Q-SECURE: A Quantum Resistant Security for Resource-Constrained IoT Device Encryption

Maurice Ngouen[*], Mohammad Ashiqur Rahman[*†], Nagarajan Prabakar[†], Selcuk Uluagac[†], and Laurent Njilla[‡]
[*]Electrical and Computer Engineering, Florida International University, USA
[†]Knight Foundation School of Computing and Information Sciences, Florida International University, USA
[‡]US Air Force Research Laboratory (AFRL), USA
{mngou002, marahman, prabakar, suluagac}@fiu.edu, laurent.njilla@us.af.mil

Abstract—**The Internet of Things (IoT) plays a significant role in shaping different aspects of our lives. IoT devices have become increasingly important due to their ability to connect, collect, and analyze data, automate processes, improve safety and efficiency, and deliver personalized experiences. However, the advancement in quantum computer development poses a significant threat to resource-constrained IoT devices. This new generation of computers can break the classic public-key cryptographic schemes and digital signatures implemented in these IoT devices. While protecting IoT devices from quantum computer attacks poses many challenges, researchers are continuously making significant progress in developing lightweight post-quantum cryptographic algorithms for efficient key exchange mechanisms and digital signature algorithms tailored to IoT devices to overcome this issue. This paper proposes Q-SECURE, a post-Quantum resistant Security Enhancing Cryptography for Unified Resource-constrained device Encryption. This novel scheme enables any IoT system to leverage the assistance of other devices in the network to gain the capability to generate any proposed post-quantum cryptographic key of a given size using distributed and parallel computing.**

Index Terms—**Post-Quantum, Internet of Things, Cryptography, 5G-IoT technologies, Lattice-based, Isogenie, Key Encapsulation, Key Generation.**

I. Introduction

Internet of Things (IoT) devices, networked gadgets that can interact and share data over the Internet, are becoming more common today than ever. However, the rapid development of quantum computers poses a security threat to these IoT devices. The world is on the approach of a massive shift in the realm of cybersecurity as quantum computing technology develops and advances [1]. Quantum computing can undermine the cryptographic methods that have served as the foundation of cybersecurity for decades [2]. As quantum computers become more powerful, traditional encryption methods will become vulnerable, compromising data security [3]. Post-quantum security has evolved as a novel technique for encryption that can survive assaults from quantum computers in this environment.

Researchers have been working on post-quantum security techniques that can survive quantum computer assaults [4]. Post-quantum security is a novel concept of encryption that employs mathematical problems thought to be difficult to answer by both conventional and quantum computers. These algorithms are meant to be resilient to quantum computer assaults, assuring data security even if quantum computers become widely available.

IoT devices are frequently developed with resource constraints in mind, as they are generally deployed in various contexts where resources such as power, processing capacity, memory, and network bandwidth may be restricted. Resource restrictions are a key part of IoT device design and operation and have consequences for their criticality. With the continuous growth of 5G technology and the Internet of Things (IoT), people rely more on using technology than ever before, e.g., for online shopping, marketing, social media, vehicles, home appliances, etc. Nowadays, almost every piece of hardware uses software to send or receive information. People have adopted these systems because they guarantee all their online activities' confidentiality, integrity, and availability.

Classical cryptography provides the integrity and confidentiality of all online activities. This security is made possible with the use of prime numbers factorization and the discrete logarithm (elliptic curve) cryptography that is used by RSA (Rivest, Shamir, Adleman), ECDSA (Elliptic Curve Digital Signature Algorithm), ECDH (Elliptic Curve Diffie-Hellman) or DSA (Digital Signature Algorithm) [5]. However, many security schemes the world has been relying on will soon end with the advancement of quantum computers. Indeed, Shor has discovered an algorithm that can easily break the cryptography schemes in polynomial time [6]. This algorithm runs on quantum computers that will be realized faster than expected, with the current exponential growth in technology.

To attain high security for cryptographic approaches and withstand future assaults from quantum computers, we need new techniques incorporating both conventional cryptography and quantum technology. In response to advances in the development of quantum computers, in 2016, the National Institute of Standards and Technology (NIST) launched a competition for the new cryptography standard called Post Quantum Cryptography (PQC). This competition is now in the fourth round, and the new cryptographic standard will be announced soon. More than one of the current contenders is anticipated to be standardized, and these algorithms will progressively take the role of RSA and ECC in applications. The major challenges that PQC brings are large key sizes, large signature and hash length, high computational complexity, and high energy consumption. All of these contradict the cryptographic schemes used for classic computers, and most IoT devices possess slow computation power and energy.

Therefore, implementing post-quantum cryptography schemes on IoT devices is still a big problem. Many implementations of post-quantum cryptography on IoT devices have been done. Some of these works are LATTICE-based standards, such as the Streamlined Non-Truncated Ring Unit (NTRU), where a Prime system with a given recommended efficient multiplication design [7]; others use the features of quantum walk to construct a new S-box method, which plays a significant role in block cipher techniques for 5G-IoT technologies, and the rest implement Isogeny based cryptography. This is the case of "Secure Data Encryption Based on Quantum Walk [8] for 5G Internet of Things Scenario," where the authors build a new S-box approach that plays a big part in block cipher techniques for 5G-IoT technologies using the characteristics of quantum walk5, and "SIKE'd Up Fast Hardware Architectures for Super singular Isogeny Key Encapsulation" in which the authors carry out the complete SIKE procedure [9]. All the approaches described above suppose the IoT device executing the post-quantum procedure has sufficient computation capability and enough power, which is not always the case in many scenarios.

