FIPS 204 (ML-DSA) [4] for digital signatures, and FIPS 205 (SLH-DSA) for hash-based signatures [5].

The urgency of migration is amplified by the *harvest- now-decrypt-later* (HNDL) threat model [6]: adversaries can capture encrypted traffic today and store it for decryption once quantum computers become available. For data with long-term confidentiality requirements—state secrets, medical records, financial data—the effective deadline for migration is not when CRQCs arrive, but when the data was first encrypted. Mosca's theorem [7] formalizes this: if the time to migrate (T_m) plus the required secrecy period (T_s) exceeds the time until CRQCs (T_q), then migration should have already begun.

Two questions are central to the migration decision:

1) How close is the threat? Can we demonstrate Shor's al- gorithm on current quantum platforms, and what resources would breaking RSA-2048 require?
2) What does the defense cost? How does ML-KEM perform relative to classical alternatives, and what are the practical migration overheads in terms of computation and band- width?

This paper addresses both questions empirically. We implement Shor's algorithm from scratch, including the Quantum Fourier Transform (QFT) and classical post-processing, exe- cute it on Qiskit's Aer simulator for $N = 15$ and $N = 21$, and quantify the gap to RSA-2048. We then benchmark ML-KEM across all three parameter sets against X25519

(Curve25519 ECDH), measuring computational performance and communication overhead with statistical rigor.

Our contributions are:

- An end-to-end, reproducible implementation of Shor's algorithm with verified QFT, controlled modular exponentiation, and continued-fraction post-processing, achieving 50.1% and 32.8% success rates for $N = 15$ and $N = 21$ respectively
- Comprehensive ML-KEM-512/768/1024 benchmarks (10,000 iterations per operation) demonstrating that ML- KEM-768 is 7–18× faster than X25519, contradicting the common assumption that PQC carries a performance penalty
- An integrated threat–defense analysis connecting the quantum threat timeline to the concrete migration cost, support- ing the case for immediate adoption of post-quantum key exchange

II. BACKGROUND AND RELATED WORK

A. Shor's Algorithm

Shor's algorithm [2] reduces integer factorization to *quan- tum period-finding*. To factor a composite $N = pq$:

1) Choose random $a < N$ with $\gcd(a, N) = 1$.
2) Find the *order* r of a modulo N: the smallest positive r such that $a^r \equiv 1 \pmod{N}$.
3) If r is even and $a^{r/2} \not\equiv -1 \pmod{N}$, then $\gcd(a^{r/2} \pm 1, N)$ yields non-trivial factors with probability $\geq 1/2$.

The quantum speedup lies in step 2. The quantum circuit creates a superposition $\frac{1}{Q}\sum_{x=0}^{Q-1}|x\rangle|0\rangle$ (where $Q = 2^{\lceil \log_2 N^2 \rceil}$), applies controlled modular exponentiation to produce $\frac{1}{Q}\sum_x |x\rangle|a^x \bmod N\rangle$, and then applies the inverse QFT to the first register. Measurement yields a value m close to a multiple of Q/r, from which r is recovered via continued fractions. The overall gate complexity is $O((\log N)^2 \cdot \log\log N)$ with optimizations [8].

B. The Quantum Fourier Transform

The n-qubit QFT implements the unitary transformation:

$$\mathrm{QFT}\,|j\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle \qquad (1)$$

This is identical to the classical Discrete Fourier Transform (DFT), but operating on quantum amplitudes. The QFT decomposes into $O(n^2)$ gates: Hadamard gates and controlled phase rotations $R_k = \mathrm{diag}(1, e^{2\pi i/2^k})$, compared to $O(N \log N)$ for the classical FFT where $N = 2^n$—an exponential speedup.

