Beyond the Breaker: Evaluating Post-Quantum Cryptographic Algorithms for Practical Smart Grid Deployment

Jacob Rydecki
jrydecki@nmsu.edu
New Mexico State University
Las Cruces, New Mexico, USA

Gaurav Panwar
gpanwar@nmsu.edu
New Mexico State University
Las Cruces, New Mexico, USA

Olga Lavrova
olavrova@nmsu.edu
New Mexico State University
Las Cruces, New Mexico, USA

Abstract

With the advancement of quantum computing, traditional public-key cryptographic algorithms are increasingly at risk of being broken. This paper investigates the applicability of post-quantum cryptography (PQC) within modernized power grid infrastructures, commonly known as smart grids, that rely heavily on interconnected and sometimes internet-facing devices and distributed control systems. To help safeguard these critical infrastructures against quantum-enabled threats, we benchmark and evaluate the performance of NIST-standardized and finalist PQC Key Encapsulation Mechanisms (KEMs) in a realistic smart grid testbed. Our evaluation includes both benchmarking of core cryptographic operations, key generation, encapsulation, and decapsulation, as well as full TLS 1.3 handshake benchmarking across a real network of embedded devices and smart grid nodes. A key contribution of this work is the development of a smart grid testbed spanning multiple buildings with distributed energy resources (DERs) such as batteries and photovoltaic arrays, enabling an in-depth, system-level evaluation of post-quantum key exchange protocols in this context. We conclude by discussing the trade-offs between algorithmic efficiency and system constraints such as bandwidth, key rotation frequency, and device capability. Our findings offer actionable recommendations for the deployment of PQC algorithms tailored to the smart grid.

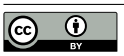
CCS Concepts

• **Security and privacy** → **Public key encryption**; **Security protocols**; • **Hardware** → **Smart grid**; • **Networks** → *Transport protocols.*

Keywords

Post-quantum cryptography, Modernized Grids, Smart Grids, IoT, Encryption

ACM Reference Format:
Jacob Rydecki, Gaurav Panwar, and Olga Lavrova. 2025. Beyond the Breaker: Evaluating Post-Quantum Cryptographic Algorithms for Practical Smart Grid Deployment. In *Proceedings of the 2025 Workshop on Quantum-Resistant Cryptography and Security (QRSEC '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3733820.3764681



QRSEC '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1908-0/2025/10
https://doi.org/10.1145/3733820.3764681

1 Introduction

With the explosive growth of the Internet over the past couple of decades and the surge in the number of connected devices such as cellphones, tablets, laptops, and personal computers, there has been a widespread push to move services online. This move has brought about a lot of security and privacy concerns that are continuously being addressed in research and industry through the proposal and development of new protocols and procedures.

These security and privacy concerns include potential access to confidential information, fraudulent commands and operations, and disruption of legitimate operations. With information routinely being exchanged over the public internet, encryption is needed to preserve the confidentiality of the data being exchanged. Likewise, a need for authentication over the network to ensure legitimate information is exchanged between devices, and fraudulent operations are detected and hampered. These concerns are of further importance in the Internet of Things (IoT), which is a network of typically household devices ("smart" devices) that connect to the internet and communicate with other user devices such as cellphones and cloud servers. The owner is typically able to monitor and manage these devices over the network and through the connection in the cloud. This has resulted in the internet now carrying not just data, but also critical commands and information with real-world consequences.

Similar to the rise of the IoT among households, energy infrastructure is also experiencing an increase in adoption of smart devices. Power companies are increasingly implementing "smart meters" and other similar devices in the power grid that allow the customer to monitor their utility usage and the company to implement automation and remote control of the power grid operations. Power grids that implement these elements for their customers and operators are called *modernized grids* or *smart grids*. Because these devices are network-connected and vulnerable to attacks, all the previously mentioned IoT security concerns extend to the smart grid with even greater urgency. Unauthorized access or disruptions can have widespread consequences, including large-scale power outages impacting citizens and critical infrastructure, breaches of sensitive customer utility data, and fraudulent use of utility services.

In all modern networks, including smart grids, various protocols and methods such as Transport Layer Security (TLS) and Internet Protocol Security (IPSec) are used to ensure data confidentiality and integrity. These protocols rely on cryptographic techniques that secure information by encrypting it with a secret key. Only entities possessing the corresponding key can decrypt and access the information, while unauthorized parties are unable to interpret it. There are two main types of cryptography: symmetric-key cryptography and asymmetric-key cryptography (also known as public-key cryptography). In symmetric-key cryptography, data is

both encrypted and decrypted with the same key, which requires the key to be shared between communicating entities to allow for secure communication. Public-key cryptography is a method in which data is encrypted using a public key and decrypted using a corresponding, but different, private key. Within both symmetric-key and public-key cryptography, various methods and algorithms based on different mathematical constructs and hardness assumptions have been proposed.

The public-key cryptographic algorithms used widely today are based on mathematical problems that are currently computationally infeasible to solve within a realistic timespan, thus preventing an adversary from undermining the security of encrypted data. The most widely used and implemented problems are the *integer factorization problem* and the *discrete log problem* (DLP). For instance, algorithms such as *Diffie-Hellman key exchange* and *ElGamal encryption* are based on DLP, while *RSA* relies on the difficulty of integer factorization. Although computing power has grown exponentially since these mathematical problems were first used in encryption algorithms, they remain computationally infeasible to solve when sufficiently large key sizes are used.

Since symmetric-key encryption and decryption operations are much faster when compared to similar public-key algorithms, network protocols use a combination of both symmetric-key and public-key cryptography to address security concerns. The most widely used protocol over the Internet is the Transport Layer Security (TLS) protocol. TLS first uses public-key cryptography to both authenticate the server to the client and to securely exchange a shared secret or key. Then, this shared secret is used by both client and server as the key for symmetric encryption for future data exchange. Therefore, the TLS protocol ensures authentication and key exchange with public-key cryptography, and efficient and secure ongoing communication with symmetric-key cryptography.

Because TLS and other similar protocols rely on both symmetric-key and public-key cryptography, the breaking of either one would result in insecure communication over the Internet. In 1999, Shor et al. [37] proposed an algorithm capable of solving both the integer factorization problem and the discrete log problem in polynomial time on quantum computers. This means that the most commonly used asymmetric encryption algorithms would be broken and invalidated, as any entity listening in on communications would be able to read the data, breaching confidentiality and integrity. However, Shor's algorithm requires a sufficiently powerful quantum computer, specifically, one with a large number of stable qubits, which is beyond the capabilities of current quantum hardware.Various methods have been proposed to effectively factor an RSA key that could take as few as $2n + 3$ qubits [3] to as many as $9n + 2$ qubits [36], or for a 2048-bit key, between 4099 and 18434 qubits. Typical quantum computers today have between 20 and 400 qubits [28], but research is rapidly accelerating, fueled by significant investments from academia, national laboratories, governments, and industry.

To address this potential invalidation of current public-key cryptography, the field of post-quantum cryptography (PQC) has emerged. PQC aims to develop cryptographic schemes based on mathematical problems that remain hard to solve, even for quantum computers. The growth in this field has been especially accelerated by the National Institute of Standards and Technology (NIST), which created a call for PQC algorithm proposals in 2016. Through this initiative, NIST aims to stimulate research on quantum-safe cryptographic schemes and to standardize algorithms resistant to quantum attacks for government and public use, both now and in the foreseeable future. At the time of writing, there have been 4 rounds of submissions that have led to the standardization of 4 algorithms for use, with one algorithm standardized mere months ago.

