

Article

Implementing Post-Quantum Cryptography to Industrial Wireless Networks

Mario Keh *  and Yuhua Chen * 

Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204, USA

* Correspondence: mkeh@cougarnet.uh.edu (M.K.); yuhuachen@uh.edu (Y.C.)

Abstract

The purpose of this research is to introduce a scalable system of an improved security for devices connected to a wireless network using the Advanced Encryption Standard with a Post-Quantum Cryptography standard FIPS203, Module Lattice–Key Encapsulation Mechanism (ML-KEM). This implementation is to address concerns regarding compromised network security or bad actors sniffing packets through a data bus to collect unintended compromised data. The ML-KEM is used to create a shared secret that is used as the symmetric key that will enable the encryption and decryption method for the ciphertexts between the client and the host. This research provides a baseline implementation of added security against Quantum Computers by using an encapsulation method for key pairs, digital signatures for data integrity, and added difficulties for side-channel attacks from unauthorized users. Devices that are older than the WiFi6-compliant standard also have additional vulnerability of not having the WiFi Protected Access (WPA) third-generation security, which this work addresses. This paper proposes an added layer of encryption security that is sufficient to protect information within the network that has been compromised by an unauthorized user. Based on the findings, new features, utilities and improvements are recommended that can modernize the needs of the industry.

Keywords: advanced encryption standard (AES); post-quantum cryptography (PQC); wireless fidelity (WiFi); user access; module lattice–key encapsulation mechanism (ML-KEM); FIPS203; local area network (LAN); Internet of Things (IoT) security

1. Introduction

Cybersecurity is a constantly evolving field, with new forms of connectivity, not just computers but Things such as gateways, sensor nodes, monitors and so many more that gather useful data. This creates interests in not only prevention and awareness, but also detection [1] using machine learning techniques. This is however a reactive response to threats which in many cases will incur losses and damages that can greatly affect businesses and industries as a whole. By introducing obscurity to data prior to transmission, the attacker faces a greater deal of difficulty when attempting to freely access information. Layering the encryption framework adds a deeper level of security [2], in many cases with requirements of more resources, runtime and complexity.

Many companies today outsource their primary storage and security to third-party applications. This also involves using kernel-level accessing software to detect adversarial behaviors using machine learning techniques; however, trusting security to external parties in itself is a major risk, especially when it comes to cloud outages [3], and even more so when the software itself becomes a risk [4–6] such as a faulty software update which can



Academic Editors: Dariusz Rzońca and Tomasz Rak

Received: 31 March 2026

Revised: 14 April 2026

Accepted: 21 April 2026

Published: 26 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

cause a mass IT outage. Still, when relying on a third-party security source, whether it is for detection of malicious activities, triage or storage of data on a cloud server, the risks are evident because of the dependencies it creates on these services. There is no guarantee of security once the data has left local or internal handling.

Though the offensive side of the cybersecurity is still in development, time is still of the essence to fortify the defensive side of Harvest Now, Decrypt Later. For that we reiterate the importance of creating a more resilient suite of defenses, building from the outside inwards. This starts from the external network where most of the possible malicious activities can occur, such as backdoors, Trojan horses, worming, credential compromises, and many other exploits. Once within the WiFi network, this relies heavily on the internal obscurities to prevent further data compromises from occurring as we have witnessed in modern times.

The most common ways in enterprise that network credentialing gets compromised are either phishing or members unintentionally revealing key identifiers that attackers can use to spoof, duplicate, manipulate or even trick the access control. Once inside the network, the attacker is free to listen to traffic within the network that is unencrypted or rather unsecured. WPA3, which was introduced in WiFi6, provides additional protection from eavesdroppers, packet sniffers, or attackers outside of the network, but it does not provide protection to the individual devices in the network.

Because of the vulnerability that is still present for unauthorized users, we introduce enhanced securities that are present exclusively for intended devices. Another layer of authentication is added by including digital hash signatures that validate that the data comes from an intended user; otherwise, the payload is ignored, preventing the defensive device from revealing any secrets. Heart-beat checks can also be included in the security measures which allows every device pair or device mesh to change their shared secrets, adding to the cryptographic security. A purely hardware-accelerated encapsulation–decapsulation architecture will speed this process up without interrupting the flow of data; however, it should be padded not to reveal a different power output which can be exploited by side-channel attacks to discover when this event happens.

Regardless, the most vulnerable activity that is present once the adversary is within the network is collecting all the data possible. The threat of Harvest Now, Decrypt Later creates vulnerabilities to all modern encrypted data that contains sensitive information using Elliptic Curve Cryptography (ECC) or Rivest–Shamir–Adleman (RSA). Attackers are waiting for Q-Day [7], and it is within the best interests of companies today to prepare their securities before the Cryptographically Relevant Quantum Computer (CRQC) becomes relevant. Once Quantum Computing becomes feasible, data that still relies on classical encryption methods will become vulnerable to brute force attacks that rely on significantly less time.

The main feature this paper introduces is an element of Post-Quantum Cryptography (PQC), a Key Encapsulation Mechanism (KEM) which is standardized in the FIPS 203. This allows two devices over an unsecured network or channel to establish a shared secret by transmitting a symmetric encryption key instead of a long encrypted message with a public key [8].

For this paper, we propose a framework for in-network encryption with an additional security to PQC prior to the introduction of CRQC [9]. The contributions of this paper are as follows:

- **Improved Data Anonymity:** This is achieved by adding an additional encryption while being transmitted through the WiFi network. Even though the attacker is within the network, it cannot see the raw data that is being transmitted without the decryption method.

- **Improved Encryption Security:** By using the Module Lattice–Key Encapsulation Mechanism, this adds a layer of difficulty in finding the shared secret that is required to decrypt the transmitting message. In short, by hiding the shared secret in a ciphertext, the attacker is required to have traded the key pair in the first step to retrieve the shared secret required to decrypt transmissions.
- **Reduced Susceptibility For Side-Channel Attacks:** By increasing the steps needed to latch onto the private network, attackers would now need to recognize if they are obtaining the public key needed to decrypt messages or if they are actually trying to decipher the first part of the transmission to obtain the shared secret.
- **Added Post-Quantum Computer Resistance:** By introducing a PQC method, ML-KEM (FIPS203), the encryption scheme now requires more effort for attackers beyond a basic Quantum Computer to begin cracking through the security of the network.

For this paper, we discuss the background of cyber threats, their importance, and the need for improvements and awareness in Section 2. We also discuss works that are similar in application, albeit with a different user group, and ways we can improve it. Then in Section 3 of this paper we present the proposed work. In Section 4, we describe the implementation and present results of this research. We discuss the significance of our findings and ways to improve the security for many industries to follow in Section 5, and conclude in Section 6.

2. Background and Related Work

2.1. Security Vulnerability of Industrial Networks

The Advanced Encryption Standard (AES) [10] is still a commonly adopted standard for encryption in use today. It is the successor to the previous standard, the Data Encryption Standard (DES), albeit a more difficult scheme to implement. Nonetheless, it is widely used to protect sensitive data passing through any network. However, with emerging advances in developing the CRQC, the possibility of Quantum Computing brings a threat to classical cryptography such as the ECC, RSA, Secure Hash Algorithm (SHA), and similar. By making these cryptographic algorithms obsolete, it brings vulnerabilities to much of the industrial data that relies on these for cybersecurity frameworks.

The vulnerability of most industry networks tends to start at their employees, when they fall victim to phishing attempts [11]. Though many cybersecurity awareness programs tend to alleviate the complacency by occasionally sending testing links via emails, this does not eliminate events where employees accidentally click links that eventually compromise their end-node access. Other unintentional revelations that employees give attackers are displays of their identification badges, trinkets or objects unique to the company. These give attackers sufficient information to gain access to the physical locations such as server rooms. As demonstrated, these do not need much sophistication to give attackers entry to sensitive areas.