C. ML-KEM (FIPS 203)

ML-KEM [3], originally proposed as CRYSTALS-Kyber [9], is a key encapsulation mechanism whose security relies on the Module Learning With Errors (MLWE) problem [10]. The MLWE problem asks: given a matrix $A \in R_q^{k \times k}$ and a vector $b = As + e$ (where s and e are small-norm secrets and errors in the polynomial ring $R_q = Z_q[X]/(X^{256} + 1)$), recover s. This is believed hard for both classical and quantum adversaries.

ML-KEM provides three parameter sets targeting NIST security levels 1, 3, and 5 (corresponding to 128, 192, and 256-bit classical security). Core operations use the Number Theoretic Transform (NTT) for efficient polynomial multiplication, yielding $O(n \log n)$ arithmetic operations.

D. Current Quantum Hardware

The current era of quantum computing, characterized by Preskill [11] as Noisy Intermediate-Scale Quantum (NISQ), features processors with 100–1,100+ qubits and two-qubit gate error rates of approximately 10^{-3}. Gidney and Ekera˚ [8] estimate that factoring RSA-2048 requires approximately 20 mil- lion noisy qubits using surface code error correction, operating for roughly 8 hours. Regev [12] has proposed an alternative algorithm with potentially reduced qubit requirements, though its practical advantage remains under investigation.

E. Related Work

Several prior studies have benchmarked post-quantum al- gorithms. Bos et al. [9] introduced CRYSTALS-Kyber and provided initial performance measurements. Bernstein and Lange [6] survey the PQC landscape comprehensively. Our work differs from prior studies by providing an *integrated* empirical analysis connecting the quantum threat (Shor's algorithm implementation and execution) to the post-quantum defense (ML-KEM benchmarks) within a single reproducible framework, quantifying both sides of the migration equation.

III. METHODOLOGY

A. Shor's Algorithm Implementation

We implement Shor's algorithm in Python using Qiskit [13] (v2.4.1) with the Aer simulator backend:

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718108

Quantum Fourier Transform. We construct the QFT circuit from Hadamard gates and controlled phase rotations, following the standard recursive decomposition. Correctness is verified against Qiskit's built-in QFT implementation by computing the statevector fidelity $F = |\langle\psi_{ours}|\psi_{Qiskit}\rangle|^2$, achieving $F > 0.999$ for $n \in \{3, 4\}$ qubits across all 2^n basis state inputs.

Controlled modular exponentiation. For $N = 15$, we employ the standard hardcoded 4-qubit unitary permutation gates for each valid base $a \in \{2, 4, 7, 8, 11, 13\}$, implementing $|x\rangle \mapsto |ax \bmod 15\rangle$ via sequences of SWAP and NOT gates. For $N = 21$, we construct the full $2^5 \times 2^5$ permutation matrix for $x \mapsto ax \bmod 21$ and decompose it using Qiskit's unitary synthesis.

Circuit architecture. The complete Shor's circuit consists of: (i) an n_{count}-qubit counting register initialized in superposition via Hadamard gates; (ii) an n_{work}-qubit work register initialized to $|1\rangle$; (iii) controlled-U^{2^j} gates for $j = 0, \ldots, n_{count} - 1$; and (iv) the inverse QFT on the counting register followed by measurement. We use $n_{count} = 8$ for $N = 15$ (12 total qubits) and $n_{count} = 10$ for $N = 21$ (15 total qubits).

Classical post-processing. Measurement outcomes m from the counting register are converted to phase estimates $\phi = m/2^{n_{count}}$. The continued fractions algorithm extracts candidate periods r as denominators of the rational approximation $\phi \approx s/r$ with $r < N$. Valid periods satisfy $a^r \equiv 1 \pmod N$, and factors are extracted as $\gcd(a^{r/2} \pm 1, N)$.

B. ML-KEM Benchmarking

Implementation. We benchmark ML-KEM-512, ML-KEM-768, and ML-KEM-1024 using liboqs [14] (v0.15.0, compiled from source with shared library support). The X25519 baseline uses the cryptography library (v48.0.0) backed by OpenSSL. For X25519, "encapsulation" is modeled as ephemeral key generation plus ECDH exchange (the func- tional equivalent of KEM encapsulation), and "decapsulation" as ECDH exchange with the received ephemeral public key.