All the research in IoT security assumes that the devices involved in generating a post-quantum cryptography key can implement it on the software or hardware levels. None of them had addressed how an IoT device could leverage the presence of other IoT on the network to improve the computational time in the key-generation process significantly. To our knowledge, this is the first approach to improve the post-quantum cryptography implementation on low-power IoT devices. In summary, our main contribution to this paper consists of the following:

- Improve the computational time in the key-generation process for a post-quantum cryptographic scheme by introducing our client-helpers framework.
- Providing all IoT devices the capability to implement any post-quantum cryptographic scheme by leveraging the computational power of other devices on the network.
- Protecting low-power IoT devices from attacks by quantum and classical computers by making them more resilient to such attacks.

All the code and its implementation and evaluation results can be reproducible with our source code available on GitHub.[1]

The remainder of the paper is organized as follows: in Section I, we present an overview of post-quantum cryptography and its protection against attacks from quantum computers. A summary of the literature review is presented in Section II. In section III, we describe our model, in Section IV we present the technical details of the implementation. We share a case study to provide insight into our suggested scheme's working principles and capabilities in Section V. Then, we demonstrate the validation with an actual prototype testbed scenario. The following section evaluates our scheme using the quantum-resistant code-based key cryptosystem. We present the conclusion and future work in Section VII.

[1]https://github.com/lerice1/IOT--Quantum-Resistant-Security

II. Related Works

Internet of Things (IoT) devices are becoming more common, but their increased use poses security issues. The threat presented by quantum computers, which can break many of the cryptographic protocols now used to safeguard IoT devices, is a specific source of concern. To solve this issue, researchers have been working on post-quantum cryptography algorithms that are supposed to be safe even against quantum computer assaults [10][11]. Before proposing our approach, we examine the state of the art in post-quantum security for IoT devices.

Despite the promise of post-quantum security, several challenges must be overcome before these algorithms can be widely adopted. One of the most difficult issues is the necessity to create new post-quantum security standards [4]. The National Institute of Standards and Technology (NIST) has led the work to draft these standards, although finalization and adoption will take several years. Meanwhile, there is a risk that quantum computers will be used to break existing encryption methods, jeopardizing sensitive data security.

One of the major issues in protecting IoT devices with post-quantum cryptography is the devices' limited processing capacity. This is the case in [12] where the authors present a post-quantum public key cryptosystem that is lightweight and tailored for IoT devices. Based on the RLCE method, the suggested cryptosystem is intended to be computationally efficient and immune to quantum attacks. Many post-quantum algorithms are computationally demanding and necessitate a significant amount of processing power and memory, which can be difficult for resource-constrained IoT devices. To solve this issue, many researchers have proposed lightweight post-quantum cryptography algorithms targeted for IoT devices [13][8][14][8][7][15]. The NIST Lightweight Cryptography project, for example, is working on a set of lightweight post-quantum cryptographic algorithms that are intended to be deployed on resource-constrained devices [10][11].

Another problem in using post-quantum cryptography to secure IoT devices is the necessity for effective key distribution techniques. Many post-quantum algorithms necessitate the exchange of large keys, which can be difficult in IoT networks with limited bandwidth [5][16]. To overcome this issue, academics have worked on efficient key exchange systems tailored particularly for post-quantum cryptography. The NewHope [17] key exchange method, for example, is a post-quantum key exchange mechanism meant to be efficient on low-bandwidth networks.

The necessity for secure firmware upgrades is a related difficulty in protecting IoT devices using post-quantum cryptography. Many IoT devices are vulnerable to attacks that take advantage of flaws in their firmware, which can be difficult to secure updates [15]. Researchers have been working on secure firmware update procedures that leverage post-quantum cryptography to assure the integrity and authenticity of firmware upgrades in order to address this difficulty[18]. For example, the NIST Post-Quantum Cryptography Standardization project is a significant initiative aiming at creating and standardizing post-quantum cryptographic algorithms that may be used to protect IoT and other post-quantum applications [19]. Other research efforts are focused on developing post-quantum cryptographic primitives, such as signature schemes, key exchange protocols, and encryption schemes, that are optimized for use in the resource-constrained environment of the Internet of Things. Lattice-based cryptographic schemes, which are noted for their efficiency and security, and code-based systems, which are supposed to be resistant to quantum assaults, are two well known examples. Overall, there is a growing corpus of research aiming at building workable and efficient post-quantum cryptography solutions that may be utilized in the future to protect IoT and other applications.