Though the entry of unwanted actors is not absolutely preventable, the Cybersecurity and Infrastructure Security Agency (CISA) regularly updates best practices regarding protection, containment, eradication and hardening defenses. However, these actions are reactive and do not prevent any damages, instead reducing the damage of data breaches. This is where it becomes crucial to be proactive in protecting sensitive data, where it should be assumed that the network has hidden bad actors. Complacency will always cause reduction in awareness, safety and reaction, which can end catastrophically.

Once attackers are within the network, there is no telling what level of damage they will cause to the company they have compromised [12]. By assuming they are always within the network, a zero-trust security shall be implemented, treating all transmissions on the network as malicious until decrypted or validated. The AES and RSA combination

encryption schemes are the most commonly implemented in the industry [13]; however, the new claim is that with Quantum Computers becoming more cryptographically relevant with the mentions of Grover and Shor's algorithms, both these schemes have their security reduced through brute force attacks. The AES has also been found to be vulnerable to side-channel attacks, while brute forcing will take significant time with classical computers. The RSA algorithm similarly is vulnerable to side-channel attacks as well, most notably the timing attack. Both algorithms can increase security by increasing the key size, but that adds to computation as a trade-off.

For side-channel attacks, there are ways to improve security against the timing attacks. Firstly, we are able to add random delays, obscure predictable times, or make all operations run at fixed amount of time. This does however sound resourcefully wasteful because of the "no operations" that occur to pad the execution time. In a real-time scenario, this can be devastating for timing critical operations, especially in Internet of Things (IoT) wireless sensor networks. Power analysis attacks can also have all their operations run on a fixed amount of radiated emissions, which again introduces waste for end-point nodes. Brute force attacks in this context are not as much of a threat because of the time it takes, unless the attacker has randomly guessed successfully in very few tries or the defensive side has very little security.

Devices using the Preshared Key (PSK) with WPA2 security have already been compromised [14] as this is normally the password used to gain access to the network. The enterprise security is also a weak point if the individual credentials have been cracked. Another issue with WPA2 security is an exploit within its protocol, KRACK [15], a replay attack discovered around 2016 while its effectiveness was studied. An event like this would cause quite some panic in IT professionals who rely heavily on WPA2 security [16], and now the WPA3 protocol, which does not guarantee a similar thing would not happen.

Regardless of the vector of attack, the bigger concern is the network being accessed. As mentioned previously, the encryption that is normally based on the AES-128 already loses its confidentiality depending on the mode of security. This emphasizes the need to place additional security for sensitive information within the network traffic, as Man-In-The-Middle attacks are typical for abusers that have hijacked the network. As mentioned above, most of the access are phished out from unsuspecting users which make it easier for attackers to gain access.

The assumption is that the data is encrypted between the device and router, which does not promise absolute secrecy to users within the network. Attacks such as packet sniffing allows the data to be eavesdropped, giving the attacker access to decrypt or decipher the message. Something like Address Resolution Protocol (ARP) spoofing by redirecting the message directly to the attacker is also a possibility. Other forms such as encryption cracking, setting up Evil Twin networks, and other variants are just a few examples of many ways that attackers attempt to retrieve unintended secrets.

News of breaches occurring shatters the trust and reputation of companies, further emphasizing the need for preventative defenses rather than reactive actions. By preventing the access pre-emptively, this reduces the chances of consequential damages occurring. For this reason, the encryption of the data, separate from the WPA protocol, is deemed necessary for added secrecy from unauthorized users.

2.2. Related Work

AES development [17–19] is still the most popular encryption framework for IoT development, with improvements being more lightweight, using less power, and fewer hardware resources. However, most of the implementations of AES do not add any further security other than stacking encryption over encryption. Once the keys have been

compromised, the algorithm can now be actively used as normal for the unintended users. Once the CRQC becomes feasible, the deployment of Grover's algorithm makes brute forcing these keys quadratically faster.

The Raspberry Pi has WiFi and Bluetooth already configured on its own UNIX Operating System, and it is quick to set up to use Python scripts, which makes it an ideal platform to demonstrate the AES security framework. Once the framework has been established, verified, validated and scalable on the platform, it makes the proof of concept much easier to deploy on other platforms, essentially by changing the language from Python to embedded software languages such as C or hardware description languages such as Verilog, or a combination of hardware and software using High-Level Synthesis.

An example of a Raspberry Pi project [20] uses only the AES to secure data in the LAN to emulate a storage server holding files within the network without the FIPS 203 ML-KEM. Without the ML-KEM to hide the key, this makes the system vulnerable to quantum attacks which would compromise the round key, which would be the shared secret. It fulfills the first part of the need in the industry, transferring files encrypted over the network; however, once the key is compromised, it defeats the security that was intended. In an industrial setting, this is far more catastrophic than consumer data being compromised from a DropBox, Google Drive or similar cloud network.

Many companies also utilize cloud storage services such as Microsoft Azure, Google Cloud, Amazon Web Services and similar. It should also be noted that these cloud services use a shared responsibility model, where the service provider is only responsible for their infrastructure, but the security within the cloud is the responsibility of the client. This creates a greater urgency to deploy encryption on the data prior to transmission that is not only part of the intranet but also when data leak to the open Internet is possible.

WiFi is a telemetry transport protocol. However, the way the data is passed along does not just have to be particular packet, but can have a modified payload sent over the socket. Sending raw data packets over WiFi requires special software or hardware configurations, and while not necessary for the applications here, it can be a useful tool to monitor traffic over the network. Also, not all modems support the monitoring mode which is necessary to send raw data.

Message Queuing Telemetry Transport (MQTT) is an IoT messaging protocol used to communicate between devices on the network (not only WiFi) using a Publication and Subscription (PubSub) model. This protocol inherently has no security built in by default, and relies on the responsibility of the architectures to implement their own forms of security. Normally, MQTT subscribers would listen to messages from publishers that are normally sensor nodes, and then publish the responses to another server or host. However, MQTT allows control of these end nodes, which can become a security risk should the publisher become compromised. This poses two problems: the first is if the publisher is able to post spoof data over the PubSub, and the second is if the compromised user is able to send malicious controls over MQTT.

Currently, the WiFi Alliance presents a security protocol WPA3 [21] which claims to include individual encryption for all devices connected to its network, further improving on its predecessor WPA2. Still, this creates an issue of trusting security based on claims rather than explicitly implementing the security on the devices themselves between its intended end nodes or destinations.

Despite being posed as a contender to the Raspberry Pi, the ESP8266 [22] with built-in WiFi has the limitation that it does not have as much accessible memory. It also does not have native secure storages for keys or credentials, presenting a vulnerability to physical attacks or unauthorized accesses if it were to have open port access to the Internet. Assuming these devices are not within physical access, without sufficient memory to implement a

digital signature, this removes an element of identity verification which opens the network to a spoofing attack.

3. Research Work

3.1. System Architecture

This paper proposes an extra layer of security beyond the WPA security protocol by encrypting the data that passes on the network. In the proposed scheme, AES, being a symmetric cryptography scheme, utilizes a shared secret provided by the ML-KEM. The encapsulation method yields a shared secret which is generated by a public key and the resulting ciphertext is transmitted to the client. The client will then be able to decapsulate the incoming ciphertext using the private key and acquire the shared secret key which is then used for the encryption and decryption of the AES algorithm. This requires the attacker to decipher three layers, adding to the complication of side-channel attacks.