Measurement protocol. Each operation (keygen, encapsu- lation, decapsulation) is timed over 10,000 iterations using Python's time.perf_counter_ns()for nanosecond pre- cision. All measurements are single-threaded on a single core.

Platform. Apple M-series (ARM64), macOS, Python 3.11.7. The platform is representative of modern consumer and developer hardware.

Statistical analysis. We report mean, median, standard deviation, 95th percentile (p95), and 99th percentile (p99). The median is preferred for central tendency due to the presence of occasional outliers from OS scheduling and garbage collection.

IV. RESULTS

A. QFT Verification

Our QFT implementation achieves statevector fidelity $F > 0.999$ against Qiskit's reference implementation for all 2^n input basis states on $n = 3$ and $n = 4$ qubits. Additionally, we verify the key property that the QFT converts a periodic input state into sharp frequency peaks: applying the QFT to a state with period $r = 4$ produces peaks at multiples of $2^n/r$, confirming its suitability for Shor's period-finding subroutine.

B. Shor's Algorithm: Simulator Results

1) $N = 15$, $a = 7$: The circuit (12 qubits, depth 11 after transpilation) was executed on the Qiskit Aer noiseless simulator with 4,096 shots. Table I presents the measurement outcomes.

TABLE I: Shor's Algorithm: $N = 15$, $a = 7$ (Simulator, 4,096 shots). The order of 7 modulo 15 is $r = 4$, yielding QFT peaks at multiples of $256/4 = 64$.

Value	Count	Phase ϕ	r_{cand}	Valid	Factors
0	994	0.000	—	×	—
64	984	0.250	4	✓	(3, 5)
128	1,051	0.500	2	×	—
192	1,067	0.750	4	✓	(3, 5)

The QFT produces four sharp peaks at the expected positions {0, 64, 128, 192}, each receiving approximately 1/4 of the total probability. Two of four peaks ($\phi = 0.25 \to r = 4$ and $\phi = 0.75 \to r = 4$) yield the correct period via continued fractions, giving a success rate of 50.1% (2,051 of 4,096 shots). The remaining peaks correspond to $\phi = 0$ (trivial, no period information) and $\phi = 0.5 \to r = 2$ (a divisor of the true period, but $7^1 \equiv 7 \not\equiv 1 \pmod{15}$, so the period check fails).

From the valid period $r = 4$: $\gcd(7^2 - 1, 15) = \gcd(48, 15) = 3$ and $\gcd(7^2 + 1, 15) = \gcd(50, 15) = 5$, correctly recovering $15 = 3 \times 5$.

We additionally verified with $a = 11$ (order $r = 2$), obtaining a 50.3% success rate with factors (3, 5).

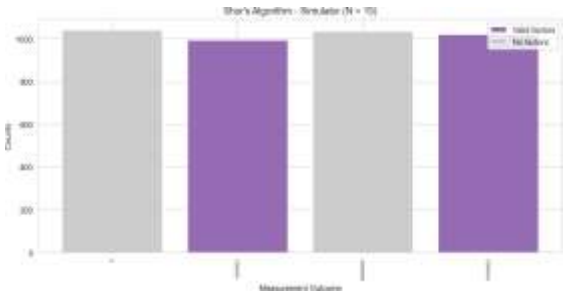

Fig. 1: Measurement outcome distribution for Shor's algorithm with $N = 15$, $a = 7$ on the Aer simulator (4,096 shots). Purple bars indicate outcomes that yield valid factors; gray bars do not. The four sharp peaks at multiples of 64 correspond to the QFT detecting the period $r = 4$.

2) $N = 21$, $a = 2$: The larger circuit (15 qubits, depth 13) was executed with 4,096 shots.

≈

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718108

The expected period is $r = 6$. Since $2^{10}/6 \approx 170.67$ is not an integer, the QFT peaks are not perfectly sharp—probability leaks into neighboring states, spreading across more measurement outcomes.