However, all of these researches focus mainly on the hardware level or the application level main that these devices must be capable of generating post-quantum cryptographic schemes on their own. In contrary, our approach leverages the presence of any computing devices on the network, and the implementation of the ”zero trust” security model. We presume that all devices and users, whether inside or outside a network’s perimeter, should not be trusted automatically. In other words, zero trust demands that every device or person seeking to access a network or a resource, regardless of location or status, be validated and allowed.

III. Scheme Description

In this section, we discuss the proposed scheme in detail.

A. *Scheme Overview*

This section will provide an overview and a description of our framework. Our framework consists of a client-helpers architecture model, in which the IoT (Client) device solicits the computation power of other IoT devices (Helpers) to speed up the post-quantum cryptography key-generation and encapsulation processes. We will refer to the IoT generating the post-quantum cryptographic key as the client, and other IoTs helping in the key generation process as helpers. We assume that all communications are encrypted using any specified post-quantum cryptographic scheme. We also assume that there are always more than two devices in the network.

When the client wants to start communication with any devices on or outside the network for the first time, it sends out requests to check the availability of other devices on the network. Other devices on the network (helpers) reply to the request by confirming their availability to provide the computation help to the client. The client generates the size of the key, splits it into multiple parts, keeps half the number of parts, and randomly sends the rest to some helpers on the network for computation. We use the example of McEliece Post-Quantum Cryptosystem where the private key of any IoT client is a combination of three matrices S, G, and P. S is a non-singular $k \times k$ matrix, referred to as the scrambler matrix. G is a $n \times k$ matrix with binary linear block code over $GF(2)$,

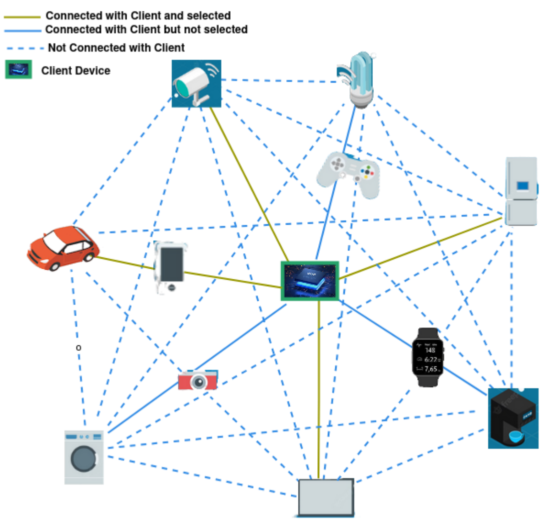
Connected with Client and selected
Connected with Client but not selected
Not Connected with Client
Client Device

Fig. 1. The Client randomly selects a subset of helpers to assist in the key generation process

with the capacity of being able to correct up to t errors with the $n - k$ parity bits [20].

This process can be done in the case of many post-quantum cryptographic schemes such as Non-truncated Ring Unit (NTRU), Learning with Error (LWE), and Isognie post-quantum public key generation mechanisms. The key size is critical because a small key size will cause the algorithm to be insecure against quantum attacks. A large key will take a long time to generate and consume too many system resources (memory, power, and processor), especially for IoT devices. With all these challenges in mind, we created a framework that will give any IoT device the capability to accomplish the key generation and encapsulation processes of any post-quantum cryptographic scheme of any size with less time and resource consumption, as shown in Fig. 1 that shows that not all the devices are selected by the client in this process.

B. *The divide and Conquer Algorithm*

We use the divide and conquer algorithm for any post-quantum cryptography key generation process alongside the zero trust security model which allows us to assume that all devices and users, whether inside or outside a network’s perimeter, should not be trusted automatically. When the zero trust model is applied to IoT devices, each device must be verified and permitted before it can connect with other devices or use network resources. This necessitates the identification and verification of each device, as well as the application of access control policies to define what actions the device is permitted to conduct within the network.

In Algorithm 1, we considered a key generation mechanism in which the client IoT uses a large key big enough for the single IoT client to efficiently generate with a limited time constraint. The divide and conquer algorithm below provides an effective method to break down the task into smaller tasks. Every IoT device has a specified or defined capability threshold $\mathcal{T}$ that provides information on how much computation it can handle or support at any given time. This

Algorithm 1: Divide and Conquer the Matrix

```
1  Function DivideAndConquerProblem($\mathcal{P}$):
2      $\mathcal{P} \leftarrow Data$
3      if $\mathcal{P} < \mathcal{T}$ then
4          Solve the problem $\mathcal{P}$ directly
5      end
6
7      else
8          while $\mathcal{P} > \mathcal{T}$ do
9              Divide the problem into smaller four
                 subproblems, $\mathcal{SP}_1$, ..., and $\mathcal{SP}_4$;
10             $solution \leftarrow$ Merge (DivideAndConquer($\mathcal{SP}_1$),
                 $\ldots$, DivideAndConquer($\mathcal{SP}_4$))
11         end
12     end
13 return
```

variable depends on the available memory, CPU frequency, etc. Algorithm 1 depicts clearly how this is done using a divide and conquer approach. The key operation in this pseudocode is DivideAndConquer, which accepts a problem as input. It determines if the problem is modest enough to be solved immediately. If so, it immediately addresses the problem. Otherwise, it divides the issue into smaller subproblems, performs the DivideAndConquer technique on each subproblem recursively, then combines the subproblem solutions to achieve the final answer. After that, the final solution is returned.

