McEliece Post Quantum Cryptography for IoT

COSC470 Research Project Report

Juliet Samandari 81955545
Supervisor: Dr. Clementine Gritti

October 2021

Abstract

Quantum computers are computers that will function in a completely different way to standard computers and due to this render all our current cryptographic standards ineffective. To combat this, post quantum cryptographic standards that will keep information and communication secure, even after quantum computers are available, are being evaluated. A thorough evaluation of post quantum cryptographic standards requires inclusion of Internet of Things (IoT) devices because these devices will need to provide continued security in a post quantum environment, but have different considerations to other devices. In this report, we outline an analysis of the McEliece post quantum cryptosystem and its possible use for IoT devices. We have concluded that there is a possibility for Classic McEliece to be used for IoT devices. However, these will need to be IoT devices that are less resource constrained in order for the use of Classic McEliece to be practical.

1 Introduction

The Internet currently uses different systems to provide secure communication. Transport Layer Security (TLS), IP Security (IPsec), and Public-Key Infrastructure (PKI) are all commonly used forms of security. TLS is designed to block attempts to tamper with messages or listen in on what is being communicated [27]. IPsec is used in Virtual Private Networks (VPNs) to authenticate and encrypt messages between two devices [34]. PKI is used to ensure that public keys for devices are available and secure to be found. The Internet use standards such as X.509 digital certificates [60].

Physical quantum computers are yet to be commercially available. However, Microsoft has already created a cloud platform that provides quantum tools [7]. It is also expected that the number of organizations working in the area of quantum computing will nearly double [37]. It has been found that using an algorithm that simulates a quantum algorithm on a classic computer is up to 10 million times slower than when the algorithm is run on a quantum computer [25]. Exact differences in computing power are yet to be known but it is expected that quantum computers will change computing.

Quantum computers will also effect the public key cryptosystems that are currently used to provide security over the Internet. When quantum computers are available, the current standards used on the Internet will no longer provide

secure communication [73]. Therefore, it is mandatory that new cryptographic standards are created for Internet communication to be secure. These cryptosystems are referred to as post quantum cryptosystems because they will provide classic computers, that rely on classical computing mechanisms for their operation, with security that is robust in the face of future attacks from quantum computers. This process of standardization has already been set in motion by the National Institute of Standards and Technology (NIST) as they established a competition for new standards to be proposed [19].

One of the finalists in this competition put together by NIST is the Classic McEliece Cryptosystem. This is based on the McEliece cryptosystem, a cryptosystem first proposed in 1978 and considered a good option for standardization [18, 59].

This competition is already under way and finalists have been announced [21]. However, many of the new cryptosystems proposed have larger storage and/or computational requirements than previous standards. This is not an issue for devices such as computers, mobile phones or servers, but applying the proposed cryptosystems to Internet of Things (IoT) devices may not be as straightforward. IoT devices are often resource constrained and therefore require different considerations to other devices.

Taking this into account then poses the following question: how will IoT devices remain secure in the coming post quantum era? And more specifically, will the Classic McEliece cryptosystem be a valid post quantum cryptosystem for IoT devices?

2 Background

2.1 Internet of Things (IoT)

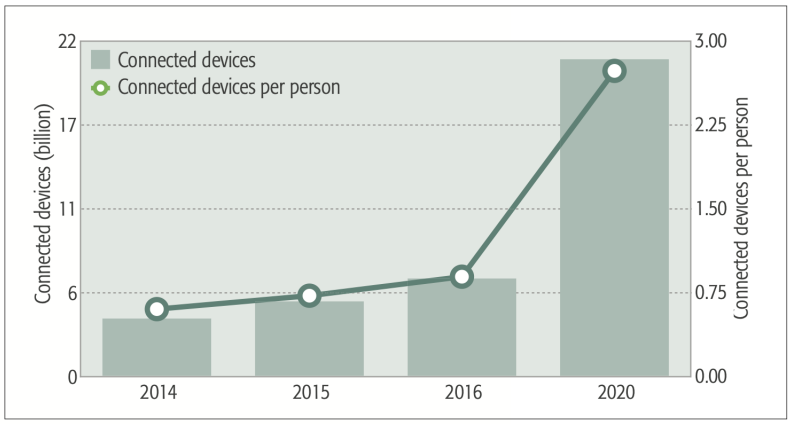
IoT is a concept outlining how there are a number of objects with communication capabilities present in society that are interconnected. Often this connection is established over the Internet, but it can sometimes occur over other communication channels [6, 77, 78].

The Internet, Radio-frequency Identification (RFID) technology and embedded computer systems, all essential elements for IoT devices, were developed in the early 1970's. However, it wasn't until 1984 that early uses of IoT technology were seen, and in 1990 the first Internet device was created. From the year 2000 it has become the standard to have Internet connectivity on all devices. Also, IoT devices as we know them now have become more prevalent[68, 70].

The number of connected devices has been greatly increasing in general, with 500 million devices connected to the Internet in 2003 compared to 50 billion de-

vices in 2020 [68]. As the overall connected devices have been increasing so have the number of IoT devices, as seen in figure 1, with devices such as virtual assistants, smart locks and smart lighting systems growing in popularity [42]. IoT is also considered one of today's trending technologies [2].

Figure 1: The number of IoT connected devices [17]



IoT has many areas in which it can be used, such as industrial and domestic applications, security, and logistics and transport. It is also expected that as the number of devices connected to the Internet increases the possible uses for IoT devices will also increase [70].

Due to the prevalence that IoT devices have, and are projected to continue having, it is imperative that there are security options when quantum computers become available [17]. This is especially crucial as one of the main concerns over the use of IoT devices is the associated possibility of security issues [70].

One other consideration for the use of IoT devices is that often in a commercial setting, such as the use of smart sensors to monitor buildings, devices are deployed and expected to work for long periods of time. With some sensors lasting three years of periodic updates without charging and their lifespan expected to be 20-30 years [1], there is a definite need to establish a viable post quantum cryptographic standard for IoT devices. Considering that these devices will likely still be in operation when quantum computers are forecasted to be commercially available, it is imperative that devices such as these will have post quantum cryptography capability as soon as possible.

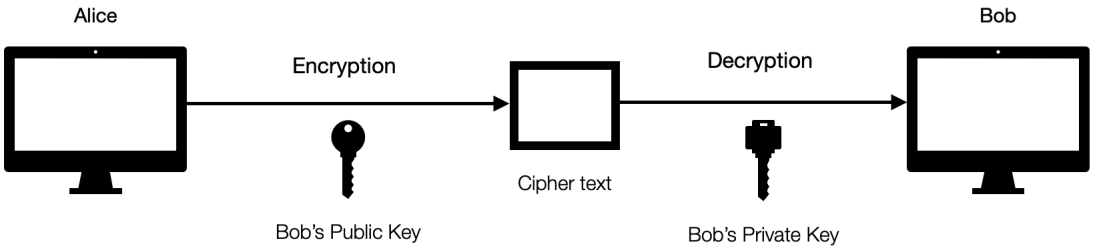
2.2 Public-Key Cryptography

Public-key encryption and digital signatures are used widely, especially to provide assurance on the Internet [8]. Public-key cryptography can be used to

provide confidentiality, integrity, authentication, and non-repudiation for individuals and organizations. Public-Key Infrastructure (PKI) uses public-key cryptosystems to provide the basis of security for email, communication and transactions on the Internet [74].

Devices that employ public-key encryption cryptosystems to manage secure communication use two different keys: one to encrypt and another to decrypt, as seen in Figure 2. In Figure 2 we assume that there are two users, Alice and Bob, communicating using public key encryption. The encryption key is publicly available, while the associated decryption key is privately held on Bob's device. This allows any device to encrypt its communication using the public key with the confidence that this communication is secure, ensuring its contents are hidden from unauthorized entities, provided the decryption key is kept secret and only known by the intended recipient, Bob [41]. An attacker will need to solve a hard mathematical problem, such as identifying the prime factors of a large number, in order to break the encryption scheme [76].

Figure 2: Diagram of a Public Key Encryption Interaction



Post quantum cryptography was first introduced in 2003 as progress was made in quantum computing; computation that has its basis in quantum mechanics [67]. It is known that the hard problems providing the basis of security for the current standard public-key encryption protocols will no longer be computationally infeasible to solve. Therefore, these encryption protocols will cease to provide security once large quantum computers are available [73]. This necessitates the creation and adoption of a new cryptographic standard that will remain secure even with the possibility of attacks from quantum computers.

It is useful to note the difference between post quantum cryptography and quantum cryptography. Quantum cryptography uses quantum mechanics and information theory in order to provide security [36] while post quantum cryptography uses classical computing to provide security from quantum computers [10].

There are many areas where it is likely that post quantum cryptography will

be used, such as hash-based cryptosystems, code-based cryptography, lattice-based cryptography and multivariate polynomial cryptography [10]. However, we will focus on code-based cryptography and specifically on the McEliece cryptosystem, one of the cryptosystems proposed to the National Institute of Standards and Technology (NIST) as a possible post quantum public key cryptography standard.

2.2.1 Digital Signature Scheme

A digital signature can be used to check the integrity of a received message and to verify the identity of the sender of the message [58]. Digital signatures provide integrity and verification by using either an encryption function or, more commonly, a hashing function. There are two actions to carry out with digital signatures: to sign a message and to verify if a received signature is valid [53, 14]. In Figure 3 we see the standard process to sign data. If we have the users Alice and Bob communicating with digital signatures, Alice will use their private key to sign the data to send and Bob will then verify the received data using Alice's public key.

Figure 3: Diagram of the Digital Certificate Signing Process

Digital signatures have been standardized since the 1990's [58]. Therefore, as digital signature schemes are also going to be affected by the availability of quantum computers, new standards are needed for digital signatures as well [19].

2.3 NIST and Post Quantum Cryptography

NIST is part of the U.S. Department of Commerce and has been providing technology and standards since 1901 [56]. NIST creates cryptographic standards and guidelines primarily for American governmental agencies to utilize for protecting their information. However, these standards are frequently used

by other organizations. NIST often engages with the cryptographic community and when an area is identified as needing a new cryptographic standard, a competition is started for researchers to share their proposed solutions [38]. NIST is looking to replace their standards for encryption, key establishment and digital signatures because they have been identified as the areas that will no longer be secure with the use of quantum computers. These standards all use public-key cryptography [57]. This competition was first proposed in 2016 but has only recently closed the submission for the third and final round in October 2020 [57].