Table II summarizes the dominant outcomes.

TABLE II: Shor's Algorithm: $N = 21$, $a = 2$ (Simulator, 4,096 shots). Selected dominant outcomes from 97 distinct measurement results.

Value	Count	Phase ϕ	r_{cand}	Valid	Factors
0	684	0.000	—	×	—
512	684	0.500	2	×	—
341	501	0.333	3	×	—
171	490	0.167	6	✓	(3, 7)
853	460	0.833	6	✓	(3, 7)
683	420	0.667	3	×	—
+ 91 additional outcomes with counts ≤ 125					

The success rate is 32.8% (1,345 of 4,096 shots), correctly finding $21 = 3 \times 7$. The reduced success rate compared to $N = 15$ is attributable to two factors: (i) the non-power-of-2 period causes spectral leakage in the QFT; (ii) only phases near $\phi \approx 1/6$ and $\phi \approx 5/6$ yield the full period $r = 6$ (phases near 1/3 and 2/3 yield $r = 3$, which fails the period check since $2^3 = 8 \equiv 1 \pmod{21}$).

C. Resource Extrapolation to RSA-2048

Table III contextualizes our results against the requirements for factoring RSA-2048.

The gap between current hardware (1,100 qubits) and the RSA-2048 requirement (20 million qubits) [8] is approximately four orders of magnitude in qubit count alone (Fig. 2). Additionally:

TABLE III: Resource Requirements: Experimental vs. RSA- 2048

Target	Qubits	Success	Gap to HW
$N = 15$ (this work)	12	50.1%	Achievable
$N = 21$ (this work)	15	32.8%	Achievable
RSA-2048 [8]	~20M	est. high	~20,000×
IBM Eagle (2023)	1,121	—	Current best
IBM Condor (2023)	1,121	—	Current best

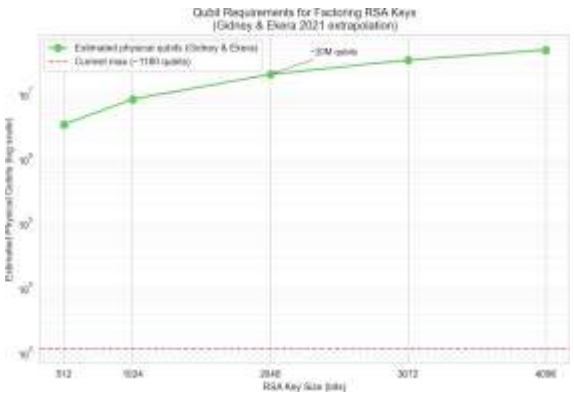

Fig. 2: Estimated physical qubit requirements for factoring RSA keys of increasing size (log scale). The horizontal dashed line indicates current IBM hardware (1,100 qubits). RSA-2048 requires 20 million noisy qubits [8], a gap of approximately four orders of magnitude.

- Gate error rates must improve from $\sim 10^{-3}$ to $\sim 10^{-6}$, or quantum error correction must compensate (adding ~1,000× qubit overhead per logical qubit)
- Coherence times must sustain computation for hours, not the current millisecond regime
- Physical connectivity constraints require extensive SWAP gate insertion for large circuits

D. ML-KEM Performance

1) Computational Performance: Table IV presents the com- plete benchmark results.

A notable result emerges: ML-KEM-768 is significantly faster than X25519 across all operations. The speedup ratios (X25519 median / ML-KEM median) are:

- Keygen: 66.3/9.6 = 6.9× faster
- Encapsulation: 199.6/11.3 = 17.7× faster
- Decapsulation: 134.4/11.6 = 11.6× faster

This contradicts the common assumption that post-quantum algorithms impose a computational penalty. Even ML-KEM-1024 (NIST Level 5, 256-bit security) is 4–13× faster than X25519. Fig. 3 visualizes this comparison for ML-KEM-768.

