IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

Evaluation of Performance, Energy, and Computation Costs of Quantum-Attack Resilient Encryption Algorithms for Embedded Devices

Basel Halak, Thomas Gibson, Millicent Henley, Cristin-Bianca Botea, Benjamin Heath, and Sayedur Khan

School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.
Corresponding author: Basel Halak (basel.halak@soton.ac.uk)

This work was supported in part in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R007268/1.

ABSTRACT The accelerated development of quantum computers poses a direct threat to all current standards of public key encryption, for example, the Shor algorithm exploits the superposition state of the qubits to solve the problem of integer factorization in polynomial time, rendering all systems whose security relies on this hard mathematical problem not secure. Public key encryption algorithms are used in a multitude of applications that from the core of the digital world (e.g., emails, banking, digital currency, defense, and communication.). The prospects of a quantum machine that can break such systems are too risky to ignore, even if such a computer still needs thirty years to build. This is because adversaries can be storing data now to decrypt later aka. SNLD attack, moreover, some systems have an operational lifetime that spans more than thirty years (e.g., defense, aviation industry). Consequently, the work has already started to develop quantum-attack resilient security schemes. The number of Internet of Things (IoT) devices is expected to be around 29 billion in 2030, forming a significant portion of all computing machines. Most of these will be implemented as embedded systems with limited resources. Consequently, assessing the energy and computational overheads of the quantum-attack resilient security schemes is vital. This work presents a comprehensive study that evaluates the energy and performance costs of the proposed solutions in resource-constrained devices, in comparison with the existing schemes. This was achieved through the development of a testbed that emulates a client-server configuration, wherein both devices perform mutual authentication and then agree on a shared key using the TLS protocol. A Raspberry Pi 3b+ was used as a server, and a client in the first set of experiments. Raspberry Pi Pico W was the client in the second group of tests. The results of the evaluation have shown that Kyber1-Dilithuim-2 is the most resource-efficient solution, it outperforms all other PQC algorithms, including the current scheme that uses elliptic curve cryptography. Our study has also shown the digital signature scheme Sphinx+ is associated with significant latency and energy costs so may not be suitable for IoT-type devices.

INDEX TERMS: Security, Post Quantum Cryptography, Embedded Devices, Internet of Things.

I. INTRODUCTION

Asymmetric encryption algorithms are essential for many digital systems for constructing key agreements and digital signature schemes. These are used in many applications including secure communication, banking, and digital currencies. The most widely used public key algorithms(i.e. RSA and Elliptic Curve) rely on the hidden subgroup problem, albeit in different settings[1, 2]. These settings include the use of one of the following three hard-to-solve mathematical problems: integer factorization, the discrete logarithm, and the elliptic-curve discrete logarithm

problem. The security of the above-mentioned public key cryptographic algorithms essentially relies on the difficulty of solving these intractable problems using classic computers. To factorize a large integer number (n) the best-known algorithm is the general number field sieve with a theoretical asymptotic running time proportional to an exponential function of n[3]. The time required to solve the other hard problems is similar.

However, in 1994 Peter Shor proposed a more efficient algorithm for integer factorization that runs in polynomial time [4], but requires access to a quantum computer. Therefore recent accelerated advances in quantum computer technology [5] pose a direct security threat to all digital infrastructure reliant on public key algorithms [6]. This has worldwide repercussions because of the increased dependence on technology and the desire for security and privacy. If quantum computers could easily break current public encryption standards, everything from banking to browsing the internet would leave end users at risk.

For now, quantum computers are still highly experimental with limited power; however, there are currently significant resources being invested around the world to further develop this technology. A recent McKinsey report predicts that some businesses with optimization problems may start to significantly benefit from quantum computer technology as early as 2026 [7].

From a cyber security threat perspective, there are two major risks associated with the development of quantum computers. The store-now-decrypt later (SNDL) attack, wherein the adversary stores the encrypted data, which they have maliciously obtained, to be decrypted in the future when a quantum computer is available. The second risk is more pressing and associated with certain type of systems that are deployed now and has an operational lifetime of over 30 years, which means there is a high probability that they will be still operational when a sufficiently powerful quantum computer is made, hence vulnerable to hacking.

The above risks made it necessary to find alternative asymmetric key encryption algorithms that are resilient to the quantum threat. This process was initiated in 2016 by the National Institute of Standards and Technology (NIST) by releasing a call for proposals for post-quantum algorithms, following its report in April of the same year that indicated that a possibility of quantum technology to render the commonly used RSA algorithm insecure by 2030 [8]. The NIST post-quantum algorithm standardization process is expected to conclude in 2024, however, this is only the first step in a long transition journey to update all systems. The history of cryptographic standards adoption shows that getting the world to migrate from one set of standards to the next can take decades.

Billions of digital systems need to be re-configured to use quantum-resistant algorithms. This transition process is particularly challenging for IoT (Internet-of-Things), edge computing, and battery-operated devices, which do not necessarily have the computation capacity nor the required energy to run post-quantum public-key algorithms.

The number of IoT devices worldwide is forecast to almost triple from 9.7 billion in 2020 to 29 billion in 2030, which will form a significant portion of all computing machines[9]. A recent study in this area has shown that selected NIST algorithms require significant additional memory and computing resources, compared to current public encryption methods [10]. What is more, such costs significantly increase if protection against differential attacks is required. For example, the work in[11] shows that implementing a countermeasure to differential attacks for the CRYSTAL-KYBER (i.e. the NIST chosen algorithm for key exchange mechanisms) leads to a more than fivefold increase in the implementation overheads. One solution, in this context, is the use of optimized dedicated hardware implementations of post-quantum functions, which may have smaller area and energy requirements [12-14], however, this approach may not always apply to the lower end of the IoT device range, which has architectural cost limitations.

With the expected large growth in resource-constraint IoT devices and the pre-existing reliance on this technology in applications such as industrial control systems and critical national infrastructure, it is imperative to ensure viable solutions to deploy quantum attack-resilient security algorithms on such devices.

Previous work in this area was limited in scope in terms of the algorithms being considered or the metrics of comparison being assessed. The contributions of this work are as follows:

- It performs a comparative analysis of the energy and computational costs of all quantum-resilient cryptographic methods, in a networked environment.
- It investigates the feasibility of implementing post-quantum public key algorithms on a resource-constrained device and explores the expected costs.
- It develops an experimental testbed to assess the performance, energy, and computational overheads, which emulate server-client key agreement scenario, one of the most widely used applications for public key cryptographic algorithms.

To the best of our knowledge, this is the first time such a comprehensive study has been conducted, in terms of the scope of the solutions being considered, the metrics being evaluated, and the test environment that emulates a practical use case of quantum-computer attack –resilient encryption algorithms(i.e. a key agreement scheme.)

The structure of the remainder of this paper is as follows. Section 2 reviews related work. Section 3 explains in detail the analysis methodology. The design of the testbeds is discussed in section 4 . Section 5 evaluates the performance of quantum-resilient solutions. Finally, conclusions are drawn in Section 6.

II. RELATED WORK

This section first introduces the main types of key agreement schemes using public key cryptography, then outlines the principles of TLS protocol used in this context. Next, an overview of the quantum resilient scheme is provided. Finally, a summary of previous studies related to this work.

A. Key Agreement using Public Key Algorithms

These methods primarily include.

1) A key Exchange Scheme such as Diffie Hellman protocol[15], where two parties wanting to communicate generate ephemeral key pairs sign their ephemeral public key with their static private key and then send their signed ephemeral public key to each other. Both parties receive the other signed ephemeral public key and verify it using the other static public key, stored by a certificate authority (CA) or in a public key infrastructure (PKI). Now they combine their ephemeral private key with the other's ephemeral public key to create a shared secret.

2) A key encapsulation method (KEM)[16] is a scheme with public and private keys, where a sender uses the public key of an intended receiver to create a ciphertext (encapsulation) containing a randomly chosen symmetric key. The ciphertext can then be decrypted the ciphertext using the receiver's private key.

KEM is a unilateral protocol whereas key exchange is a bilateral protocol, i.e. both parties take part in constructing the shared key, however, in some cases, KEM is easier to design and build, and they are being proposed for the new post-quantum public key algorithms.

B. Transport Layer Security (TLS)

Transport Layer Security (TLS) is a network protocol that allows secure data communication across the internet. The objectives of the TLS protocol are threefold, providing data privacy, which is typically achieved using symmetric encryption algorithms, authentication of communicating parties using public key cryptography, and providing data integrity using message authentication codes. TLS comprises two protocols referred to as Handshake and Record, respectively. The Handshake protocol employs public-key cryptography to establish a shared secret key between the client and the server, it is also used to provide mutual authentication. The Record protocol uses the secret key established in the handshake protocol to protect communication between the client and the server. The TLS protocol is the standard approach for public key-based key exchange schemes [10, 17-20], hence its adoption in this study for comparison.

