ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

Performance Evaluation of Standardized and Candidate Post-Quantum Algorithms on Resource-Constrained Virtualized IoT

Yahaya Salisu[1*]; Sandeep Kumar[2]; Surajo Nuhu Umar[3]; Muhammad Dini Ibrahim[4]; Umar Muhammad Tasiu[5]

[1]Department of Computer Science and Engineering, Shobhit Institute Engineering, Meerut, India.
[2]Department of Computer Science and Engineering, Shobhit Institute Engineering, Meerut, India.
[3]Department of Computer Science and Engineering, Shobhit Institute Engineering, Meerut, India.
[4]Department of Computer Science and Engineering, Shobhit Institute Engineering, Meerut, India.
[5]Department of Computer Science Federal University Dutsinma, Dutsinma, Nigeria.

Corresponding Author: Yahaya Salisu[1*]

Publication Date: 2026/08/18

Abstract: Quantum computing poses a significant threat to Internet of Things (IoT) communications that rely on classical cryptographic algorithms such as RSA, Diffie–Hellman, and Elliptic Curve Cryptography (ECC). These algorithms are vulnerable to Shor's algorithm, making long-term data security uncertain, particularly for IoT devices with extended operational lifespans. Post-quantum cryptographic (PQC) algorithms offer resistance to quantum attacks; however, their deployment on resource-constrained IoT devices remains challenging due to limitations in processing power, memory, and energy consumption. This study benchmarks seven Key Encapsulation Mechanisms (KEMs) ML-KEM, Kyber, NTRU, BIKE, Classic McEliece, FrodoKEM, and sntrup761 across key generation, encapsulation, and decapsulation operations. Experiments were conducted using the liboqs library in a virtualized Linux environment running WSL2 to simulate constrained computing conditions. Results revealed substantial performance differences among the algorithms. ML-KEM-512 achieved the best overall performance, completing all operations in under 14 µs, while Kyber-512 showed similar efficiency. NTRU provided the fastest encapsulation but incurred significantly higher key-generation costs. BIKE, FrodoKEM, and Classic McEliece exhibited execution times unsuitable for real-time IoT applications. Overall, lattice-based schemes, particularly ML-KEM and Kyber, emerged as the most practical PQC solutions for resource-constrained IoT environments. The findings also demonstrate the usefulness of virtualized benchmarking for deployment planning.

Keywords: *IoT Security, ML-KEM, CRYSTALS-Kyber, NTRU, BIKE, FrodoKEM.*

How to Cite: Yahaya Salisu; Sandeep Kumar; Surajo Nuhu Umar; Muhammad Dini Ibrahim; Umar Muhammad Tasiu (2026) Performance Evaluation of Standardized and Candidate Post-Quantum Algorithms on Resource-Constrained Virtualized IoT. *International Journal of Innovative Science and Research Technology*, 11(8), 800-812. https://doi.org/10.38124/ijisrt/26aug297

I. INTRODUCTION

Modern computing has incorporated the Internet of Things (IoT) into its foundation. There are now billions of IoT devices connecting an array of applications, including healthcare systems, smart cities, industrial plants, agricultural operations, and energy grids. Continuous data collection is done by these devices through autonomous coordination and sharing of data over networks. They lack the capability to provide adequate support for them [1], [2]. Most IoT devices do not support security because they have limited processing power, reduced memory, low storage capacity, and very small amounts of power available to work with. Thus, the security and operational feasibility trade-off is not a theoretical issue; it is simply a matter of engineering that exists in every IoT deployment [3].

Classical cryptographic schemes sit at the centre of this problem. RSA, Diffie-Hellman, and ECC all depend on mathematical problems that quantum computers can solve efficiently. Shor's algorithm breaks both integer factorisation and discrete logarithm in polynomial time, which means any scheme built on those problems becomes vulnerable once a sufficiently powerful quantum computer exists [1], [4]. For general computing systems, this is a future concern. For IoT, it is more pressing, because devices deployed today

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

may still be running in fifteen years. An adversary can collect encrypted traffic now and decrypt it later; the encryption does not need to be broken in real time for the data to be exposed [4].

Post-quantum cryptography is the response to this. PQC schemes are built on mathematical problems that quantum algorithms do not currently solve efficiently. They run on conventional hardware and require no quantum infrastructure, which makes them compatible with existing IoT deployments[4], [5] . NIST completed its first standardisation round in 2024, publishing ML-KEM as the primary key encapsulation standard and continuing evaluation of several candidate schemes, including NTRU, BIKE, Classic McEliece, FrodoKEM, and sntrup761 [5], [6], [7]. Key encapsulation specifically matters for IoT because session establishment with gateways, cloud services, and peer nodes happens frequently and under latency pressure [8].

While quantum resistant algorithms will not by themselves provide a means by which devices connected to the Internet of Things (IoT) can be secured, quantum resistant algorithms (PQC) typically require at least an order of magnitude more compute power [9] and memory than classical algorithms that they replace, and how much more compute power and memory are required varies greatly depending upon both the algorithm family and the specific security level being used within a family of algorithms. For example, ML-KEM and Kyber, both of which are lattice-based algorithms, tend to perform better than code-based algorithms (such as BIKE and McEliece) or plain LWE implementations (such as FrodoKEM). However, the amount of compute power or memory used will also depend on the underlying hardware platform and how it is being used [5], [6], [7]. Consequently, when evaluating the suitability of a particular algorithm for use on an IoT device, consideration must be given to not only which algorithm is being used but also how it will be utilized.

Existing benchmarking research is valuable, as it provides valuable information; however, it may be limited in scope either by having a narrow focus that is the algorithms or by testing only one method. Additionally, limited studies enforce the use of strict resource limit requirements based on the IoT requirement [2], [7], [10]. Performance comparisons on desktop hardware do not transfer cleanly to constrained deployments, and aggregate results hide the operation-specific bottlenecks that are most relevant for deployment decisions [6], [11]. There is no published study that evaluates all major standardised and candidate KEMs together, at multiple security levels, with per-operation granularity, under consistently enforced resource constraints [8], [11].

This study runs that evaluation. Seven KEMs are benchmarked using liboqs in a virtualised, software-constrained Linux environment. Key generation, encapsulation, and decapsulation are each measured separately across parameter variants, producing a dataset that supports direct cross-family and cross-security-level comparison under conditions that reflect constrained IoT execution.

The goal is practical: to identify which algorithms are actually deployable in resource-limited IoT contexts, which ones have specific operational costs that might still make them useful in targeted scenarios, and which ones are not viable under constraint. The data supports deployment decisions that go beyond theoretical security guarantees.

II. RELATED WORK

The goal of securing IoT with post-quantum cryptography involves combining two different fields, which both have limitations of their own. PQC algorithms are usually designed to have maximum theoretical strength, while IoT devices commonly do not possess the resources necessary to support these algorithms [12]. There has been a great deal of literature published about this since the NIST started its standardization process; however, it has been sporadic, with some areas well developed and others not. This chapter will discuss the literature in four areas related to securing IoT devices against the quantum threat, reviewing each of the design features for KEMs that are applicable to IoT devices, reporting on what has been achieved via benchmarking of these algorithms on constrained hardware, and identifying where the remaining gaps exist.