There were originally over 50 possible post quantum cryptographic standards proposed in the first round of the NIST competition [20]. There are now four public key encryption schemes with five alternates proposed, as well as three digital signature finalists and three alternates [21]. Numbers were reduced at the end of every round and comments made for improvements to the proposed standards. Changes are still being made to these cryptosystems before standardization will take place. The draft of these standards is expected some time between 2022 and 2024 [22].

When the competition was first proposed, in 2016, only designs for quantum computers were available. Yet, it was known that their existence and use would leave systems encrypted using current cryptographic standards unsecured [57]. There are now existing examples of functioning quantum computers, such as the quantum computer Google created with a 72-qubit quantum processor and a very low error rate [73] or the quantum computer from Honeywell with quantum value 64 [45]. These devices are still considered not very computationally powerful but are the largest quantum computers known to exist. These quantum computers, due to the interference that takes place in quantum computers, are considered more powerful than a classic computer with the same number of bits for computation [69]. While quantum computers are unlikely to be prevalent in the near future, the advancements we have seen require there to be a viable alternative cryptosystem available soon so that there is adequate time to widely implement this secure alternative.

NIST has identified that it usually takes around 20 years for the framework of new cryptography standards to be rolled out. This makes it imperative to begin considering possibilities now, as predictions have been made that quantum computers at a scale large enough to make current security ineffective will be possible in roughly 20 years [19].

NIST plans to reassess the threat quantum computers pose to our current cryptosystems as soon as quantum-resistant standards for public-key cryptography are available. This would allow them to analyze whether it is necessary to discontinue the use of any current cryptographic standards [16].

2.4 Types of Post Quantum Cryptography

There are four main types of post quantum cryptosystems proposed [10].

The first type of post quantum cryptography is hash-based digital signatures. Digital signatures are commonly used for authentication and identification purposes but do not carry out any encryption. Therefore, this does not provide confidentiality but does provide integrity. Hash-based digital signatures provide security based on the collision resistance of the hash function; no two different documents will be able to produce the same digital signature. However, the number of signatures that can be produced is limited unless the size of the signatures is increased. An example of a hash-based scheme is the Merkle Signature Scheme (MSS) [14, 16].

The second type is lattice-based cryptosystems. A lattice is a set of points periodically structured in an n-dimensional space. Lattice-based problems are hard even for quantum computers and hence they provide security. Some lattice-based cryptosystems can be proven secure under the conditions of worst case hardness, which has resulted in a lot of interest in this type of cryptosystem. They are also comparatively simple, parallelizable and efficient to implement. However, it is difficult to have precise estimates of the security; while the worst case may be easy to compute, the average case is not. Some examples of lattice-based cryptography are the Goldreich–Goldwasser–Halevi/Hermitian Normal Form (GGH/HNF) public key cryptosystem, the N-th degree Truncated polynomial Ring Units (NTRU) public key cryptosystem and the Learning with errors based (LWE-based) cryptosystem [16, 54].

The third type of cryptography is code-based cryptography. These cryptosystems use error correcting codes as their underlying function to provide security. Large keys are needed to provide security and while changes have been suggested to reduce key length, this has compromised the security of the cryptosystem. However, the originally proposed code-based systems have yet to be broken and are fast to implement, even given the long key length. Some examples of code-based cryptosystems are the McEliece cryptosystem, the Niederreiter variant of McEliece and the Cryptographic File System (CFS) signature scheme [59, 16].

The final type of post quantum cryptography is multivariate polynomial cryptography. This type of cryptography uses a set of polynomials over a finite field as its basis. Solving nonlinear equations over a finite field is NP-hard, thus providing the security for these cryptosystems. Many multivariate encryption cryptosystems of this type have been proposed and then broken, so problems of this type are now being considered as more useful for digital signatures [29, 16]. Rainbow is one of the digital signature finalists for NIST's competition and is an example of a multivariate cryptosystem [28].

There have been other proposed post quantum cryptographic schemes that have not fallen under these categories. However, there is not as much confidence in these cryptosystems due to a lack of research [16]. We have, therefore, chosen not to include them in our work. Based on current research we believe that a code-based, and specifically the McEliece, cryptosystem is one of the best options for a post quantum cryptographic standard.

2.5 The Challenges of Post Quantum Cryptography

Currently, there have been three major challenges identified with implementing post quantum cryptography. The first of these challenges is efficiency. Of the proposed post quantum cryptographic schemes, none are as efficient as the currently used cryptosystems. Research has already improved the efficiency of the proposed cryptosystems. However, further research is still needed [10].

The second challenge is confidence. Many post quantum cryptosystems are newer and therefore have not had as much research done into possible attacks. This results in there being less confidence in their security. It is therefore imperative to have increased experience with post quantum cryptosystems and also for more post quantum cryptanalysis to take place [10].

The third challenge of post quantum cryptography is usability. Encryption mechanisms and digital signatures first need to be well defined and standardized by an organization such as NIST. Then a software implementation that can be used over many devices needs to be available in order for the cryptosystems to be usable [10].

The challenge specifically for the use of post quantum cryptography to provide security for IoT devices is that, due to large key sizes, many cryptography algorithms are not feasible [55]. This is especially a concern as many post quantum cryptosystems provide their added security through the use of larger keys.

We will be looking specifically at the McEliece post quantum public key cryptosystem as it addresses the three major challenges outlined above. Due to it having been proposed forty years ago, there has been plenty of research conducted with no serious threats to security yet found. It has also been found to be very efficient, even though it has large key sizes, and is well documented [59].

While McEliece is considered efficient, it is important to check if this is feasible with IoT devices and their specific limitations, such as their storage, computational power and communication.

3 The McEliece Cryptosystem

The McEliece cryptosystem was first proposed in 1978 as a possible code-based public-key cryptosystem. Code-based cryptosystems use error correcting codes as the basis for the encryption function, a function that cannot be decrypted without knowledge of the key, which provides the security of the encryption [18, 59].

The McEliece cryptosystem is based on the use of Goppa codes. Goppa codes employ modular arithmetic in order to carry out error correction [72]. These Goppa codes form the foundation for creating a publicly accessible generator matrix that other devices can use to communicate securely with the original device [51].

McEliece was the first algebraic code-based public-key cryptosystem proposed. Although there has been over 20 years of cryptanalysis of the McEliece cryptosystem, it has yet to be broken [67]. The McEliece cryptosystem uses randomization when encrypting and was the first public-key cryptosystem to do so [52].

The McEliece cryptosystem is the most researched cryptosystem of all the participants in the competition held by NIST; it has been frequently researched since its proposal over 40 years ago [51, 18]. This is one reason why McEliece is being considered a possible option for a new standard. While the key sizes are large, it is very fast to encrypt. It has also been well documented due to its history and no serious threats to the cryptosystem have been found [59].

3.1 Definitions

A ***linear code*** is an error-correcting code where linear operations are used to represent encoding and decoding, often using a generator matrix and parity check matrix respectively [24].

A ***parity check matrix*** of a linear code is a matrix that outlines what rules the elements of a linear code must satisfy [43].

The ***identity matrix*** is the matrix where if the column is equal to the row, the cell value is 1, otherwise it is 0. For example:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is the 3×3 identity matrix. Identity matrices are unique, as any matrix multiplied by the identity will remain unchanged [33].

A matrix G is a ***generating matrix*** of a linear code if its rows form a basis of that code matrix C of size $k \times n$. Conversely, given a generating matrix G, $\langle G \rangle$ gives the F-vector subspace spanned by the rows of the generating matrix. k' is the rank of the generating matrix and $\langle G \rangle$ is a code of type $[k', n]$. Therefore, $\langle G \rangle = C$ where C is the code matrix [67].

A matrix is a ***systematic generator matrix*** of a code when a matrix has the first $k \times k$ values as the identity matrix and then the next $k \times (n - k)$ such that it is a generating matrix for the code. This means the generator matrix is now in systematic form [13].

A matrix is a ***non-singular matrix*** if it has an inverse. Hence, there exists an $n \times n$ matrix B for the $n \times n$ matrix A such that

$$AB = BA = I$$

where I is the identity matrix [33].

A ***transpose vector*** is a vector that has either been transposed from a column vector to a row vector or vice versa [15]. For example,

$$V = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \quad V^T = (10011)$$

A ***permutation matrix*** is when an identity matrix has one or more rows swapped [33].

A ***codeword*** is a linear code with dimension k and length n. It is defined over a field that is a k-dimensional subspace of a set of n-dimensional vectors [72].

The ***Hamming distance*** is the number of bits in two codewords that differ from each other [72]. For example, given the codeword $x = (100011)$ and $y = (100110)$, x and y have a Hamming distance of two as only the fourth and sixth bits differ.

The ***Hamming weight*** is the distance of a codeword from the zero string, a string of all zeroes of the same length as the codeword. In other words, the Hamming weight is the number of bits that are not zero in the codeword [72]. Using the same codeword $x = (100011)$ from above, this codeword would have a Hamming weight of three as there are three one values in this codeword.

Goppa codes are the linear error correcting codes that provide the basis of the McEliece cryptosystem [51]. They are used for encryption and decryp-

tion [72].

A ***Goppa polynomial*** is a polynomial $g(z)$ with coefficients in $GF(q^m)$ where q is a prime number and m is an integer. L is a subset of elements in $GF(q^m)$ that are not roots of $g(z)$. $|L|$ gives the length of the Goppa polynomial $g(z)$ [9]. The length of the Goppa code, n is dependent on the value of L. The resulting Goppa code is defined as

$$R_c(z) = \sum_{i=1}^{n} \frac{c_i}{x - a_i}$$

where $L = a_1, a_2, ..., a_n$ and $c = (c_1, c_2, ..., c_n)$ is a vector of n codewords such that the polynomial $g(z)$ divides $R_c(z)$ [72].