Figure 1 shows that the message is encrypted prior to transmission; thus, if the attacker is within the network, they require both the digital signature and decryption method to decipher the original message. The destination device is then able to verify the sender with the hash algorithm, which can be the SHA-256 or the PQC method, FIPS 204, Module Lattice–Digital Signature Algorithm (ML-DSA) based on Crystals-Dilithium [23], or alternatively use a Message Authentication Code (MAC) tag to validate the sender. Without a sender authentication tag, it would be good security practice to discard the last unauthenticated payload as it could have been tampered with.

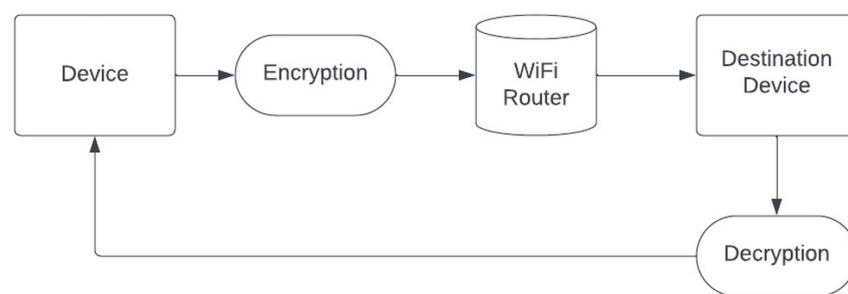


Figure 1. Path of data flow.

Figure 2 shows a common scenario that occurs in a cyberattack. The threat is within the network, and is able to sniff, spoof, divert, manipulate or otherwise tamper with the intended transmission. With the data being encrypted, the attacker must not only be able to obtain the shared secret to perform the decryption, but also be able to successfully replicate the hash function to authenticate the transmission to the target destination. A diverted message can look like a dropped transmission, easing the effort to detecting a bad actor within a network, and a failed authentication can immediately spotlight an intruder. With that detection, an immediate deauthentication can start to eradicate the threat immediately from the network.

The encapsulation and decapsulation flow paths are demonstrated in Figure 3. This step is done at every authentication handshake between devices prior to starting a WiFi session. The key pairs are created and distributed, with the public key for the encapsulation and the private key for the decapsulation. The shared secret between the devices is then used to encrypt and decrypt all messages. This handshake may occur as many times as needed, though the encapsulation normally takes the longest to perform.

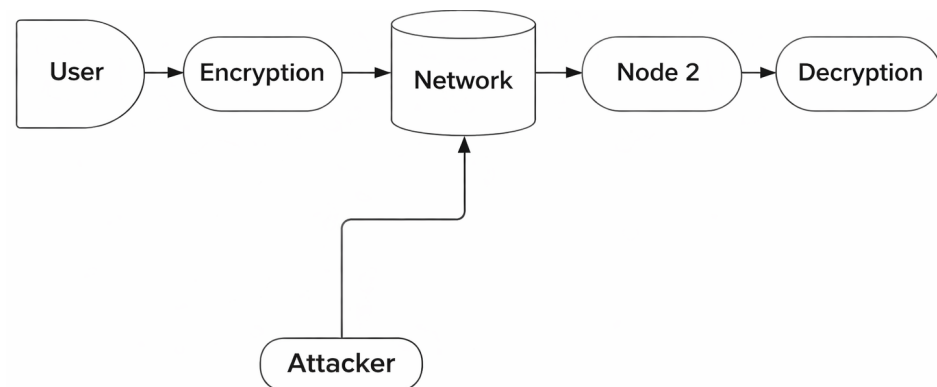


Figure 2. Possible threat scenario.

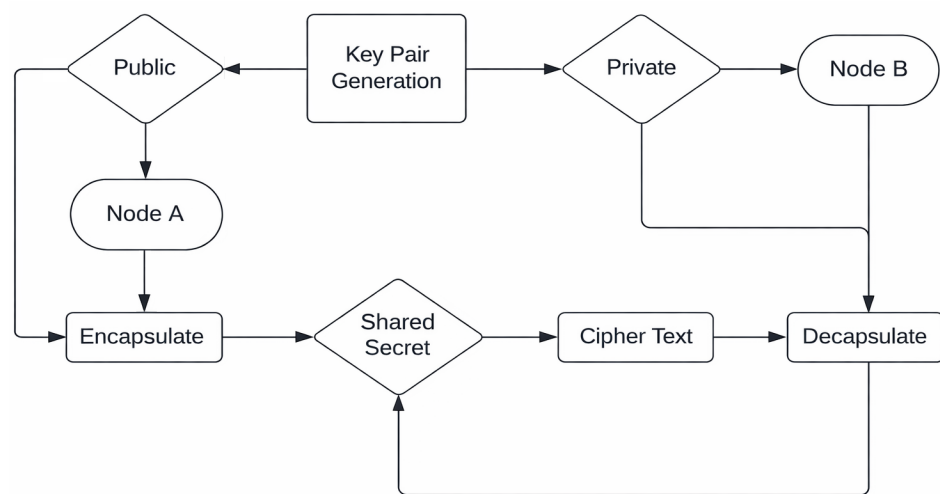


Figure 3. ML-KEM flow chart.

3.2. Advanced Encryption Standard

The main encryption standard used in this paper is the AES, which encrypts and decrypts data based on various key lengths of 128, 196 and 256 bits. The input and output lengths are of the same size, allowing padding to match the key sizes. The key sizes determines the number of rounds performed by the algorithm.

The AES in Figure 4 uses 4 steps per round until it reaches the last round where it does not perform the MixColumn function. The first step is substitution, where each byte is substituted by another byte. This is based on a lookup table called an S-Box. The substituted byte should never be itself or a compliment of the current byte. The result of this step is a 16-byte (4×4) matrix.

The next step is to shift rows in a particular way. These shifts are done by a multiple of the row minus one to the left. For example, the fourth row is shifted left three times, the third shifted left twice, the second shifted left once and the first row not shifted at all.

The third step is a matrix multiplication where each column is multiplied with a specific matrix. The resultant column changes the positions of each byte as a result. This step is skipped in the last round.

Finally, the last step of the round is to add the round keys. The resultant output is now XOR-ed with the corresponding round key. In this particular step, the data is no longer treated as a byte of data as previously seen, but operated in a bit-wise fashion. Therefore, the chunks are not seen as 16 bytes but 128 bits instead. The decryption of this algorithm is done in reverse starting with adding the round key. Then the ShiftRow function comes

after and finally, the reverse of the substitution is performed to essentially undo everything the encryption has done.

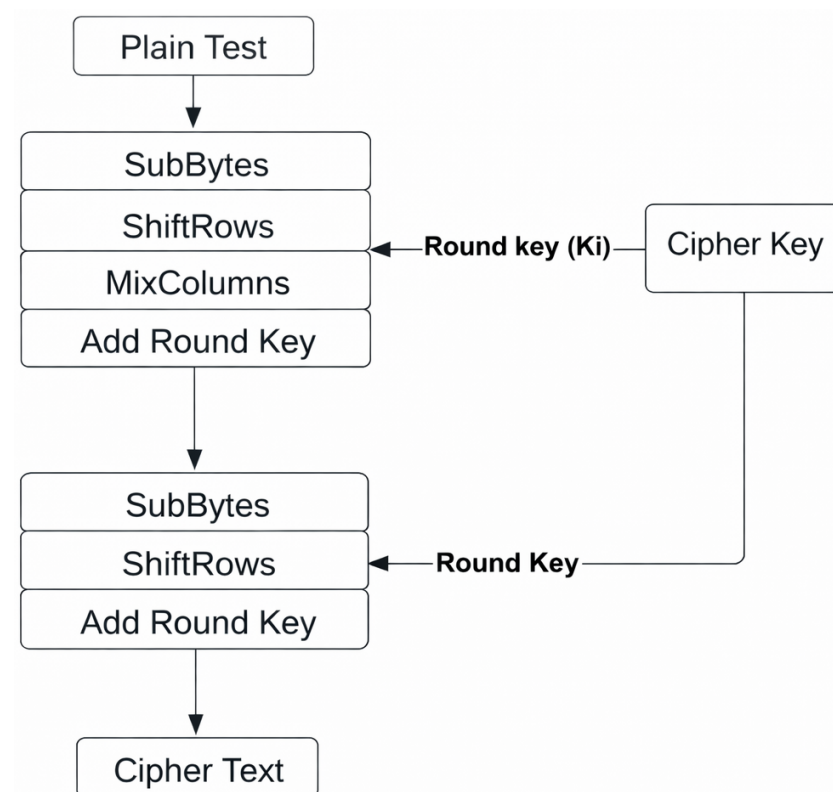


Figure 4. Generalized flow of AES.