- *PQC and IoT Security*

IoT faces a REAL quantum threat. RSA and ECC are the two main cryptographic protocols that provide the basis for most IoT devices to establish sessions today, but they are based on problems that can be solved by using Shor's algorithm in polynomial time [4]. The longer-term, and deeper issue facing IoT is that devices endure for a long time. An example is that the smart meter or industrial sensor that is deployed today could potentially still be operating 15 years from now. As a result, any data being captured from that device as it is being encrypted using a classical system today can be stored by someone with malicious intent and decrypted when quantum computing technology matures. Therefore, this concept of "harvest now, decrypt later" has been explicitly included in national-level migration guidance [4], [13].

PQC is the best practical response as it is implemented using standard silicon, hence no quantum channel is needed for interoperability and no requirement for a new infrastructure to be built [5]. Asif's 2021 survey [5] is at the forefront by providing baseline metrics upon which other works can build. He has examined lattice schemes for IoT and argued that Ring-LWE offers the best overall balance in that it uses smaller keys than code-based schemes and that the execution time will be shorter for software embedded implementations. While the findings from Asif's survey provide a good reference point in their own right, they also help to establish a common framework that allows future work to build upon similar comparisons.

New research has helped clarify the practical limitations of using quantum-safe algorithms in the Internet of Things (IoT). In particular, Shirisha et al. [1] identified three particular tensions that have been seen repeatedly across different deployment scenarios: the amount of computation-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

al overhead that can be tolerated compared to latency; the size of keys and ciphertexts compared to the available memory of the device; and implementation complexity versus the ability for a developer to continue maintaining it over time. They point out that there is no family of algorithms that can resolve all three of these issues for all classes of devices, establishing the case for making algorithm decisions based on context of use rather than simply based on their own level of security. Chawla and Mehra [13] similarly address these issues and point out that the bandwidth constraints associated with protocols like MQTT and CoAP mean that standard parameter sets for post-quantum cryptography will not work as they currently are defined.

Specific examples of this work have been demonstrated in literature that applies to specific domains. As noted by Mansoor et al. [14] in their examination of how e-health IoT devices operate, they found that increases in handshake latency from a few milliseconds had real, measurable effects on how well real-time patient monitoring systems operated; in other words, the delay caused by the increased latency in the handshake alone caused small changes to the patient monitoring pipeline that had huge ramifications. Kumari et al. [15] tested an approach to combining lattice-based authentication with code-based encryption and concluded that while the approach would limit memory overhead on the devices, the increase in implementation complexity would likely not be sustainable for most developers. Ma et al. [16] designed a cryptosystem using BRLWE-based keys that were resistant to side-channel attacks and demonstrated that the additional time required to thwart power analysis attacks is not prohibitively burdensome to the runtime performance of most IoT devices.

Taken together, this work confirms that PQC deployment in IoT is viable but not uniform. It also sets up a recurring requirement: algorithm selection decisions cannot be made without performance data collected under conditions that actually resemble the target environment.

➢ *Standardised and Candidate PQC KEMs*

The establishment of a shared secret is the first step of a secure session and is handled by a Key Encapsulation Mechanism (KEM). In 2024, NIST established ML-KEM (Module Lattice-Based KEM, built on CRYSTALS-Kyber) as its primary KEM standard [6]. Many other options exist as KEMs, as they were evaluated during the NIST process or created independently; however, they have varying security assumptions and performance benchmarks.

- *ML-KEM / CRYSTALS-Kyber*

ML-KEM is built on the Module-LWE problem and targets three security levels equivalent to AES-128, AES-192, and AES-256 [5], [6]. Operations are converted into polynomial multiplies in small rings and can be easily implemented on general-purpose processors, therefore providing a natural decomposition to take advantage of SIMD acceleration. Ehsan et al. [7] compared Kyber to NTRU using embedded hardware; they discovered that through testing all levels of security, encapsulation and decapsulation times were consistently faster with Kyber, and the costs for creating keys were approximately equal between both systems. In their implementation of ML-KEM in MQTT and CoAP, Almutairi and Sheldon [17] found that this was manageable under normal conditions; however, due to packet loss, there existed compounding retransmission costs with respect to the ciphertext size, resulting in a much steeper degradation of performance than with several alternatives.

- *NTRU*

NTRU is an older form of a lattice-based scheme, which uses truncated polynomial rings for its polynomial mathematical operations [5]. Compared to ML-KEM, the generation of keys can be fairly expensive; however, encapsulation and decapsulation of a key can be done quickly and produce a small amount of ciphertext. These characteristics are beneficial for situations where keys are infrequently generated, but sessions are created often. Because NTRU relies upon a different hardness assumption than schemes such as LWE, there is some diversity in terms of security arguments. The sntrup761 variant has made some modifications to the original design and is working within OpenSSH; however, this comes at the cost of somewhat larger keys [10].

- *BIKE*

BIKE is a code-based KEM derived from quasi-cyclic moderate-density parity-check codes [11], [18]. Its public keys are small, but its ciphertexts are large, and the decoding step at the core of decapsulation is iterative. Fitzgibbon and Ottaviani [11] found that BIKE decapsulation times varied substantially across invocations, which is a problem in latency-sensitive applications where worst-case timing matters as much as average-case. Kumar [18] noted that BIKE scales poorly with security level compared to lattice-based schemes, widening the gap at higher parameter settings.

- *Classic McEliece*

Classic McEliece has the oldest security foundation of any evaluated scheme, random linear code decoding, studied since the 1970s, and no structural attacks are known against it [11], [18]. The cost is key size and public keys run from hundreds of kilobytes to several megabytes depending on the parameter set, and key generation is orders of magnitude slower than any other scheme in comparable benchmarks [19] [11]. For devices with kilobytes of RAM, that is not viable for online use. Classic McEliece is realistic only where key generation can happen offline on more capable hardware.

- *FrodoKEM*

FrodoKEM is intentionally built without any ring or module organization such that FrodoKEM can utilize only plain LWE to ensure avoiding any potential algebraic attacks that may arise against structured lattices [20]. The downside of taking this approach is FrodoKEM does provide significantly lower performance indicators when compared to the same security level as ML-KEM, e.g. larger keys, larger ciphertexts, lower operational speed, etc. Hanna et al. [21] conducted a performance assessment of FrodoKEM on consumer IoT devices and determined it

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

would not be suitable for real-time session establishment in either memory-constrained or bandwidth-constrained environments. As with the Classic McEliece scheme, FrodoKEM is a more effective option for offline provisioning than for real-time session establishment.

Septien-Hernandez et al. [20] have concluded that the performance of various post-quantum cryptographic algorithms depends more on the family of algorithms selected rather than on the security level of the algorithm chosen within that family, which mirrors the results of Satrya et al. [7] in their review of energy monitoring systems, which found that lattice-based schemes routinely outperformed code-based schemes for all three key establishment operations when implemented in software and should be regarded as a valid basis for future planning.

- *Benchmarking PQC in Constrained Environments*

The benchmarking data is inconsistent across studies. The individual studies may be rigorous in their methodology, but they are sufficiently different such that comparisons between studies are not meaningful. Differences in platform selection, the definition of "constrained," the set of operations to measure, and whether the measurement was taken with a real hardware platform or via emulation cause significant variability, and there is no common methodology in the literature that allows for results to build cumulatively to form a coherent understanding of performance.