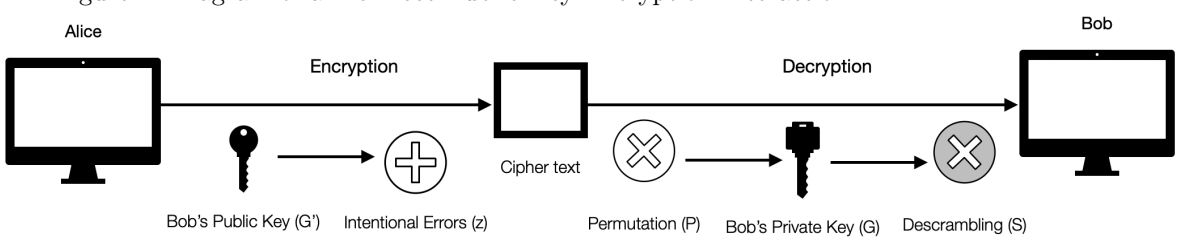
A ***binary Goppa code*** is a Goppa code where the Goppa polynomial $g(z)$ has coefficients in $GF(2^m)$ [72]. Therefore, the code has only values zero or one.

Other codes have been tried in variants of the McEliece cryptosystem but none of the other error-correcting codes used have had the same level of security as Goppa codes [52].

3.2 The McEliece Cryptosystem

In this section, we outline the structure of the McEliece Cryptosystem. The encryption and decryption processes for users Alice and Bob to communicate using the McEliece cryptosystem can be seen in Figure 4.

Figure 4: Diagram of a McEliece Public Key Encryption Interaction



First, a public key and its corresponding private key must be generated for Bob. Our variables t, k and n are integers, where t is defined as the number of errors that can be corrected, and k and n are the dimensions of the generator matrix. We produce a $k \times n$ generator matrix called G for the linear code. A random, binary, non-singular $k \times k$ scrambler matrix S as well as a random $n \times n$ permutation matrix P is selected to generate our public key.

The public generator matrix $G'$is computed by the following equation:

$$G' = SGP \tag{1}$$

Therefore, the public key is now (G', t) and the private key is (S, G, P) [52, 51].

To encrypt the message m to be sent, it must be represented as a binary string of length k. A randomly generated binary vector, z, with length n and maximum t one values is used to compute the binary vector, the ciphertext c:

$$c = mG' + z$$

This binary vector c is then sent [52, 51].

To decrypt the message m from the received encrypted value c, the inverse of the permutation matrix P needs to be calculated. This is then used to compute

$$c' = cP^{-1}$$

The decoding algorithm is then used to calculate m' from c'. The message m can then be found using the transpose vector of m and the equation:

$$m = m'S^{-1}$$

where S^{-1} is the inverse of the non-singular matrix S used for the public key generation [52, 51].

3.3 McEliece Example

In this section we shall outline an example of the McEliece Cryptosystem [71].

The Generator Matrix,

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

The Scrambler Matrix,

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

The Permutation Matrix,

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The Public Generator Matrix,

$$G' = SGP = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

We then choose our message to send as $m = (1101)$ and set our random intentional error matrix to $z = (0000100)$.

This then results in the calculated ciphertext c to send.

$$\begin{aligned} c &= mG' + z \\ &= (0110010) + (0000100) \\ &= (0110110) \end{aligned}$$

When this is received, the value $c' = cP^{-1}$ is calculated, where:

$$P^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This gives the result $c' = (1000111)$.

We can then decode c' using $y' = (1110)^T$, the transpose vector of m. This gives the receiver, Bob, the code word $m' = (1000110)$ and given the matrix G chosen it is now known that $mS = (1000)$. Therefore we can calculate the original matrix by using the matrix S^{-1} where:

$$S^{-1} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

This multiplication of mS and S^{-1} gives $m = (1101)$, the original message sent by the sender Alice [71].

3.4 Classic McEliece Proposal for NIST

The Classic McEliece cryptosystem is the variant of the McEliece cryptosystem used for the NIST Competition and is now one of the finalists to choose a standard for public-key encryption that will remain secure once quantum computers can be used for attacks [21].

The Classic McEliece cryptosystem utilizes the Niederreiter variant of the McEliece cryptosystem. This variant uses the parity check matrix of the chosen generator matrix G as the public key, rather than the generator matrix G itself. This reduces the length of the produced column vector c from n to $n-k$, where the dimension of the above matrices is $k \times n$. This reduction in ciphertext size results in notably small ciphertexts for a post quantum cryptosystem [18].

The Classic McEliece cryptosystem also specifies that the generator matrix G chosen for the ciphertext c containing the private key must be unique and in the systematic form. This reduces the key size from kn bits to $k(n-k)$ bits. These reductions in ciphertext length and key length increase the efficiency of running the McEliece key and ciphertext generation algorithms [18].

These improvements of efficiency are important initial steps for the McEliece cryptosystem to be computationally reasonable to use for public-key cryptography. However, it is not known if these improvements are enough to make the Classic McEliece scheme practical to carry out on devices that have limited storage and computational power, such as Internet of Things (IoT) devices.

These changes do bring a slight loss in security of maximum 2 bits, which given the improvement in efficiency is considered acceptable. It does result in there being a slightly smaller pool of valid keys. Therefore, it is possible that more than one key generation attempt is needed before a valid key is generated. On average 3.4 attempts are needed to generate the key but this is considered an acceptable overhead due to the reduced public key size and as the key is often kept for a period of time, depending on the sensitivity of the data [18]. If constant sensitive communication is taking place, keys will likely need to be changed more rapidly than if the McEliece cryptosystem is being used to encrypt a database that is not considered particularly sensitive.

Another consideration is that key generation takes time when carried out in software. Therefore, applications must use a public key for a certain period of time before the time needed to generate and distribute the keys is considered acceptable. However, if key generation occurs in hardware it is quite fast [18].

4 Security

The McEliece cryptosystem has been considered as a viable option for post quantum cryptography as only slight improvements to attacks, using Grover's algorithm, seem to be possible with the use of a quantum computer [13]. Grover's algorithm is a quantum search algorithm that allows an item in a group of n to be found in $\sqrt{n}$ steps, rather than $n/2$ steps [50]. More specifically, this is a quantum algorithm that allows an unordered list to be searched with complexity $o(\sqrt{n/m})$ where n is the number of items and m is the number of solutions. This is a major improvement over the fastest search algorithm, which is $o(n)$ as mentioned prior [64].

When assessing the security of the McEliece cryptosystem, the encrypted message to be sent c and the public key G' are known. Decoding the sent codes is an NP-complete problem. This makes it easier to find a solution by randomly selecting k coordinates from the message m at random. If there are no errors in the randomly selected coordinates then the equation $\bar{x} = m\bar{G}$, where $\bar{x}$ is a vector of selected x entries and $\bar{G}$ is a sub matrix of G' the public generator matrix [67]. This possibility requires a relatively long key to be used in order for security to be assured.

Two algorithms to decode McEliece encrypted messages have been proposed: Berlekamp and Patterson. The Berlekamp algorithm has the better performance of the two, and hence is the algorithm most commonly researched. The security level, in terms of the number of operations to decode, when using the Berlekamp algorithm, is comparable with current cryptography standards [49].

Patterson algorithm is a decoding algorithm used for binary Goppa codes. This algorithm finds the polynomial syndrome from the ciphertext c and depends on the random error z. This can then be used to calculate the value of the random error z and therefore decrypt the message as the addition of the random error can then be reversed [35].

Another decoding algorithm proposed to be employed to decode the McEliece cryptosystem uses information set decoding. This is considered one of the most efficient decoding techniques. An attack was carried out using this algorithm but it was not efficient enough to be able to decode McEliece ciphertexts within a reasonable period of time, provided long keys are used. The cryptanalysis also concluded that there is little room for improvement using this method. Therefore, a better attack would only be likely using a different method of decoding [32].

It can be seen that attacks on the McEliece cryptosystem using different decoding algorithms have been attempted and none have been successful. Therefore, there are no serious threats to the security of the cryptosystem known [59].

4.1 Classic McEliece Security Considerations

The Classic McEliece cryptosystem uses the established best practices in order to ensure security. Uniform, random input is used for the public key establishment and the private key G is computed by calculating the hash of this input. Also, if decryption fails, rather than returning a key encapsulation mechanism failure, a separate private key and the ciphertext are hashed together and this value is then returned. The hash function used is SHAKE256 [18]. SHAKE256 is one of the two expendable output functions from the SHA-3 standard given by NIST [30].

Hash functions are designed so that they are easy to compute but difficult to reverse. This means that hashing data is straightforward. However, finding the data from a hashed value should not be possible. This assurance is based on the assumption that the hash function is well constructed so collisions will not take place; meaning that it is not possible for two different data values to have the same hash value [23].

The Classic McEliece cryptosystem is designed to provide security such that the encrypted value is indistinguishable from random values under adaptive chosen ciphertext attacks (IND-CCA2 security) [18]. This means that even if an attacker is able to obtain many chosen ciphertext and plaintext pairs they can not use them to decode other ciphertexts [48].

It should also be taken into account that the Classic McEliece Cryptosystem was first proposed for the NIST competition. Therefore, this variant does not have the same history of cryptanalysis as the original McEliece cryptosystem.

5 McEliece System for IoT Devices

The McEliece cryptosystem is quite fast but requires a significant amount of storage, as the public keys are very long. For instance, a 460,647-bit key is necessary to have the same level of security as a 1,024-bit public key for RSA, a current public key cryptosystem used to secure communication over the Internet [40], or a 160-bit elliptic curve key, another currently used public key cryptosystem [4]. It is discussed in a recent paper that the large matrices required for both the public and private keys are the main disadvantage of the McEliece cryptosystem for IoT devices [31]. This storage requirement is likely not an issue for standard computers but may be a concern for many IoT devices, that have become popular due to the fact that they are inexpensive. This results in many IoT devices having limited processing power and storage capacity in order to minimize the cost of production. This means the McEliece cryptosystem is likely infeasible for IoT devices [46].

As can be seen in Table 1, even in comparison with other post quantum cryptosystems the McEliece cryptosystem requires significantly larger keys in

Table 1: Approximate Key Size Summary for the Post Quantum Cryptosystem Types [44, 61, 11, 18]

Algorithm	Public Key Size	Private Key Size
Code Based	1 MB	14 KB
Lattice Based	766 B	843 B
Multivariate Based	124 KB	95 KB
Hash Based	32 B	64 B

order to provide security [47].