C. Overview of Quantum-Computer-based Attack Resilient Methods

1) Post-Quantum Public Key Algorithms

There are several different families of post-quantum algorithms which differ by the type of mathematical problem they are based on. These include hash-based, lattice-based, code-based, and supersingular elliptic curve isogeny [21], as explained below.

a) Code-Based

Code-based cryptography encompasses cryptosystems that are based on an error-correcting code. Wherein errors are used intentionally to obscure messages, in such a way that the errors can only be corrected if the recipient of such messages has access to a private key. The security of this scheme comes from the hard problem of decoding an erroneous codeword from a random-looking codeword without the use of the private key. This type of cryptography is well established with McEliece being introduced in [22].

b) Hash-Based

Hash-based cryptography encompasses cryptosystems based on the security of hash functions. The latter is a non-reversible function with an input of a string of any length, producing a fixed-length output[23]. These schemes use a tree data structure to combine a collection of One-Time-Signature schemes (OTS). The cryptosystem signs a message using an OTS, which should never be used twice to retain security. Implementations of such a scheme can be either stateful or stateless. The former must remember which OTSs have been used, hence requiring a state management procedure. The latter uses a very large tree, where OTSs are chosen at random. With a sufficiently large tree, the probability of a repeated OTS is low. The strongest candidate for a post-quantum hash-based scheme is SPHINCS+, which is stateless.

c) Lattice-Based

Lattice-based cryptography encompasses cryptosystems that are based on the conjectured intractability of lattice problems. There are two types of lattice problems used by the finalists. These are the NTRU problem [24] and the Learning-With-Errors problem (LWE)[25]. The security of LWE comes from the hard problem of solving an errored simple linear algebraic problem, not too dissimilar to code-based cryptography. LWE requires large public keys, therefore all remaining candidates that make use of LWE use variants that allow for reduced key size. NTRU-based encryption uses a mixing system based on polynomial algebra and reduction modulo to encrypt data. Decryption uses an un-mixing system with validity depending on elementary probability theory.

d) Super Singular Elliptic Curve Isogeny

Isogeny-based cryptography encompasses public key cryptosystems that use maps between elliptic curves. Although traditional public key cryptosystems using elliptic curves are vulnerable to attack by a quantum computer, isogeny-based schemes do not rely on the hidden subgroup problem. These were discovered in [26] and use the mathematics of elliptic curves with certain specific properties, called Supersingular elliptic curves. These are different from

the widely used elliptic curves from classical cryptography because they are non-commutative. However, recently a vulnerability was exposed in the more prominent proposed algorithm of this type, SIKE [27], and was subsequently dropped from the NIST standardization process.

2) NIST-Post Quantum Algorithms for Key Agreement

a) CRYSTALS-KYBER

Kyber is chosen for standardization for key encapsulation. It is a lattice-based algorithm that gets its security from the hardness of solving the learning-with-errors problem on module lattices (Module-LWE). This algorithm is fast, has efficient factorization and constant-time implementation, and is designed for low memory consumption so it can be implemented on embedded devices. To construct a key encapsulation mechanism (KEM), it first encrypts messages using IND-CPA (indistinguishability under chosen plaintext attack)secure public-key encryption [28], then it transforms this using a modified Fujisaki-Okamoton transform[29].

Saber was one of the three NIST finalists, it is also a structured lattice scheme and has a very similar performance to Kyber. NIST determined that there was no compelling reason to standardize multiple different structured lattice KEMs and chose KYBER instead of Saber. One factor that led to this decision was NIST's assessment that the MLWE problem, which accounts for most of the security of KYBER, is better studied than the MLWR problem on which the security of Saber is entirely based. Saber was included in this for comparison.

b) Classic McEliece

This cryptosystem is an evolution upon the first code-based cryptosystem introduced in 1978 by McEliece [22], with few differences. Firstly, it is a Key Encapsulation Mechanism (KEM)[30] that is IND-CCA2 secure against all Random Oracle Model (ROM) attacks, which means high security, most importantly against quantum computers. Secondly, the KEM is constructed from a PKE designed for OW-CPA security (One-Wayness against Chosen-Plaintext Attack). This means that attacks cannot efficiently find the codeword from a ciphertext and public key when the codeword is chosen randomly. Whilst McEliece's system is OW-CPA secure, the base PKE used in the construction of the KEM isn't the original McEliece PKE, but rather a dual variation created by Niederreiter[31]. The Niederreiter PKE is on par with the original scheme regarding security, both providing quantum resistance, however, encryption in the Niederreiter PKE is roughly ten times faster than the McEliece PKE, hence why it is the basis for the Classic McEliece cryptosystem.

Classic McEliece, along with its predecessors, has the advantage of high performance and security strength [32]. It is the most researched candidate, so it has the highest level of assurance. There are no known classical or quantum attacks on a McEliece cryptosystem with a sub-exponential running time [33]. It also has the smallest ciphertexts compared to other KEM candidates. However, a drawback regarding the potential adoption of these cryptosystems lies in the use of the Goppa codes to determine the public key, which results in a large public key matrix[34]. Consequently, key generation is the slowest and most resource-intensive aspect of the system, which makes it less feasible for embedded devices, therefore it was not included in this study, although it is still part of the NIST selection process.

c) Alternate Candidates

There are two additional alternate candidates submitted for evaluation as part of the fourth round in the PQC standardization process, namely, BIKE (Bit Flipping Key Encapsulation) and HQC (Hamming Quasi-Cyclic) are two code-based cryptosystems. The submitters of BIKE were not confident enough at the end of PQC round two to claim it provided CCA(Chosen Ciphertext Attack) security. There are also questions from NIST regarding BIKE's side-channel protections. Whilst HQC provides strong security assurances, it lacks in performance and has an unfavorable width when compared to BIKE. HQC's key generation and decapsulation are comparatively much faster than BIKE's. These two algorithms are still being considered so they have not been included in this study, however, previous studies have indicated their implementation costs are expected to be higher than schemes based on the lattice problem, hence less relevant to embedded and IoT systems. A comparison of the parameters of the different NIST algorithms is shown in Table 1 below.

TABLE I
PUBLIC KEY, SECRET KEY, CIPHERTEXT, AND SHARED SECRET SIZES IN BYTES OF NIST KEY ENCAPSULATION MECHANISM SCHEMES

Algorithm	**Public Key**	**Secret Key**	**Ciphertext**	**Shared Secret**
KYBER 1	800	1632	768	32
KYBER 3	1184	2400	1568	32
KYBER 5	1568	3168	1568	32
Saber 1	672	1568	736	32
Saber 2	996	2304	1088	32
Saber 3	1312	3040	1472	32
Classic McEliece 1	261120	6492	96	32
Classic McEliece 3	1044992	13932	208	32

3) NIST-Post Quantum Algorithms for Digital Signatures

a) CRYSTALS-Dilithium

Dilithium is a compact lattice-based quantum-resistant signature algorithm[35]. The security of this algorithm is given by the hardness of finding the shortest vector in lattices. Within its implementation, Dilithium uses the SHAKE128 and SHAKE256 hash functions and Number-Theoretic Transform (NTT) for polynomial multiplication. Instead of using a discreet Gaussian distribution which was shown to be insecure against side-channel attacks[36], Dilithium uses uniform sampling to generate random numbers. This was selected by NIST for standardization in round 3 and was recommended to be the first algorithm to be standardized based on its security properties and compact implementation.

b) FALCON

FALCON is the second lattice-based signature chosen for standardization. This signature uses the 'fast Fourier sampling' trapdoor over NTRU lattices. The signatures of this scheme are shorter than Dilithium's while the public keys are of similar size. Falcon is also scalable and fast because of the use of Fourier sampling. In addition, it can be used on embedded devices with memory constraints, due to the key generation algorithm used. This scheme gets its security from the hardness of solving the short integer problem (SIS) over NTRU lattices. The issue with FALCON is that it requires a 53-bit floating point unit[1] which may not be supported by all IoT devices. Alternatively, this can be achieved with the software but at a significant drop in performance. This also makes the key generation slower than the other algorithms[2] as most constrained devices are 32-bit or smaller and will need multiple clock cycles per instruction when calculating the key. It was decided to not include Falcon in this evaluation because of all the above-mentioned downsides.

c) SPHINCS+

SPHINCS+ is a stateless hash-based signature framework that was chosen for standardization. In comparison to its predecessor SPHINCS, this method is faster and has a smaller signature as well as using a signature framework instead of a signature scheme. SPHINCS+ was selected for benchmarking in this work because of its flexibility, which is particularly advantageous on embedded devices. In our tests, we have used a robust tweakable hash function with an estimated security level of 1, which means that the high-level construction of SHPINCS+ is done using Construction 6 presented in [37]. Table 2 compares the algorithm parameters for the selected NIST digital signature schemes.