C. Network Agnostic

IoT (Internet of Things) networks are a pivotal component of the modern digital landscape, enabling a wide range of devices to connect and communicate over the internet. One of the key advantages of our design is that many IoT devices are built with network capabilities. This provides our system with the ability to enhance efficiency and convenience[21].

Our system is network agnostic. This proposed solution will work on any network. We implemented our prototype on the WIFI network. Our solution is designed to be IoT network agnostic, meaning it can be implemented on any type of IoT network, whether it is a LoRaWAN, NB-IoT, Zigbee, or any other standard. Our approach is adaptable and compatible with a wide range of IoT communication protocols. This flexibility ensures that our solution can seamlessly integrate with various existing and emerging IoT infrastructures. However, the choice of the network should depend on factors such as range requirements for the network, power constraints, and cost considerations.

IV. Technical Implementation

In the following section, we will go through the scheme's implementation.

A. System Overview

The main idea is to speed up the key generation and encapsulation processes on IoT resource-constrained devices. Because most post-quantum cryptographic schemes use large keys to maintain the security resistance of the system and data against Indistinguishable under Chosen-Plaintext Attack (IND-CPA) [22][23]. The larger the key, the more resistant it is against attacks, and the harder this key is to be generated by low computational power IoT devices on the network. The importance of IoT devices may be understood from numerous angles. Below are some of the main four.

Power Management. IoT devices are frequently battery-powered or operate in inaccessible or distant regions with restricted power sources. Power consumption management is crucial for guaranteeing the lifetime and dependability of IoT devices. Low-power modes, sleep modes, duty cycling, and energy harvesting are popular techniques used to optimize power consumption and extend the operating life of IoT devices [24].

Bandwidth and Network Connectivity. Depending on the deployment context, IoT devices may have restricted network access, such as low bandwidth, inconsistent connectivity, or excessive latency. This can have an impact on IoT devices' real-time data transfer, reaction time, and overall performance. For IoT devices to function successfully in resource-constrained network contexts, efficient data transfer, data compression, and adaptive communication protocols are required [25][26].

Fault Tolerance and Resilience. IoT devices working in resource-constrained contexts must be resilient to a wide range of failures and faults, including hardware breakdowns, network interruptions, and data corruption. Fault tolerance methods, redundancy, error recovery, and robust data processing are critical for assuring IoT device dependability and availability in resource-constrained contexts [27][28].

Processing Capacity and Memory. Because of their compact form size and resource restrictions, IoT devices may have limited processing capability and memory. Because IoT devices must work within certain constraints, this might have an influence on their performance and usefulness. Optimizing algorithms, data processing, and memory use is critical to ensure IoT devices operate efficiently in resource-constrained contexts [29][30].

B. Technical Implmentation

To maintain the same security level when implementing post-quantum cryptography IoT devices, we created a system that leverages the computation power of any IoT device by taking advantage of other devices and systems on the network. We leveraged a parallel and distributed computing environment that permits low computation and energy power IoT devices on the network to send parts of their key generation and encapsulation process to a subset of devices on the network.

To accomplish this strategy, we implemented our scheme on top of the Dispycos parallel and distributed computing framework[31]. This Python-based framework allows for distributed computing using asynchronous programming, where tasks are distributed across multiple nodes in a cluster or network. It is built on top of the popular Dispy framework, which allows for parallel execution of Python code across multiple CPUs or nodes in a network. Dispycos framework

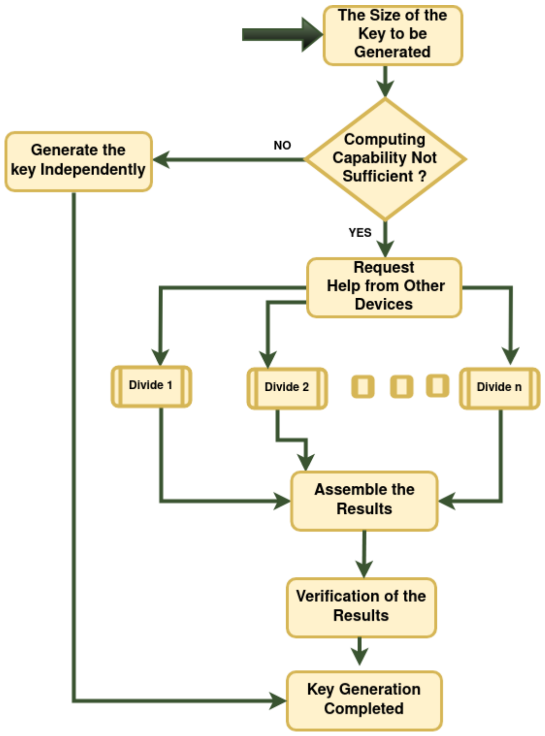
The Size of the Key to be Generated
Computing Capability Not Sufficient ?
NO
Generate the key Independently
YES
Request Help from Other Devices
Divide 1
Divide 2
Divide n
Assemble the Results
Verification of the Results
Key Generation Completed

Fig. 2. The flow diagram shows different steps that a client takes in the key generation process

extends the functionality of Dyspy by adding the ability to write asynchronous code using the ”asyncio” library in Python [32]. This allows for efficient use of resources in a distributed environment, as well as the ability to write complex distributed programs.