3.3. Key Encapsulation Mechanism

The ML-KEM, based on the CRYSTALS-KYBER, an IND-CCA2 with a hardness based on Module Learning With Errors (MLWE), was selected for the FIPS 203 standardization. There are slight differences between ML-KEM and CRYSTALS-Kyber, the first being that the standard has a fixed length of 256 bits for the shared secret key while Kyber allows for variable length. One of the ways to defeat this algorithm is using side-channel or fault attacks. The survey in [24] had countermeasures, and reference [25] also introduced a more thorough survey on this issue. Side-channel attacks exploit the nature of devices and implementation with unintentional data leakages which are typically observed through power analysis and reactions [26,27].

The purpose of this paper is not to go into details of the architecture of the Kyber algorithm, but rather to implement its functionality. As with many other Post-Quantum Cryptography methods, its primary mathematical algorithm is based on the Numeric Theory Transform (NTT) which simplifies the system to solve polynomial multiplication. Without proper hardware, this algorithm can be very power-intensive and consume the majority of resources of the system being used.

In Algorithm 1 we introduce the NTT algorithm which is essential in implementing the most computationally heavy portion of the ML-KEM standard. The mathematical algorithm is a specially modified version of the Discrete Fourier Transform (DFT), allowing for a faster, exact and efficient pathway to deal with polynomial multiplication and convolution. Compared to the DFT, this deals with integers rather than floating points, which adds more relief on power intensity.

When considering hardware accelerations, the butterfly hardware can effectively speed up the calculations for implementing the NTT and its inverse NTT (INTT). The library

used in this paper uses the software implementation. The algorithms for the CT and GS butterflies are available [28].

Algorithm 1 NTT algorithm.

```

1:  $j \leftarrow 0$ 
2: for  $j$  from 0 to 255 do
3:    $\hat{w}[j] \leftarrow w_j$ 
4:  $m \leftarrow 0$ 
5:  $len \leftarrow 128$ 
6: while  $len \geq 1$  do
7:    $start \leftarrow 0$ 
8:   while  $start < 256$  do
9:      $m \leftarrow m + 1$ 
10:     $z \leftarrow \text{zet}as[m]$ 
11:    for  $j$  from  $start$  to  $start + len - 1$  do
12:       $t \leftarrow (z \cdot \hat{w}[j + len]) \bmod q$ 
13:       $\hat{w}[j + len] \leftarrow (\hat{w}[j] - t) \bmod q$ 
14:       $\hat{w}[j] \leftarrow (\hat{w}[j] + t) \bmod q$ 
15:     $start \leftarrow start + 2 \cdot len$ 
16:   $len \leftarrow \lfloor len/2 \rfloor$ 
17: return  $\hat{w}$ 

```

3.4. Hashing Algorithm

Hashing algorithms such as SHA-256 [29] are used to generate a hash, digest or, generally speaking, a digital signature. This is a one-way algorithm that does not reverse back into its original input. The output generated, however, can be used to verify the integrity of the data that is sent through the transmission. An example of this is shown in Figure 5.

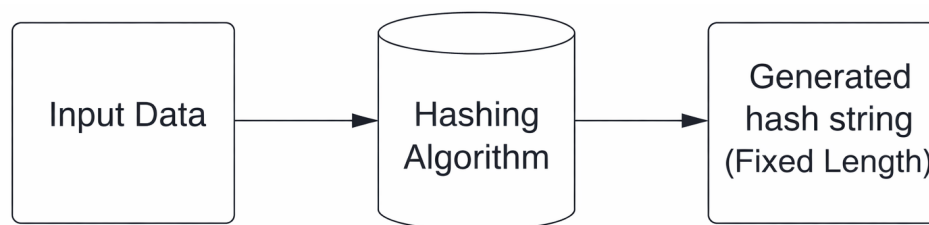


Figure 5. Example of hashing.

Hashing collision can occur when the function maps two different keys to the same index in a hash table. This requires the usage of resolution techniques, but it would be better to find ways to create algorithms that do not have this issue [30]. This event also occurs due the principle of the pigeonhole, as theoretically the inputs are unlimited, while the outputs are finite. Because of its fixed length output, the number of inputs that output the same hash are virtually infinite. This vulnerability can introduce a Birthday Attack [31], which is an unfortunate trait of weak hash algorithms such as MD5 and SHA-1 [32]. With the addition of the hashing function, Encrypt-then-MAC is now an even higher level of security for the system. The MAC is calculated using a MAC key that can be generated using the same shared secret over the ciphertext which then contains the MAC tag to verify the integrity of the data before decryption [33]. Using this method prevents an opening for padding oracle attacks [34] while also maintaining confidentiality. This is typically used in IPsec for VPN encryption, but there is no reason why it cannot be stacked again before going through the Transport Layer Security (TLS).

Figure 6 presents the flow chart of the baseline architecture, introducing the encryption prior to transmission along with a MAC tag that is generated using the hash algorithm. Once transmitted, the destination side can verify the MAC tag using the same hash algorithm and reject the payload if the tag is not verified. This is to ensure that malicious messages are not further processed, and that adversarial access does not pass through the line. More securities can be added to this architecture at the discretion of the designer.

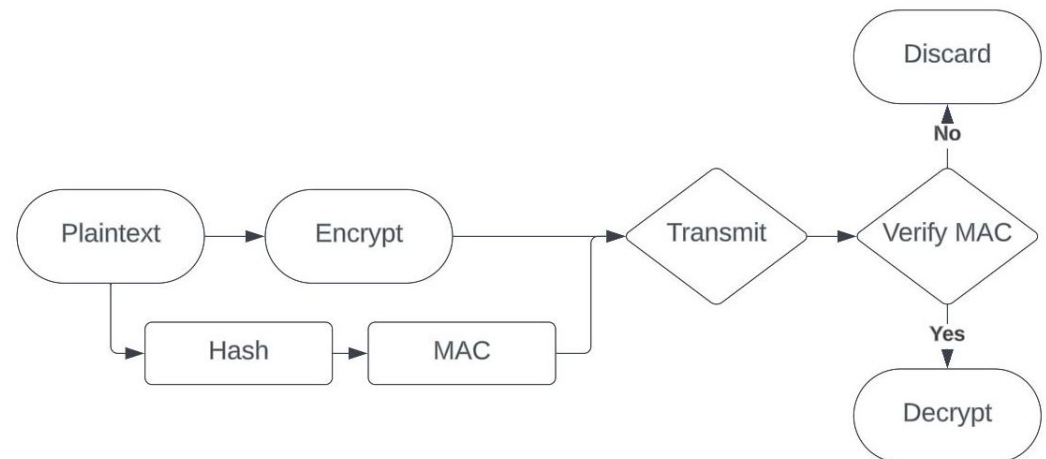


Figure 6. Flow chart of system.