[1] https://csrc.nist.gov/csrc/media/Presentations/2022/benchmarking-and-analysing-nist-pqc-lattice-based/images-media/session4-howe-benchmarking-analysing-pqc2022.pdf

TABLE II
PUBLIC KEY, SECRET KEY, AND SIGNATURE SIZES IN BYTES OF THE NIST DIGITAL SIGNATURE SCHEME FINALISTS FOR DIFFERENT NIST LEVELS OF SECURITY.

Digital signature	Public Key	Secret Key	Signature
Dilithium 2	1312	2528	2420
Dilithium 3	1952	4000	3293
Dilithium 5	2592	4864	4595
Falcon 1	897	1281	690
Falcon 5	1793	2305	1330
SPHINCS+ 1	32	64	17088
SPHINCS+ 3	48	96	35664
SPHINCS+ 5	64	128	49856

D. Related studies on post-quantum algorithms Implementation Costs.

1) NIST Security Level Definitions

Previous work in this area mainly focused on the comparison between public key algorithms that are part of the NIST selection process[20, 38-40]. Such comparison needed to be made between algorithms that provide the same level of security, therefore, NIST has provided a collection of broad security strength categories as outlined in Table 3. A given cryptosystem may be instantiated using different parameter sets to fit into different categories. Any attack that breaks the relevant security level must require computational resources comparable to or greater than those required for the specified type of search on a block cipher key or hash function of a certain size.

TABLE III
NIST SECURITY LEVEL DEFINITIONS

Security Level	Search Type	Search Performed On
1	Key Search	Block cipher with a 128-bit key
2	Collision Search	256-bit hash function
3	Key Search	Block cipher with a 192-bit key
4	Collision Search	384-bit hash function
5	Key Search	Block cipher with a 256-bit key

[2] https://csrc.nist.gov/csrc/media/Presentations/2022/falcon-update/images-media/session-1-prest-falcon-pqc2022.pdf

NIST has declared Level 1 to be the benchmark entry-level quantum resistance strength which new public key cryptography must be able to meet. Table 4 lists the NIST security levels for which each candidate has provided a parameter set.

TABLE IV
PARAMETER SETS PROVIDED BY PQC FINALISTS

PQC Candidate	NIST Security Level				
	1	2	3	4	5
Classic McEliece	✓	✓	✓	✓	✓
CRYSTALS-KYBER	✓	×	✓	×	✓
NTRU	✓	×	✓	×	✓
SABER	✓	×	✓	×	✓
CRYSTALS-DILITHIUM	×	✓	✓	×	✓
FALCON	✓	×	×	×	✓

2) Performance Comparison

There have been several studies in the literature aimed at evaluating the performance of proposed PQC algorithms. For example, the work in [40] used the toolkit SUPERCOP[3] to test the performance of key generation, signing, and verifying procedures of various digital signature schemes including two of the NIST finalists Dilithium and Falcon. A 16-core Intel Core i7-10700 clocked at 2.9GHz was used to run this toolkit. It was found that Falcon took longer to sign while Dilithium was slower at verifying.

The Open Quantum Safe project[41] has also profiled various properties of post-quantum algorithms such as performance and memory usage using three different architectures, x86 64, Apple M1, and aarch64 (ARM64). In terms of key encapsulation mechanisms (KEM) speed, Kyber was found to have done more operations per second for each of the categories of key generation, encapsulation, and decapsulation, than any other NIST algorithms considered. The same study also assessed the performance of digital signatures, where Dilithium was found to have faster keypair generation and signing speed than the rest. While SPHINCS+ trails Dilithium regardless, it outperforms Falcon when using its fast variant for keypair generation. The verifying speed between Dilithium and Falcon was comparable, with Falcon taking the edge over its counterpart at NIST level 5 security.

3) Energy Consumption Comparison

This is an important metric to evaluate when considering the application for which these cryptosystems may be used such as mobiles and other battery-operated devices. Energy consumption is also important in high-performance computing environments where energy consumption directly relates to maintenance and cooling. The work in [38] has done an estimation of the energy consumption of NIST algorithms for both signature schemes and Key encapsulation mechanisms when the cryptosystems were run on an Intel Core i7-6700 CPU. Their analysis of KEM showed that KYBER and SABER consume a fraction of energy compared to Classic MCEleiece and NATRU schemes. For digital signatures, FALCON consumed less energy compared to CRYSTALS-DILITHIUM.

Other researchers targeted more constrained devices[19, 42]. For example, the work in [19] implements NIST first-round reference algorithms in the open-source embedded TLS library mbedTLS4 and measures the performance of the post-quantum primitives on four different embedded platforms with three different ARM processors(including M0) and an Xtensa LX6 processor. This was subsequently compared to a classical TLS variant using elliptic curve cryptography (ECC). While exploratory research has been completed on post-quantum algorithms' implementation into TLS on resource-constrained embedded devices, there is a significant gap in research on more constrained platforms. A wealth of papers has been devoted to implementing post-quantum TLS on the Raspberry Pi 3b+ and 4b and their respective powerful Cortex-A53 and A72 processors; a smaller proportion devoted to ARM Cortex M4 across various development boards which NIST noted as a constrained platform target for algorithm developers, and even less research has been performed on heavily constrained embedded devices at the lowest end of the spectrum including processors such as the ARM Cortex M0+ or Xtensa LX6. While these devices are low in computational power and resources, they are widely used in embedded devices in the exponentially growing IoT space. This is evidenced by the success of the new Cortex M0+-based RP2040 processor with over 10 million fabricated in their first year of production (2021). The RP2040 has only 264KB of SRAM and supports up to 16MB of flash and yet is highly in demand. What is more, none of the previous studies performed a comparative analysis with quantum resilient cryptographic schemes that are based on symmetric ciphers. Investigating the performance of all quantum-resilient cryptographic methods and resourced-constrained devices is crucial to safeguarding networking-enabled embedded devices' security over the decades to come.

III. METHODOLOGY

The essence of the proposed testbed design is to emulate realistic use cases of the quantum attack resilient algorithms. The work in [43] has shown that public key-based TLS protocol used for establishing secure connections on the web

[3] https://bench.cr.yp.to/supercop.html
[4] https://www.trustedfirmware.org/projects/mbed-tls/

consumes more than 92% of the global energy incurred by all public-key cryptography applications. Therefore, the testbed was constructed to emulate a secure link setting and allow the measurements of performance and energy overheads for all key agreement and digital signature schemes being considered.

The overall testbed architecture shown in Figure 1 consists of a client-server configuration, connected through a networking environment, wherein the two devices need to perform mutual authentication and agree on a shared Key.

FIGURE 1. The testbed architecture

A. Hardware Architecture Development

The server is assumed to be an unconstrained device, therefore Raspberry Pi 3b+ was employed as it has high computational power and RAM size; it is also suitable for editing, compiling, and debugging software to run using its operating system Raspberry Pi OS[5].

Two types of clients are considered in this study. A high-end embedded device (e.g. Raspberry Pi 3b+), and a low-end IoT-constrained device. The latter should be representative of a modern resource-constrained embedded device and its corresponding limitations. Table 5 lists several examples of such devices.

The ARM Cortex M0+ is a constrained processor with basic arithmetic features including a 32-bit multiplier and is claimed to be the most energy-efficient ARM processor available for constrained embedded applications, hence it is a popular choice for the lower end of spectrum IoT devices.

Therefore, it makes more sense to use a device incorporating this type of processor in the proposed testbed as opposed to platforms that use more powerful processors such as ((Cortex-M4 & Cortex-M7).

The Raspberry Pi Pico W and Arduino Nano RP2040 Connect both use the MCU on a development board with networking capability and significant flash memory. The Pico W has support for external Serial Wire Debugging (SWD) using OpenOCD and GDB on a host device and has exposed pin headers for this, however, the Arduino device does not have accessible pin headers (using pads instead) and depends on Arduino Framework libraries making integration of new code an additional challenge.

Consequently, the Raspberry Pi Pico W was selected as the constrained device based on its representative constrained MCU and Processor, low cost, documentation, and ease of programming and debugging.