We used the Dispycos framework to create computing nodes on the network that can run on separate devices and then submit tasks to these nodes that we call helpers for execution. These tasks are parts of the post-quantum key generation functions that can be executed asynchronously, and using the framework, we distribute the tasks to the available nodes and the communication among these nodes. The results are returned to the IoT device (client) that needed the help to execute the code for key generation or encapsulation.

We also implemented the zero-trust security model approach in which all devices and users, whether within or outside a network's perimeter, are not immediately trusted. In other words, zero trust demands that every device or person seeking to access a network or a resource, regardless of location or status, be validated and then allowed. In this model for IoT devices, security controls are enforced at the network and device level perimeters. This means that each device should have its own security controls in place to ensure that it is not vulnerable to attack, and it is properly configured to adhere to the network's security policies. Zero trust for IoT devices also requires continuous monitoring and visibility into device activity, as well as rapid response capabilities to mitigate any possible security incidents.

We combined this with the divide and conquer algorithm that helps us break down the inversion of larger matrices into smaller, more manageable sub-matrices, solving each sub-matrix independently and then combining the solutions to the sub-matrices to solve the original matrix. This approach is often used when a problem is too complex to be solved in a straightforward manner. Therefore, the problem is divided into smaller parts that are easier to solve. Each of these smaller parts can be solved independently, using the same algorithm or a different one, and the solutions are combined to obtain the final solution. The divide and conquer algorithm is commonly used in various fields such as computer science, mathematics, and engineering to solve problems such as sorting, searching, and matrix multiplication [33]. Fig 2 illustrates this approach.

C. Implementation

This divide and conquer method accepts an array A, a left index l, and a right index r as input. It initially determines if the left and right indices are equal. If so, the method returns the element at index l. Otherwise, the array is divided in half at index $m = (l + r)/2$. It then recursively calls itself on both the left and right halves of the array. Finally, the results of the two recursive calls are combined.

For the implementation, we utilized four Raspberry Pis, which we had installed Ubuntu 20.04 LTS Desktop. One of the Raspberry Pis is configured as a client node, while others are configured as helper nodes, as shown in Fig 3.

Fig. 3. The setup of our system on Raspberry Pi

We chose the Raspberry Pis because these were the closest to the IoT devices we could access. However, the same code can run on other low-computational power IoT devices.

V. Case Study

In this section, we use the example of McEliece Post-Quantum Cryptosystem where the private key of any IoT client is a combination of three matrices S, G, and P. S is a non-singular $k \times k$ matrix, referred to as the scrambler matrix. G is a $n \times k$ generator matrix that can correct up to t errors on a binary linear block code over $GF(2)$. this means G is able to correct up to t errors with the $n - k$ parity bits [20].

The McEliece cryptosystem is based on the difficulty of decoding a randomly generated linear code. The fundamental concept is to encode a message as a vector and then add random mistakes to the vector. The vector that results is then broadcast through a noisy channel. By correcting the vector flaws, the receiver can decode the message. The McEliece cryptosystem's security is predicated on the difficulty of decoding a random linear code. This challenge is considered computationally infeasible for both classical and quantum computers if the code is sufficiently long and the errors are randomly sampled.

Our main goal is to perform the computations involved in the key generation process using the three matrices S, G, and P. This process corresponds to the McEliece private key and public key generation, respectively. Algorithm 2 below shows the pseudocode for the divide and conquer algorithm in the case of McEliece post-quantum cryptography.

Algorithm 2: Divide and Conquer Algorithm for McEliece Key Generation.

```
block size $b$ (power of 2), number of levels $l$ (such that $b^l$ is
  the dimension of the code)
random generator matrix $G$
**Function** DivideandConquer($\mathcal{M}$):
    **if** $\leftarrow 0\$$ **then**
        **return** random $b \times b$ matrix $\mathcal{M}$
    **end**
**return**
**else**
    $M_1 \leftarrow GenerateMatrix[i-1]$
    $M_2GenerateMatrix[i-1]$
    Divide $\mathcal{M}_1$ and $\mathcal{M}_2$ into $b \times b$ submatrices
**end**
**for** $j \in range(b^i)$ **do**
    **for** $k \in Range(b^i)$ **do**
        $N_{j,k} \leftarrow randomb \times b\mathcal{M}$
    **end**
**end**
$P_{j,k} \leftarrow N_{j,k}M_1 + N_{j,k+b^i}M_2$ for $1 \leq j,k \leq b^i$
Combine $P_{j,k}$ into a single $b^i \times b^i$ matrix $M$
**return** $M$
```