Silva et al. [22] demonstrated that execution time and memory consumption demonstrate different characteristics on embedded vs. general-purpose platforms, as well as demonstrating that measurements taken in one environment do not transfer to another. This conclusion is also true for measurements conducted in the context of PQC, and therefore it is critical to use platforms that are cross-platform and support reproducibility.

Fitzgibbon and Ottaviani [11] performed a multi-algorithm benchmark of various PQCs on constrained devices using liboqs, which is an API that is consistent across all leading PQCs and allows for comparison under identical conditions. Their results indicate that the encapsulation process is consistently the fastest relative to other types of encryption algorithms, while the key generation process and the decapsulation process both have significant differences between groups of encryption algorithms. Similarly, Khan et al. [2] conducted comparable analyses on consumer electronic devices and found that clock frequency and memory bus width were two of the primary characteristics of the hardware that caused large differences in performance among the various encryption algorithms.

Considering several platforms, Abbasi et al. [23] performed a cross-environment study and discovered that even though relative ranks were consistent between benchmarks, the absolute time it took to execute the benchmark could vary greatly from platform to platform (by orders of magnitude). Therefore, this finding has substantial implications; it suggests that relative performance can be trusted across different hardware, while absolute values from one piece of hardware can't be transferred to another.

There hasn't been as much research conducted on virtualized and emulation-based benchmarking as empirical or real-world benchmarks, but studies such as Tran-Hoang and Nakajo's [10] have provided some positive evidence; this study evaluated their implementation of Kyber using both real hardware and QEMU emulation via an experimental setup where the rankings of the two methods' performance were consistent. Almutairi and Sheldon [17] have suggested that containerized benchmarking with artificially imposed limits on resources can produce more consistent results than traditional hardware variables. Satrya et al.'s [7] comparison between their software-only evaluation of the performance of multiple different algorithm implementations and an associated testbed is an example of how some software can successfully replace real hardware in determining comparative performance. Different performance characteristics are offered by hardware-based accelerators since they entail different compromises; for example, Akcay & Yalcin [24] used ASIPs designed for lattice-based post-quantum cryptography (PQC) to show their value when compared to software implementations through significantly faster processing speeds but also found that a major obstacle to the adoption and use of ASIPs is high development costs, as custom hardware is fully fixed-function hardware. Additionally, work done by Nguyen & Chen [25] shows that Ascon-based PQC can be implemented on automotive FPGAs and distributed. Shahbazi & Ko [26] demonstrated that even modest hardware-accelerated implementations would reduce energy consumed by more than 60% on IoT devices. However, the findings of these three studies provide useful upper boundaries, but they only describe accelerated implementations of PQC rather than software-only platforms or deployments to which the majority of IoT devices will ultimately be limited in terms of PQC implementations.

Lastly, there have been very few studies indicating that benchmarking frameworks should include security characteristics beyond run time and memory. Ji & Dubrova [27] performed a side-channel attack on a masked implementation of the Kyber PQC algorithm to illustrate the vulnerability of a given algorithm to side-channel attacks, illustrating that whilst a given PQC algorithm may provide real-time computational speed advantages, it may still present a vulnerability to side-channel attacks, due to leakage of information through timing or power consumption. Opilka et al. [9], studied the total PQC signature overhead of applicable mutual authentication protocols and revealed their findings to be relevant in IoT-based applications, where devices must mutually authenticate to each other, not just to a server.

The overall picture is a literature that has produced good individual data points but lacks a unified evaluation. No study covers all three KEM operations across both standardised and candidate schemes under consistently enforced resource constraints on the same platform. Results sit in separate papers with incompatible setups, and synthesis is difficult.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

➢ *Research Gap*

Three areas in the literature continually demonstrate the same gaps, and these gaps are intertwined and therefore should not be considered separately.

First, the scope of algorithms. A typical study studies one or two schemes extensively, which are normally either ML-KEM or Kyber schemes, with variable inclusion of NTRU in comparison [8], [11]. Studies that do include larger numbers of schemes, such as Fitzgibbon and Ottaviani [11] and Abbasi et al. [23], focus on desktop or server-based hardware and therefore cannot be represented within the resource constraints of the IoT platform. There is no available study that compares all the KEM algorithms that are standardized to NIST alongside the main candidate schemes under constrained physical resource conditions.

Secondly, there is a lack of commonality across the definition and application of the concept of "constrained" as it relates to various methods of defining and applying the constraints [28]. Clearly defined physical benchmarks are real-world examples of hardware that are difficult to reproduce, associated with location and device. The virtualization methods described in [10], [17] are efficient for a broad KEM comparison but have not previously been utilized as the basis for comparison across all candidate algorithms and all security levels within a comprehensive study across all algorithms and security levels. An alternative approach would be to impose container-based resource throttling as the means of reproducing results without the logistical challenges that would accompany moving physical hardware. No studies have thus far used container-based resource throttling as the method of comparison across all KEM algorithms.

The absence of operation-level granularity is the third issue. A significant number of studies report on the aggregate runtime or focus on a single operation. Thus, it is not possible to find out where each algorithm's bottleneck lies in the context in which it has been utilized [1], [18]. This is important because the operations are stressed differently by different IoT use cases: a device is going to have different operational requirements if it creates keys once when it is provisioned and encapsulates them routinely or if it rotates its keys on an ongoing basis. Therefore, without data based on each operation, you will not have data to rely on when making decisions.

Hasija et al. [29] verified in their bibliometric analysis that comparative benchmarking of PQC research beneath realistic constraints is one of the least covered areas of PQC relative to its practical importance. Moreover, Almutairi and Sheldon [8] illustrated this point from a protocol perspective, where they established that performance data from unconstrained hardware do not provide critical information for MQTT or CoAP integration decisions.

This study addresses all three gaps. Seven KEMs are evaluated: ML-KEM, Kyber, NTRU, BIKE, Classic McEliece, FrodoKEM, and sntrup761 using liboqs under a containerised environment with enforced CPU and memory limits. Key generation, encapsulation, and decapsulation are all measured at multiple security levels alongside resource consumption metrics. The result is operation-level, multi-algorithm, constraint-aware data that the literature currently does not provide.

III. METHODOLOGY

The evaluation uses experimental benchmarking to measure how selected post-quantum KEMs perform under a virtualized constrained IoT environment. The objective is to produce execution data that is reproducible and comparable across algorithms, without relying on specific physical hardware. A software-based constraint model simulates the resource limits typical of IoT edge nodes.

➢ *Experimental Setup*

Tests were conducted on a Linux environment running under Windows Subsystem for Linux 2 (WSL2) on an x86_64 host. Cryptographic implementations were built and executed using the open quantum safe liboqs library, which provides a unified benchmarking interface across a wide range of post-quantum algorithms. GCC was used as the compiler with optimisation flags enabled.

The environment was configured with OpenSSL support and available processor-level acceleration features. Table 1 summarizes the configuration obtained during execution.

Table 1 Experimental Configuration

Parameter	Specification
Target platform	x86_64-Linux-5.10.16.3-microsoft-standard-WSL2
Compiler	GCC 11.4.0
Build optimisation	-O3, -fomit-frame-pointer, gc-sections
OQS version	0.15.0
OpenSSL support	Yes (OpenSSL 3.0.2)
AES support	AES-NI
SHA-2 implementation	OpenSSL
SHA-3 implementation	ADX, AES, AVX, AVX2, BMI1, BMI2, PCLMULQDQ, POPCNT, SSE, SSE2, SSE3AVX2
CPU extensions active	

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

The speed_kem utility located within the Wide Area Network Security library (libOQC) was utilized to carry out a series of runs of key generation, encapsulation, and decapsulation multiple times for each Key Encapsulated Mechanism (KEM) over a specific period of time as a means of determining statistical validity of the timing and cycle counts regardless of the number of cycles executed during the other three operations above. The overall benchmarking workflow adopted in this study is illustrated in Figure 1.