The use of the McEliece system for IoT has previously been investigated, specifically in the context of RFID tags. However, this research was undertaken purely as a theoretical calculation. The authors approached the issue from a specific perspective and they did not implement any practical testing [46]. Additionally, the McEliece cryptosystem has previously been tested on a Raspberry Pi 3 and it was concluded that the main threat to the cryptosystem's operation was the processing power required [66].

There is, however, no research on whether the efficiency improvements made to the McEliece system since that study was conducted will mitigate the issue noticed above with a Raspberry Pi 3.

The submission package for the Classic McEliece scheme also discusses the possibility of compression for private keys. Although private keys are much smaller than public keys, this could be helpful in the event of wanting to reduce the necessary storage space. Four different kinds of truncation have been proposed [18].

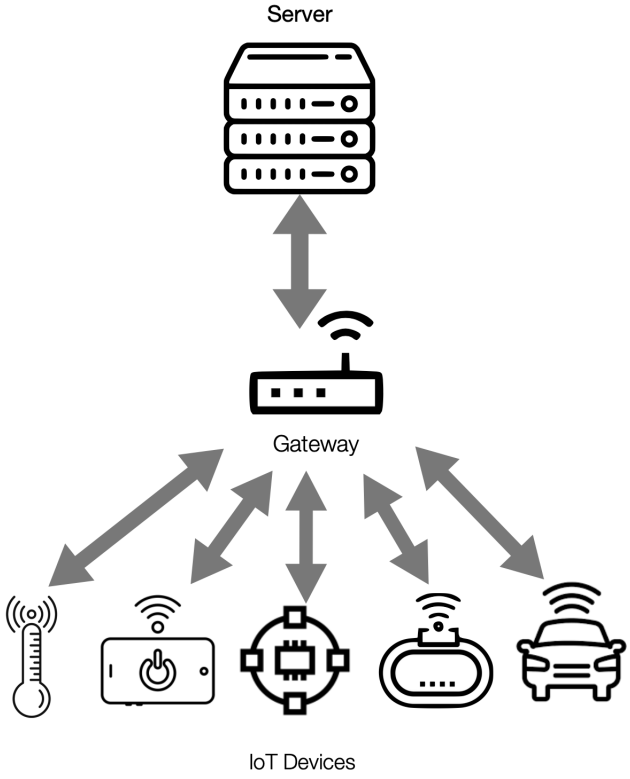
The private key for the Classic McEliece variant has five elements. Depending on the truncation option, the final compressed private key can have one, two, three, or four of the original five elements. Each reduction in private keys saves significantly more space. However, it can be quite computationally expensive if the removed value needs to be recomputed. Therefore, all truncation options have a trade-off between how much space is saved and how much work is required to regenerate public key values if needed again later [18].

6 Design

Our first consideration when designing our experiment is what set up we assume for our IoT devices. We decided on a layout where there are many IoT devices all connecting to one central gateway that then communicates with the organization's other servers for storage or processing. This is seen in Figure 5.

This is referred to as a Device-to-Gateway communication model. This model provides better interoperability between the many IoT devices. It is also possible for a gateway to even be a smart phone if it has the correct application installed [68]. We assume that our gateway is able to receive and process received packets from the IoT devices in such negligible time that the many connected devices will not differ in processing time to just having one IoT device connected.

Figure 5: The connection setup of our IoT devices to a central gateway



We also had to decide on what device we would use to model an IoT device for our experiment. We decided to use a Raspberry Pi to model an IoT device as it allowed us to use a less powerful device but also one that would be general, given

that many IoT devices run on specific embedded systems or micro-controllers.

6.1 IoT Communication Protocols

We then looked into existing communication protocols for IoT devices. The majority of the common communication protocols have security included as part of the protocol. We found that there were two overarching categories of protocols: Low Power Wide Area Networks (LPWANs) and short range networks. One consideration common to all the protocols was the need for communication to be low power, requiring a low power consumption [3]. This highlighted the importance of power consumption when running protocols on an IoT device and showed that the power required to use the Classic McEliece cryptosystem will be an important consideration.

7 Implementation

In order to carry this out we had to get a Raspberry Pi, prepare it and then run the suggested standardization implementation on the Raspberry Pi.

We used the Raspberry Pi 3B+ to carry out our experiments. This model of Raspberry Pi has the specifications: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC at 1.4GHz 1GB LPDDR2 SDRAM 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN [62]. We installed the operating system image for our Raspberry Pi onto a USB stick with 4 GB of memory.

The implementation we used was the Classic McEliece cryptosystem code provided as part of the submission package to NIST. This setup will be what is used as a future standard, with possible minor adjustments, if this cryptosystem is selected. This is an implementation with over 10 modules of code and is written in the C programming language [18]. We chose to use the mceliece348864 implementation as it is the smallest key size possible for the required security level by NIST and therefore has the shortest implementation time. This makes this version of the McEliece cryptosystem the most practical to be used for IoT and other resource constrained devices.

This version of Classic McEliece cryptosystem has public keys of size 261,120 bytes, private keys of size 6,492 bytes, a ciphertext of size 128 bytes and a session key of 32 bytes. This version of the Classic McEliece cryptosystem has IND-CCA2 KEM, Category 1 security protection [18]. This means that even if it is possible to submit several ciphertexts to a decoding oracle that returns the related plaintext it is still unable to derive the private key used [26].

The Classic McEliece system implementation also required the use of other standard NIST libraries for things such as hash generation, which were difficult to find and had undergone changes in the available git repositories since when

this implementation was first proposed. The eXtended Keccak Code Package (XKCP) collates the open source implementations for the Keccak defined cryptographic systems into a repository [39]. Keccak is the name of the hash function used and the standards are named SHA3

It was also necessary to install OpenSSL in order for the implementation to function as this is used for secure communication. OpenSSL is an open source software for the Transport Layer Security (TLS) protocol, which was previously known as the Secure Sockets Layer (SSL) protocol [63].

There is also the situation that this proposal has only just been put forward. As a result, previous work had not been done on the Classic McEliece cryptosystem. Therefore, information about its implementation was limited solely to the submission package provided to NIST for the post quantum cryptography competition [18].

7.1 The Effect of COVID-19

Due to our lockdown this year it was necessary to move our implementation setup from the lab environment to be run on a laptop. This required a few extra steps to be taken. First, it was necessary to find the relevant connectors and ensure that SSH was still possible with the new device on a different network. Second, a virtual machine running a Linux operating system was set up and used in order to be able to run the code locally on our personal device. This was needed as it was not possible to add the necessary libraries for connection and hashing on a MacOS Operating System. Finally, we setup our Raspberry Pi disk image on a USB stick rather than a secure digital (SD) card as there was no SD card reading or writing capability in our home setup but changes needed to be made to permissions in order for the Raspberry Pi to connect to our personal device.

7.2 Experiment

Our experiment was carried out on a Raspberry Pi 3B+ to test the time taken to generate keys, encrypt messages, and decrypt ciphertexts using this device. Our Raspberry Pi was running the Raspberry Pi OS Lite and had a CPU frequency of 1.4 GHz [62].

For comparison, the same steps of key generation, encryption, and decryption were run and the time taken recorded while using our own laptop. This was carried out on a Linux Mint 19.3 Virtual Machine on a 2017 MacBookPro running MacOS Big Sur. The Linux virtual machine has a base memory of 3072 MB and 2 CPU processors and runs at a CPU frequency of 2.9 GHz.

We ran the full suite of the Classic McEliece cryptosystem (key generation, encryption, and decryption) 15 times and recorded the time taken for each

Table 2: Table of the Specifications of Devices Used in Our Experiment

Device Type	CPU	RAM
Raspberry Pi 3	1.40 GHz	4 GB
Laptop	2.90 GHz	16 GB
titan0 [18]	3.50 GHz	32 GB

different action. We ran and recorded the time taken for these three steps individually and averaged them to match the values submitted to NIST in the Classic McEliece submission package.

The Classic McEliece cryptosystem submitted to NIST implemented and analyzed the run time on a super computer. This computer is called titan0 and has an Intel Xeon E3-1275 v3 (Haswell) CPU that is running at 3.50GHz. It also is on a machine with 32GB of Random Access Memory (RAM) [18]. We will compare the results given by our experiments on devices with different processing powers with those written in the submission package.

An overview of the specifications of the three different devices used is seen in Table 2.

8 Results

8.1 Experimental Performance Analysis

Running the Classic McEliece cryptosystem on our laptop, we had the mean value of 0.143s and a median value of 0.148s for key generation. Key generation also had a range of 0.14s and a standard deviation of 0.0437s. For encryption, our laptop had a mean of 0.00016s and a median of 0.0001s as well as a range of 0.0009s and a standard deviation of 0.0002s. Decryption had a mean time of 0.0246s and median time of 0.0268s. The range for decryption is 0.0276s and the standard deviation is 0.0069s.

Carrying out the implementation on the Raspberry Pi gave a mean value of 5.95s for key generation and a median of 5.43s, with a range of 2.66s and a standard deviation of 0.8996s. For encryption, the mean time taken was 0.0428s and the median time was 0.03913s. Also, the range is 0.0038 and the standard deviation is 0.0013s. Finally, decryption had a mean of 1.0619s and a median of 0.9697s with a range of 0.489s and a standard deviation of 0.1657s.