TABLE V
EXAMPLES OF RESOURCES-CONSTRAINED IOT DEVICES

Development Board	MCU (Processor)	Freq. (MHz)	SRAM (KB)	Flash (KB)
Arduino Nano 33 IoT	SAMD21 (ARM Cortex-M0)	48	32	256
Arduino Nano RP2040 Connect	RP2040 (2 ARM Cortex-M0+)	125	264	16000
Arduino Nano 33 BLE Sense	nRF52840 (ARM Cortex-M4)	64	256	1000
Raspberry Pi Pico W	RP2040 (2 ARM Cortex-M0+)	125	264	16000
ESP32-S2-DevKit	ESP32-S2 (Xtensa LX7)	240	512	384
STM32-F4 Series	STM32F4 (ARM Cortex-M4)	180	384	512-2056
STM32-F7 Series	STM32F7 (ARM Cortex-M7)	216	64-248	256-512

B. Software Implementation

list There are several open-source software solutions capable of connecting two devices using TLS and quantum-safe algorithms however the majority focus on supporting devices less constrained than the Pico W. This work initially considered using WolfSSL because it supports embedded devices and is an active member of the PQC research community, however, support for the Pico W was found to be unsatisfactory, as a result, an alternative software called Mbed TLS was adopted. The latter is an open-source C library that implements the TLS protocol and associated cryptographic

[5] https://www.raspberrypi.com/software/

primitives. It is part of the Trusted Firmware project[6] and is designed for embedded devices. While well established, it has a smaller user base than WolfSSL and support for PQC is experimental. A prototype implementation of TLS 1.3 alongside support for PQC via labors can be found on a branch of the project on GitHub[7]. While Mbed TLS can be added to the Pico-SDK[8] as well as run on a Raspberry Pi 3b+ the quantum-safe algorithms are not directly located in the library itself and therefore, an alternative approach was needed. The authors of [19] show an approach to integrate quantum-safe algorithms into Mbed TLS, this was used as a starting point. Next, the latest NIST round three versions of the Saber KEM and Dilithium DS were integrated. In addition, the proprietary version of Mbed TLS was installed on a Raspberry Pi 3b+ and added to the Pico-SDK-ready development of client and server applications.

The cipher suites supported by the Mbed TLS PQC branch from [19] are as follows (additional schemes implemented in this work are denoted by *): Sphincs+ (1), Dilithium (2, 3, 5)*, Kyber (1, 3, 5), Saber (1, 3, 5)*, ECDSA, and ECDHE. The curve used for both ECDHE and ECDSA is the secp256r1 curve as recommended by NIST

Another issue that motivated the use of Mbed TLS is the networking of the embedded device. The Pico W lacks an operating system, resulting lack of needed library support for aspects such as sockets. A solution to this is using a TCP/IP stack. The most likely choice for this would be lwIP (lightweight IP)[9], which is open-source and made for embedded systems. Given our choice of Mbed TLS, lwIP support was guaranteed by the fact that the Pico SDK contains an implementation of it, with wrapper libraries for Mbed TLS[10].

C. Benchmarking

In addition to the development and integration of the software needed to complete a TLS handshake on the chosen hardware, further software is needed to collect data on compute resource usage and power consumption. This section explains the metrics used for evaluation, the measurement methods, and the data collection process.

1) Metrics

To compare the overheads of each solution, three main metrics are going to be used as follows:

a) Latency refers to the time it takes to complete a specific operation. The latter is either a complete handshake protocol required to establish a secure connection, or one of its steps (e.g., verification of a digital signature, key encapsulation). This was measured using the standard timer functions available in C and the Pico-SDK respectively by repeating a set number of handshakes and finding the mean and standard deviation for each test.

a. Random Access Memory Usage refers to the RAM resources required by a specific operation. A significant portion of this is used for heap memory (a part of RAM where the programmer manages memory allocation) and stack memory (the remaining part of RAM where function variables are stored inside stack frames). The metric is an important indicator of computational resource requirements, hence its selection for evaluation.

On the Raspberry Pi device, the memory used was evaluated using an external profiling application called Massif from the Valgrind suite of applications. This takes a C program and its arguments as input and runs it while capturing information on certain metrics or about certain aspects of the program. In the case of Massif, it measures heap usage and has an option for stack usage. To enable the use of Massif and efficient data collection bash scripts were developed to automate the benchmarking process on the client and the server. As part of this, the client and server applications were modified to only run a single handshake with the KEM, and DS specified by program arguments.

b. Energy dissipation: This is the energy required by a device to complete a specific operation (e.g. a TLS handshake). This was evaluated by measuring the instantaneous power of each device using the R&S HMC8012 unit, a digital multi-meter device, and calculating the energy figures based on the time required for the execution of each of the tasks. To accurately measure the energy associated with each task, the base power consumption was also estimated, so that only the additional power related to the task in question is taken into consideration.

2) Data Collection Process

To enable the collection of a large amount of data, a benchmarking program was designed that runs TLS handshakes repeatedly. To run a single handshake requires a server to be waiting to accept a connection and a client to initiate the handshake. During the handshake data is collected where appropriate and then the connection is closed. The benchmarking programs repeat this process a given number of times and handle processing statistics and outputting the data to a file.

[6] https://www.trustedfirmware.org/projects/mbed-tls/
[7] https://github.com/hannestschofenig/mbedtls
[8] https://github.com/peterharperuk/pico-sdk/tree/add mbedtls
[9] https://www.nongnu.org/lwip/2 1 x/index.html
[10] https://raspberrypi.github.io/pico-sdk-doxygen/group pico lwip.html

D. Implementation

The implementation process included the construction of the testbeds, installing and configuring Mbed TLS, and integrating PQC algorithms, where necessary. The Dilithium signature scheme was integrated using the process set out in [44]. This was also followed to implement the Saber KEM whilst using the Kyber KEM implementation in [19]for reference. The source code was taken from the open-source GitHub repository[11]. Furthermore, the SPHINCS+ implementation from [19] was used as a reference.

This work has used self-signed certificates a. The X.509 standard, specified in [45] defines the format of public key certificates. For certificate files, the Privacy-Enhanced Mail (PEM) format, formalized in[46] is used. The implementation process also included the development of benchmarking applications and libraries. The rest of this section briefly explains the architecture of the testbeds and the experimental techniques used.

1) Raspberry Pi Client Testbed

This testbed uses two Raspberry Pi 3b+ devices linked directly with a Cat5e Ethernet Cable. One device is configured to act as a server, while the other acts as a client. Each Raspberry Pi has the 64-bit version of Raspberry Pi OS installed and for software development may connect to the internet using the device's built-in Wi-Fi. As the Pi has an operating system that includes features for networking, text editing, compiling software, and a terminal for executing programs, all benchmarking is internal to the Raspberry Pi for both server and client. For power measurements, additional setups were required to allow the use of the multi-meter, which can only be controlled by a device with a supported operating system (Windows or Ubuntu). Therefore, an x86 Linux machine running Ubuntu was used for data collection and control flow management throughout, the machine triggered the start and stop of a handshake cycle, and the collection of power measurement, using a Python script written for this purpose. A diagram of the power measurement experimental setup is shown in Figure 2. The device on the test is connected to the multi-meter via the front measurement inputs. We have spliced into a power supply for a pi 3b+ to do this.

The Linux machine connects to a USB port on the back of the multi-meter to control it. It sends SCPI commands to the multi-meter to start or stop the logging function. The multi-meter outputs its recordings to a USB drive using another USB port on the front.

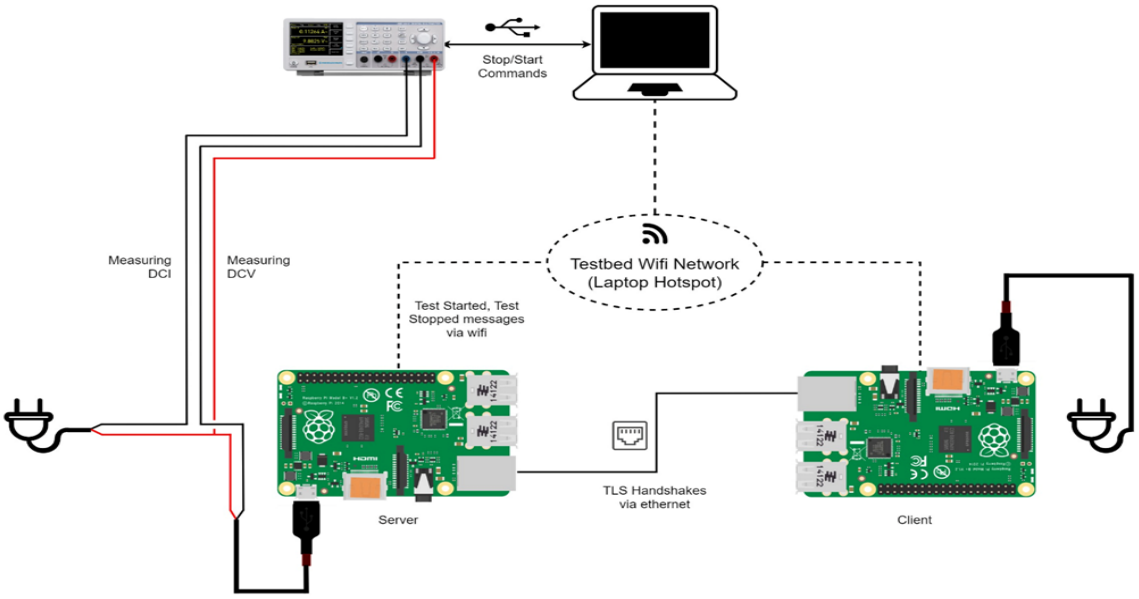

FIGURE 2. Power Measurements Testbed Experimental Setups for a Raspberry Pi 3b+ Client