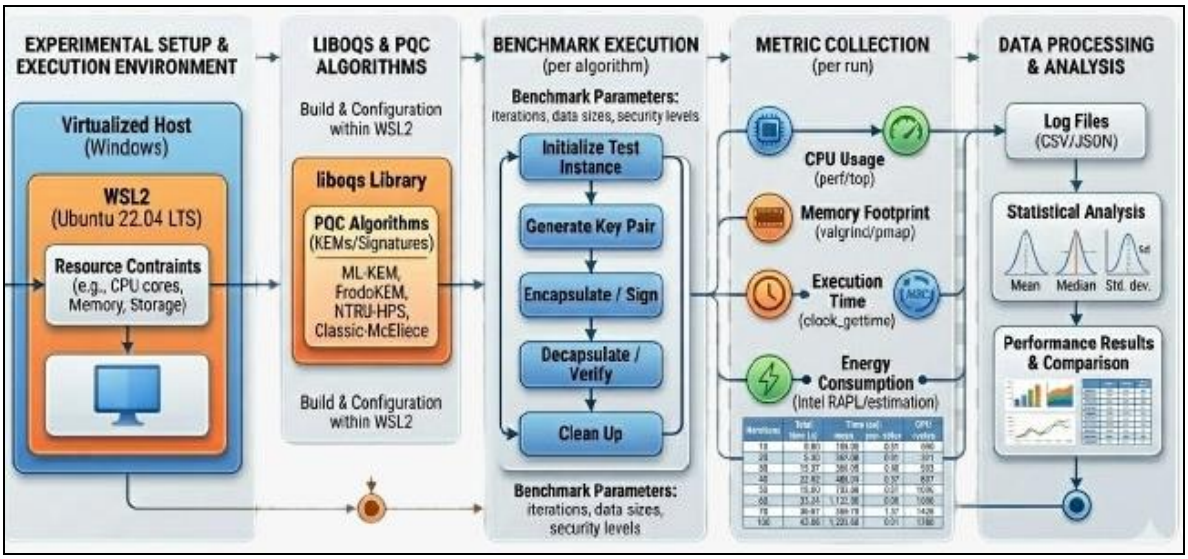


Fig 1 The Workflow Showing how the Experiment was Carried Out in this Post-Quantum Cryptography Benchmarking Framework for Analyzing Suitability in Constrained IoT Environments.

➢ *Evaluated Algorithms*

This project evaluated KEMs because they are directly responsible for the secure exchange of keys needed to encode and send data across the Internet of Things (IoT) network. Algorithms selected from NIST standards-based and those that were nominated for consideration, representing the primary post-quantum families across multiple designs, levels of security, and computing profiles, were in the review of the following:

ML-KEM (ML-KEM-512, ML-KEM-768, ML-KEM-1024)

Kyber (Kyber512, Kyber768, Kyber1024)

BIKE (BIKE-L1, BIKE-L3, BIKE-L5)

Classic McEliece (multiple parameter sets)

NTRU-HPS (2048-509, 2048-677, 4096-821, 4096-1229)

NTRU-HRSS (701, 1373)

sntrup761

FrodoKEM (640, 976, 1344; AES and SHAKE variants

This work contains details of the most commonly being considered post-quantum schemes in terms of their cryptographic strength and efficiency. Lattice based schemes are classified as being most popular amongst systems with respect to the amount of active endorsement and trials conducted using them, which includes ML-KEM, Kyber, NTRU, sntrup761, FrodoKEM as well as code based (Classic McEliece, BIKE). Table 3.2 highlights both standardised and candidate post-quantum algorithms so that comparisons can be made between all current possibilities in relation to their safety and speed.

Table 2 Evaluated post-quantum KEM algorithms and cryptographic families

Algorithm	Parameter Set(s)	Family	Characteristics	IoT Relevance
ML-KEM	ML-KEM-512/768/1024	Lattice	NIST-standard; balanced	Primary IoT candidate
Kyber	Kyber512/768/1024	Lattice	Efficient; widely benchmarked	Baseline comparison
BIKE	BIKE-L1/L3/L5	Code	Fast encapsulation; heavy decapsulation	Code-based trade-off analysis
Classic McEliece	348864 to 8192128 variants	Code	Very slow keygen; large keys	Offline provisioning only
NTRU-HPS	2048-509/677, 4096-821/1229	Lattice	Efficient except keygen	Infrequent-keygen deployments
NTRU-HRSS	701, 1373	Lattice	Moderate overhead overall	Constrained suitability test
sntrup761	sntrup761	Lattice	Compact; OpenSSH-deployed	Lightweight candidate
FrodoKEM	640/976/1344 AES+SHAKE	Lattice	Conservative security; heavy	Identifies impractical options

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

➢ *IoT Constraint Modelling*

Testing utilized a combination of virtual and actual device configurations. The targeted environment is for operations of low-power Internet of Things (IoT) edge nodes, having limitations for processor capability, available memory and hardware crypto acceleration (other than the processing capability of the host CPU). The virtual platform used for testing was based on a Linux subsystem environment with host-level resource restrictions and software emulator support.

By providing some physical realism, this approach provides researchers with significant increases in reproducibility and accessibility. The assumption of constraints is based on 3 common characteristics of constrained IoT devices:

Lower processor capacity compared to conventional computing hardware.

Strict latency constraints that prevent (high) overhead crypto operations.

Resource load sensitivity that makes any algorithm with heavy key generation or decapsulation cost impractical on lightweight edge nodes.

Emulation-based testing is extremely useful in instances where access to specific embedded hardware is limited and when there is a need for comparing multiple algorithms across many different testing configurations. The relative performance order exhibited in the results of the virtual benchmark studies is very consistent with the results produced on physical hardware, as supported by previous research [10], thus making this method of conducting the comparative analysis indicated for this study. Figure 3.2 presents the conceptual model used to simulate constrained IoT execution conditions within the virtualized benchmarking environment.