Both legacy and smart power grids rely on network protocols such as TLS for security. Thus, the threat posed by quantum computers to these cryptographic systems extends to the power grid which is an especially serious concern given the potential for widespread disruption. This risk is amplified by known efforts from hostile nation-states to exploit power grids as tools for cyberattacks against adversaries. Hence, it is important to address the implementation of PQC algorithms and methods in the realm of smart grids. As smart grids continue to expand—and in some cases remain in their early stages—it is crucial to address these security challenges proactively. This allows for a timely transition to quantum-safe mechanisms and protocols, which in some cases will take significant efforts from stakeholders and operators, before quantum computers reach the capability to threaten power infrastructure.

Contributions: In this work, we: i) Develop a novel smart grid test network incorporating real-world equipment, infrastructure, and communication protocols, enabling realistic protocol benchmarking, security assessments, and facilitating future data collection for accurate digital twin simulations; and ii) Benchmark, analyze, and evaluate the performance of selected NIST post-quantum cryptography (PQC) candidate algorithms within the smart grid context by deploying them on various resource-constrained and low-capability IoT devices in our smart grid testbed.

The rest of this paper is organized as follows. In Section 2, we discuss the related work in the areas of PQC and smart grids, and describe the key evaluation criteria for smart grids and our testbed in Section 3. In Section 4 we describe our extensive experimental analysis and include our discussion about the results. We finally conclude with Section 5 and provide some directions for future work in Section 6.

2 Related Work

Research has increasingly explored both post-quantum cryptography and smart grids, including their intersection. In the area of post-quantum cryptography, there has been significant development since Shor's publication. In 2005, Regev et al. [29] proposed the *learning with errors* (LWE) problem over a lattice-based cryptosystem [29] for which no efficient solution is currently known, either by classical or quantum computation. Based on this mathematical problem, Langlois et al. [17] proposed the *module learning with errors* (M-LWE) problem, which generalizes the LWE problem by introducing modules as a mathematical structure to increase efficiency while maintaining security. This formulation retains the hardness assumptions of LWE while enabling cryptographic constructions that are more practical for implementation in real-world systems. M-LWE became the basis for the NIST PQC standard *Module Lattice Based Key Encapsulation Mechanism* (ML-KEM) in 2024 [25]. Likewise, a few encryption schemes have been proposed based on Berlekamp et al.'s [5] coding theory work from 1978 that resulted in the *Syndrome Decoding Problem* (SDP) [38]. This problem does not

Table 1: Algorithms Tested

Algorithm	NIST Security Level	Cryptographic Foundation
ML-KEM-512	1	Lattice-Based
ML-KEM-768	3	
ML-KEM-1024	5	
HQC-128	1	Code-Based
BIKE-L1		
Classic-McEliece-348864		
HQC-192	3	Code-Based
BIKE-L3		
Classic-McEliece-460896		
HQC-256	5	Code-Based
BIKE-L5		
Classic-McEliece-6688128		

have any known efficient quantum or classical solution and thus is a candidate for post-quantum cryptography algorithms. NIST evaluated three different algorithms for standardization where the security relies on the hardness of variants of the Syndrome Decoding Problem (SDP): *Hamming Quasi-Cyclic* (HQC), *Classic McEliece*, and *Bit Flipping Key Encapsulation* (BIKE) and as of March 2025, announced its decision to standardize HQC in this category.

In the power grid realm, there have been significant efforts to secure various aspects of power grid systems. In response to various power grid attacks there is an increasing concern about the security of these systems because of the large impact the compromise of such systems can have. One such attack was in 2015, when Ukraine's power grid across three provinces had various power outages due to a cyberattack. There have been works exploring mitigation techniques to further secure all power grids based on the analysis of past incidents [39] and security frameworks such as NIST's Smart Grid Framework [27] and application of MITRE's security framework [22] in power grids that aim to help stakeholders secure the power grid. Given the history of successful attacks on power grids, smart grids, with their expanded use of data, communications, and control systems, present an even larger attack surface and are more vulnerable to cyberattacks. Thus, various techniques and protocols for securing smart grids [7, 19, 21] have been proposed in the literature. These works cover various aspects of security, such as secure access control, intrusion detection, encryption, and authentication.

There has been some recent interest in applying PQC algorithms to smart grid settings where some works look at PQC implementation specifically at the neighborhood-level [8], while other works focus on such implementation at the physical power generation level [24]. Although there have been other efforts to benchmark PQC algorithms, they often overlook key evaluation factors. Nakka et al. [24] tested and compared individual PQC algorithms with current non-PQC standards in the distributed energy resources (DER)-data context, but didn't compare multiple PQC algorithms with each other. Satrya et al. [35] compared multiple PQC algorithms with each other, but their benchmarked algorithms have been dropped from NIST consideration for standardization since publication. Several papers have also presented custom implementations of post-quantum encryption schemes [4, 8], but these often lack the public scrutiny and independent verification necessary for reliable deployment and adoption.

To our knowledge, there has been no work that both compares multiple standardized PQC algorithms (or those being considered for standardization) and also evaluates their performance based on smart-grid-specific needs and requirements in a real-world smart grid testbed, which is a key contribution of our work.

3 Approach and Setup

When evaluating the performance of PQC algorithms for smart grid systems, it's important to consider the specific requirements of the various entities within the system. Key performance metrics include key generation latency, encapsulation/decapsulation latency, key length, and ciphertext length. Each of these metrics corresponds to particular needs or constraints of the smart grid infrastructure. For example, system bandwidth limits the feasible key and ciphertext sizes, while the computational capabilities of individual nodes influence acceptable key generation and encapsulation/decapsulation latencies. Additionally, a device's role within the system, and its required key rotation frequency, combined with its processing power, determines suitable key generation latency. Likewise, the frequency of communication and the level of security needed for a device will impact acceptable encapsulation and decapsulation times from a particular algorithm.

3.1 Algorithms Tested

The full set of algorithms we analyzed and tested can be found in Table 1. Each of these algorithms has an associated NIST security category, or level. Per NIST categorization standards [2], security Levels 1, 3, and 5 correspond to 128, 192, and 256-bit security, respectively. Likewise, the lowest security level we see in our algorithms, Level 1 (128-bit security), is projected to be sufficient through 2031 and beyond [2]. The majority of the results discussed in this paper pertain to the Level 1 algorithms, but we conducted experiments comparing the different algorithms' performance at all three security levels. For the majority of the experiments, the performance of each algorithm with respect to the others was the same when changing security levels, but simply with a longer runtime as the levels increased. Although NIST anticipates that Level 1 security will be sufficient for the foreseeable future, evaluating and implementing higher security levels enables operators and manufacturers to identify suitable devices for deployment in smart grid environments based on specific security requirements.