[11] https://github.com/KULeuven-COSIC/SABER

2) Constrained IoT Device Testbed

The constrained hardware testbed has a more complex setup. The Raspberry Pi Pico W has no on-chip debugger and must use an external device and the Serial Wire Debug (SWD)12 port for hardware debugging. The external device also acts as a slave UART RX/TX signal between the host serial debugs terminal and Pico W's outputs. The Picoprobe setup uses an additional Raspberry Pi Pico device which has been flashed with binary programming to act as a hardware debugger. The picoprobe binary13 is written as a branch of OpenOCD14, the Open-source On-Chip Debugger that acts as an intermediary between embedded device and host debugging software. OpenOCD provides the control signals for the Picoprobe to interface with the Pico W's SWD Port for flashing the Pico W with a binary, setting breakpoints, communicating debug symbols, and interfacing with device memory. The Picoprobe's USB output is connected to a host device that is executing an instance of the GNU Debugger (GDB)15 initialized to interface with the Picoprobe device through USB. The host device may also interface with the Pico W through the Picoprobe's UART port using a serial terminal on the host device. This serial terminal is used to receive terminal and statistical output from the Pico W when benchmarking is taking place. The picoprobe wiring diagram is shown in Figure 3.

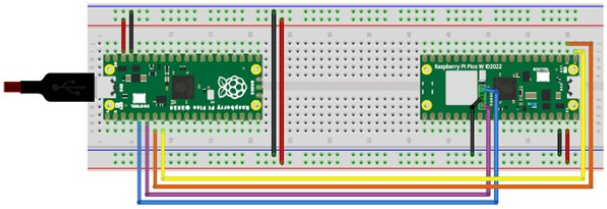

FIGURE 3. Pico-probe Wiring Diagram

For power measurement using the constrained testbed, a similar setup to what was depicted in Figure 2 was used, as shown in Figure 4.

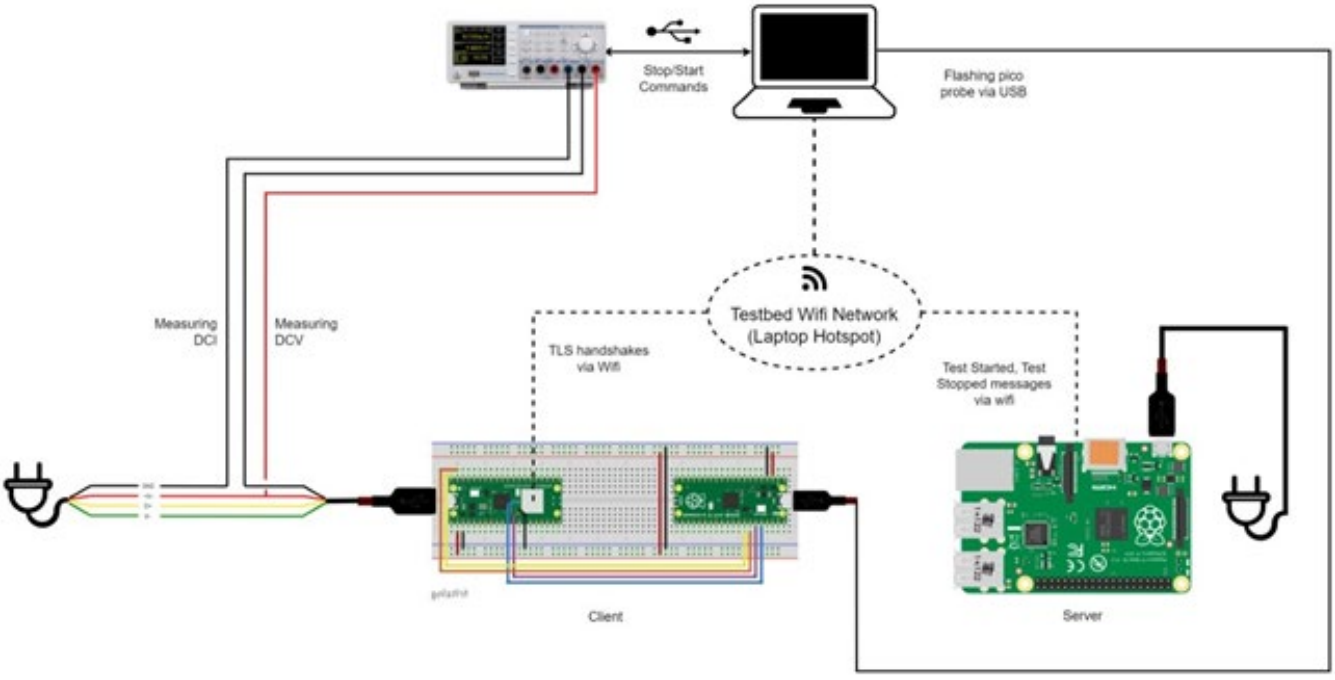

FIGURE 4. Power Measurements Testbed Experimental Setups for a Raspberry Pi Pico W Client

[12] https://developer.arm.com/documentation/101761/0100/Debug-and-trace-interface/Serial-Wire-

[13] https://github.com/raspberrypi/picoprobe

[14] https://openocd.org/

IV. EVALUATION

For This section explains the rationale of the experiments conducted in this study and outlines the results from the two testbeds discussed in section 4.

A. Design of Experiments

Two sets of experiments were carried out. The goal of the first group of tests is to compare the performance, energy, and memory usage of the newly developed public key algorithms with existing solutions based on Elliptic curves systems, as well as, with alternative schemes based on the use of quantum secure symmetric ciphers such as AES 256. These experiments were performed using the testbed with Raspberry Pi client, as all the studied algorithms have implementation support for this platform. The server and client were connected using an Ethernet cable to avoid WiFi-related latency fluctuations. The same configuration was used for the symmetric scheme.

The goal of the second set of tests is to investigate the feasibility of implementing the new public key algorithms on the resource-constrained device. This was performed using the constrained IoT device testbed as outlined above, wherein the devices were connected using the local WIFI network, as the Pico W has no Ethernet capabilities.

B. Results for the Raspberry Pi Client Testbed

To perform a comprehensive comparison between available solutions for quantum-resilient cryptographic algorithms, all possible combinations of newly developed key encapsulation mechanisms and digital signature schemes were implemented and evaluated. In each case, latency, memory usage, and energy dissipation were calculated as explained in section 4.

1) Latency

Figures 5 and 6 show the overall latency of a complete TLS handshake on the client and server, respectively. This was estimated by measuring the time required to complete the TLS-related operation on each node. The sum of latencies on the client and server sides is the total time needed for a complete handshake cycle.

The y-axis is grouped according to KEMs, showing Saber, Kyber, and ECDHE. Each bar represents a DS, including Dilithium, Sphincs+, and ECDSA. The number next to each algorithm name represents the security level of the implementation used according to Table 2. For readability, the scale in these two graphs is logarithmic, and units are in milliseconds.

The latency estimations include all communication and computational tasks. The latency figures use the mean value of all performed measurements. The standard deviation results for all schemes ranged between 8-10%.

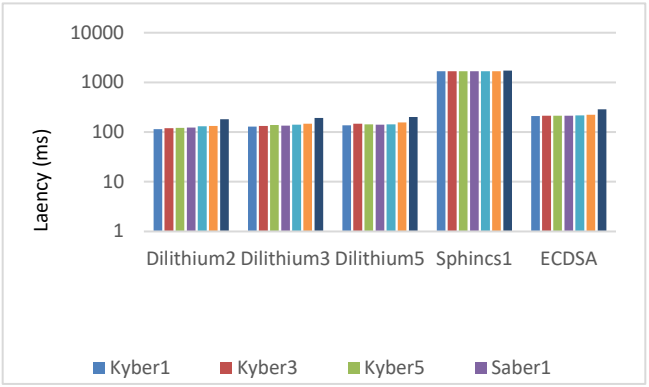

FIGURE 5. TLS Handshake Latency on the Raspberry Pi 3b+ (Client)

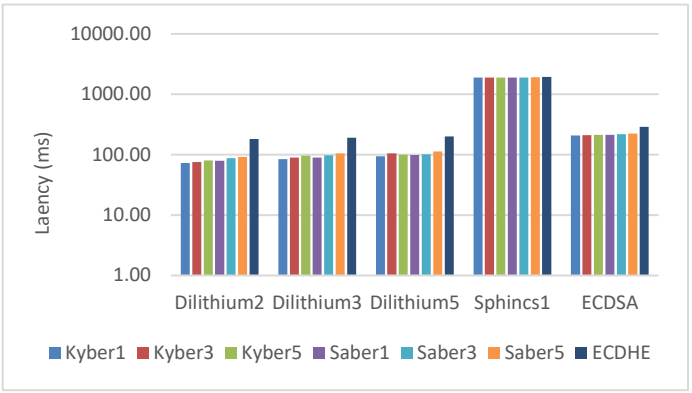

FIGURE 6. TLS Handshake Latency on the Raspberry Pi 3b+ (Server)

The first significant insight from the above figures is that most schemes constructed using newly developed post-quantum public key algorithms cause significantly less latency compared to conventional systems based on the use of elliptic key cryptography. This is partly because key agreement using ECDHE uses Diffie-Hellman key exchange, while the other algorithm uses a key encapsulation mechanism instead, which has less delay. The only exception is the schemes that use Sphinx algorithms for digital signature due to the complexity of its computation. The results also show that the Kyber1-Dilithium2 scheme yields the best performance on both the client and server sides.