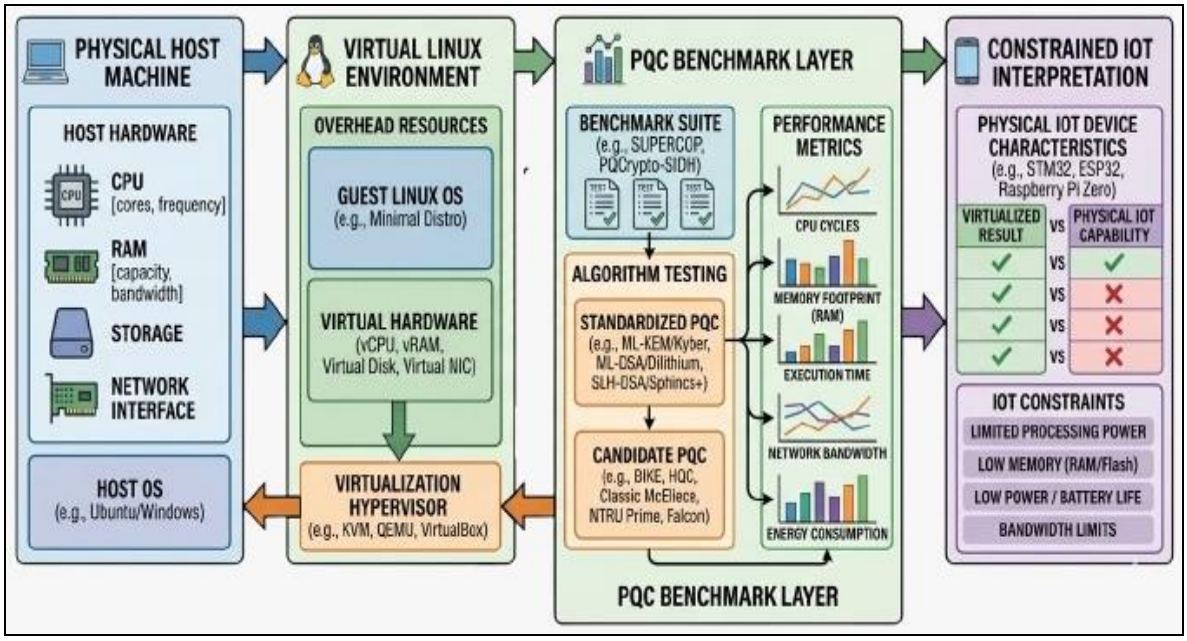


Fig 2 A Conceptual Model of Constrained IoT Execution Conditions, Simulated Using a Virtual Software Benchmarking Environment

➢ *Performance Metrics*

The performance data is obtained from the automatic generation of a benchmark output using the liboqs speed evaluation utility. The performance metrics collected during benchmarking and their interpretation in constrained IoT environments are summarized in Table 3. In general there are three actions that are measured for every scheme: key generation, encapsulation and decapsulation. The following four metrics are recorded for each of the three actions:

- Average execution time (μ sec/operation): This metric captures the mean amount of time that it takes to complete one action. It is the most straightforward measure of latency and serves as the primary metric to evaluate an IoT device's suitability.
- CPU cycles/operation: This measures the average number of processor cycles consumed for each operation (one operation). This will provide an insight into the computational cost of each scheme from an abstract level and is independent of any variations in clock frequency.
- Iteration count: This will provide an indirect measure of how many iterations of the benchmark were completed within the fixed measurement period. This information provides a relative measure of throughput.
- The population standard deviation is recorded for both execution time and CPU cycles for each of the three actions to provide a measure of performance stability across repeated iterations of the benchmark. A High standard deviation indicates poor time stability and is problematic in real-time IoT situations when the worst-case latency is critical.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

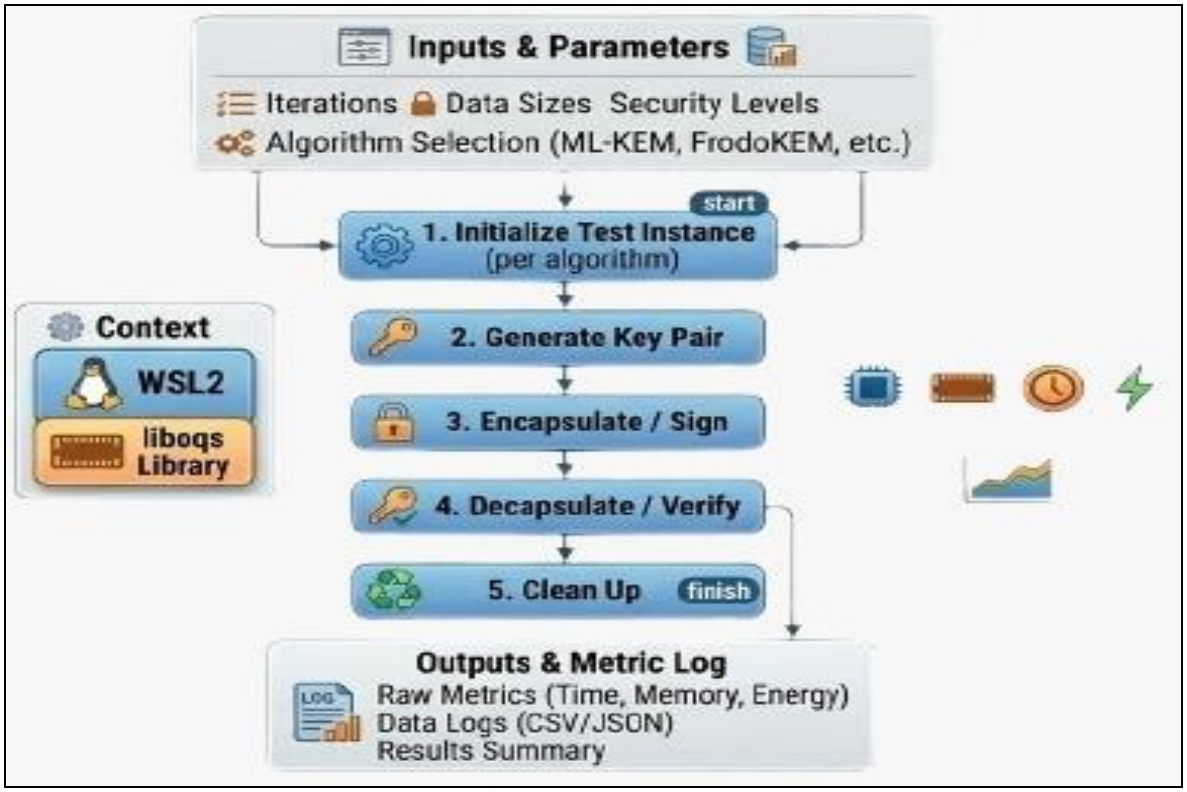
Table 3 Performance Metrics and their Interpretation in Constrained IoT Contexts

Metric	Description	Unit	IoT Interpretation
Key Generation Time	Mean time to generate public and secret keys	µs/op	Lower = faster setup, less initialisation overhead
Encapsulation Time	Mean time to generate ciphertext and shared secret	µs/op	Lower = better session establishment responsiveness
Decapsulation Time	Mean time to recover the shared secret	µs/op	Lower = faster secure communication on receiver side
CPU Cycles	Mean processor cycles per operation	Cycles/op	Lower = less burden on constrained processors
Iteration Count	Completed benchmark runs in measurement interval	Count	Higher = greater operational throughput
Std. Deviation	Variability across repeated runs	µs/op or cycles/op	Lower = more predictable execution behaviour

➢ *Benchmark Procedure*

The complete benchmarking procedure followed in this study is illustrated in Figure 3. A structured six-step process was used to allow for consistent and repeatable results through each of the test schemes.