Table 1 also has a column for *Cryptographic Foundation*, which refers to the underlying mathematical hardness assumption that each algorithm depends on. The currently proposed algorithms fall under two categories, lattice-based and code-based, which correspond to the previously mentioned M-LWE problem and SDP, respectively. NIST has already standardized a lattice-based algorithm, ML-KEM, which is why we only chose to benchmark one lattice-based algorithm. At the time of testing, none of the three code-based algorithms had been selected by NIST for standardization. However, since then, NIST has chosen to standardize HQC, a decision that aligns with our experimental results, which show that HQC outperforms the other code-based schemes in most performance metrics. Standardizing algorithms based on diverse cryptographic schemes is desirable, as it mitigates the risk associated with a potential break in any single underlying mathematical structure. Currently, there

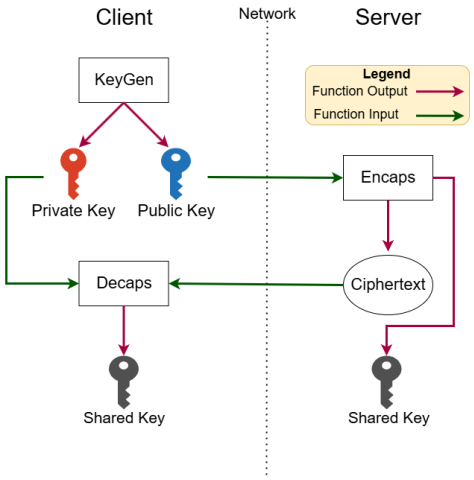


Figure 1: Simplified Key Encapsulation Mechanism (KEM) Protocol Outline

are no known feasible attacks on the M-LWE problem; however, if one is found, there will be alternative PQC schemes available. In Section 4, we examine the theoretical and empirical strengths and weaknesses of the three code-based algorithms, comparing them with one another as well as with the lattice-based scheme.

A KEM scheme involves key generation (*KeyGen*), encapsulation (*Encaps*), and decapsulation(*Decaps*) operations to establish a shared secret using post-quantum asymmetric cryptography and is illustrated in Figure 1. *KeyGen* generates a public and private key pair at the client. This public key can be openly published to other entities in the system. *Encaps* algorithm at the server uses the public key and results in a ciphertext and a shared key. The server sends this ciphertext to the client over an unprotected network where other entities can possibly intercept the ciphertext. The *Decaps* algorithm at the client takes the private key and ciphertext as input and outputs the shared key. This shared key is now only available at the client and the server and all future communication can use this shared key in a secure symmetric encryption algorithm.

3.2 Smart Grid Testbed

In an effort to obtain accurate and useful metrics, the PQC algorithms were tested and benchmarked in a testbed that has been created to accurately represent a real smart grid network. We deployed various network switches and a central router across multiple buildings, which house photovoltaic (PV) arrays or solar panels, battery backup systems, electric vehicle chargers, and other IoT and smart grid devices. The design of this network is based on the proposed NIST Framework and Roadmap for Smart Grid Interoperability Standards [13]. To simulate the various domains found in the draft, virtual local area networks (VLANs) were utilized. Each VLAN acts similarly to a domain, only allowing applicable and predefined network traffic and communication across different VLANs. Figure 2 illustrates the first iteration of our testbed network that connected 4 different buildings in our testing facility.

The implemented VLANs include the Neighborhood VLANs (110/120), which represent the networks that span the buildings and

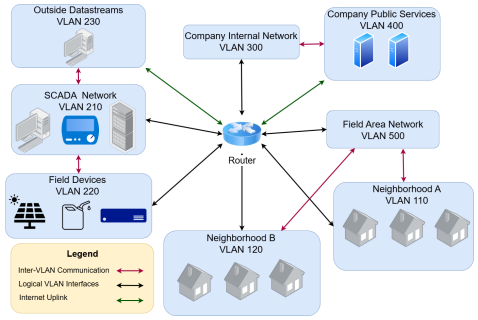


Figure 2: Logical Smart Grid Testbed Topology

include devices such as Smart Meters, and IoT devices.These VLANs can communicate with the Field Area Network (FAN) VLAN (500), which contains systems that aggregate data from the neighborhood VLANs and allow communication to the grid company. The SCADA VLAN (210) was implemented to hold various SCADA devices, such as Human Machine Interfaces (HMI), Remote Terminal Units (RTU), and other control servers. This network communicates with devices on the Field Device VLAN (220), which contains the actual power system devices such as photovoltaic (PV) arrays, generators, and transformers. The Outside Datastreams VLAN (230) was also created to account for any information coming from outside the SCADA Network that needs to be processed. For example, we placed a workstation on this VLAN that receives data from a local power company, which was then processed by our SCADA devices. Finally, we implemented company VLANs (300/400) where company IT resources and systems are located, such as the visualization web dashboards for the smart grid status and database servers that store aggregated information about the grid.

One notable feature of this design is the assumption that this network is privately owned and operated by the power company. This is the same model that most power companies operate and the utility information and data is not passed over the public internet, but only over a network that the power provider has complete control over. While there are various advantages and disadvantages of both this model and a model that has information traveling over the internet, small-scale smart grids utilize the private network model. The debate between using the public internet versus deploying widespread private networks remains an active area of research, but it is beyond the scope of this paper. Importantly, the security and privacy concerns discussed for smart grids are relevant to both network models. This is because both involve IoT end-user devices and publicly exposed components, such as smart meters and transformers, that often lack strict physical access controls to prevent tampering by malicious actors.

This network was deployed at a dedicated facility for microgrid research with the physical equipment and layout seen in Figure 3. This testbed network spans four buildings using wireless point-to-multipoint communication systems. The central building with the core router has a Ubiquiti Rocket 5AC Lite BaseStation. This base station connects three Ubiquiti LiteBeam bridges installed on the remaining 3 buildings. Each building includes multiple VLANs, providing network segmentation and security similar to that of strictly

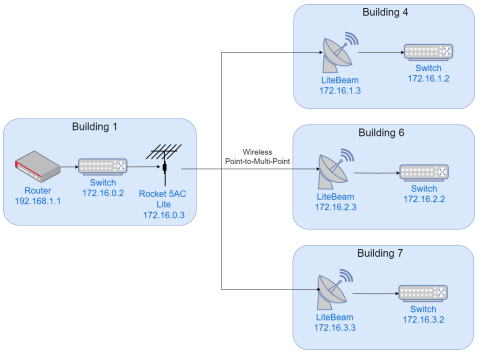


Figure 3: Smart Grid Testbed Network Topology

wired connections. The central router, located in the main building, enforces firewall rules to control traffic between the VLANs of different buildings and across buildings.

3.3 Threat Model

The highly connected nature of a smart grid network allows multiple attack surfaces to be exposed to potentially malicious actors. It is important to identify the intricate communication interfaces of the network and implement security and privacy mechanisms to limit the abilities of adversaries. NIST standardized key exchange and digital signature PQC algorithms might be implemented at any entity that uses asymmetric cryptography in the smart grid networks, but careful attention should be paid to the latency incurred by the different algorithms and the operational needs of the devices being upgraded to PQC. Following is a list of a few important devices and entities in the smart grid networks that need to be secured [30]:

Smart Meters facilitate two-way communication between the power company and the customer, transmitting data such as power usage and generation statistics. Physically installed on customers' homes and commercial buildings, these devices are easily accessible and therefore vulnerable to tampering. Given their direct connection to critical power infrastructure, securing smart meters is essential to maintaining the integrity and safety of the system.