A. Key Generation Process

The generator matrix$G$ is created for a binary linear code as part of the McEliece key creation procedure, and a random error is then added to the matrix to obtain a public key. This process could be improved by computing the private key as the collection of $P, G, andS$, and the public key **G** is the product of these private key matrices ($P \times G \times S$) using the divide and conquer approach. By recursively splitting the matrix into smaller submatrices and creating random matrices for each submatrix, the divide and conquer technique can be used to build a random generator matrix. The random generator matrix for the full matrix may be created by combining the random matrices. This algorithm performs as below:

- Select a level of l and a block size b, a power of two such that bl is the code's dimension.
- For each leaf node of the recursion tree, produce a random $b \times b$ matrix.
- Combine the matrices from the prior level to get a new set of matrices in three steps. i) Create $b \times b$ submatrices from the parent matrix, ii) Produce a random $b \times b$ matrix to make a new submatrix for every pair of submatrices, and iii). Create a new matrix for the current level by combining the new submatrices.
- The ultimate generating matrix for the code is the root of the recursion tree.

This approach may be used to construct a random generator matrix for a binary linear code, which can subsequently be used to generate a McEliece public key. Because the divide and conquer approach assures that the matrix is more random and has a better structure, the resultant key is safer than a key created using a fundamental random matrix. However, this approach may need more processing resources than a simple random matrix-generating algorithm.

Algorithm 3: MatrixInverse

```
  **Data:** Matrix $A$ of size $n \times n$
  **Result:** Inverse matrix $A^{-1}$
**if** $n = 1$ **then**
**return** $\left[\frac{1}{A_{11}}\right]$
**else**
    Divide $A$ into four sub-matrices $A_{11}$, $A_{12}$, $A_{21}$,
      and $A_{22}$;
    $B_{11} \leftarrow$ MatrixInverse($A_{11}$);
    $B_{22} \leftarrow$ MatrixInverse($A_{22}$);
    $B_{12} \leftarrow -B_{11} \times A_{12} \times B_{22}$;
    $B_{21} \leftarrow -B_{22} \times A_{21} \times B_{11}$;
    $C \leftarrow B_{22} + B_{22} \times A_{21} \times B_{11} \times A_{12} \times B_{22}$;
    $D \leftarrow B_{11} + B_{12} \times C \times A_{21}$;
    $E \leftarrow B_{22} \times (A_{11} - A_{12} \times B_{22}^{-1} \times A_{21})^{-1}$;
    $F \leftarrow -B_{12} \times C \times E$;
    $G \leftarrow -B_{22} \times A_{21} \times B_{11} \times (B_{22} \times A_{11} - A_{12} \times B_{21})^{-1}$;
    $H \leftarrow E \times A_{12}$;
    $I \leftarrow B_{12} \times C \times H$; $J \leftarrow E \times A_{21}$;
    $K \leftarrow G \times A_{12}$;
    $L \leftarrow G \times A_{21}$;
    $M \leftarrow B_{22} \times J + B_{22} \times L \times B_{11}$;
    $N \leftarrow D - I$; $O \leftarrow J \times M$;
    $P \leftarrow -G \times M$;
    $Q \leftarrow F \times N$;
    $R \leftarrow D + Q$;
    $S \leftarrow J \times R$;
    $T \leftarrow -G \times R$;
    $U \leftarrow O + S$;
    $V \leftarrow P + T$;
    $W \leftarrow G \times U$; $X \leftarrow W - H$;
    $Y \leftarrow V - K$;
**return** $\left[X \quad Y\; G \quad E\right]$;
```

This technique employs a recursive approach to compute the inverse of a matrix A. When the matrix has the size of one, the inverse is just the reciprocal of the solitary element in the matrix. Otherwise, the matrix is divided into submatrices, and the inversion of these smaller matrices is computed recursively.

TABLE I
PRIMARY DIFFERENCES BETWEEN THE EXECUTION IN A SINGLE RASPBERRY PI AND EXECUTION ON THREE RASPBERRY PI.

Matrix size (n x n)	Execution time Single Raspberry Pi in Second	Execution time in three Raspberry Pi in Second
64	0.74158087267151	0.707228292465
128	6.4137382420104	5.8750646697256
256	53.956084489822	40.759545955444
512	288.46228933334357	199.762945595544

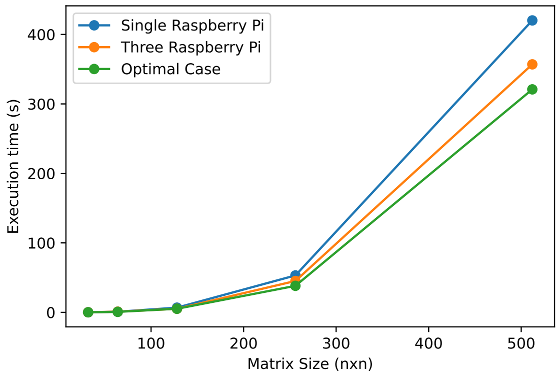

Fig. 4. The execution time at different number of devices (Raspberry Pis).

The inversed submatrices are then concatenated using matrix operations to generate the original matrix's inverse.