Table 6 below gives more insights into the performance overheads associated with key encapsulation algorithms being considered, it includes the latency associated with the TLS computation tasks of these schemes, at the client and server sides. These figures show that Kyber 1 has the best performance among the newly proposed public key algorithms, while Saber5 incurs the largest delay. In the same Table, we can also see that the increase in security levels for the PQ KEM leads to more latency, which is consistent with the fact that computational overheads are typically higher for schemes with a higher level of security. When comparing

Kyber and Saber together it is shown that Kyber takes on average a shorter amount of time to run its algorithm
Note, that the ECDHE has not been included in Table 5 because it doesn't apply here.

TABLE VI
COMPARATIVE ANALYSIS OF LATENCY OF KEY ENCAPSULATION SCHEMES IN TLS

Algorithm	Latency (milliseconds)		
	Key Encapsulation (Client)	Key Decapsulation (Server)	Key Generation (Server)
Kyber1	2.68	2.4	1.57
Kyber3	4.52	3.93	2.78
Kyber5	6.88	5.44	4.15
Saber1	6.62	6.12	3.34
Saber3	11.99	10.45	6.52
Saber5	14.99	15.97	10.98

Table 7 compares the performance overhead associated with the digital signature schemes being considered. it includes the latency associated with the TLS signature verification step on the client and server sides. The results demonstrate that Sphinics1-based verification consumes significantly more time than all other schemes combined, this is because of computational complexity.

TABLE VII
COMPARATIVE ANALYSIS OF LATENCY OF SIGNATURE VERIFICATION IN TLS

Algorithm	Latency (milliseconds)	
	Signatures Verification (Client)	Signatures Verification (Server)
Dilithium2	4.78	14.46
Dilithium3	8.37	24.35
Dilithium5	9.63	28.49
Sphincs1	241.21	1749.3
ECDSA	81.1	42.63

The last consideration for latency is between the computation time and the overall handshake latency. This allows for a comparison between the sources of latency for specific algorithms. The communication overhead is determined by the difference between the values for the sum of all computational tasks and overall handshake latency.

For all cipher suites except the ones containing a conventional ECC algorithm, the sum of computation latency is a fraction of the total latency. This observation indicates that overall latency post-quantum KEMs in a TLS handshake is mostly determined by bandwidth requirements as opposed to computation.

2) Memory Usage

Figures 7 and 8 show the memory heap usages for the TLS-based handshake on the client and server, respectively. The y-axis is grouped according to KEMs, showing Saber, Kyber, and ECDHE. Each bar represents a DS, including Dilithium, Sphincs+, and ECDSA

A clear pattern for the heap usage can be deduced from these results, where the schemes that employ Dilithium level 1 have the lowest usage, and those using SPHINCS+ have the highest usage, with the only exception being ECDHE where the lowest value is for ECDSA. ECDHE in combination with Dilithium uses about twice as much heap compared with all the other KEM, using between 23K and 28K, while the highest recorded values between the other algorithm are 11K and 16K as can be seen in Figure 7. In the same graph, we can also see that the increase in security levels for the PQ KEM is directly related to a slight rise in heap usage. The results from the server side in Figure 8 have similar trends.

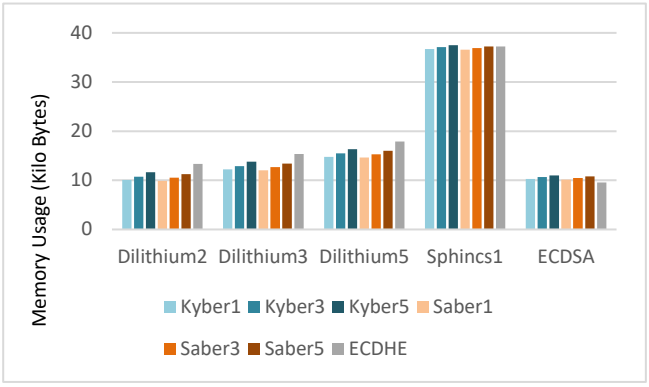

FIGURE 7. Memory Heap Usage for a TLS Handshake (Client)

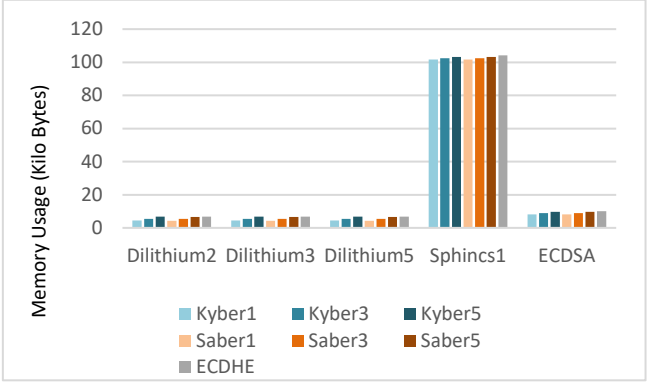

FIGURE 8. Memory Heap Usage for a TLS Handshake (Server)

Figures 9 and 10 show the memory stack usages for the TLS-based handshake on the client and server, respectively. The amount of stack used is closely related to the digital signature scheme, as there is little to no change between the different KEMs. The only exception to this rule is the classic ECDSA, where the stack increases with the security level of the encapsulation methods. Overall, there is a clear trend where ECDSA has the lowest stack usage followed by SPHINCS+, Dilithium levels 1, 3, and 5. This pattern can also be noticed in the Open Quantum Safe PQC profiling results in [16]

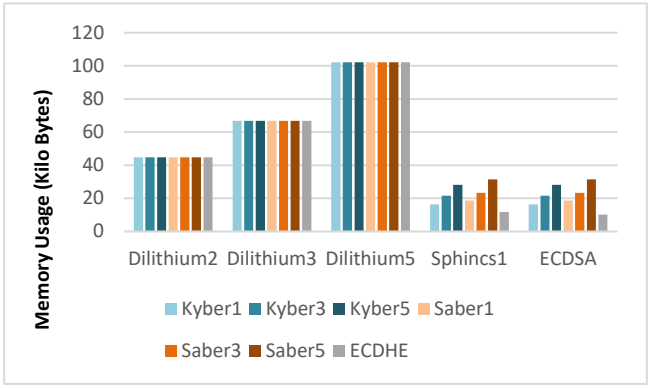

FIGURE 9. Memory Stack Usage for a TLS Handshake (Client)

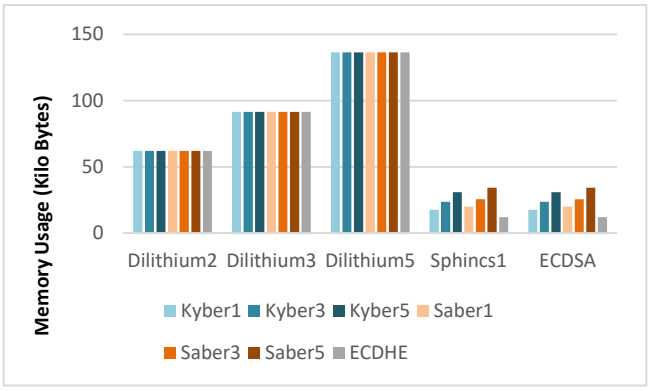

FIGURE 10. Memory Stack Usage for a TLS Handshake (Server)

3) Energy Dissipation

This was estimated by measuring the power consumption for consecutive TLS handshakes and computing the average energy over the measurement time. The results shown in Figures 11 and 12 confirm the previous trend, with kyber1-Dilithium 2 being the most energy efficient, while the use of the Sphinics1 digital signatures scheme consistently leads to more energy dissipation for all KEMs being considered. There are slight variations between the client and the server sides, but the overall trend is the same. An algorithmic scale is also used in these two figures for better readability.