Geographic Information Systems (GIS) collect and aggregate data from geographical features, which can then be used to assess power generation capabilities and potential in various regions [12]. Based on the insights provided by these systems, power generation decisions are made. Consequently, manipulation of GIS data can lead to inaccurate aggregations and poor decision-making, thus highlighting the importance of securing GIS infrastructure.

Supervisory Control and Data Acquisition (SCADA) systems collect data from various monitoring and energy-generating devices and are able to manipulate power generation output and configurations. SCADA systems include both hardware and software components, such as circuit breakers, sensors, programmable logic controllers (PLCs), and human-machine interfaces (HMIs). SCADA systems typically have more control over power grid operations, and thus are critical to the smooth operation of a smart grid.

Power Generation systems produce electricity for distribution to customers and vary based on the type of power plant, such as wind, tidal, hydro, and photovoltaic (PV). These systems may contain vulnerabilities in their software, hosted services, or authentication mechanisms [15]. Since such vulnerabilities can be exploited remotely over the network, securing communication channels within the smart grid is essential to maintaining the reliability and safety of power generation operations.

Table 2: Devices Tested

Device	Processor	Cores	Frequency
Raspberry Pi 5	ARM Cortex-A76	4	2.4 GHz
NVIDIA Jetson Orin Nano	ARM Cortex-A78AE	6	1.7 GHz
Raspberry Pi 400	ARM Cortex-A72	4	1.8 GHz
Raspberry Pi 3 Model B	ARM Cortex-A53	4	1.2 GHz
Desktop	Intel Core i7-6700	4	3.4 GHz

4 Experimental Analysis

We conducted our benchmarking experiments across a range of devices to obtain a comprehensive evaluation of the tested PQC algorithms' performance. Table 2 details the devices and their configurations that were used in our experiments. All devices used the latest Ubuntu-based Linux distributions for the corresponding devices. These devices were selected to reflect the computational capabilities and resource constraints typical of embedded systems and devices commonly used in the smart grid and Distributed Energy Resource (DER) systems.

We conducted two main types of experiments. The first involved benchmarking the three core operations of each KEM algorithm: key generation (*KeyGen*), encapsulation (*Encaps*), and decapsulation (*Decaps*). The second set of experiments evaluated the algorithms within the context of a TLS handshake, as detailed in Section 4.2. The code for our experiments can be found on Github. [1]

4.1 KEM Operation Experiments

In the first experiment, each operation of every algorithm was executed 200 times, and the average runtime was calculated. The results for the *KeyGen*, *Encaps*, and *Decaps* operations are presented in Figures 4, 6, and 5, respectively. We utilized Open Quantum Safe's *liboqs* C library [31], which provides implementations of multiple PQC algorithms.

Based on our experiments, ML-KEM was universally and significantly faster than all other tested algorithms, and thus necessitated the use of a logarithmic scale on the y-axis to present our results. Among all the devices tested, ML-KEM's runtime was under 1 *ms* and much faster than the next fastest algorithm. The only outlier was Classic McEliece's *Encaps* operation, where Classic McEliece was only slightly slower. The following analysis focuses on comparing the performance of the three code-based algorithms, motivated by the need to evaluate non-lattice-based PQC alternatives, in the event that, lattice-based cryptography is compromised.

While the figures present the performance of individual operations across each of the four devices, the following discussion focuses on comparing the average performance of the algorithms across all devices. Among the code-based cryptography algorithms in both the *KeyGen* (Figure 4) and *Decaps* (Figure 5) operations,

[1] https://github.com/NMSU-Prism/PQC-MG

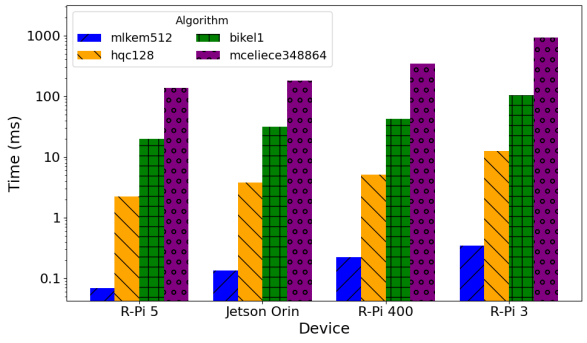


Figure 4: *KeyGen* Operation Runtime (Level 1) (Logarithmic)

HQC performed the best, next followed by BIKE, then finally Classic McEliece. This difference is most significant in the *KeyGen* stage, where on average, HQC (5.9 *ms*) was about 8.4x faster than BIKE (49.7 *ms*) and, about 67.3x faster than Classic McEliece (397.5 *ms*).

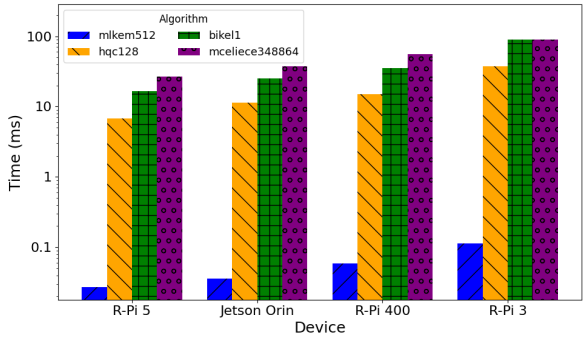


Figure 5: *Decaps* Operation Runtime (Level 1) (Logarithmic)

This is in contrast to the performance gaps in the *Decaps* operation, where, HQC (17.7 *ms*) is only about 2.3x faster than BIKE (41.5 *ms*) and about 3x faster than Classic McEliece (52.4 *ms*).

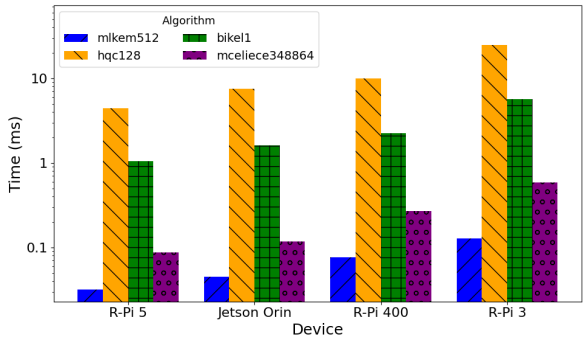


Figure 6: *Encaps* Operation Runtime (Level 1) (Logarithmic)

In contrast to the previous operations, the performance changes in the *Encaps* operation shown in Figure 6. Classic McEliece performed the best, next followed by BIKE, and finally HQC. On average, Classic McEliece incurred 0.266 *ms* delay, being 10.1x faster than BIKE (2.7 *ms*) and, on average, 44x faster than HQC (11.7 *ms*).

4.2 TLS Handshake Experiments

In the second set of experiments, we evaluated the performance of the different PQC algorithms when incorporated into a TLS connection. The TLS experiment measures the time it would take to establish a secure channel of communication over an insecure network, such as the internet, using PQC algorithms. The TLS handshake contains all three steps of the first test: *KeyGen*, *Encaps*, and *Decaps*, and is performed between a client and a server, in order to establish a shared key for future communication. This handshake also involved the server signing its message with ML-DSA, the PQC signature algorithm standardized by NIST. We utilize Open Quantum Safe's *oqsprovider* [33], an OpenSSL provider, which is essentially a plug-in for OpenSSL that allows a device to use custom algorithms, in our case, PQC algorithms, in TLS connections. In using OpenSSL, an industry default and standard, we are able to get more accurate performance comparisons for the TLS handshake, as any real-world OpenSSL management overhead will be included in the results of the different schemes.