Table 3: Table of the Experimental Average Time Taken to Generate Keys, Encrypt, and Decrypt on Different Machines

Device Type	Key Generation	Encryption	Decryption
Raspberry Pi 3	5.952 s	0.0428 s	1.0619 s
Laptop	0.1433 s	0.00016 s	0.02463 s
titan0 [18]	16.58 s	0.0125 s	0.0383 s

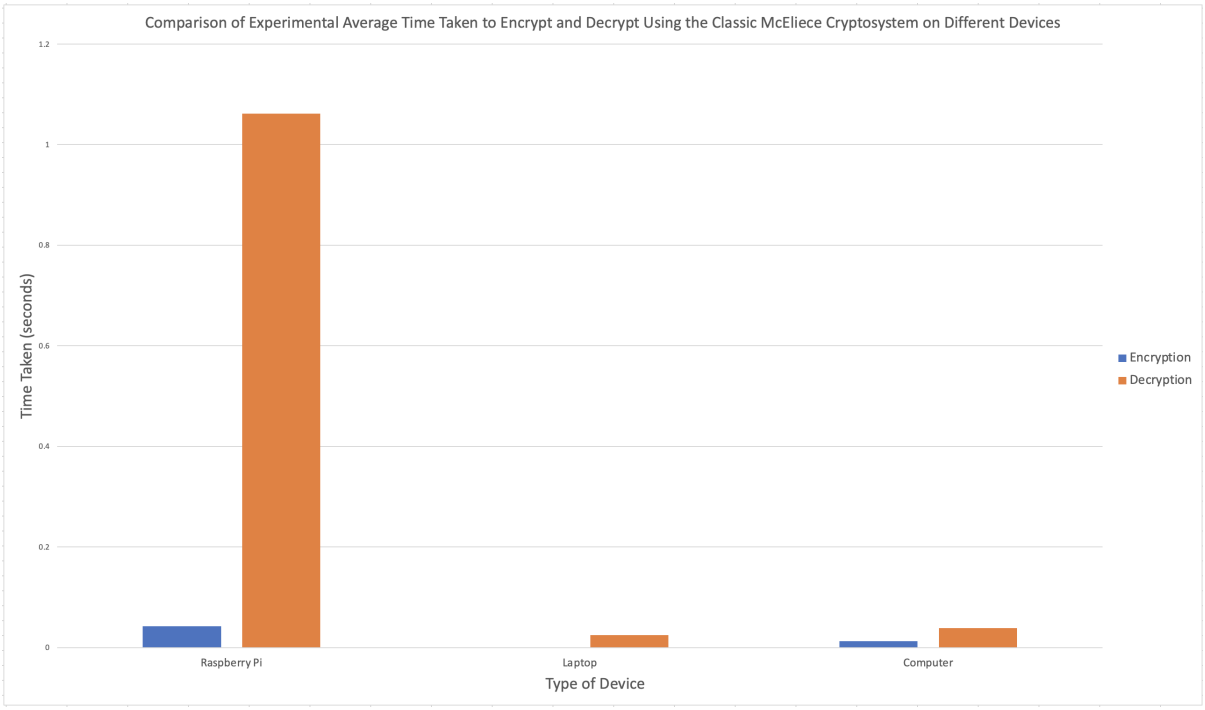
The provided results with the Classic McEliece cryptosystem submission package, for the version mceliece348864, are a mean of 16.58s (46,526,112 clock cycles) and a median of 13.29s (58,034,411 clock cycles) for key generation, where key generation is successful 29% of the time [18]. For encryption, the submitted performance analysis says that encryption has a mean of 0.0127s (43,832 clock cycles) and a median of 0.0125s (44,350 clock cycles), when run on the titan0 computer [18]. The submission package outlines the time taken for decryption is a mean of 0.0385s (134,184 clock cycles) and a median of 0.0383s (134,745 clock cycles) using their computer [18].

An overview of the average values for the three different devices can be seen in Table 3.

The trends in encryption and decryption can be seen in Figure 6. The figure shows how decryption is always more time consuming than encryption, though the difference in time taken varies depending on the device. The Raspberry Pi has the greatest difference in time by a large margin. It can also be seen that the time taken to encrypt and decrypt on a laptop has taken less time than the computer, which is the opposite of what we would have expected due to the higher processing power of the computer.

Similarly, for the time taken for key generation we can see in Figure 7 that the laptop has taken less time than the computer to generates keys. However, in addition the Raspberry Pi also generated keys in a shorter period of time than the computer, in just over a third of the time, which is quite different to what would be expected. The standard deviation for key generation with the Raspberry Pi is quite high so it may be because of this high value that we are not getting a value matching our expectation.

Figure 6: Comparison of the Experimental Average Time Taken to Encrypt and Decrypt on Different Machines



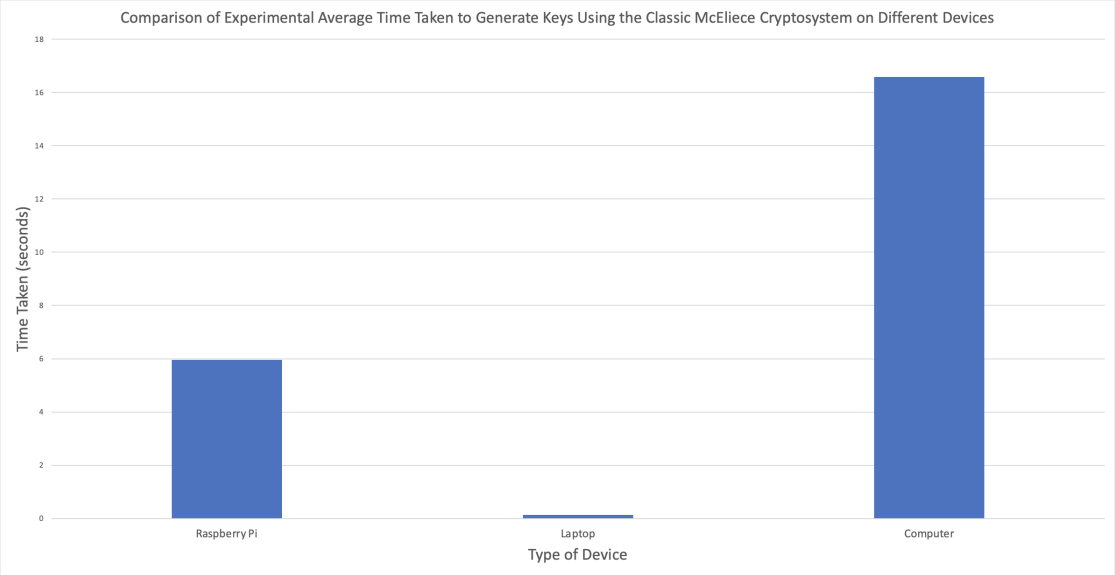
In all of the sets of data we have been provided we can see a general increase in time taken for the experiment as devices become more constrained. However, this progression is not as clear as what we would expect. In order to assess how different our values are to the expected values when running this implementation of the Classic McEliece cryptosystem we calculated a theoretical performance analysis to compare with our values.

8.2 Theoretical Performance Analysis

The Classic McEliece submission package has included in their performance analysis the number of Central Processing Unit (CPU) cycles required on average for the different tasks: key generation, encryption and decryption [18].

This allows us to propose the theoretical time taken for different types of devices given the frequency of their CPU is known. This is because the frequency gives the number of clock cycles in a second. Therefore, in the following section, we will compare the time taken to carry out the three steps on a Raspberry Pi 3B+, an IoT device with a higher clock speed (300 MHz), and an IoT device

Figure 7: Comparison of the Experimental Average Time Taken to Generate Classic McEliece Cryptosystem Keys on Different Machiines



with a lower clock speed (8 MHz). Frequency of IoT devices are taken from the maximum and minimum frequencies for IoT devices suggested at NIST's post quantum cryptography standardization conference [5].

Table 4: Table of the Theorized Average Time Taken to Generate Keys, Encrypt, and Decrypt on Different Device Types

Device Type	Key Generation	Encryption	Decryption
Laptop	16.04 s	0.015 s	0.046 s
Raspberry Pi 3	41.45 s	0.032 s	0.099 s
High CPU IoT Device	193.45 s	0.148 s	0.449 s
Low CPU IoT Device	2418.10 s	1.848 s	5.614 s

For a laptop, such as the one we are using for our experiments, that runs at

a CPU frequency of 2.9 GHz average values of 16.04s for key generation, 0.015s for encryption, and 0.046s for decryption are expected when running the Classic McEliece cryptosystem.

For the Raspberry Pi 3B+ the following average values are expected when running the Classic McEliece cryptosystem: 41.45s for key generation, 0.032s for encryption, and 0.099s for decryption.

For the higher CPU IoT device, the following average values are expected when running the Classic McEliece cryptosystem: 193.45s for key generation, 0.148s for encryption, and 0.449s for decryption.

For the lower CPU IoT device, the following average values are expected when running the Classic McEliece cryptosystem: 2418.10s for key generation, 1.848s for encryption, and 5.614s for decryption.

A comparison of the time taken for encryption and decryption for these three devices is seen in Figure 8. It can be seen that the time taken for decryption is slightly more than encryption on all devices but as the CPU frequency decreases the difference between the two values also increases. It can also be seen that the difference in run time between a 1.4GHz and a 300MHz CPU is less noticeable as the difference between a 300MHz and 24 MHz CPU.

We can also see in Figure 9 the general trend of there being a marked increase, especially for the IoT device with a low CPU. Also of note is that, even though the trend is similar, the scale is quite different as the low CPU IoT device is forecasted to take 2,418 seconds, or approximately 40 minutes to generate keys, while decryption would only take 5.614 seconds.

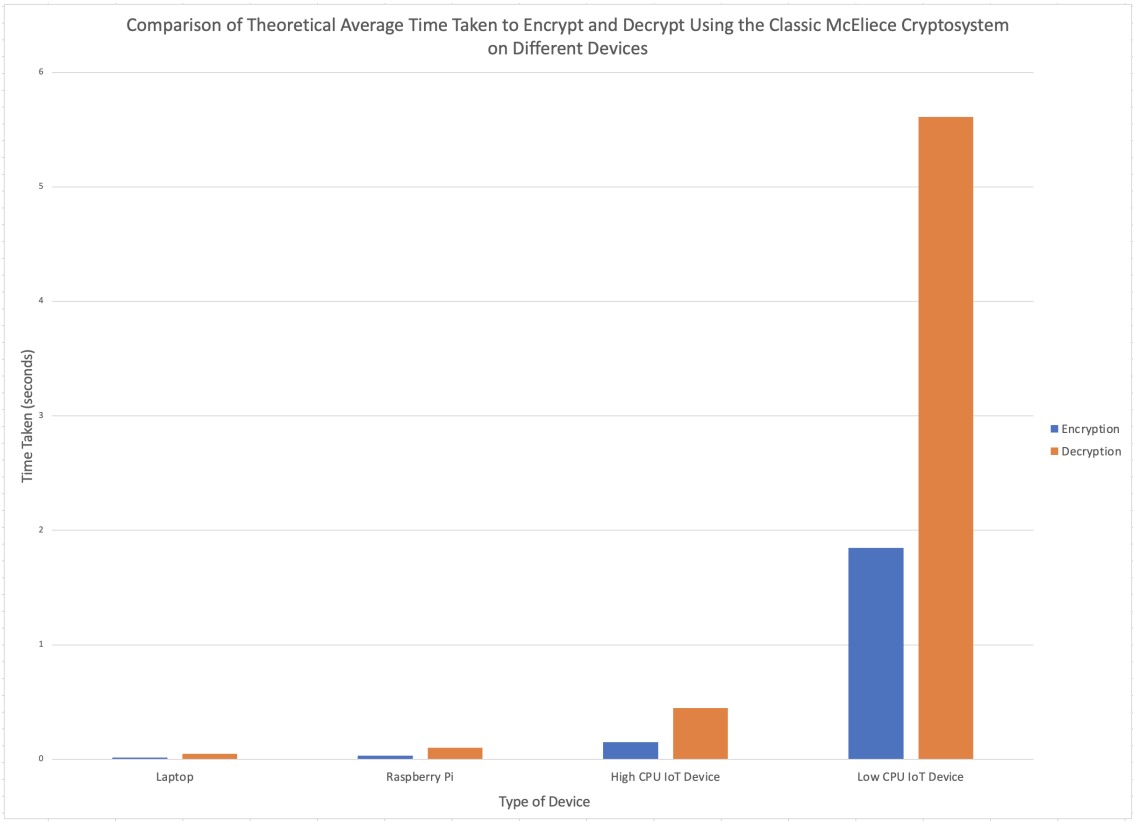
8.3 Comparison of Theoretical and Experimental Performance Analysis

We then used the values found through experimenting on our Raspberry Pi and laptop as comparative points to the theoretical values calculated to view the similarity, or differences, between these values.