2) Performance Scaling: ML-KEM operations scale approximately linearly with the module rank k:

- ML-KEM-512 ($k = 2$): ~7 µs per operation
- ML-KEM-768 ($k = 3$): ~11 µs per operation

TABLE IV: ML-KEM and X25519 Performance (10,000 itera- tions each, Apple Silicon ARM64). All times in microseconds

Algorithm	Op.	Mean	Median	P95	P99
ML-KEM-512	KG	6.2	6.2	6.8	7.9
ML-KEM-512	Enc	7.6	7.3	8.2	9.3
ML-KEM-512	Dec	7.6	7.7	8.0	9.8
ML-KEM-768	KG	9.8	9.6	10.4	12.2
ML-KEM-768	Enc	11.1	11.3	11.6	14.0
ML-KEM-768	Dec	11.6	11.6	12.5	14.6
ML-KEM-1024	KG	13.9	13.9	14.6	17.3
ML-KEM-1024	Enc	15.2	15.2	15.6	18.9
ML-KEM-1024	Dec	16.7	16.8	17.3	19.3
X25519	KG	66.4	66.3	69.0	74.1
X25519	Enc	200.1	199.6	211.6	232.0
X25519	Dec	132.6	134.4	143.1	155.1

KG = keygen, Enc = encapsulate, Dec = decapsulate.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718108

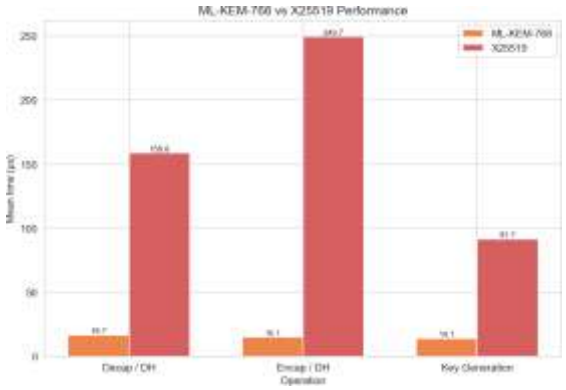

Fig. 3: Performance comparison of ML-KEM-768 vs. X25519 across keygen, encapsulation, and decapsulation operations (10,000 iterations each). ML-KEM-768 is 7–18 faster across all operations.

- ML-KEM-1024 ($k = 4$): ~15 µs per operation

The linear scaling confirms that the NTT-based polynomial multiplication dominates the computation, and each additional module dimension adds a constant overhead.

3) Timing Distribution: ML-KEM operations exhibit remarkably low variance: standard deviations are consistently below 1 µs for encapsulation and decapsulation (except ML-KEM-512 encapsulation, which shows a single outlier at 2905 µs, likely from OS scheduling). In contrast, X25519 keygen has $\sigma = 101$ µs due to variable-time scalar multiplication and occasional cache misses. This consistency is advantageous for real-time applications where worst-case latency matters.

4) Key and Ciphertext Sizes: ML-KEM-768 requires transmitting 2,272 bytes vs. X25519's 64 bytes—a 35× increase (Fig. 4). While the size difference is significant in relative terms, the absolute overhead of ~2 kB per key exchange is modest in context:

- A TLS 1.3 ClientHello adds ~2 kB, fitting within two TCP

TABLE V: Key and Ciphertext Sizes (bytes). Wire cost is the total bytes transmitted per key exchange (public key + ciphertext).

Alg.	PK	SKCT	SS	Wire
X25519	32	32	3232	64
ML-KEM-512	800	1,632	76832	1,568
ML-KEM-768	1,184	2,400	1,08832	2,272
ML-KEM-1024	1,568	3,168	1,56832	3,136

PK = public key, SK = secret key, CT = ciphertext, SS = shared secret.