Our experiments included i) a simple TLS handshake within the device to eliminate the network overhead and reflect the isolated overhead of PQC algorithms and OpenSSL operations; ii) TLS handshake between two devices (R-Pi 5) over a five-hop and a seven-hop building-to-building network; iii) evaluation of symmetric key operations using the key established during the TLS handshake; iv) evaluation of non-PQC operation overhead during a TLS handshake; and v) TLS Handshake runtime comparison of ML-KEM vs current OpenSSL algorithms for different levels of security.

To further provide insight into the performance of the PQC algorithms, we tested one additional algorithm included in OpenSSL, X25519. X25519 is an implementation of Elliptic Curve Diffie-Hellman (ECDH) key exchange, and although not post-quantum secure, it is the current default algorithm for key exchange in TLS 1.3. TLS 1.3 is utilized by most standard browsers and devices on the internet today and thus helps us compare the performance of PQC algorithms against a standard algorithm in wide use today.

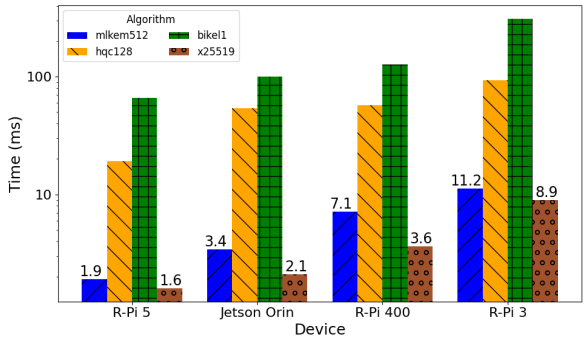


Figure 7: TLS Handshake Runtime (Level 1) (Logarithmic)

We did not include Classic McEliece for comparison in this experiment and discuss this choice more in detail in Section 4.3.2. In these sets of experiments, the TLS handshake was done on the same machine to eliminate any additional network transfer time. Therefore, our TLS handshake results represent the entire KEM process with any OpenSSL overhead, but without network overhead.

The results for our first TLS experiment are shown in Figure 7 and show ML-KEM outperforming the other PQC algorithms, again necessitating our use of a log scale on the y-axis to present our results. Interestingly, X25519 did outperform every PQC algorithm on all devices, but is closely followed by ML-KEM. X25519 (4.1 *ms*) was only, on average, 1.4x faster than ML-KEM (5.9 *ms*), which suggests that most real-world applications might be easily able to replace X25519 with ML-KEM in their TLS implementation without a significant performance hit. Among the PQC algorithms, ML-KEM (5.91 *ms*) was on average 9.5x faster than HQC (56 *ms*) which in turn was on average 2.7x faster than BIKE (151.2 *ms*).

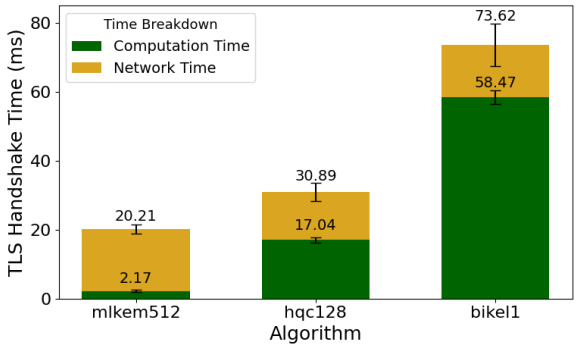


Figure 8: TLS handshake delay over five-Hop network From Building 1 to Building 7 on Raspberry Pi 5 (Level 1).

Our next experiment involved a TLS handshake between two devices (R-Pi 5) across a five-hop building-to-building network. The tests involved two R-Pi 5 devices in the same VLAN communicating over the smart grid testbed, thus simulating wireless communication between two SCADA devices that are physically separated but on the same network and thus incurring some network delay. Thus, by measuring the average time it takes to perform the handshake over a five-hop network, as shown in Figure 8, along with the previous experiments, we can quantify both the computation time, processing time, and the network transfer time incurred during a TLS handshake, which can then be used to support accurate smart grid digital twin simulations.

Table 3: Storage and network overhead (all values in Bytes)

Algorithm	Public Key Size	Private Key Size	Ciphertext Size	Total Bytes Over Network
ML-KEM-512	800	1632	768	1568
HQC-128	2249	2305	4433	6682
BIKE-L1	1541	5223	1573	3114
Classic-McEliece-348864	261120	6492	96	261216

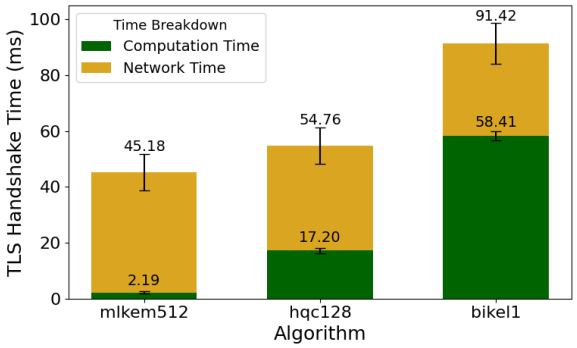


Figure 9: TLS handshake delay over seven-hop network from Building 4 to Building 7, bridged by Building 1, on Raspberry Pi 5 (Level 1).

Next, we repeated this experiment on a seven-hop path, from an edge building, Building 4, to another edge building, Building 7. These results can be seen in Figure 9. In this layout, the data packets had to traverse the link from Building 4 to the central building's omnidirectional antenna before being wirelessly routed to the target building (Building 7). As expected, the results reflect an increase in the network time in the latter experiments. However, both sets of experiments show ML-KEM's network time to be higher than the network time for HQC and BIKE. This is in direct contrast to ML-KEM's performance in terms of computation time and the amount of data to be transferred as reflected in Table 3. In Figure 8, ML-KEM-512 sends 1568 bytes of cryptographic information over the network during the handshake and takes on average 18 *ms*. But, HQC-128, which sends 6682 bytes of cryptographic information over the network, took an average network time of 13.8 *ms*.

We ran the experiments in an isolated network under our control and our investigations included network monitoring and Wireshark analysis of all traffic from the devices to eliminate the possibility of other processes and devices affecting our results. Our investigations into the different messages sent during TLS, such as ClientHello and other metadata information, did not indicate the reason for this extra delay. We hypothesize that the OS, in an attempt to optimize network bandwidth, is delaying the sending of the smaller ML-KEM packets. Mechanisms such as Nagle's algorithm, that attempt to consolidate multiple socket writes into a single packet near the maximum segment size (MSS), may be buffering ML-KEM packets, which are smaller than the MSS, to lower the number of packets sent in the network. Whereas HQC and BIKE, whose packets are much larger than the MSS, are not affected. Although further investigation remains part of our future work, this behavior suggests that efficient algorithms like ML-KEM may require special consideration when integrating into operating systems and cryptographic libraries to ensure their performance advantages are fully realized.