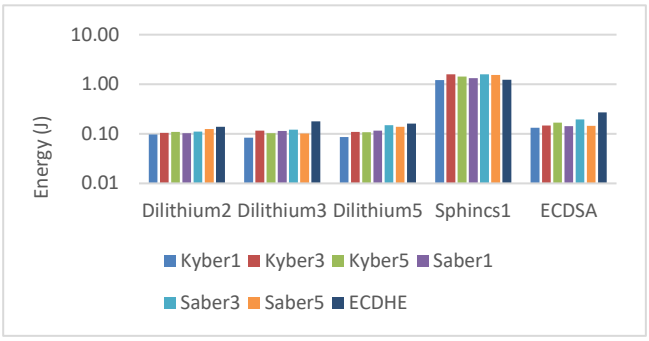

FIGURE 11. Energy Dissipation for a TLS Handshake (Client)

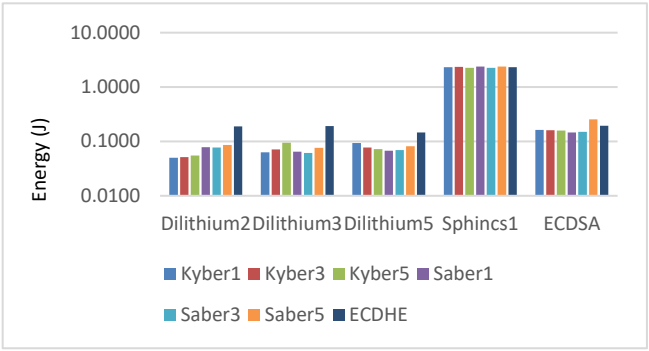

FIGURE 12. Energy Dissipation for a TLS Handshake (Server)

C. Results for the Constrained IoT Client Testbed

To evaluate the feasibility of implementing newly developed public key algorithms on constrained IoT devices. All possible combinations of key encapsulation mechanisms and digital signature schemes were implemented on this testbed. The focus, in this case, is to assess the latency and energy dissipation on the client side, as this is likely to be battery-operated with limited computational resources. The device used as a server was the same as the first testbed, so its corresponding results were not included as they did not provide any additional insights into what was already discussed previously.

1) Latency

Figure 13 shows the overall latency of a complete TLS handshake. These include all communication and computational tasks. The latency figures use the mean value of all performed measurements. The standard deviations were larger (up to 30%) in this experiment due to WiFi fluctuations.

[16] https://openquantumsafe.org/benchmarking/visualization/memsig.html

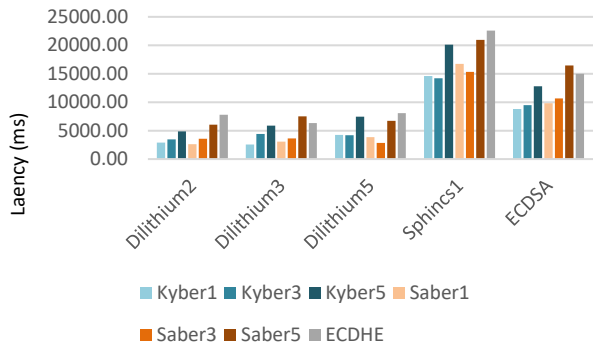

FIGURE 13. TLS Handshake Latency on the Raspberry Pico W (Client)

The above results show that the TLS handshake takes significantly longer to complete compared to the time required by an un-constrained client.

The trend observed previously still holds here with schemes that use Sphinics1 as a digital signature algorithm having the largest latency, and the kyber1-Dilithuim 2 scheme being the most performant.

Table 8 below gives more insight into the performance overhead associated with key encapsulation algorithms being considered. These figures show that Kyber 1 has the best performance among the newly proposed public key algorithms, while Saber5 incurs the largest delay. These findings are consistent with the results from the Raspberry Pi 3b client testbed.

TABLE VIII
COMPARATIVE ANALYSIS OF LATENCY OF KEY ENCAPSULATION SCHEMES IN TLS FOR THE RASPBERRY PI PICO W CLIENT

Algorithm	**Latency (milliseconds) of Key Encapsulation at the Client**
Kyber1	19.43
Kyber3	33.19
Kyber5	51.54
Saber1	23.23
Saber3	42.14
Saber5	66.23

Table 11 compares the performance overhead associated with the TLS signature verification step on the client side. Again, on the Pi 3b+, it takes Sphincs1 longer to run. The large magnitude of the latencies for these DSs takes up a large proportion of the overall latency time, hence being the more important of the two algorithms when timings are considered. Dilithium consistently provides a faster signature verification, likely due to its smaller signature size. This is reflected both on the Pico W and the Pi 3b+.

TABLE IX
COMPARATIVE ANALYSIS OF LATENCY OF SIGNATURE VERIFICATION IN TLS KEY ENCAPSULATION SCHEMES IN TLS FOR THE RASPBERRY PI PICO W CLIENT

Algorithm	**Signatures Verification (Client)**
Dilithium2	42.4
Dilithium3	68.9
Dilithium5	113.7
Sphincs1	2908.8
ECDSA	2454.2

2) Energy Dissipation

This was estimated by measuring the power consumption for consecutive TLS handshakes and computing the average energy over the measurement time. The results shown in figures 11 and 12 confirm the previous trend, with kyber1-Dilithium 2 being the most energy efficient, while the use of Sphinics1 digital signatures schemes consistently leads to more energy dissipation for all KEMs being considered.

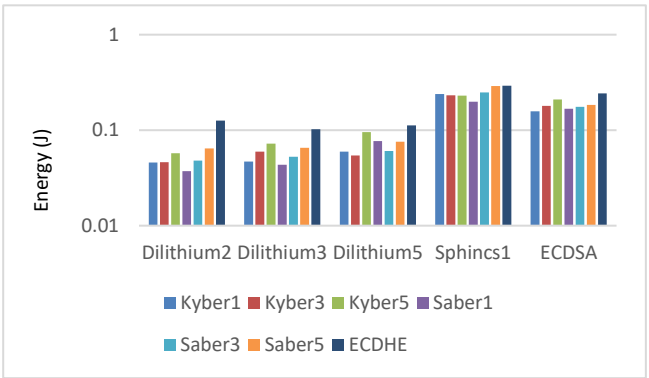

FIGURE 14. Energy Dissipation for a TLS Handshake of a Raspberry Pi Pico W Client

V. CONCLUSIONS

The fast-paced development of quantum computing technology has driven the need for new public key cryptographic algorithms that are resilient to quantum-computer-based attacks. This is essential to maintain the security of key establishment and digital signature schemes, which are core elements of all digital infrastructure. The

National Institute of Standards and Technology (NIST) is leading the process of developing new post-quantum public key algorithm standards. This work has developed two testbeds to investigate the energy and computational costs of all proposed quantum computer attack-resilient schemes. The first testbed aimed to perform a comparative analysis between available solutions, including the classical elliptic curve-based scheme. To achieve this, a server-client configuration was used to measure and calculate the latency, memory usage, and energy dissipation for a key agreement cycle. The results have shown that a TLS handshake based on post-quantum public key algorithms can be faster and consume less memory and computation resources, compared to an existing TLS scheme that uses elliptic key cryptography (ECDHE-ECDSA). This is because conventional schemes adopt a Diffie-Helman key exchange that incurs more latency than the key encapsulation mechanism adopted by the post-quantum algorithms.

Kyber level 1 and Dilithium level 2 were found to be the most efficient key encapsulation mechanism and digital signature scheme respectively.

The goal of the second testbed was to investigate the feasibility of implementing post-quantum public key algorithms on a resource-constrained device. A Raspberry Pi Pico W was chosen as its hardware architecture is typical of such devices. The work has demonstrated these new schemes can run on highly resource-constrained embedded networked devices. Extensive measurements and analysis on the constrained client for many different benchmarks as well as relative performances between the Raspberry Pi Pico W and the Raspberry Pi 3b+ were also presented. The results have shown that the constrained device has significantly larger latency, however, their relative performance was very similar to the Raspberry Pi 3b+ client. Overall, The results of the evaluation have shown that Kyber1-Dilithuim-2 is the most resource-efficient solution, it outperforms all other PQC algorithms, including the current scheme that uses elliptic curve cryptography. Our study has also shown the digital signature scheme Sphinx+ is associated with significant latency and energy costs so may not be suitable for IoT-type devices.

Future work will consider research challenges associated with the adoption of these new algorithms and its integration with existing Internet standards and other applications of public key cryptography such as cryptocurrencies, digital certificates, and virtual private networks.

REFERENCES

[1] U. S. D. o. Commerce, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography: NIST SP 800-56B Rev. 2," National Institute of Standards and Technology, Gaithersburg, 2019.

[2] U. S. D. o. Commerce, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography," National Institute of Standards and Technology, Gaithersburg, 2018.