VI. EVALUATION

This part summarizes the findings from the system under consideration and the practicality of applying our suggested model. As described in Section IV, we developed a post-quantum scheme that enables IoT devices to generate any post-quantum cryptographic key regardless of the key size. We evaluated our system on three aspects: the execution time, the optimal number of devices, and the submatrix threshold size (i.e., the smallest size until which our algorithm keeps dividing a matrix).

A. Impact of the Matrix Size on the Execution Time

Since McEliece's post-quantum cryptography key-generation and encapsulation processes depend much more on the matrix multiplications and inverses, we simulated a program that works in that regard. Table I shows the computational time of execution on one IoT device versus three IoT devices. From Table I, we can derive that the computational time on a single Raspberry Pi is consistently getting slower compared to the one on three Raspberry Pis. This result allows us to confidently conclude that our proposed scheme, Q-SECURE, improves the computation time when using multiple IoT devices on a distributed and parallel computing network system.

The same conclusion is seen in Fig. 4. The execution time on a single Raspberry Pi is higher than on our Q-SECURE environment distributed with three Raspberry Pi.

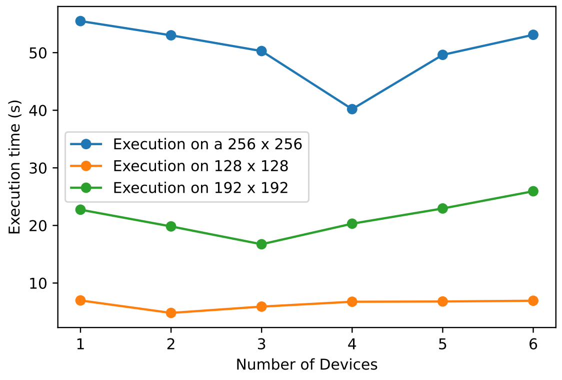

Fig. 5. The key generation time in different numbers of devices.

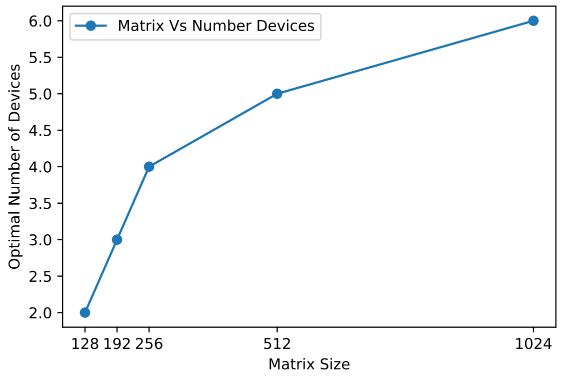

Fig. 6. The optimal number of devices for different matrix sizes.

B. Impact of the Number of Devices on the Execution Time

The next part of our evaluation is to find the optimal outcome (number of IoT devices) needed to accomplish a specific task in the shortest time possible. To realize this, we set up a network with different devices (Raspberry Pis and Virtual machines). The Virtual machines are configured with less computing power and low memory to match those of Raspberry Pis as closely as possible. Our exploration shows that for a specific matrix size, there should be a predefined number of IoT device helpers that the client must request to minimize the execution time. The result is presented in Fig. 5. In this case, we defined a squared matrix M of size 256. Fig. 5 shows that we must use four devices (three IoT helpers and one client) to compute a key generation of size 256 in about 40.2 seconds.

From our experimental results in Fig. 5, we identify the optimal number of IoT devices for different matrix sizes with the lowest execution time. This information is presented in Fig. 6, where the optimal number of IoT devices needed increases with the increase in the matrix size.

C. Impact of Submatrix Size Threshold on the Execution Time

The optimal submatrix matrix size (that we refer to as threshold) is the size $\mathbf{M}$ of the input matrix that triggers our program's divide and conquer algorithm. A threshold is an arbitrary option that can change depending on the hardware and the size of the matrices for which the determinant and inverse are to be computed. In general, the threshold should be adjusted so that, for smaller matrices, the recursive calculation is quicker than the lower triangular matrix (L) and an upper triangular matrix (U) factorization, also known as LU decomposition. Still, it is not so big that memory or processing time becomes a barrier for larger matrices.

VII. Conclusion

Post-quantum security is an important field of study for protecting IoT devices from quantum computer assaults. While there are many challenges to overcome in this area, researchers are making significant progress in developing lightweight post-quantum cryptographic algorithms, efficient key exchange mechanisms and secure firmware update mechanisms tailored to IoT devices. In this work, we put our attention on implementing post-quantum schemes on resource-constrained Internet of Things devices and effectively improving their computational efficiencies by leveraging the power of distributed and parallel computing of devices on the same network.

This work does not cover all the possible ways of improving the computation power of IoT devices when it comes to implementing such algorithms. For instance, one approach this paper could improve is that all devices present on the network do not have the same computation power and other characteristics that should be taken into consideration when distributing the work to different helper nodes. In the future, we will address this heterogeneity issue. Additionally, there is a need to implement different post-quantum cryptography systems and the importance of conducting rigorous assessments that could help in choosing the appropriate post-quantum cryptography scheme that will minimize resource consumption in accordance with the IoT device specifications and constraints.