Next, we evaluate the performance of the devices in handling symmetric key encryption and decryption after a shared key has been established during TLS. In this experiment, a single 256-byte message was encrypted on the server side and decrypted on the

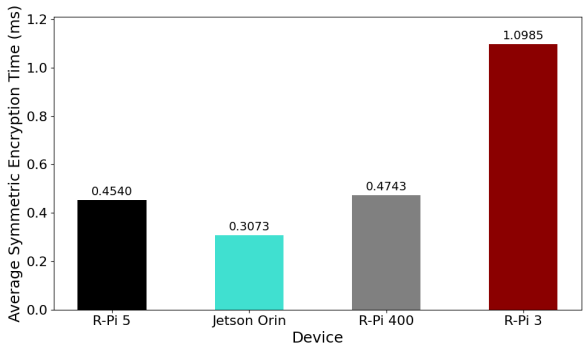


Figure 10: Computation time for Symmetric Encryption and Decryption operations after TLS Handshake

Table 4: Percentage of TLS handshake runtime spent on Non-PQC operations (OpenSSL operations, inter-process communication, OS scheduling, etc.)

Device	mlkem512	hqc128	bikel1
R-Pi 5	93.3%	29.9%	43.6%
Jetson Orin	93.7%	58.1%	42%
R-Pi 400	94.9%	47.2%	37.2%
R-Pi 3 Model B	94.8%	20%	35.8%
Average	94.2%	38.8%	39.6%

client side. This message length was selected based on the maximum payload size supported by Modbus—a widely used protocol in SCADA systems, which is 256 bytes [14]. By measuring the time required for encryption and decryption, we obtain an approximate upper bound for the runtime of future communications that occur after key exchange. Since the message is secured using symmetric cryptography, this additional time should depend only on the device's performance and remain independent of the specific PQC algorithm used during key exchange. As shown in Figure 10, the symmetric encryption and decryption times for the three newer devices were similar, ranging from 0.3 to 0.5 *ms*.

Next, we evaluate the overhead for OpenSSL operations and inter-process communication operations. This experiment does not include the network delay and was conducted by having the devices do a TLS handshake on a local port. We took the resultant TLS handshake time and eliminated the KEM operations delay to result in the non-PQC overhead, such as time spent on OpenSSL operations, OS level RPC calls, and scheduling. Although one might expect these operations to vary greatly depending on the operating system and device, we discovered surprisingly consistent results within each algorithm. The percentage of time spent on operations not directly related to PQC can be seen in Table 4. BIKE-L1 was more consistent than HQC-128, with on average spending 39.6% of its TLS handshake on non-PQC operations. For HQC-128, the percentage was not nearly as consistent between devices, but on average spent 38.8% of its TLS handshake time on non-PQC operations. One should expect faster algorithms to have a higher percentage in this context because the handshake would be bottlenecked by the OpenSSL operations and OS communications, which is reflected in ML-KEM, with non-PQC operation time of 94.2% on average.

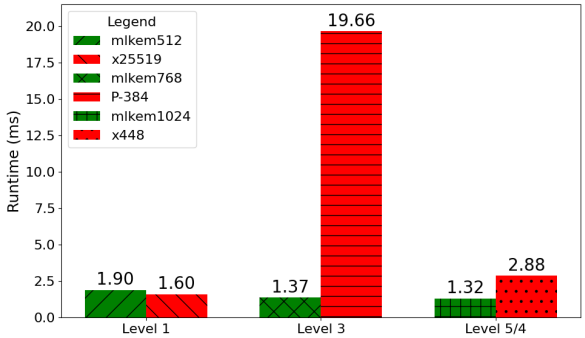


Figure 11: TLS handshake runtime comparison of ML-KEM (PQC) and currently used algorithms over different NIST Security Levels.

Lastly, we compare the performance of the best-performing PQC algorithm, ML-KEM, with the current OpenSSL algorithms categorized by security level on a R-Pi 5. The results were obtained by running the different algorithms in TLS handshakes on a local port to avoid network latency measurements in our evaluation. Currently, OpenSSL supports P-384, another Elliptic Curve (EC) algorithm, which claims to offer 192-bit security (NIST Security Level 2). OpenSSL also supports X448, also an EC algorithm, which claims to offer 224-bit security (NIST Security Level 4). We note that Level 4 is the highest security level offered by non-PQC algorithms in OpenSSL. As reflected in Figure 11, in both Level 1 and Level 5/4, the PQC algorithm performed similar to the non-PQC algorithm. ML-KEM performed 0.3 *ms* slower in the lower level and was 1.56 *ms* faster than X448 in the higher level. Notably, P-384 performed significantly worse than all the ML-KEM and other EC algorithms for Level 3. This is likely because P-384 operates over a 384-bit prime field using a Weierstrass curve form, unlike X25519 and X448, which are based on Montgomery curves optimized for performance. Montgomery Curves, proposed by Peter Montgomery [23], allow for efficient and secure scalar multiplication using the Montgomery ladder technique, which reduces the number of required field operations. The absence of this optimization in P-384 likely contributes to its slower performance. These findings highlight that ML-KEM, even at its highest security level, is capable of outperforming or closely matching the best algorithms currently supported by OpenSSL.

4.3 Findings and Implications

In both the KEM operations experiment and the TLS handshake experiment, ML-KEM was much faster than the other tested PQC algorithms. ML-KEM is the only lattice-based cryptographic scheme that was tested, which we believe played a part in its better runtime. One of the fundamental mathematical operations that is used in ML-KEM (and lattice-based cryptography more broadly) is the Number Theoretic Transform (NTT) which allows more efficient

polynomial multiplication [34]. Accordingly, NTT can be improved further with the use of CPU instruction set architecture (ISA) extensions. There has been work in this field to improve NTT with such extensions including Longa et al. [20] using AVX2 instructions to increase performance and Lei et al. [18] using AVX512 instructions to do the same, and both Alkim et al. [1] and Dolmeta et al. [11] specifically accelerating ML-KEM on RISC-V architectures. Finally, in addition to ISA extensions, there has been other work on accelerating and optimizing NTT computations by Bisheh-Niasar et al. [6]. Because the NTT is a fundamental and frequent operation in ML-KEM, and there has been work into creating efficient, CPU-level, implementations of it, we believe ML-KEM received a major boost in its performance. Likewise, the creators of *liboqs*, Open Quantum Safe, specifically note that ML-KEM indeed utilizes these CPU extensions [32]. For these reasons, further analysis below excludes discussion and consideration of ML-KEM, as it universally outperformed all the other PQC algorithms.

4.3.1 Standalone Algorithm Operation. In the KEM operations experiment, the slowest operation was *KeyGen* across all algorithms. On our slowest device, the Raspberry Pi 3, there was a difference of 915.83 *ms*, almost an entire second, between HQC and Classic McEliece, and a difference of 91.98 *ms* between HQC and BIKE. The differing performance profiles of the algorithms across *KeyGen*, *Encaps*, and *Decaps* operations highlight the nuanced considerations smart grid operators must account for when selecting an algorithm for a specific application. Although NIST has selected HQC for standardization, BIKE may still be suitable in certain contexts depending on operational constraints. As seen in the performance figures, the *Decaps* operation tends to have a greater runtime impact than *Encaps*, but remains less significant than *KeyGen*, which is the most computationally intensive of the three.