4. Implementation and Results

4.1. Implementation

The proposed method is an additional layer of security by including another encryption that happens before the transmission of raw data between devices. Specifically, the protocol stack being used here is IEEE 802.11 [35] with WPA2 security since the device under test has not been introduced to the WiFi 6 standard [36]. No direct interactions are made to the 802.11 protocol after the Application Layer, where the data is encrypted before being formed into packets for transmission. By default, the data is transmitted using the Transmission Control Protocol (TCP) for its reliability and prevention of dropped packets en route to its destination. We observe the Wireless Fidelity protocol stack shown in Figure 7 in the following paths:

- **Application Layer**—Ignoring the Presentation and Session Layers of the OSI model, this is the interface between the user (device node) and the network services. In this context, we are using the WebSocket protocol in general form. In other use cases, this can also use the HTTP, FTP, MQTT and many others to deliver data.
- **Presentation Layer**—Though this is done by the WiFi protocol, we introduce the encryption prior to this layer, in essence adding an encryption to a ciphertext before it is sent via TLS.
- **Session Layer**—As normal, this layer is handled by the protocol and not by the user.
- **Transport Layer**—Here, we use the TCP protocol for its reliability in delivering data, while also relying on its error, flow and connection controls.
- **Network Layer**—Though this system relies on a closed network, separated from the Internet, devices on the network are still assigned IP addresses for which the end nodes are required to assign a source and destination for data packets. For default large networks, devices are assigned the 10.x.x.x subnets.
- **Data Link Layer**—This layer contains the Logical Link Control and Media Access Control. The first handles the flow control and the second for device addressing and

medium access. This is where the data goes through framing, error detection and correction, flow and media access controls.

- Physical Layer—Finally, for this system we use the wireless transmission to pass data between devices, nodes and endpoints on this network.

For brevity, the Presentation Layer is in the Application Layer as this is where both the protocol encryption and intended AES-256 with ML-KEM are implemented, and the Session Layer is with the Transport Layer. In the Data Link Layer, the authentication is handled by the protocol; however, the additional hashing function if implemented is handled on the receiver-side Application layer prior to decryption, authenticating the TLS signature.

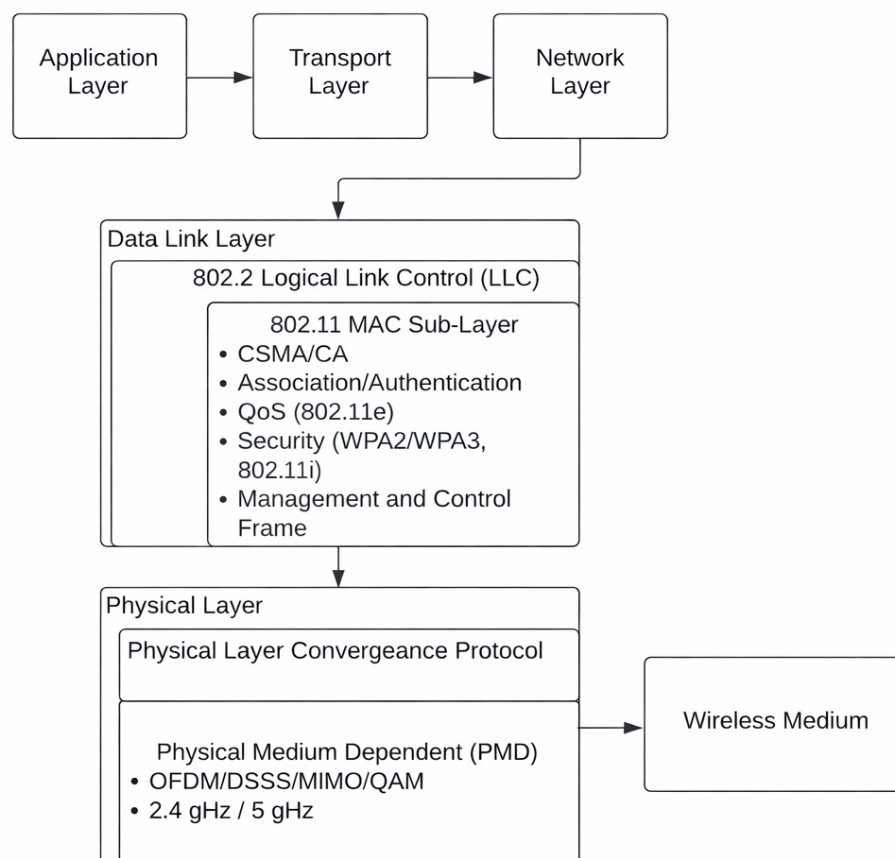


Figure 7. WiFi Protocol Stack.

Figure 8 shows the baseline model of what was implemented. There are other transport methods that can be implemented to better match the use case of the devices, nodes and gateways. A popular message transport protocol would be the MQTT, which in itself has either an open communication between devices [37], its own Quality of Service (QoS) or security protocols that can be stacked on top of the already encrypted data. Others such as ZigBee [38] pose their own sets of challenges, especially designing for low power when they are on their own mesh network. An interesting insight to this would be to handle the decryption at the hub instead of the Zigbee nodes as it may be too resource-constrained to handle it. In the end, if the devices are able to handle the additional computational workload, the feasibility of using both the ML-KEM and AES-256 would be beneficial in adding security to the devices instead of having them exposed to attackers.

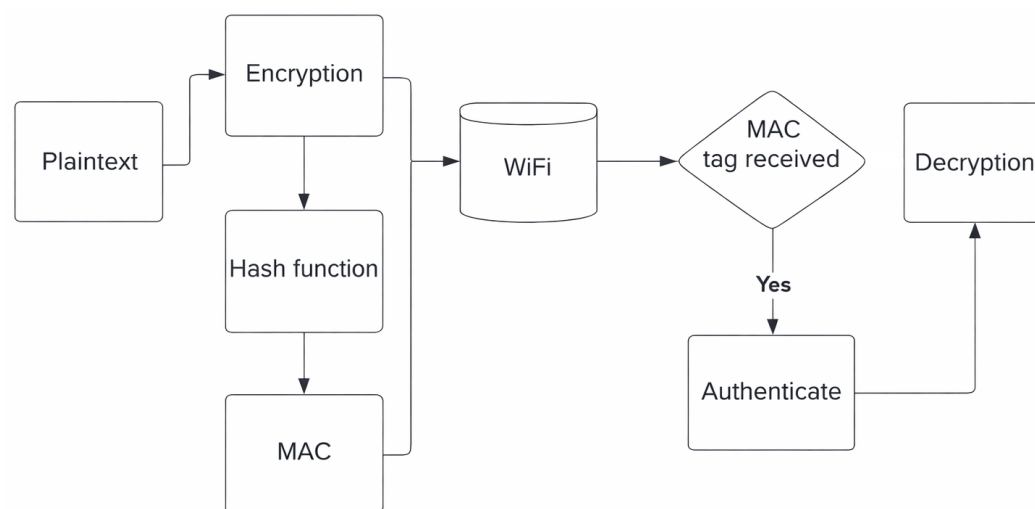


Figure 8. Implemented data flow.

4.2. Results

The implementation of the AES-256 algorithm was done using the Python Crypto library which includes the AES module. The key generation was done using the mlkem library for python and implemented using the cython version to speed up the build as this was purely a software implementation. The AES used the Cipher Block Chaining (CBC) mode, PKCS7 padding scheme, and the initialization vector generated using a random string generator of 16 bytes.

Table 1 shows the test bench results of implementing the ML-KEM to perform the key pair generation, encapsulation and decapsulation method, which are heavily affected by the computational power of the processor being used. The 2021 Apple M1 MacBook Pro manufactured by Foxconn in China, was the benchmark claimed by the author of the ML-KEM library using Python 3.13. The setup used for this research was Python 3.11, and the test bench was executed 10 times for each part to approximate a general average execution time in Python using C implementation. There are many implementations of the ML-KEM [39], while most popular software implementations are done using the Python language; however, advances in research regarding performance, resource utility, and power are made primarily on the hardware description languages. The speed of the MacBook Pro was published by the author of the Cython mlkem library which was tested using the built-in benchmark tests created by python.