- Step 1(Test Environment Setup): A WSL2/Linux test environment was established containing CMake, GCC, liboqs and OpenSSL. All libraries were installed to the same version level, and no performance benchmarks were run before this preparation was completed.
- Step 2 (liboqs Compilation): liboqs was built from source with optimization flags applied and with cryptographic acceleration if supported by the underlying hardware (i.e., processor).
- Step 3 (Benchmarking): Key generation, encapsulation and decapsulation for each KEM algorithm were processed using the integrated speed_kem tool at an approximately 3-second duration for each operation.
- Step 4 (Results Collection): For each KEM algorithm and for each operation, the speed_kem tool recorded the total time taken for the operation, the average time taken for the operation, the average number of CPU cycles used, the variance of cycle counts, the number of iterations, and the standard deviation of cycle counts.
- Step 5 (Benchmark Comparison): The data recorded in Step 4 were grouped into tabular format and compared across KEM algorithms for latency, computational costs, consistency of execution, and feasibility for implementation in IoT devices.
- Step 6 (Feasibility for IoT Device Implementation): The KEM algorithm results were assessed against the expected constraints that are typical of IoT devices to organize the KEM algorithms into lightweight, moderate-overhead, and heavyweight candidates.



ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

Fig 3 A Step-by-Step Benchmarking Process Used to Evaluate the Performance of Post- Quantum

➢ *Reproducibility*

Each test was performed with a common set of software tools and configuration. The primary benchmarking interface for all tests was liboqs; this will help eliminate any possible discrepancies due to each implementation being developed separately. By benchmarking over a set duration instead of number of iterations, it will also limit the impact of any temporary system loads present during each benchmark on timing results. Even though there was no direct way to measure all tests against the same hardware due to virtualisation vs bare metal, this methodology provides a repeatable comparison of post-quantum performance that is applicable to more devices with limited processing power than the same class of devices with more processing power.

IV. RESULTS AND DISCUSSION

This section presents benchmark results for the evaluated KEMs and discusses what they mean for IoT deployment. The analysis focuses on key generation, encapsulation, and decapsulation measured separately for each algorithm using mean execution time in microseconds from liboqs. Results are grouped by comparison type: overall performance, security-level scaling, and operation-specific bottlenecks.

➢ *Comparative Execution Time*

Table 4 summarizes mean execution time for representative algorithms from each major PQC family. The differences are substantial, spanning several orders of magnitude across families.

Table 4 Mean Execution Time (µs) for Representative Evaluated KEMs

Algorithm	Key Generation (µs)	Encapsulation (µs)	Decapsulation (µs)
ML-KEM-512	11.664	12.767	13.719
ML-KEM-768	18.493	19.311	21.366
ML-KEM-1024	27.316	31.575	30.247
Kyber512	13.207	15.185	10.451
Kyber768	19.662	22.543	17.013
Kyber1024	26.730	30.854	24.329
NTRU-HPS-2048-509	61.112	10.918	13.845
NTRU-HRSS-701	101.388	11.908	22.300
sntrup761	381.696	20.398	21.802
BIKE-L1	427.915	64.033	1128.958
BIKE-L3	1038.697	146.130	3619.356
BIKE-L5	2627.443	298.527	8094.119
Classic McEliece-348864	80388.053	26.317	59.497
Classic McEliece-6688128	343486.111	82.123	126.348
FrodoKEM-640-AES	511.672	782.898	808.555
FrodoKEM-976-SHAKE	2802.083	3121.878	3052.556
FrodoKEM-1344-SHAKE	5521.513	5776.933	5773.719

ML-KEM and Kyber are the fastest schemes overall. ML-KEM-512 completes key generation in 11.664 µs, encapsulation in 12.767 µs, and decapsulation in 13.719 µs. Kyber512 is comparable: 13.207 µs, 15.185 µs, and 10.451 µs respectively. Both families maintain low, balanced overhead across all three operations, which is exactly what constrained IoT devices need.

NTRU tells a more mixed story. NTRU-HPS-2048-509 achieves the fastest encapsulation of any scheme tested at 10.918 µs, and decapsulation at 13.845 µs is also competitive. Key generation, however, costs 61.112 µs roughly five times the ML-KEM-512 equivalent. NTRU-HRSS-701 is similar: excellent session-establishment performance, but key generation at 101.388 µs starts to look expensive. For IoT deployments where keys are provisioned infrequently but sessions are established often, NTRU is still a viable option. For devices that rotate keys regularly, the key generation cost is a real constraint.

BIKE, FrodoKEM, and Classic McEliece all have costs that would be problematic for most constrained IoT use cases. BIKE-L5 decapsulation reaches 8094.119 µs over 8 milliseconds on the receiving side of every session. FrodoKEM-1344-SHAKE exceeds 5.7 ms on all three operations. Classic McEliece is the most extreme case: key generation for the Classic McEliece-6688128 parameter set reaches 343,486 µs, and even the smallest parameter set (348864) costs over 80 ms for keygen alone. Encapsulation and decapsulation for Classic McEliece are moderate, which means it could potentially work in scenarios where keys are generated offline and reused, but online key establishment is not feasible.

➢ *Security-Level Scaling Within Algorithm Families*

Raw performance numbers only tell part of the story. Understanding how overhead grows as security level increases matters for deployment planning, since selecting a higher-security parameter set is often non-negotiable in real applications. Table 5 shows how execution time scales across security levels for ML-KEM, BIKE, and FrodoKEM.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

Table 5 Security-Level Scaling within Selected Algorithm Families

Family	Variant	Key Generation (µs)	Encapsulation (µs)	Decapsulation (µs)
ML-KEM	512	11.664	12.767	13.719
ML-KEM	768	18.493	19.311	21.366
ML-KEM	1024	27.316	31.575	30.247
BIKE	L1	427.915	64.033	1128.958
BIKE	L3	1038.697	146.130	3619.356
BIKE	L5	2627.443	298.527	8094.119
FrodoKEM	640-AES	511.672	782.898	808.555
FrodoKEM	976-SHAKE	2802.083	3121.878	3052.556
FrodoKEM	1344-SHAKE	5521.513	5776.933	5773.719

ML-KEM scales well. Moving from ML-KEM-512 to ML-KEM-1024, key generation grows from 11.664 µs to 27.316 µs and decapsulation from 13.719 µs to 30.247 µs roughly 2.3x increases for a step up from AES-128 to AES-256 equivalent security. The absolute numbers remain low enough that even the highest security level is practical on constrained hardware.

BIKE scales badly. BIKE-L5 key generation is 6.1x higher than BIKE-L1, and decapsulation is 7.2x higher. At L5, both values are in the range where real-time session establishment becomes impractical on low-power devices. Lower BIKE levels might still be usable in less latency-sensitive contexts, but the trajectory of cost growth limits BIKE's headroom.

FrodoKEM is the clearest case of a scheme that does not scale for constrained use. FrodoKEM-976-SHAKE is already over 5 times more expensive than FrodoKEM-640-AES across all three operations, and FrodoKEM-1344-SHAKE pushes all three above 5.7 ms. For an IoT device where milliseconds of session establishment latency matter, those numbers rule out FrodoKEM at any security level above the minimum.

➢ *Operation-Specific Performance Behaviour*

Knowing which operation is the bottleneck for a given algorithm is as important as knowing its overall cost. Table 6 identifies the dominant computational bottleneck for selected algorithms.

Table 6 Dominant Computational Bottleneck by Algorithm

Algorithm	Dominant Bottleneck	Observation
ML-KEM-512	Balanced	Low cost across all operations
Kyber512	Balanced	Efficient in all three phases
NTRU-HPS-2048-509	Key Generation	Session operations remain efficient
BIKE-L1	Decapsulation	High receiver-side overhead
BIKE-L5	Decapsulation	Cost grows severely at high security
Classic McEliece-348864	Key Generation	Extremely high setup cost
FrodoKEM-640-AES	Encapsulation/Decapsulation	Heavy runtime across both sides
FrodoKEM-1344-SHAKE	All operations	Uniformly high overhead

This operation-level view changes how some algorithms should be assessed. Classic McEliece looks unusable from its key generation number, but its encapsulation and decapsulation are moderate. That means it is only ruled out for online key establishment it could still work in a deployment model where a gateway generates and distributes keys offline, and the constrained device only runs encapsulation and decapsulation at runtime. The bottleneck is real, but it is also avoidable in the right architecture.