4.3.2 TLS Handshake Performance. Classic McEliece was excluded from the TLS handshake experiment for two main reasons. Firstly, it is not implemented in Open Quantum Safe's *oqsprovider*, and a custom implementation would be inconsistent with the current OQS code, making comparisons unfair. Secondly, Classic McEliece performs poorly in the *KeyGen* step relative to other schemes, significantly skewing results. This limitation is inherent and acknowledged, as the algorithm is not designed for frequent key generation. TLS sessions, which typically last several minutes (with OpenSSL's default timeout set to 300 seconds [16]), upon a session expiration, require a full handshake including all KEM operations. Classic McEliece is better suited for scenarios with infrequent key rotation and where efficient *Encaps* operations are prioritized. Therefore, it was not included in the TLS handshake evaluation.

As for the two other code-based algorithms tested, HQC and BIKE, the results for the entire TLS handshake are consistent with expected results based on individual KEM operation runtimes. HQC outperformed BIKE on each device, HQC being on average 2.7x faster than BIKE. On the slowest device being a difference of 216.5 *ms*, and on the fastest device a difference of 47.2 *ms*. These results are consistent with Section 4.3.1's reasoning on ranking algorithms based on their order in higher-impact operations. HQC outperformed BIKE in the more significant operations, and thus HQC outperformed BIKE also in the entire handshake (performing all KEM operations).

The network time in TLS connections is heavily dependent on physical network infrastructure. In our experiments over the five-hop and seven-hop networks, our wireless connection bandwidths were typically between 150 and 250 Mbps. Smart grid equipment, especially at the field area network levels, may have much lower bandwidths due to their usage of low power protocols/devices such as LoRaWAN and Wi-SUN [10]. The differences in data transferred in each algorithm may be more pronounced in such systems. So, when analyzing which of the tested algorithms works best in smart grid systems, the available bandwidth should be considered as well. HQC-128 transfers 6682 bytes over the network, while BIKE-L1 transfers 3114 bytes, 3568 less bytes than HQC-128. For example, LoRaWAN operates at bandwidths ranging from 10 to 50 Kbps [9, 26]. Assuming the lower end of 10 Kbps, HQC-128 would take approximately 2.85 seconds longer to transmit than BIKE-L1, while at the higher end of 50 Kbps, the difference drops to around 570 *ms*. Whether this additional delay is acceptable depends on the specific system requirements, and in long-range wireless networks, where latency is inherently higher, such differences may be negligible.

5 Conclusion

As smart grids continue to evolve, ensuring their security against emerging threats – particularly quantum computing – is of great importance. This paper has explored the impact of post-quantum cryptography (PQC) within the context of smart grids, benchmarking several leading PQC algorithms to assess their viability in real-world smart grid environments. Our analysis demonstrates that ML-KEM consistently outperforms other PQC algorithms in terms of efficiency, making it the most suitable candidate for integration into smart grid infrastructures. However, in scenarios where diversification of cryptographic foundations is desired to mitigate risks associated with potential future breakthroughs in cryptography, code-based algorithms must be considered. Our analysis demonstrates that HQC is best suited to this role of secondary PQC algorithm, provided that the modest increase in bandwidth requirements, compared to BIKE, can be supported by the smart grid network and devices. While BIKE offers advantages in terms of smaller ciphertext sizes and lower bandwidth usage, HQC consistently outperforms it in overall cryptographic operation performance, making it the more practical choice for smart grid applications with adequate bandwidth.

6 Future Work

Our work opens several promising avenues for future research. The gathered results and the testbed can be used in network simulations to facilitate accurate and realistic digital-twin smart grid simulations. The experiments can be expanded to a physical field area network using previously mentioned protocols (LoRaWAN, Wi-SUN, etc.). This would involve embedded implementations of the PQC algorithms and help provide insights into the comparison and applicability of BIKE and HQC in such highly resource-constrained environments. With access to such a network, the various Low-Power Wide-Area Network communication protocols could be evaluated in the context of PQC. The PQC algorithms can be incorporated within networks that have real-time SCADA devices and sensor communications. In doing so, the effectiveness

of using the TLS protocol vs real-time Operational Technology (OT) protocols can be evaluated. We also plan to expand the testbed by integrating additional smart grid devices and implementing PQC algorithms in real-world smart grid communication scenarios.

Acknowledgements

This research was partially funded by the US National Science Foundation under grant #2417062, and the US Department of Energy award numbers DE-CR0000061 and DE-OE-0000947. Any opinions, findings, conclusions, or recommendations expressed in this material are solely those of the authors and do not necessarily reflect the views of the US federal agencies.

References

[1] Erdem Alkim, Hülya Evkan, Norman Lahr, Ruben Niederhagen, and Richard Petri. 2020. ISA Extensions for Finite Field Arithmetic - Accelerating Kyber and NewHope on RISC-V. Cryptology ePrint Archive, Paper 2020/049. https://eprint.iacr.org/2020/049

[2] Elaine Barker. 2020. *Recommendation for Key Management.* Technical Report NIST Special Publication (SP) 800-57, Rev. 5, Includes updates as of May 04, 2020. National Institute of Standards and Technology, Gaithersburg, MD. doi:10.6028/NIST.SP.800-57pt1r5

[3] Stephane Beauregard. 2003. Circuit for Shor's algorithm using 2n+3 qubits. arXiv:quant-ph/0205095 [quant-ph] https://arxiv.org/abs/quant-ph/0205095

[4] Basudeb Bera and Biplab Sikdar. 2024. Securing Post-Quantum Communication for Smart Grid Applications. In *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm).* 555–561. doi:10.1109/SmartGridComm60555.2024.10738045

[5] E. Berlekamp, R. McEliece, and H. van Tilborg. 1978. On the inherent intractability of certain coding problems (Corresp.). *IEEE Transactions on Information Theory* 24, 3 (1978), 384–386. doi:10.1109/TIT.1978.1055873

[6] Mojtaba Bisheh-Niasar, Daniel Lo, Anjana Parthasarathy, Blake Pelton, Bharat Pillilli, and Bryan Kelly. 2023. PQC Cloudization: Rapid Prototyping of Scalable NTT /INTT Architecture to Accelerate Kyber. In *2023 IEEE Physical Assurance and Inspection of Electronics (PAINE).* 1–7. doi:10.1109/PAINE58317.2023.10318029

[7] Shehzad Ashraf Chaudhry, Jamel Nebhan, Khalid Yahya, and Fadi Al-Turjman. 2022. A Privacy Enhanced Authentication Scheme for Securing Smart Grid Infrastructure. *IEEE Transactions on Industrial Informatics* 18, 7 (2022), 5000–5006. doi:10.1109/TII.2021.3119685

[8] Chi Cheng, Yue Qin, Rongxing Lu, Tao Jiang, and Tsuyoshi Takagi. 2019. Batten Down the Hatches: Securing Neighborhood Area Networks of Smart Grid in the Quantum Era. *IEEE Transactions on Smart Grid* 10, 6 (2019), 6386–6395. doi:10.1109/TSG.2019.2903836