Table 1. Test bench resulting time averages for key generation, encapsulation, and decapsulation combined.

System	MLKEM-512-C	MLKEM-768-C	MLKEM-1024-C
MacBook Pro	0.544 s	0.794 s	1.095 s
Ryzen 7 3800X	1.138 s	1.644 s	2.263 s
RPi4	1.558 s	2.366 s	3.287 s

The energy calculation in Table 2 assumes an average Thermal Design Power (TDP) consumption of 105 Watts for the AMD Ryzen 3800X, and measured using the pyrapl library for Python. This library utilizes the hardware sensors that were originally supported for the Intel Sandy Bridge architectures onward; therefore, its measurement on the AMD, an equivalent 8086 architecture, is an approximation rather than precise measurement. Software optimizations will also improve the energy efficiency readings from this library, other threads running on the chip will also add to the noisy energy measurement.

Table 3 contains the throughput of a 10 Megabyte payload for encryption with a MAC and decryption with a verification.

Table 2. Execution speed average for ML-KEM1024.

System	Encap (Python)	Encap (C ext)	Handshake	AES + HMAC	Energy
AMD 3800X	47.14 ms	0.81 ms	2.45 ms	0.0113 ms	0.180 J
RPi 4	160.82 ms	3.01 ms	9.74 ms	0.2755 ms	0.000 mJ
Rpi 5	51.08 ms	1.47 ms	2.45 ms	0.0113 ms	3.622 mJ

Table 3. The 10 MB throughput.

System	Encryption + MAC	Decryption + Verify
AMD 3800X	387.78 MB/s (25.79 ms)	618.15 MB/s (16.18 ms)
RPi 4	29.04 MB/s (344.30 ms)	31.09 MB/s (321.69 ms)
Rpi 5	361.67 MB/s (27.65 ms)	492.58 MB/s (20.30 ms)

We attempt to simulate a packet-sniffing attack, where the attacker captures the Initialization Vector (IV), ciphertext and tag. It was observed that this was vulnerable to a replay attack, but was easily countered using an expiration timestamp or sequence counter. This prevents the same packet from being used twice as each payload has now become unique. An example of packet sniffing is shown in Figure 9.

In future work, it is possible to implement the popular message telemetry protocol MQTT. Since this PubSub message telemetry by default is an unsecure communication protocol between machines on a network, relying on the TLS/SSL for encryption and authentication, it is possible to implement additional security such as client certificates and user access controls. There are many open-source brokers such as HiveMQ and RabbitMQ that can implement this broker feature and are available in many languages.

```

--- Starting ML-KEM-1024 Handshake ---
Handshake Latency: 2.45 ms

--- Packet Capture Simulation ---
Plaintext: Hello World
KEM Overhead: 1568 bytes
AES IV:      a0c185889fac5f75938a5d29cf5cfbda
Ciphertext:  07df07452d7316fa5bbbece81a6d2c5d
HMAC Tag:    abf74b958fe6aa42c9b2110e0fe47d3dddc8f46e4850bb16683106a63cad0e39

Decrypted Message: Hello World
Verification: SUCCESS

--- Performance ---
Throughput (AES-256 + HMAC): 562.99 MB/s

```

Figure 9. Packet capture example.

5. Discussion

This paper proposes a security architecture designed to alleviate an ongoing industrial crisis where adversarial actors enter the network and perform data theft. Though many third-party security softwares offer detection, triage and data loss prevention, reliance on these third parties can prove to be a fatal mistake. In 2024, a data breach on an elastic data server caused catastrophic damages, especially in reputational damages, which has prompted new research in enhancing securities between endpoints and cloud services [40].

The Raspberry Pi 4, referred to from here on as Client, receives the private key over the WiFi network from the Host PC as the first step of the implementation. The Host PC, which runs on an AMD Ryzen 7 3800X, generates the shared secret and then encapsulates it with the public key. The Host then sends the encapsulation over the socket, where the Client then receives the shared secret as a ciphertext to be decapsulated using the private key that was received in the first step. This shared secret is now the key used in the symmetric encryption and decryption scheme via the AES-256 algorithm. Therefore, instead of using a random number generator to create a 32-byte value, this is instead the shared secret. Now the cipher can be performed using the shared secret known only to the Host and Client, and to outsiders, the key is 1184 bytes in size compared to the 32 bytes which hides within the encapsulation of the ML-KEM768 algorithm.

Table 4 shows the sizes of the output from the encapsulation and decapsulation algorithm. Regardless, it produces a 32-byte shared secret that is used in the encryption and decryption.

Table 4. Data size of encapsulation, decapsulation, ciphertext and shared secret based on parameter set.

Parameter Set	Encap Size	Decap Size	Cipher	Shared Secret
ML-KEM-512	800 bytes	1632 bytes	768 bytes	32 bytes
ML-KEM-768	1184 bytes	2400 bytes	1088 bytes	32 bytes
ML-KEM-1024	1568 bytes	3168 bytes	1568 bytes	32 bytes

Industries with particularly sensitive data [41], especially in healthcare, are more susceptible to becoming targets of hackers. But this still does not deter attempted phishing attacks via emails, links and misdirection, which exploit human error and mishandling, giving the attackers an easier time to compromise a device to gain access to the network. Though software such as Crowdstrike (Released on 19 July 2024) enhances detection of malicious activities, a faulty software update caused a catastrophic shutdown of many Windows operating systems, consequently causing a lot of financial damage for the time these devices were down. Companies constantly train their employees to become more cyber-aware, while also depending on third-party software to monitor the networks for malicious activities. These programs tend to be resource-hungry due to their machine learning algorithms, which tend to reduce productivity because of the sluggishness they cause. A more recent incident demonstrated how one buggy software update caused an absolute IT nightmare for millions of users.

Detection of malicious activities is definitely a good step in prevention, but it would be better to design secrecy first between devices on networks to reduce or essentially remove eavesdroppers. This paper introduces encryption prior to transmission to WiFi, and enhances it further by introducing a MAC tag that is required by the sender to validate their identity. The MAC tag can also be replaced with the FIPS 204 standard, ML-DSA to verify through a hash signature, enhancing the post-quantum security.

This paper uses the AES along with the FIPS 203 to generate the key pair. The key pair is then transmitted to respective users to use for encapsulation and decapsulation, and neither the encryption nor decryption that is typical of key pairs. Instead, the shared secret is used in place of the keys. The encapsulation is done prior to the WiFi handshake, or rather every connection establishment as this would normally take the most time to perform compared to the decapsulation. Attackers attempting to sniff the transmission would have to guess if the WiFi is sending actual data or if it is watching the encapsulation

being sent. For this paper, the ML-KEM-768 is used, which yields an 1184-byte-wide payload with a ciphertext size of 1088 bytes.

It is apparent that different hardware will take different amounts of time to perform the totality of the encapsulation, decapsulation and key pair generation; however, with most of the time being spent on encapsulation, this does not matter much unless the architecture requires frequent connection establishment which calls the encapsulation each time a handshake is requested. The library used for this paper uses a standard Cooley–Tukey and Gentlemen–Sande software butterfly to handle the NTT and INTT algorithms required for the ML-KEM. There are Extended Harvey Butterflies which introduce an extensible software butterfly to further improve the processing speed when handling the polynomial multiplication required for the algorithm.

Regardless, attackers harvesting raw data now, then planning to decrypt it later when CRQC becomes available is a looming threat for current data that is relying on the modern ECC and RSA securities. The requirements of creating a more robust and secure zero-trust network becomes more important as data breaches are becoming more prevalent in industries with sensitive data such as healthcare and government. By introducing a post-quantum-resistant algorithm to the encryption scheme, it makes it difficult for Quantum Computers to crack the security required to get to data.