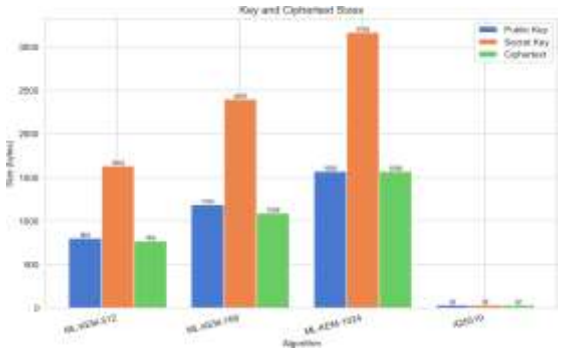

Fig. 4: Key and ciphertext sizes across algorithms. ML-KEM keys and ciphertexts are 25–50× larger than X25519, but the absolute overhead (~2 KB) is modest for most applications.

segments

- A typical HTTPS page load transfers 2–5 MB; the PQC overhead is < 0.1%
- For bandwidth-constrained IoT, the overhead may require protocol-level optimization

V.DISCUSSION

A. Threat Assessment

Our experiments confirm that Shor's algorithm is correct and implementable on current quantum computing frame- works. The 50.1% success rate for $N = 15$ precisely matches the theoretical expectation: of four QFT peaks, two ($\phi = 1/4$ and $\phi = 3/4$) yield the full period $r = 4$, while $\phi = 0$ is trivial and $\phi = 1/2$ yields a period divisor. The degradation to 32.8% for $N = 21$ is due to spectral leakage from the non-power-of-2 period, not hardware noise—the simulator is noiseless.

The significance of our simulator results is that they establish a *lower bound* on the algorithm's correctness: any failure on real hardware beyond these rates is attributable to noise, not algorithmic error. Published hardware demonstrations of Shor's algorithm for small numbers [?] have historically shown significant degradation due to decoherence and gate errors, consistent with NISQ-era limitations [11].

The RSA-2048 gap remains vast: ~20,000× in qubit count, ~ 1,000 in error rate, and orders of magnitude in coherence time. Even with optimistic projections, CRQC's are estimated to be 15–30 years away [7]. However, the HNDL threat means that data encrypted today with RSA or ECDH may be compromised retroactively.

B. Defense Readiness

Computational cost is negative. Our finding that ML-KEM-768 is 7–18 faster than X25519 on Apple Silicon is attributable to:

1) NTT-based arithmetic: ML-KEM's core operations use the Number Theoretic Transform over $Z_{3329}[X]/(X^{256} + 1)$, which decomposes into cache-friendly, vectorizable butterfly operations
2) No big-integer arithmetic: Unlike RSA's modular expo- nentiation or X25519's scalar multiplication on a 255-bit curve, ML-KEM operates on fixed-size 16-bit coefficients
3) Constant-time by design: ML-KEM's operations have inherently data-independent timing, avoiding the variable- time patterns that increase X25519's variance

This speedup is consistent with other platform bench-marks [9] and suggests that PQC migration may actually *improve* server-side throughput for applications performing many key exchanges.

Size cost is modest but real. The 35 increase in wire cost (Table V) is the primary tangible cost of migration. For most web and API applications, the additional 2 kB per handshake is negligible. For constrained environments (low-bandwidth IoT, satellite links, smart cards with limited memory), the larger keys and ciphertexts may require protocol adaptation.

C. Migration Implications

NIST has published concrete deprecation timelines [15]: RSA-2048 and 112-bit security to be deprecated by 2030 and disallowed by 2035 for federal use. The recommended migration path is *hybrid key exchange*—combining ML-KEM- 768 with X25519 [16]—which maintains security even if one algorithm is later found vulnerable. This approach is already deployed in Chrome and Firefox. Given our findings:

1) The computational cost of ML-KEM is zero or negative (it's faster)
2) The bandwidth cost is 2 kB per exchange (negligible for most applications)
3) The HNDL threat creates urgency independent of CRQC timelines

The primary barrier to migration is ecosystem readiness, not performance.