First, we compared the time taken for encryption, as seen in Figure 10. For this process, the experimental value was larger than the theoretical when using the Raspberry Pi, by about 0.2 seconds. However, the experimental value for encryption on the laptop was considerably less than the theoretical value, less than a tenth of that value.

We then compared the time taken to decrypt in our experiment compared to the theorized time, seen in Figure 11. It can be seen that for the laptop the values are very similar, with the experimental value being slightly smaller. In

Figure 8: Comparison of the Theorized Average Time Taken to Encrypt and Decrypt on Different Device Types

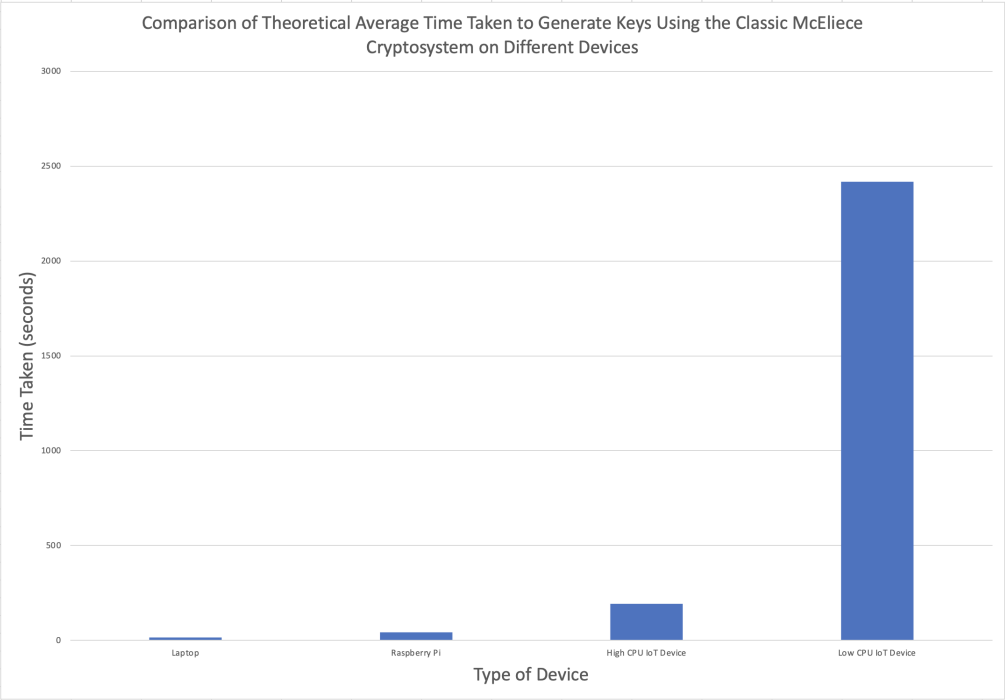


contrast, we can see that the experimental time taken to decrypt on the Raspberry Pi is noticeably longer than it was theorized. This could possibly be due to the fact that the implementation has a correctness check when decrypting to ensure that all decrypted ciphertexts match the messages that have been encrypted. However, as this difference is only seen on the Raspberry Pi and not on the laptop it is difficult to make any conjecture on its cause.

Finally, we compared the time taken for key generation in our experiment and the expected times, as can be seen in Figure 12. It can be seen that the time taken for key generation in our experiment is a fraction of what is expected in the implementation.

Through this comparison we noted that except for one case, the experimental time for decryption on a Raspberry Pi, the experimental values are smaller than the theoretical ones. This is unexpected but is encouraging that perhaps the

Figure 9: Comparison of the Theorized Average Time Taken to Generate Classic McEliece Cryptosystem keys on Different Device Types



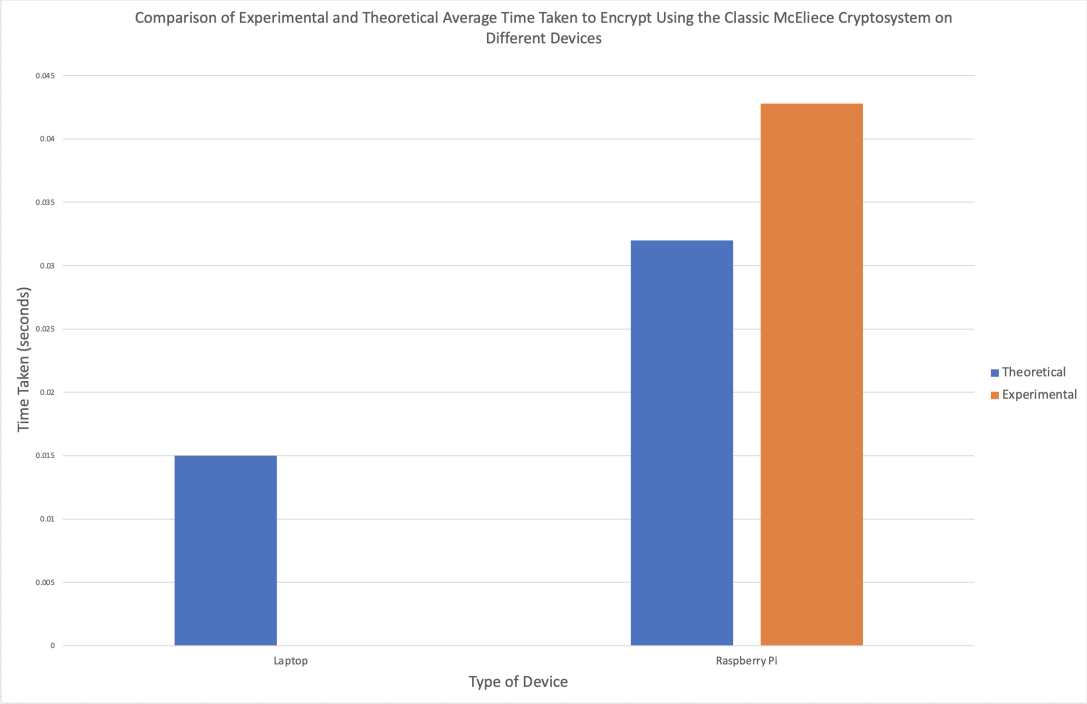
same will hold for the smaller IoT devices and their theorized delay will in fact be shorter than projected.

9 Discussion

One of the most unexpected areas of our results was that the values for the time taken for key generation were significantly shorter than what would be expected given the submission package. This may possibly be due to the fact that the seeds for random number generation used for the experiment were the ones suggested in the implementation files. Therefore, they could have possibly been chosen so that they would always successfully generate keys.

The Classic McEliece cryptosystem submitted to NIST implemented and analyzed the run time on a computer with a CPU AT 3.5 GHz and 32 GB of RAM, while many laptops have only 8GB or 16GB of RAM and a CPU running at less than 3 GHz [18]. This means that the provided performance analysis is in fact not particularly realistic when keeping in mind most devices would not have this capability. It is seen that the submission package has tried to mitigate

Figure 10: Comparison of the Theorized and Experimental Average Time Taken to Encrypt Using the Classic McEliece Cryptosystem on a Raspberry Pi and Laptop

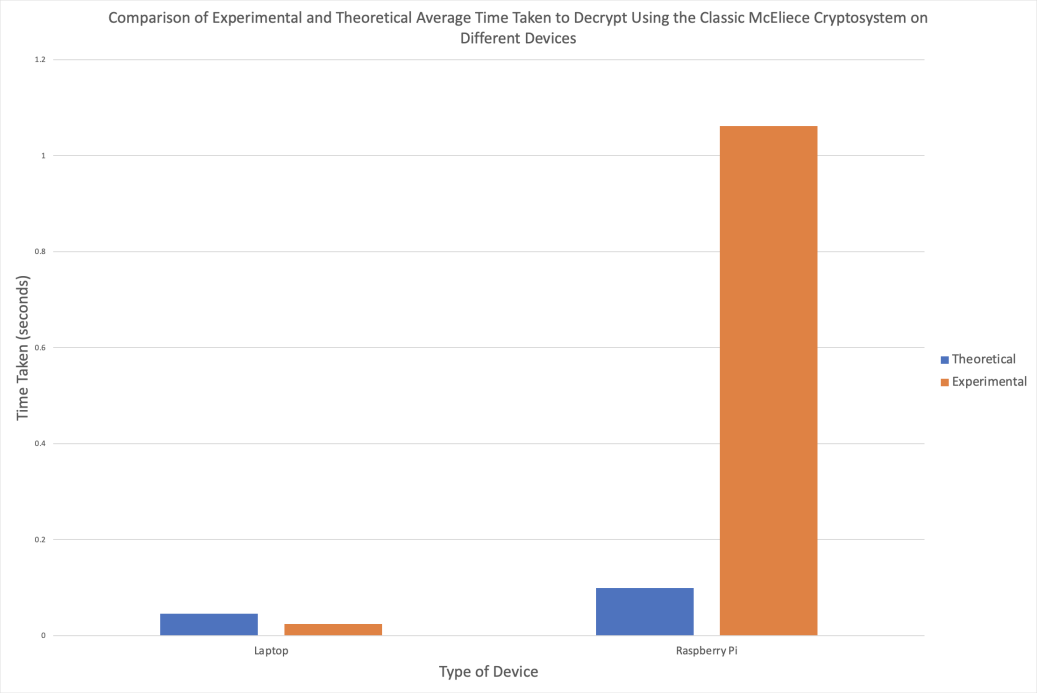


this by providing values, such as timing, according to the number of clock cycles so that it can be related to other devices easily.