For sensitive data that requires secrecy and security throughout transmission, homomorphic encryption can be implemented. However, fully homomorphic encryption itself has some drawbacks, including the possibility of compromising accuracy of data integrity. This is primarily due to the noise created through multiplicative operations, which consequently creates noise, increasing the odds of decryption failure. Bootstrapping the data has been a proposed solution to this problem; however, it is computationally expensive. Therefore, this technique is not popular to use on low-end network systems.

The TLS and the current WPA3 standard rely on Elliptic Curve Diffie–Hellman (ECDH) or RSA to agree on keys, neither of which are quantum-resistant. Even if they are secure to current standards, this does not remove the hazard of the Harvest Now, Decrypt Later issue that is still present today. Furthermore, an attacker that is present on the same trusted network or a compromised router will expose the data traffic. By providing End-to-End Encryption (E2EE) protection, the attacker would only see the ciphertexts.

The Authenticated Encryption with Associated Data (AEAD), while preferred currently with the AES-GCM, poses a risk if using the same IV. The attacker is able to recover the authentication keys and spoof messages once successful. By using Encrypt-then-MAC using HMAC-SHA256, it creates a unique MAC for every transaction.

Finally, there are many other ways to add to the security of the network, not limited to adding post-quantum security. By including an external device outside the accessible network which requires a handshake for every transaction for the first encryption, it can add confusion to the attackers as the data they retrieve would still be encrypted by the first AES pass on the data. The end device would still store the data as encrypted, but it would be able to decrypt the message locally as the keys are stored directly on that device. A similar idea to SIPRNET or JWICS is implemented by government agencies: an intranet that is isolated from the open unsecure Internet but can only be accessed through strict controls would deter intruders by utilizing more physical securities rather than elements implemented through software.

6. Conclusions

This paper proposed an implementation of an additional layer of security on top of the recently developed WiFi6 WPA3 security protocol implemented on a modern router, and WPA2 for the Raspberry Pi. The baseline encryption method used was AES-256 in

CBC mode that was implemented using the cryptography library in the Python language. The payload transport was then sent using a standard 802.11 socket protocol using the Python library socket.

This research introduced an additional layer of security in lieu of claimed security to increase the difficulty of cracking vulnerabilities of the system. The usage of cython greatly improves the performance speed of the Python libraries. Through this demonstration, it is apparent that optimization through hardware acceleration will further improve the performance and reliability of each payload transmission up to the theoretical limit of the network bandwidth.

Additional research in implementing faster executions of the ML-KEM utilizing FPGAs, ASICs and hardware abstractions can add more feasibility for speed-constrained projects that may desire additional security on a WiFi network. This system is not limited to just Raspberry Pis or PCs. Other devices may count as the host and client, or in other terms referred to as nodes on a WiFi network.

The implementation of AES and ML-KEM is not limited to Python, and can be done in other languages that may provide advantageous speed; however, the necessary implementation here is the WiFi socket which is explicitly a handler on the operating system. This layered protocol may also be used on other telemetry such as cellular, Bluetooth, LoRa, and many other wireless networks. This proposes additional avenues of development for nodal security for sensor networks, and most importantly, company assets in industries that contain sensitive data.

Author Contributions: Conceptualization, M.K. and Y.C.; methodology, M.K.; investigation, M.K.; writing—original draft preparation, M.K.; writing—review and editing, M.K. and Y.C.; visualization, M.K.; supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
CISA	Cybersecurity and Infrastructure Security Agency
JWICS	Joint Worldwide Intelligence Communication System
SIPRNet	Secret Internet Protocol Router Network
AEAD	Authenticated Encryption with Associated Data
PQC	Post-Quantum Cryptography
DES	Data Encryption Standard
ECC	Elliptic Curve Cryptography
DFT	Discrete Fourier Transform
CBC	Cipher Block Chaining
CRQC	Cryptographically Relevant Quantum Computer
IV	Initialization Vector
ML-KEM	Module Lattice–Key Encapsulation Mechanism
ML-DSA	Module Lattice–Digital Signature Algorithm
MLWE	Module Learning With Errors
Elliptic Curve Diffie–Hellman	ECDH
RSA	Rivest–Shamir–Adleman
IoT	Internet of Things

WiFi	Wireless Fidelity
MAC	Message Authentication Code
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
NTT	Numeric Theory Transform
INTT	Inverse Numeric Theory Transform
QoS	Quality of Service
TDP	Thermal Design Power
SSL	Secure Socket Layer
TLS	Transport Layer Security
TCP	Transmission Control Protocol
WPA	Wifi Protected Access

References

1. Liu, Y.; Li, S.; Hulayyil, S.B. Intelligent detection of cyber attack patterns in industrial IoT using pretrained language models. *Electronics* **2025**, *14*, 4094. [\[CrossRef\]](#)
2. Alharbi, S.; Awad, W.; Bell, D. HECS4MQTT: A Multi-Layer Security Framework for Lightweight and Robust Encryption in Healthcare IoT Communications. *Future Internet* **2025**, *17*, 298. [\[CrossRef\]](#)
3. Khern-am nuai, W. Key Lessons Learned for Technology Managers From CrowdStrike Global IT Outage. *IEEE Eng. Manag. Rev.* **2025**, *53*, 25–27. [\[CrossRef\]](#)
4. Mugu, S.R.; Zhang, B.; Kolla, H.; Balaji, S.R.A.; Ranganathan, P. Lessons from the CrowdStrike Incident: Assessing Endpoint Security Vulnerabilities and Implications. In Proceedings of the 2024 Cyber Awareness and Research Symposium (CARS), Grand Forks, ND, USA, 28–29 October 2024; pp. 1–10. [\[CrossRef\]](#)
5. Sobbb, T.; Turnbull, B. CrowdStrike: Lessons in Mission Assurance to Build Resilience for Complex Cyber–Physical–Social Systems. *IEEE Access* **2025**, *13*, 205324–205337. [\[CrossRef\]](#)
6. Singh, B.; Dutta, P.K.; Lal, S.; Arora, M.K.; El-kenawy, E.S.M. Beyond CrowdStrike: A technical analysis of SHIELD’s cyber resilience platform in mitigating large-scale outages. *IET Conf. Proc.* **2025**, *2024*, 409–415. [\[CrossRef\]](#)
7. Mikale, E.; Waters, S.; Swiney, L. *Stabilizing Qubits with Dynamic Frequencies, the Implications on Post-Quantum Encryption Protocols, and a Hybrid Quantum Internet*; IEEE: New York, NY, USA, 2024; pp. 1–12.
8. Dent, A.W. A Designer’s Guide to KEMs. In *Proceedings of the IMA International Conference on Cryptography and Coding*; Springer: Cirencester, UK, 2003; pp. 133–151.
9. Cherkaoui Dekkaki, K.; Tasic, I.; Cano, M.D. Exploring post-quantum cryptography: Review and directions for the transition process. *Technologies* **2024**, *12*, 241. [\[CrossRef\]](#)
10. Dworkin, M.J.; Barker, E.; Nechvatal, J.R.; Foti, J.; Bassham, L.E.; Roback, E.; Dray, J.F., Jr. *Advanced Encryption Standard (AES)*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001.
11. Bhavsar, V.; Kadlak, A.; Sharma, S. Study on phishing attacks. *Int. J. Comput. Appl.* **2018**, *182*, 27–29. [\[CrossRef\]](#)
12. Tsiknas, K.; Taketzis, D.; Demertzis, K.; Skianis, C. Cyber threats to industrial IoT: A survey on attacks and countermeasures. *IoT* **2021**, *2*, 163–186. [\[CrossRef\]](#)
13. Hasib, A.A.; Haque, A.A.M.M. A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography. In Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, Republic of Korea, 11–13 November 2008; Volume 2, pp. 505–510. [\[CrossRef\]](#)
14. Alhamry, M.; Alomary, A. Exploring Wi-Fi WPA2-PSK protocol weaknesses. In Proceedings of the 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Sakhir, Bahrain, 25–26 October 2022; pp. 190–195. [\[CrossRef\]](#)
15. Fehér, D.J.; Sándor, B. Effects of the WPA2 KRACK Attack in Real Environment. In Proceedings of the 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 13–15 September 2018; pp. 000239–000242. [\[CrossRef\]](#)
16. Radivilova, T.; Hassan, H.A. Test for penetration in Wi-Fi network: Attacks on WPA2-PSK and WPA2-enterprise. In Proceedings of the 2017 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), Odessa, Ukraine, 11–15 September 2017; pp. 1–4. [\[CrossRef\]](#)
17. Vaz, Y.S.; Mattos, J.C.B.; Soares, R.I. Lightweight AES Algorithm for Internet of Things: An Energy Consumption Analysis. In Proceedings of the 2024 XIV Brazilian Symposium on Computing Systems Engineering (SBESC), Recife, Brazil, 26–29 November 2024; pp. 1–6. [\[CrossRef\]](#)