[9] Phui San Cheong, Johan Bergs, Chris Hawinkel, and Jeroen Famaey. 2017. Comparison of LoRaWAN classes and their power consumption. In *2017 IEEE Symposium on Communications and Vehicular Technology (SCVT).* 1–6. doi:10.1109/SCVT.2017.8240313

[10] Claudio Ferreira Dias, Lucas Diogo De Mendonça, Karoline Ferreira Tornisiello, Andre Saito Guerreiro, Eduardo Rodrigues De Lima, and Gustavo Fraidenraich. 2022. A Straightforward Method to Promote Effective Interoperability in Wi-SUN FAN Smart Grid Networks. In *2022 IEEE Latin-American Conference on Communications (LATINCOM).* 1–5. doi:10.1109/LATINCOM56090.2022.10000542

[11] Alessandra Dolmeta, Emanuele Valpreda, Maurizio Martina, and Guido Masera. 2024. Implementation and integration of NTT/INTT accelerator on RISC-V for CRYSTALS-Kyber *(CF '24 Companion).* Association for Computing Machinery, New York, NY, USA, 59–62. doi:10.1145/3637543.3652872

[12] Lavanya Gnanasekaran and Sean Monemi. 2018. GIS Role in Smart Grid. In *2018 IEEE Conference on Technologies for Sustainability (SusTech).* 1–5. doi:10.1109/SusTech.2018.8671383

[13] Avi Gopstein, Cuong Nguyen, Cheyney O'Fallon, Nelson Hastings, and David A. Wollman. 2021. NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 4.0. doi:10.6028/NIST.SP.1108r4

[14] Vasile Gheorghiță Găitan, Ionel Zagan, and Nicoleta Cristina Găitan. 2025. Modbus RTU Protocol Timing Evaluation for Scattered Holding Register Read and ModbusE-Related Implementation. *Processes* 13, 2 (2025). doi:10.3390/pr13020367

[15] Jonathan Gerardo Hurtado, Christian Birk Jones, Jay Johnson, and Brian J. Wright. 2024. *DEReliction: A Cybersecurity Vulnerability Assessment Methodology for Distributed Energy Resources.* Technical Report. Sandia National Lab. (SNL-NM), Albuquerque, NM (United States). doi:10.2172/2480142

[16] OpenSSL Software Services Inc. 2016. https://docs.openssl.org/master/man3/SSL_get_default_timeout/

[17] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography* 75, 3 (June 2015), 565–599. doi:10.1007/s10623-014-9938-4

[18] Douwei Lei, Debiao He, Cong Peng, Min Luo, Zhe Liu, and Xinyi Huang. 2023. Faster implementation of ideal lattice-based cryptography using avx512. *ACM Transactions on Embedded Computing Systems* 22, 5 (2023), 1–18.

[19] Xu Li, Xiaohui Liang, Rongxing Lu, Xuemin Shen, Xiaodong Lin, and Haojin Zhu. 2012. Securing smart grid: cyber attacks, countermeasures, and challenges. *IEEE Communications Magazine* 50, 8 (2012), 38–45. doi:10.1109/MCOM.2012.6257525

[20] Patrick Longa and Michael Naehrig. 2016. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. Cryptology ePrint Archive, Paper 2016/504. https://eprint.iacr.org/2016/504

[21] Anthony R. Metke and Randy L. Ekl. 2010. Smart Grid security technology. In *2010 Innovative Smart Grid Technologies (ISGT).* 1–7. doi:10.1109/ISGT.2010.5434760

[22] MITRE. 2023. MITRE - Ensuring a Resilient, Sustainable Power Grid for a New Energy Landscape. https://www.mitre.org/news-insights/impact-story/ensuring-resilient-sustainable-power-grid-new-energy-landscape

[23] Peter L. Montgomery. 1987. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Math. Comp.* 48, 177 (1987), 243–264.

[24] Kalyan Nakka, Seerin Ahmad, Taesic Kim, Logan Atkinson, and Habib M. Ammari. 2024. Post-Quantum Cryptography (PQC)-Grade IEEE 2030.5 for Quantum Secure Distributed Energy Resources Networks. In *2024 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT).* 1–5. doi:10.1109/ISGT59692.2024.10454235

[25] National Institute of Standards and Technology (NIST). 2024. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard.* Federal Information Processing Standards Publication 203. National Institute of Standards and Technology. https://csrc.nist.gov/pubs/fips/203/final Accessed: 2025-01-15.

[26] Jorge Navarro-Ortiz, Sandra Sendra, Pablo Ameigeiras, and Juan M. Lopez-Soler. 2018. Integration of LoRaWAN and 4G/5G for the Industrial Internet of Things. *IEEE Communications Magazine* 56, 2 (2018), 60–67. doi:10.1109/MCOM.2018.1700625

[27] NIST. 2016. NIST - Smart Grid Framework. https://www.nist.gov/ctl/smart-connected-systems-division/smart-grid-group/smart-grid-framework

[28] Timothy Proctor, Kevin Young, Andrew D Baczewski, and Robin Blume-Kohout. 2025. Benchmarking quantum computers. *Nature Reviews Physics* (2025), 1–14.

[29] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing* (Baltimore, MD, USA) *(STOC '05).* Association for Computing Machinery, New York, NY, USA, 84–93. doi:10.1145/1060590.1060603

[30] Javier Sande Ríos, Jesús Canal Sánchez, Carmen Manzano Hernandez, and Sergio Pastrana. 2024. Threat analysis and adversarial model for Smart Grids. arXiv:2406.11716 [cs.CR] https://arxiv.org/abs/2406.11716

[31] Open Quantum Safe. 2025. liboqs. https://github.com/open-quantum-safe/liboqs. Accessed: 2025-02-17.

[32] Open Quantum Safe. 2025. ML-KEM Specification. https://openquantumsafe.org/liboqs/algorithms/kem/ml-kem.html. Accessed: 2025-02-20.

[33] Open Quantum Safe. 2025. oqsprovider. https://github.com/open-quantum-safe/oqs-provider. Accessed: 2025-02-17.

[34] Ardianto Satriawan, Rella Mareta, and Hanho Lee. 2024. A Complete Beginner Guide to the Number Theoretic Transform (NTT). *Cryptology ePrint Archive* (2024).

[35] Gandeva Bayu Satrya, Yosafat Marselino Agus, and Adel Ben Mnaouer. 2023. A Comparative Study of Post-Quantum Cryptographic Algorithm Implementations for Secure and Efficient Energy Systems Monitoring. *Electronics* 12, 18 (2023). doi:10.3390/electronics12183824

[36] A Shaheed Nehal, Mubasheer Farhan, and YS Sunad. 2020. Quantum Cryptography-Breaking RSA Encryption using Quantum Computing with Shor's Algorithm. *International Journal For Technological Research in Engineering* 8, 1 (2020).

[37] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332. http://www.jstor.org/stable/2653075

[38] Jacques Stern. 1994. A new identification scheme based on syndrome decoding. In *Advances in Cryptology — CRYPTO' 93*, Douglas R. Stinson (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 13–21.

[39] David E. Whitehead, Kevin Owens, Dennis Gammel, and Jess Smith. 2017. Ukraine cyber-induced power outage: Analysis and practical mitigation strategies. In *2017 70th Annual Conference for Protective Relay Engineers (CPRE).* 1–8. doi:10.1109/CPRE.2017.8090056