BIKE has the opposite problem. Encapsulation is fast enough, but decapsulation is where the cost concentrates and decapsulation happens on the receiver side, which in IoT is often the constrained device. A system where the constrained node always decapsulates would feel the full weight of BIKE's iterative decoder on every session.

ML-KEM and Kyber are the only families where overhead is balanced and low across all three operations. That uniformity is not just a performance advantage; it also simplifies deployment planning because there is no architectural workaround needed to avoid an expensive operation.

➢ *Implications for Constrained IoT Deployment*

The results do not produce a single answer, they're rather illustrative of the different algorithms' classifications and where they sit in the tier system of classifications for the constraints of IoT.

ML-KEM and Kyber are clearly the main contenders for the majority of constrained IoT deployments since their execution times never dip beneath 35 microseconds at their highest security level of operation and the execution times for all three operations are nearly equal across all three algorithms. Additionally, transitioning to a higher security parameter from one of these algorithms is low-cost and predictable in nature. This means that the use of low-power Edge devices, establishment of simple sessions, and the de-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

mands of latency-sensitive communications make this group of algorithms a suitable option.

In certain scenarios, NTRU represents a valid second option. NTRU's speed in encapsulating and decapsulating data But none of the three is a practical candidate for online, real-time key exchange on a power- or memory-limited IoT node. The broader implication is that performance data at this level of granularity matters. Security level alone does not determine IoT suitability. The algorithm family, the distribution of cost across operations, and how that cost scales with security level are all factors that only emerge from benchmarking under conditions that resemble the target environment.

V. CONCLUSION AND FUTURE WORK

This study evaluated seven post-quantum key encapsulation mechanisms ML-KEM, Kyber, NTRU, BIKE, Classic McEliece, FrodoKEM, and sntrup761 under a virtualised, software-constrained IoT environment using liboqs. The goal was to produce operation-level, reproducible performance data that can inform deployment decisions for resource-limited IoT devices.

The results have indicated a distinct performance hierarchy across the families of algorithms. The two algorithms with the best performance on KEMs, ML-KEM and Kyber, achieved consistent, low latencies of less than 35 µs for all security levels tested, along with a minimal amount of overhead, making them good candidates for use in low-power, latency-sensitive IoT environments. Although NTRU has competitive performance for establishing sessions, it has a costly key-generation

The other algorithms tested (BIKE and FrodoKEM) each have their own unique operating characteristics (e.g., BIKE's iterative decapsulation; Frodo's large, constant-time execution overhead; Classic McEliece's large key generation overhead), which disqualify them from being used online in real time on constrained hardware, although they may be feasible in some instances within the targeted architecture.

Additionally, the study has demonstrated that virtualised, software-based benchmarks are consistent and reproducible. By using liboqs as a common interface among the different schemes, implementation-based variance can be avoided and algorithms can be compared directly and systematically. This creates an important methodological contribution to the existing literature, as physical implementations are not easily replicated and frequently only allow comparisons among a small number of algorithms.

Not every KEM that is standardised by NIST, or proposed as a candidate, is suitable for deployment in constrained IoT environments. The level of security alone cannot be used in its own right to determine whether or not it is suitable to deploy in that environment; rather, the overall distribution of computational cost across operations as well as the scaling behaviour of different types of security need to be taken into account. Using this approach, this study provides a granularity level for seven different KEM schemes evaluated over multiple parameter sets, which is not currently accounted for as one unified data set in the literature.

This work can be extended further in multiple directions. Testing KEMs by enforcing stricter CPU and memory constraints through containerised environments and emulating the KEMs on ARM architectures using the QEMU emulator would yield results applicable to specific types of IoT hardware. Further characterisation of the pre-existing datasets with memory footprint metrics, communication overhead metrics and energy consumption metrics would present systematic gaps in the reported metrics. Integrating KEMs into the ongoing work to develop IoT communication protocols (e.g., TLS, DTLS, MQTT, CoAP) would demonstrate the impact of performance at the algorithmic level on end-to-end protocol overhead. Finally, including post-quantum digital signature schemes in future evaluations would provide more complete understanding of quantum-safe IoT security issues.

➢ *Author Contributions*

- Yahaya Salisu: Conceptualization, methodology, software implementation, experimental design, data collection, benchmarking, formal analysis, visualization, investigation, writing original draft preparation, and manuscript revision.
- Sandeep Kumar: Supervision, conceptual guidance, methodology review, validation, critical review and editing of the manuscript, and academic oversight of the research process.
- Surajo Nuhu Umar: Literature review, methodology support, experimental design support, data interpretation, validation of results, manuscript review, and editorial contributions.
- Muhammad Dini Ibrahimi: Investigation, literature review support, literature analysis, data verification, manuscript review, and contributions to the interpretation of findings.
- Umar Muhammad tasiu critical review of thesis configuration of Virtual Liboqs, and parameter setups in WSL ubuntu

All authors participated in the development of the study, critically reviewed the manuscript for important intellectual content, approved the final version of the manuscript for publication, and agree to be accountable for all aspects of the work, ensuring that questions related to the accuracy or integrity of any part of the research are appropriately investigated and resolved.

➢ *Data Availability Statement (DAS)*

The data underlying this study were generated through controlled benchmark experiments using the Open Quantum Safe (liboqs) framework. All benchmark outputs, processed results, and supporting materials used in the analysis are available from the corresponding author upon reasonable request. The experimental procedures, software configura-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

tions, and evaluation methodology have been described in sufficient detail to facilitate independent replication of the study.

REFERENCES

[1]. N. Shirisha, H. M. Manoj, S. J. Hussain, R. Kotoju, R. Kolikipogu, and A. Mohan, 'Post-quantum security framework for resource-constrained systems: emerging trends, challenges, sustainability, and future directions', Discov Computing, vol. 29, no. 1, p. 85, Feb. 2026, doi: 10.1007/s10791-026-09982-2.

[2]. M. A. Khan, F. Noor, S. Javaid, and J. Żywiołek, 'Implementation and performance of post-quantum cryptography for resource constrained consumer electronics', Discov Internet Things, vol. 5, no. 1, p. 139, Nov. 2025, doi: 10.1007/s43926-025-00238-x.

[3]. V. L. R. D. Costa, J. López, and M. V. Ribeiro, 'A System-on-a-Chip Implementation of a Post-Quantum Cryptography Scheme for Smart Meter Data Communications', Sensors, vol. 22, no. 19, p. 7214, Sep. 2022, doi: 10.3390/s22197214.

[4]. K. Cherkaoui Dekkaki, I. Tasic, and M.-D. Cano, 'Exploring Post-Quantum Cryptography: Review and Directions for the Transition Process', Technologies, vol. 12, no. 12, p. 241, Nov. 2024, doi: 10.3390/technologies12120241.