VI.LIMITATIONS

1) Small factoring instances: Our Shor's implementation is limited to $N \in \{15, 21\}$, which are far from crypto- graphically relevant. The modular exponentiation circuits use hardcoded permutations rather than general modular arithmetic.
2) Simulator only: Due to IBM Quantum Platform's migra- tion from the legacy API to IBM Cloud during the course of this work, we were unable to obtain hardware results.

Our simulator results establish algorithmic correctness but do not capture the effects of real quantum noise.

3) Single platform: ML-KEM benchmarks are from Ap- ple Silicon (ARM64) only. Performance on x86-64 (with AVX2/AVX-512), server-class ARM, and embedded plat- forms may differ.
4) Library-level measurement: We benchmark through Python bindings (liboqs-python), which add overhead vs. native C. Absolute timings may be faster in production deployments.
5) No protocol integration: We measure primitive-level per- formance, not end-to-end protocol impact (TLS handshake latency, certificate chain size with PQ signatures, etc.).

VII.CONCLUSION AND FUTURE WORK

We presented an empirical study bridging both sides of the quantum cryptographic challenge. On the threat side, Shor's algorithm successfully factors $N = 15$ (50.1% success) and $N = 21$ (32.8%) on a quantum simulator, validating the al- gorithmic threat, while RSA-2048 remains 20,000 beyond current hardware capabilities. On the defense side, ML-KEM- 768 is 7–18 faster than X25519 across all operations, with the primary cost being a 35 increase in key/ciphertext sizes (2 kB per exchange).

Our integrated analysis leads to a clear conclusion: the quantum threat is real but distant, the defense is ready and fast, and the migration cost is minimal. Under the HNDL threat model, organizations handling long-lived sensitive data should begin migrating to post-quantum key exchange now, using hybrid ML-KEM + X25519 deployments as a transition strategy.

Future work includes: (i) hardware execution of Shor's on IBM Cloud Quantum when platform access is restored; (ii) benchmarking ML-DSA and SLH-DSA post-quantum signa- tures; (iii) cross-

platform benchmarks on x86-64 and embed- ded ARM; (iv) end-to-end TLS 1.3 handshake measurements with PQ key exchange; and (v) extending Shor's to larger instances using iterative phase estimation to reduce qubit requirements.

Reproducibility

All code, data, and analysis are publicly available at https://github.com/utkarshc-sudo/btech-pqc-shors, with pinned dependency versions and automated Makefile targets for full reproduction.

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.

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1994, pp. 124–134.

[3] National Institute of Standards and Technology, "Module-lattice-based key-encapsulation mechanism standard (FIPS 203)," NIST, Tech. Rep. FIPS 203, 2024.

[4] ——, "Module-lattice-based digital signature standard (FIPS 204)," NIST, Tech. Rep. FIPS 204, 2024.

[5] G. Alagic *et al.*, "Status report on the third round of the NIST post- quantum cryptography standardization process," *NIST Internal Report 8413-upd1*, 2022.

[6] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[7] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[8] C. Gidney and M. Ekera˚, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021.

[9] J. W. Bos *et al.*, "CRYSTALS—Kyber: A CCA-secure module-lattice- based KEM," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018, pp. 353–367.

[10] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *Journal of the ACM*, vol. 60, no. 6, pp. 1–35, 2013.

[11] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quan- tum*, vol. 2, p. 79, 2018.

[12] O. Regev, "An efficient quantum factoring algorithm," in *arXiv preprint arXiv:2308.06572*, 2023.

[13] Qiskit Contributors, "Qiskit: An open-source framework for quantum computing," 2024.

[14] Open Quantum Safe Project, "liboqs: C library for prototyping and experimenting with quantum-resistant cryptography," 2024. [Online]. Available: https://openquantumsafe.org/

[15] National Institute of Standards and Technology, "Transition to post- quantum cryptography standards," NIST, Tech. Rep. IR 8547, 2024.

[16] A. Langley, "Post-quantum key agreement in TLS 1.3," in *IETF Internet- Draft*, 2024, x25519Kyber768Draft00.