18. Vishwakarma, A.; Singh, B. Implementation Study of AES Standard for IoT Systems. In Proceedings of the 2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT), New Delhi, India, 23–25 September 2022; pp. 1–6. [\[CrossRef\]](#)
19. Xiang, L.; Zeng, Y.; Hu, J.; Yang, K.; Hanzo, L. Multi-Domain Polarization for Enhancing the Physical Layer Security of MIMO Systems. *IEEE Trans. Commun.* **2024**, *72*, 1502–1515. [\[CrossRef\]](#)
20. Lagunzad, H.C.; Gonzaga, M.V.; Samonteza, C.M. Development and Implementation of File Security Using Raspberry pi with an Advanced Encryption Standard (AES) Algorithm. In Proceedings of the 2023 IEEE 6th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), Haikou, China, 18–20 August 2023; pp. 1195–1200. [\[CrossRef\]](#)
21. Halbouni, A.; Ong, L.Y.; Leow, M.C. Wireless Security Protocols WPA3: A Systematic Literature Review. *IEEE Access* **2023**, *11*, 112438–112450. [\[CrossRef\]](#)
22. Stoev, I.; Zaharieva, S.; Borodzheva, A.; Staevska, G. An Approach for Securing MQTT Protocol in ESP8266 WiFi Module. In Proceedings of the 2020 XI National Conference with International Participation (ELECTRONICA), Sofia, Bulgaria, 23–24 July 2020; pp. 1–4. [\[CrossRef\]](#)
23. Huang, T.; Lu, J.; Li, A.; Chen, L.; Li, K.; Zhang, J.; Wang, M.; Mei, X.; Li, H.; Liu, D. A Lightweight and Efficient Post Quantum Crypto-Processor for ML-DSA. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2025**, *73*, 2595–2607. [\[CrossRef\]](#)
24. Angom, A.; Kar, N.; Debbarma, T.; Biswas, P. A survey on Lattice-Based Key Establishment Schemes: Types, Evolution and Advances. In Proceedings of the 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 6–8 March 2025; Volume 3, pp. 1–6. [\[CrossRef\]](#)
25. J, R.; N, E.D.; Annadurai, S. Side-Channel and Fault Attacks on ML-KEM: A Survey of Vulnerabilities and Countermeasures. In Proceedings of the 2025 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, 20–22 March 2025; pp. 1–6. [\[CrossRef\]](#)
26. Ghiban, C.; Choudary, M.O. Improved Correlation Power Analysis Attack on the Latest Cortex M4 Kyber Implementation. *Cryptography* **2025**, *9*, 19. [\[CrossRef\]](#)
27. Shrivastava, S.; Kaur, A.; Singh, G. Advancing Green Communication: Energy-Efficient Lightweight RSA Implementation on FPGA Platforms. In Proceedings of the 2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 13–15 November 2024; pp. 1102–1106. [\[CrossRef\]](#)
28. Scott, M. A note on the implementation of the number theoretic transform. In *Proceedings of the IMA International Conference on Cryptography and Coding*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 247–258.
29. National Institute of Standards and Technology. Secure Hash Standard (SHS), FIPS Publication 180-4, Update 1. Federal Information Processing Standards Publication. 2015. Available online: <https://csrc.nist.gov/pubs/fips/180-4/upd1/final> (accessed on 20 April 2026).
30. Cherckesova, L.V.; Safaryan, O.A.; Lyashenko, N.G.; Korochentsev, D.A. Developing a new collision-resistant hashing algorithm. *Mathematics* **2022**, *10*, 2769. [\[CrossRef\]](#)
31. Iwamoto, M.; Peyrin, T.; Sasaki, Y. Limited-birthday distinguishers for hash functions: Collisions beyond the birthday bound can be meaningful. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 504–523.
32. Bellare, M.; Kohno, T. Hash function balance and its impact on birthday attacks. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 401–418.
33. Abdoun, N.; El Assad, S.; Manh Hoang, T.; Deforges, O.; Assaf, R.; Khalil, M. Authenticated encryption based on chaotic neural networks and duplex construction. *Symmetry* **2021**, *13*, 2432. [\[CrossRef\]](#)
34. Pallakonda, A.; Kaliyannan, K.; Sumathi, R.L.; Raj, R.D.A.; Yanamala, R.M.R.; Napoli, C.; Randieri, C. AI-Driven Attack Detection and Cryptographic Privacy Protection for Cyber-Resilient Industrial Control Systems. *IoT* **2025**, *6*, 56. [\[CrossRef\]](#)
35. Hiertz, G.R.; Denteneer, D.; Stibor, L.; Zang, Y.; Costa, X.P.; Walke, B. The IEEE 802.11 universe. *IEEE Commun. Mag.* **2010**, *48*, 62–70. [\[CrossRef\]](#)
36. Mozaffariahrar, E.; Theoleyre, F.; Menth, M. A Survey of Wi-Fi 6: Technologies, Advances, and Challenges. *Future Internet* **2022**, *14*, 293. [\[CrossRef\]](#)
37. Azzedin, F.; Alhazmi, T. Secure data distribution architecture in IoT using MQTT. *Appl. Sci.* **2023**, *13*, 2515. [\[CrossRef\]](#)
38. Ghobakhlou, A.; Al-Hamid, D.Z.; Zandi, S.; Cato, J. A Comprehensive Analysis of Security Challenges in ZigBee 3.0 Networks. *Sensors* **2025**, *25*, 4606. [\[CrossRef\]](#)
39. Nagy, N.; Alnemer, S.; Alshuhail, L.M.; Alobiad, H.; Almulla, T.; Alrumaihi, F.A.; Ghadra, N.; Nagy, M. Module-Lattice-Based Key-Encapsulation Mechanism Performance Measurements. *Sci* **2025**, *7*, 91. [\[CrossRef\]](#)

40. Nageshwaran, V. Enhancing Snowflake Security: A Novel Framework Integrating HGWCS and CANN for Improved Threat Detection and Response in Cloud Environments. In Proceedings of the 2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI), Erode, India, 20–22 January 2025; pp. 1950–1955. [[CrossRef](#)]
41. Seh, A.H.; Zarour, M.; Alenezi, M.; Sarkar, A.K.; Agrawal, A.; Kumar, R.; Ahmad Khan, R. Healthcare data breaches: Insights and implications. *Healthcare* **2020**, *8*, 133. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.