However, as seen in the theoretical analysis the difference in CPU for IoT devices compared to even a Raspberry Pi, let alone a computer, creates a large impact on the time taken for implementation of the Classic McEliece cryptosystem. Consider how the theoretical time for key generation on a low power CPU shows it would take about 40 minutes to generate the keys. Additionally, this time doesn't take into account the possible need to carry out key generation several times due to key generation being unsuccessful. This quite obviously makes key generation impractical on a device this constrained.

One of the advantages for the Classic McEliece cryptosystem is that the ciphertext sizes are all very small; considerably smaller than the other post quantum cryptosystems with the same level of security. They also are consistent for each level of security; only when the key size increases does the ciphertext size increase. Also, this increase is markedly less; a public key that is approximately

Figure 11: Comparison of the Theorized and Experimental Average Time Taken to Decrypt Using the Classic McEliece Cryptosystem on a Raspberry Pi and Laptop



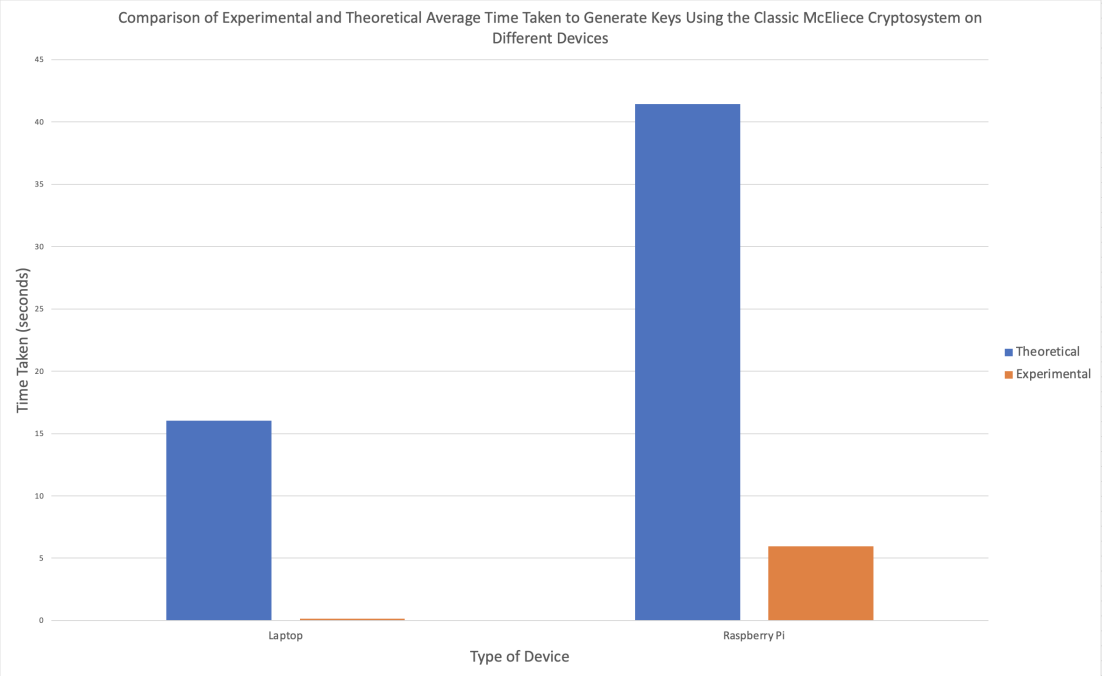
twice the size has a 60 Byte increase [18].

Another advantage is that the time taken for encryption and decryption is still quite low, even for the lower CPU of an IoT device. This means that the more repeated actions will not have as much of an overhead, as key generation should only need to occur periodically rather than every time a message is sent or received.

There is also the possibility to not need any RAM to be used by the device using the McTiny algorithm. This algorithm has clients use a small network server that will stream ephemeral keys and set up the new sessions. This would increase the practicality of using the Classic McEliece cryptosystem for IoT devices, as there will be less of a storage requirement for a specific device because it would be distributed to the central server [12]. This luckily mitigates the concern of storage for these large keys, but unfortunately still doesn't take into account the concerns around their generation.

We can consider these times in relation to the period of information to send.

Figure 12: Comparison of the Theorized and Experimental Average Time Taken to Generate Keys Using the Classic McEliece Cryptosystem on a Raspberry Pi and Laptop



If an IoT device is sending information back to the gateway server every 10 minutes or every hour then delays of several seconds would likely have little effect on the communication between the IoT device and the gateway. However, if the IoT device is sending updates over a period of a minute, or shorter, these delays will become noticeable and start affecting the functioning of the sensor and its communication.

The feasibility of using the Classic McEliece cryptosystem will also depend on the sensitivity of the data being transmitted by the IoT device to the gateway. If the data is considered confidential, the public key will be changed more often, perhaps even every day. However, if the data is less sensitive it may be considered acceptable to keep the same key for several days or even months, provided the system hasn't been compromised in any way. This variance in key generation will also affect whether the overhead for generating public keys will be considered worth the expense. In general, it would be best to generate new keys as infrequently as possible.

It should also be taken into account that the probability that key generation

is successful is approximately 29%. Therefore, it would be expected that on average 3-4 attempts would be needed until a key was successfully generated. Using the geometric formula we can calculate $E = 1/P$ where P is the probability of success and E is the expected number of attempts [75]. This gives that it is expected that 3.45 attempts ($1/0.29 = 3.45$) are needed before the key has been generated successfully. This would mean that the actual time taken to generate a valid key would be approximately calculated after $3.45 * t$ where t is the time taken to generate a key.

Overall, the marked difference between our experimental results and the theoretical results has highlighted the importance of collecting different types of information to calculate the computational power required to run this cryptosystem. It also raises the question of what other factors, other than the CPU, will affect the time taken to implement the Classic McEliece cryptosystem.

10 Future Work

This work could be used to implement another proposed post quantum cryptosystem, such as lattice based cryptosystems, and compare its suitability for use with IoT devices. This could also be conducted with an actual IoT device, such as a device with some of the standard IoT micro controllers. NIST has recommended the Arm Cortex-M4 [65]. This would be done to provide a better understanding of the use of this cryptosystem with IoT devices.

Another possible area for future work would be to carry out the experiment looking at factors other than time, such as more detailed processing power information or battery consumption information when using the Classic McEliece cryptosystem on a more resource constrained device.

This experiment could also be carried out with several IoT devices, or Raspberry Pis, all connecting to one central server to see if the multiple devices change the interaction between each IoT device and the server.

It would also be important to look into how post quantum secure digital signatures could be created and used by IoT devices and possibly how the inclusion of encryption and a digital signature may affect an IoT device.

11 Conclusion

Through our report we have analyzed and explained the need for, and use of, post quantum cryptography. We have also considered how the requirements for a cryptosystem used by IoT devices will be different than one that is used by other types of devices. We have looked at possible post quantum options and selected the Classic McEliece cryptosystem as a viable option. We have analyzed

its use for IoT through its implementation on a Raspberry Pi and considered how this differs to its use for standard computers and super computers.

We have found that there is a possibility for the Classic McEliece cryptosystem to be used for IoT devices. With a device such as a Raspberry Pi delays were acceptable and for other IoT devices with a higher CPU the Classic McEliece cryptosystem could be a good idea for post quantum security. However, for other more constrained IoT devices it will likely not be a practical choice due to the large delays.

This report has outlined how the McEliece cryptosystem generates keys, encrypts and decrypts data. It also discusses the changes that the Classic McEliece cryptosystem has made and how these will be helpful for its use, especially for IoT devices. It also looks at the implementation time on different device types and how it differs depending on the device.

The main areas we have discussed are: the importance of creating post quantum cryptographic standards that can also be used for IoT devices, and the importance of analyzing the performance of a cryptosystem on many types of devices, including a more resource constrained device in order to have a more holistic understanding of the functionality of that cryptosystem.

This work highlights not only the work that needs to continue being done in the area of post quantum cryptography but also the steps that are already taking place. This work has shown that it is possible to use post quantum cryptographic standards for IoT device. It also takes into account the differences there are between devices and how the cryptographic scheme will function on each device. We have also contextualized how the smaller CPU values of IoT devices will seriously affect the time taken to complete tasks.

References

[1] Lorawan sensor battery: Long lifespan sensor battery: Milesight. https://www.milesight-iot.com/blog/lorawan-sensor-battery/, Jun 2021. Retrieved October 19, 2021.

[2] Accenture. Accenture technology vision 2018, 2018. Retrieved October 1, 2021 from "https://www.accenture.com/t20180227T215953Z__w__/us-en/_acnmedia/Accenture/next-gen-7/tech-vision-2018/pdf/Accenture-TechVision-2018-Tech-Trends-Report.pdf".

[3] Shadi Al-Sarawi, Mohammed Anbar, Kamal Alieyan, and Mahmood Alzubaidi. Internet of things (iot) communication protocols. In *2017 8th International conference on information technology (ICIT)*, pages 685–690. IEEE, 2017.

[4] Moncef Amara and Amar Siad. Elliptic curve cryptography and its applications. In *International workshop on systems, signal processing and their applications, WOSSPA*, pages 247–250. IEEE, 2011.

[5] Derek Atkins. Third pqc standardization conference. NIST, Jun 2021. Retrieved September 17, 2021 from `https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/atkins-requirements-pqc-iot-pqc2021.pdf`.

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[7] Kevin Babitz. Quantum computing is now publicly available, Feb 2021. Retrieved October 11, 2021 from `https://medium.com/swlh/quantum-computing-is-now-publicly-available-df1abbc38578`.