[5]. R. Asif, 'Post-Quantum Cryptosystems for Internet-of-Things: A Survey on Lattice-Based Algorithms', IoT, vol. 2, no. 1, pp. 71–91, Feb. 2021, doi: 10.3390/iot2010005.

[6]. M. A. Ehsan, W. Alayed, A. U. Rehman, W. U. Hassan, and A. Zeeshan, 'Post-Quantum KEMs for IoT: A Study of Kyber and NTRU', Symmetry, vol. 17, no. 6, p. 881, Jun. 2025, doi: 10.3390/sym17060881.

[7]. G. B. Satrya, Y. M. Agus, and A. B. Mnaouer, 'A Comparative Study of Post-Quantum Cryptographic Algorithm Implementations for Secure and Efficient Energy Systems Monitoring', Electronics, vol. 12, no. 18, p. 3824, Sep. 2023, doi: 10.3390/electronics12183824.

[8]. M. Almutairi and F. T. Sheldon, 'Resilience of Post-Quantum Cryptography in Lightweight IoT Protocols: A Systematic Review', Eng, vol. 6, no. 12, p. 346, Dec. 2025, doi: 10.3390/eng6120346.

[9]. F. Opiłka, M. Niemiec, M. Gagliardi, and M. A. Kourtis, 'Performance Analysis of Post-Quantum Cryptography Algorithms for Digital Signature', Applied Sciences, vol. 14, no. 12, p. 4994, Jun. 2024, doi: 10.3390/app14124994.

[10]. K. Tran-Hoang and H. Nakajo, 'Kyress: a secure, scalable, and resource-efficient CRYSTALS-Kyber cryptosystem for low-cost embedded devices', J Supercomput, vol. 82, no. 2, p. 101, Jan. 2026, doi: 10.1007/s11227-025-08178-7.

[11]. G. Fitzgibbon and C. Ottaviani, 'Constrained Device Performance Benchmarking with the Implementation of Post-Quantum Cryptography', Cryptography, vol. 8, no. 2, p. 21, May 2024, doi: 10.3390/cryptography8020021.

[12]. A. Kia, A. W. Storey, and M. Imtiaz, 'Advanced Hardware Security on Embedded Processors: A 2026 Systematic Review', Electronics, vol. 15, no. 5, p. 1135, Mar. 2026, doi: 10.3390/electronics15051135.

[13]. D. Chawla and P. S. Mehra, 'A roadmap from classical cryptography to post-quantum resistant cryptography for 5G-enabled IoT: Challenges, opportunities and solutions', Internet of Things, vol. 24, p. 100950, Dec. 2023, doi: 10.1016/j.iot.2023.100950.

[14]. K. Mansoor, M. Afzal, W. Iqbal, Y. Abbas, S. Mussiraliyeva, and A. Chehri, 'PQCAIE: Post quantum cryptographic authentication scheme for IoT-based e-health systems', Internet of Things, vol. 27, p. 101228, Oct. 2024, doi: 10.1016/j.iot.2024.101228.

[15]. S. Kumari, M. Singh, R. Singh, and H. Tewari, 'A post-quantum lattice based lightweight authentication and code-based hybrid encryption scheme for IoT devices', Computer Networks, vol. 217, p. 109327, Nov. 2022, doi: 10.1016/j.comnet.2022.109327.

[16]. C. Ma, A. Shankar, S. Kumari, and C.-M. Chen, 'A lightweight BRLWE-based post-quantum cryptosystem with side-channel resilience for IoT security', Internet of Things, vol. 28, p. 101391, Dec. 2024, doi: 10.1016/j.iot.2024.101391.

[17]. M. G. Almutairi and F. T. Sheldon, 'Methodology and Architecture for Benchmarking End-to-End PQC Protocol Resilience in an IoT Context', IoT, vol. 7, no. 1, p. 17, Feb. 2026, doi: 10.3390/iot7010017.

[18]. M. Kumar, 'Post-quantum cryptography Algorithm's standardization and performance analysis', Array, vol. 15, p. 100242, Sep. 2022, doi: 10.1016/j.array.2022.100242.

[19]. J. Choi and J. Lee, 'Secure and Scalable Internet of Things Model Using Post-Quantum MACsec', Applied Sciences, vol. 14, no. 10, p. 4215, May 2024, doi: 10.3390/app14104215.

[20]. J.-A. Septien-Hernandez, M. Arellano-Vazquez, M. A. Contreras-Cruz, and J.-P. Ramirez-Paredes, 'A Comparative Study of Post-Quantum Cryptosystems for Internet-of-Things Applications', Sensors, vol. 22, no. 2, p. 489, Jan. 2022, doi: 10.3390/s22020489.

[21]. Y. Hanna, J. Bozhko, S. Tonyali, R. Harrilal-Parchment, M. Cebe, and K. Akkaya, 'A comprehensive and realistic performance evaluation of post-quantum security for consumer IoT devices', Internet of Things, vol. 33, p. 101650, Sep. 2025, doi: 10.1016/j.iot.2025.101650.

[22]. N. B. F. Silva, D. F. Pigatto, P. S. Martins, and K. R. L. J. C. Branco, 'Case studies of performance evaluation of cryptographic algorithms for an embedded system and a general purpose computer', Journal of Network and Computer Applications, vol. 60, pp. 130–143, Jan. 2016, doi: 10.1016/j.jnca.2015.10.007.

[23]. M. Abbasi, F. Cardoso, P. Váz, J. Silva, and P. Martins, 'A Practical Performance Benchmark of Post-Quantum Cryptography Across Heterogeneous Computing Environments', Cryptography, vol. 9, no. 2, p. 32, May 2025, doi: 10.3390/cryptography9020032.

[24]. L. Akçay and B. Ö. Yalçın, 'Lightweight ASIP Design for Lattice-Based Post-quantum Cryptography

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug297

Algorithms', Arab J Sci Eng, vol. 50, no. 2, pp. 835–849, Jan. 2025, doi: 10.1007/s13369-024-08976-w.

[25]. H. P. Nguyen and Y. Chen, 'Lightweight, Post-Quantum Secure Cryptography Based on Ascon: Hardware Implementation in Automotive Applications', Electronics, vol. 13, no. 22, p. 4550, Nov. 2024, doi: 10.3390/electronics13224550.

[26]. K. Shahbazi and S.-B. Ko, 'Area and power efficient post-quantum cryptosystem for IoT resource-constrained devices', Microprocessors and Microsystems, vol. 84, p. 104280, Jul. 2021, doi: 10.1016/j.micpro.2021.104280.

[27]. Y. Ji and E. Dubrova, 'A side-channel attack on a masked hardware implementation of CRYSTALS-Kyber', J Cryptogr Eng, vol. 15, no. 1, p. 7, Apr. 2025, doi: 10.1007/s13389-025-00375-7.

[28]. F. Oberhansl, T. Fritzmann, T. Pöppelmann, D. Basu Roy, and G. Sigl, 'Uniform instruction set extensions for multiplications in contemporary and post-quantum cryptography', J Cryptogr Eng, vol. 14, no. 1, pp. 1–18, Apr. 2024, doi: 10.1007/s13389-023-00332-2.

[29]. T. Hasija, K. R. Ramkumar, A. Kaur, and M. S. Bali, 'Exploring the landscape of post quantum cryptography: a bibliometric analysis of emerging trends and research impact', J Big Data, vol. 12, no. 1, p. 225, Sep. 2025, doi: 10.1186/s40537-025-01269-5.