VIII. Acknowledgment

This work is partially supported by the National Security Agency (NSA) under Award H98230-21-1-0324 and Award H98230-22-1-0327.

References

[1] V. Bindu, "Cyber security analysis for quantum computing," *Journal of IoT in social, mobile, analytics and cloud*, vol. 4, 2022.

[2] V. Bhatia *et al.*, "An efficient quantum computing technique for cracking rsa using shor's algorithm," in *2020 IEEE 5th international conference on computing communication and automation (ICCCA)*.

[3] V. Hassija *et al.*, "Forthcoming applications of quantum computing: Peeking into the future," *IET Quantum Communication*, vol. 1, no. 2, 2020.

[4] G. Alagic *et al.*, "Status report on the second round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, vol. 2, 2020.

[5] T. M. Fernandez-Carames and P. Fraga-Lamas, "Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks," *IEEE access*, 2020.

[6] T. Monz *et al.*, "Realization of a scalable shor algorithm," *Science*, 2016.

[7] H. Wu and X. Gao, "Efficient multiplier and fpga implementation for ntru prime," in *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, 2021.

[8] A. A. Abd El-Latif *et al.*, "Secure data encryption based on quantum walks for 5g internet of things scenario," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, 2020.

[9] B. Koziel *et al.*, "Sike'd up: Fast hardware architectures for supersingular isogeny key encapsulation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, 2020.

[10] H. Madushan *et al.*, "A review of the nist lightweight cryptography finalists and their fault analyses," *Electronics*, vol. 11, no. 24, 2022.

[11] M. S. Turan *et al.*, "Status report on the second round of the nist lightweight cryptography standardization process," *National Institute of Standards and Technology Internal Report*, vol. 8369, 2021.

[12] H. Cheng, D. Dinu, J. Großschädl, *et al.*, "A lightweight implementation of ntru prime for the post-quantum internet of things," in *Information Security Theory and Practice: 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11–12, 2019, Proceedings 13*, Springer, 2020.

[13] A. Kumar *et al.*, "Securing the future internet of things with post-quantum cryptography," *Security and Privacy*, vol. 5, no. 2, p. e200, 2022.

[14] Z. Liu, Choo, *et al.*, "Securing edge devices in the post-quantum internet of things using lattice-based cryptography," *IEEE Communications Magazine*, vol. 56, no. 2, 2018.

[15] C. A. Lara-Nino *et al.*, "Post-quantum cryptography for embedded systems," in *2022 IEEE Mexican International Conference on Computer Science (ENC)*.

[16] S. Paul *et al.*, "Towards post-quantum security for cyber-physical systems: Integrating pqc into industrial m2m communication," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*, Springer.

[17] E. Alkim *et al.*, "Post-quantum key exchange-a new hope.," in *USENIX security symposium*, 2016.

[18] S. Paul *et al.*, "Tpm-based post-quantum cryptography: a case study on quantum-resistant and mutually authenticated tls for iot environments," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021.

[19] L. Chen *et al.*, "Nist post-quantum cryptography standardization," *Transition*, vol. 800, no. 131A, 2017.

[20] M. Kumar and P. Pattnaik, "Post quantum cryptography (pqc)-an overview," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*.

[21] M. Aboubakar *et al.*, "A review of iot network management: Current status and perspectives," *Journal of King Saud University-Computer and Information Sciences*, 2022.

[22] Y. Wang *et al.*, "Quantum misuse attack on frodo," *Entropy*, vol. 24, no. 10, 2022.

[23] D. Kostic, "Analysis of the bike post-quantum cryptographic protocols and the legendre pseudorandom function," tech. rep., EPFL, 2020.

[24] A. H. Sodhro *et al.*, "Power-management strategies for medical information transmission in wireless body sensor networks," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, 2020.

[25] S. Alam *et al.*, "Internet of things (iot) enabling technologies, requirements, and security challenges," in *Advances in Data and Information Sciences: Proceedings of ICDIS 2019*, Springer, 2020.

[26] S. M. P. Dinakarrao *et al.*, "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2019.

[27] M. A. Shahid *et al.*, "Towards resilient method: An exhaustive survey of fault tolerance methods in the cloud computing environment," *Computer Science Review*, vol. 40, 2021.

[28] M. T. Moghaddam and H. Muccini, "Fault-tolerant iot: A systematic mapping study," in *Software Engineering for Resilient Systems: 11th International Workshop, SERENE 2019, Naples, Italy, September 17, 2019, Proceedings 11*, Springer, 2019.

[29] A. Imteaj *et al.*, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, 2021.

[30] V. A. Thakor *et al.*, "Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities," *IEEE Access*, vol. 9, 2021.

[31] "Distributed and parallel computing framework with." https://github.com/pgiri/dispy, 2022.

[32] C. Hattingh, *Using Asyncio in Python: understanding Python's asynchronous programming features.* " O'Reilly Media, Inc.", 2020.

[33] Y. Zhu *et al.*, "A high-performance hardware implementation of saber based on karatsuba algorithm," *Cryptology ePrint Archive*, 2020.