[8] Messaoud Benantar. The internet public key infrastructure. *IBM Systems Journal*, 40(3):648–665, 2001.

[9] Elwyn Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, 1973.

[10] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[11] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2129–2146, 2019.

[12] Daniel J. Bernstein and Tanja Lange. Mctiny: Fast high-confidence post-quantum key erasure for tiny network servers. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1731–1748. USENIX Association, 2020. Retrieved October 19, 2021.

[13] Daniel J Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. In *International Workshop on Post-Quantum Cryptography*, pages 31–46. Springer, 2008.

[14] Johannes Buchmann, Erik Dahmen, and Michael Szydlo. Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–93. Springer, 2009.

[15] Aydin Buluç, Jeremy T Fineman, Matteo Frigo, John R Gilbert, and Charles E Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 233–244, 2009.

[16] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[17] Chi Cheng, Rongxing Lu, Albrecht Petzoldt, and Tsuyoshi Takagi. Securing the internet of things in a quantum world. *IEEE Communications Magazine*, 55(2):116–120, 2017.

[18] Tung Chou, Carlos Cid, Simula UiB, Jan Gilcher, Tanja Lange, Varun Maram, Rafael Misoczki, Ruben Niederhagen, Kenneth G Paterson, Edoardo Persichetti, et al. Classic mceliece: conservative code-based cryptography. Oct 2020. Retrieved June 1, 2021 from `https://classic.mceliece.org`.

[19] Information Technology Laboratory Computer Security Division. Post-Quantum Cryptography: CSRC, Mar 2021. Retrieved June 1, 2021 from `https://csrc.nist.gov/projects/post-quantum-cryptography`.

[20] Information Technology Laboratory Computer Security Division. Round 1 submissions - post-quantum cryptography, 2021. Retrieved October 1, 2021 from `https://csrc.nist.gov/Projects/post-quantum-cryptography/Round-1-Submissions`.

[21] Information Technology Laboratory Computer Security Division. Round 3 submissions - post-quantum cryptography: Csrc, Mar 2021. Retrieved June 1, 2021 from `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[22] Information Technology Laboratory Computer Security Division. Workshops and timeline - post-quantum cryptography, 2021. Retrieved October 1, 2021 from `https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline`.

[23] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 430–448, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[24] John Daintith and Edmund Wright. *A dictionary of computing*. Oxford University Press, 2008.

[25] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3), Aug 2016.

[26] Alexander W Dent. Ecies-kem vs. psec-kem. Technical report, Citeseer, 2002.

[27] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. *RFC*, 5246:1–104, 2008.

[28] Jintai Ding, Ming-Shing Chen, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. 2020. Retrieved October 10, 2021 from `https://www.pqcrainbow.org/`.

[29] Jintai Ding and Bo-Yin Yang. Multivariate public key cryptography. In *Post-quantum cryptography*, pages 193–241. Springer, 2009.

[30] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. *Federal Information Processing Standards Publication*, 2015.

[31] Tiago M Fernández-Caramés. From pre-quantum to post-quantum iot security: A survey on quantum-resistant cryptosystems for the internet of things. *IEEE Internet of Things Journal*, 7(7):6457–6480, 2020.

[32] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. Cryptology ePrint Archive, Report 2009/414, 2009. Retrieved June 1, 2021 from `https://eprint.iacr.org/2009/414`.

[33] William H. Ford and Inc Books. *Numerical linear algebra with applications: using MATLAB*. Elsevier/Academic Press, 2015.

[34] Sheila Frankel, Karen Kent, Ryan Lewkowski, Angela D Orebaugh, Ronald W Ritchey, and Steven R Sharma. Guide to ipsec vpns. *US Department of Commerce, Technology Administration, National Institute of Standards and Technology*, 2005.

[35] Mariya Georgieva and Frédéric de Portzamparc. Toward secure implementation of mceliece decryption. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 141–156. Springer, 2015.

[36] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Reviews of modern physics*, 74(1):145, 2002.

[37] Jonathan Greig. 6 experts share quantum computing predictions for 2021, Nov 2020. Retrieved October 12, 2021 from `https://www.techrepublic.com/article/6-experts-share-quantum-computing-predictions-for-2021/`.

[38] Cryptographic Technology Group. NIST Cryptographic Standards and Guidelines Development Process, March 2016. Retrieved June 1, 2021 from `https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.7977.pdf`.

[39] Seth Hoffert Michaël Peeters Gilles Van Assche Guido Bertoni, Joan Daemen and Ronny Van Keer. Xkcp. `https://github.com/charlespwd/project-title`, 2021. Retrieved October 19, 2021.

[40] Hamed Hellaoui, Mouloud Koudil, and Abdelmadjid Bouabdallah. Energy-efficient mechanisms in security of the internet of things: A survey. *Computer Networks*, 127:173–189, 2017. Retrieved June 1, 2021 from `https://www.sciencedirect.com/science/article/pii/S1389128617303146`.

[41] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[42] Software Testing Help. 18 most popular iot devices in 2021 (only noteworthy iot products), Apr 2021. Retrieved June 1, 2021 from `https://www.softwaretestinghelp.com/iot-devices/`.

[43] Raymond Hill. *A first course in coding theory*. Oxford University Press, 1986.

[44] Philip S Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing ntruencrypt parameters in light of combined lattice reduction and mitm approaches. In *International Conference on Applied Cryptography and Network Security*, pages 437–455. Springer, 2009.

[45] Honeywell. The world's highest performing quantum computer is here. Retrieved June 1, 2021 from `https://www.honeywell.com/us/en/news/2020/06/the-worlds-highest-performing-quantum-computer-is-here`.

[46] F. Ivanov, E. Krouk, and A. Kreshchuk. On the lightweight mceliece cryptosystem for low-power devices. In *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, pages 133–138, 2019.

[47] Rickard Johansson and Thomas Strahl. *Post-quantum secure communication on a low performance iot platform*. PhD thesis, Master's thesis, Lund University, 2016.

[48] Eike Kiltz and John Malone-Lee. A general construction of ind-cca2 secure public key encryption. In *IMA International Conference on Cryptography and Coding*, pages 152–166. Springer, 2003.

[49] Matthias Leclerc. An implementation of the mceliece-cryptosystem. *ACM SIGSAC Review*, 9(2):1–4, 1991.

[50] Shu-Shen Li, Gui-Lu Long, Feng-Shan Bai, Song-Lin Feng, and Hou-Zhi Zheng. Quantum computing. *Proceedings of the National Academy of Sciences*, 98(21):11847–11848, 2001.

[51] Robert J McEliece. A public-key cryptosystem based on algebraic coding theory. *The Deep Space Network Progress Report*, 42-44:114–116, 1978.

[52] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press, Boca Raton, 1997.

[53] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[54] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[55] Effy Raja Naru, Hemraj Saini, and Mukesh Sharma. A recent review on lightweight cryptography in iot. In *2017 international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*, pages 887–890. IEEE, 2017.

[56] NIST. About NIST, June 2017. Retrieved June 1, 2021 from `https://www.nist.gov/about-nist`.

[57] NIST. NIST Asks Public to Help Future-Proof Electronic Information, Jan 2018. Retrieved June 1, 2021 from `https://www.nist.gov/news-events/news/2016/12/nist-asks-public-help-future-proof-electronic-information`.

[58] Corporate Nist. The digital signature standard. *Communications of the ACM*, 35(7):36–40, 1992.

[59] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In *Post-quantum cryptography*, pages 95–145. Springer, 2009.

[60] Radio Perlman. An overview of pki trust models. *IEEE network*, 13(6):38–43, 1999.

[61] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. Selecting parameters for the rainbow signature scheme-extended version-. *IACR Cryptol. ePrint Arch.*, 2010:437, 2010. Retrieved August 23, 2021.

[62] Raspberry Pi. Buy a raspberry pi 3 model b. Retrieved October 17, 2021 from `https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/`.

[63] Matt Caswell Andy Polyakov Bodo Moeller David von Oheimb Richard Levitte, Stephen Hensen. openssl. `https://github.com/openssl/openssl.git`, 2021. Retrieved October 17, 2021.

[64] Juanjo Rué and Sebastian Xambó. Mathematical essentials of quantum computing. In *Preprint. Seminar on Quantum Processing*, 2010.

[65] Markku-Juhani O Saarinen. Mobile energy requirements of the upcoming nist post-quantum cryptography standards. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 23–30. IEEE, 2020.

[66] Gokay Saldamli, Levent Ertaul, and Krishnaraj MA Menon. Analysis of mceliece cryptosystem on raspberry pi 3. In *Proceedings of the International Conference on Security and Management (SAM)*, pages 42–48, 2018.

[67] Narcís Sayols and Sebastià Xambó-Descamps. Computer algebra tales on goppa codes and mceliece cryptography. *Mathematics in Computer Science*, pages 1–13, 2019.

[68] Neha Sharma, Madhavi Shamkuwar, and Inderjit Singh. The history, present and future with iot. In *Internet of Things and Big Data Analytics for Smart Generation*, pages 27–51. Springer, 2019.

[69] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[70] Priya Suresh, J Vijay Daniel, Velusamy Parthasarathy, and RH Aswathy. A state of the art review on the internet of things (iot) history, technology and fields of deployment. In *2014 International conference on science engineering and management research (ICSEMR)*, pages 1–8. IEEE, 2014.

[71] Math Department UC Denver. The mceliece cryptosystem). Retrieved October 5, 2021 from `http://www-math.ucdenver.edu/~wcherowi/courses/m5410/ctcmcel.html`.

[72] Ashley Valentijn. Goppa codes and their use in the mceliece cryptosystems. *Syracuse University Honors Program Capstone Projects*, (845), 2015.

[73] Petros Wallden and Elham Kashefi. Cyber security in the quantum era. *Communications of the ACM*, 62(4):120–120, 2019.

[74] Joel Weise. Public key infrastructure overview. *Sun BluePrints OnLine, August*, pages 1–27, 2001.

[75] Eric W. Weisstein. Geometric series. Retrieved October 17, 2021 from `https://mathworld.wolfram.com/GeometricSeries.html`.

[76] Michael J Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information theory*, 36(3):553–558, 1990.

[77] Felix Wortmann and Kristina Flüchter. Internet of things. *Business & Information Systems Engineering*, 57(3):221–224, 2015.

[78] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International journal of communication systems*, 25(9):1101, 2012.