[3] M. Mumtaz and L. Ping, "Forty years of attacks on the RSA cryptosystem: A brief survey," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 22, no. 1, pp. 9-29, 2019.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review,* vol. 41, no. 2, pp. 303-332, 1999.

[5] V. Hassija *et al.*, "Present landscape of quantum computing," *IET Quantum Communication,* vol. 1, no. 2, pp. 42-48, 2020, doi: https://doi.org/10.1049/iet-qtc.2020.0027.

[6] D. Stebila and M. Mosca, "Post-quantum key exchange for the Internet and the open quantum-safe project," in *Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, 2017: Springer, pp. 14-37.

[7] A. O. Ménard, I.; Patel, M.; Volz, D., "A game plan for quantum computing. ," McKinsey Q. 2020, 7–9. . [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/a-game-plan-for-quantum-computing (accessed on 8 August 2023).

[8] "NIST Released NISTIR 8105, Report on Post-Quantum Cryptography," National Institute of Standards and Technology United States, 2016. [Online]. Available: https://csrc.nist.gov/News/2016/NIST-Released-NISTIR-8105,-Report-on-Post-Quantum

[9] L. S. Vailshery, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030," Transforma Insights, 2020. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[10] J. Barton, N. Pitropakis, W. Buchanan, S. Sayeed, and W. Abramson, "Post-quantum cryptography analysis of TLS tunneling on a constrained device," 2022.

[11] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical CCA2-secure and masked ring-LWE implementation," *Cryptology ePrint Archive,* 2016.

[12] S. Ebrahimi, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT," *IEEE Internet of Things Journal,* vol. 6, no. 3, pp. 5500-5507, 2019.

[13] J. Hu, M. Baldi, P. Santini, N. Zeng, S. Ling, and H. Wang, "Lightweight key encapsulation using LDPC codes on FPGAs," *IEEE Transactions on Computers,* vol. 69, no. 3, pp. 327-341, 2019.

[14] Y. Kim, J. Song, and S. C. Seo, "Accelerating Falcon on ARMv8," *IEEE Access,* vol. 10, pp. 44446-44460, 2022.

[15] E. Rescorla, "Diffie-hellman key agreement method," 2070-1721, 1999.

[16] T. Saito, K. Xagawa, and T. Yamakawa, "Tightly-secure key-encapsulation mechanism in the quantum random oracle model," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part III 37*, 2018: Springer, pp. 520-551.

[17] M. Schöffel, F. Lauer, C. C. Rheinländer, and N. Wehn, "On the energy costs of post-quantum KEMs in TLS-based low-power secure IoT," in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 158-168.

[18] S. Sarıbaş and S. Tonyalı, "Performance Evaluation of TLS 1.3 Handshake on Resource-Constrained Devices Using NIST's Third Round Post-Quantum Key Encapsulation Mechanisms and Digital Signatures," in *2022 7th International Conference on Computer Science and Engineering (UBMK)*, 2022: IEEE, pp. 294-299.

[19] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider, "Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 841-852.

[20] P. Schwabe, D. Stebila, and T. Wiggers, "Post-quantum TLS without handshake signatures," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1461-1480.

[21] F. Borges, P. R. Reis, and D. Pereira, "A Comparison of Security and its Performance for Key Agreements in Post-Quantum Cryptography," *IEEE Access,* vol. 8, pp. 142413-142422, 2020, doi: 10.1109/ACCESS.2020.3013250.
[22] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv,* vol. 4244, pp. 114-116, 1978.
[23] L. Lamport, "Constructing digital signatures from a one way function," 1979.
[24] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory: Third International Symposium, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*: Springer, 2006, pp. 267-288.
[25] J.-P. D'Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, "Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM," in *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10*, 2018: Springer, pp. 282-305.
[26] H. Hasse, "Zur Theorie der abstrakten elliptischen Funktionenkörper III. Die Struktur des Meromorphismenrings. Die Riemannsche Vermutung," 1936.
[27] W. Castryck and T. Decru, "An efficient key recovery attack on SIDH (preliminary version)," *Cryptology ePrint Archive,* 2022.
[28] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography,* vol. 75, no. 3, pp. 565-599, 2015.
[29] R. Avanzi *et al.*, "CRYSTALS-Kyber: Algorithm specifications and supporting documentation (2020)," *URL: https://web. archive. org/web/20211007045636/https://pqcrystals. org/kyber/data/kyber-specification-round3. pdf. Citations in this document,* vol. 1, no. 1.1, p. 1.1.
[30] X. S. W. G. F2F, "Key Encapsulation: A New Scheme for Public-Key Encryption," 2009. [Online]. Available: http://lists.w3.org/Archives/Public/public-xmlsec/2009May/att-0032/Key_Encapsulation.pdf
[31] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Prob. Contr. Inform. Theory,* vol. 15, no. 2, pp. 157-166, 1986.
[32] A. Vambol, V. Kharchenko, O. Potii, and N. Bardis, "Post-Quantum Network Security: McEliece and Niederreiter Cryptosystems Analysis and Education Issues," *WSEAS Transactions on Systems and Control,* vol. 15, pp. 627-634, 2020.
[33] D. Engelbert, R. Overbeck, and A. Schmidt, "A summary of McEliece-type cryptosystems and their security," *Journal of Mathematical Cryptology,* vol. 1, no. 2, pp. 151-199, 2007.
[34] A. Kuznetsov, M. Lutsenko, M. Bagmut, and V. Zhora, "Performance Evaluation of the Classic McEliece Key Encapsulation Algorithm," in *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2021, vol. 2: IEEE, pp. 755-760.
[35] V. Lyubashevsky *et al.*, "Crystals-dilithium," *Algorithm Specifications and Supporting Documentation,* 2020.
[36] T. Espitau, P.-A. Fouque, B. Gérard, and M. Tibouchi, "Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1857-1874.
[37] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS+ signature framework," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2129-2146.
[38] C. A. Roma, C.-E. A. Tai, and M. A. Hasan, "Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms," *IEEE Access,* vol. 9, pp. 71295-71317, 2021.
[39] K. Hines *et al.*, "Post-Quantum Cipher Power Analysis in Lightweight Devices," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 282-284.
[40] R. Kuang, M. Perepechaenko, R. Toth, and M. Barbeau, "Benchmark Performance of a New Quantum-Safe Multivariate Polynomial Digital Signature Algorithm," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022: IEEE, pp. 454-464.
[41] D. S. a. M. Mosca., " Post-Quantum Key Exchange for the Internet and
the Open Quantum Safe Project," 2017.
[42] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4," 2019.
[43] B. Halak, Y. Yilmaz, and D. Shiu, "Comparative analysis of energy costs of asymmetric vs symmetric encryption-based security applications," *IEEE Access,* vol. 10, pp. 76707-76719, 2022.
[44] A. B. Ventosa, "Secure IoT for the Future," Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria …, 2020.
[45] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2070-1721, 2008.
[46] S. Josefsson and S. Leonard, "Textual encodings of PKIX, PKCS, and CMS structures," 2070-1721, 2015.

BASEL HALAK is an Associate Professor of secure electronics and the Director of the Cyber Security Academy at the University of Southampton. He is a Visiting Scholar with the Technical University of Kaiserslautern, an Industrial Fellow of the Royal Academy of Engineering, a Senior Fellow of the Higher Education Academy, and a National Teaching Fellow of Advance HE U.K. He has published more than 100 refereed conference and journal papers and authored five books on the security and reliability of electronic devices and systems. His research interests include hardware security, digital design, and embedded systems. He is with several technical program committees, such as HOST, IEEE DATE, DAC, IVSW, ICCCA, ICCCS, MTV, and EWME. He is a member of the Hardware Security Working Group of the World Wide Web Consortium (W3C). He is an Associate Editor of IEEE ACCESS and the Editor of the IET Circuits, Devices, and Systems.

THOMAS GIBSON received a B.Sc. degree in electronics and computer engineering from the University of Southampton, U.K., in 2021, where he is currently pursuing a postgraduate degree in computer engineering. His research interests include embedded system security and cryptography.

MILLICENT HENLEY received a B.Sc. degree in electronics and computers from the University of Southampton, U.K., in 2021, where she is currently pursuing a postgraduate degree in computer engineering. her research interests include embedded systems and security.

CRISTIN-BIANCA BOTEA received a B.Sc. degree in electronics and computers from the University of Southampton, U.K., in 2021, where she is currently pursuing a postgraduate degree in computer engineering. Her research interests include embedded system security and cryptography.

BENJAMIN HEATH received a B.Sc. degree in electronics and computers from the University of Southampton, U.K., in 2021, where he is currently pursuing a postgraduate degree in computer engineering. His research interests include embedded systems and Internet of Things technologies.

Sayedur Khan received a B.Sc. degree in electronics and computers from the University of Southampton, U.K., in 2021, where he is currently pursuing a postgraduate degree in computer engineering. His research interests include embedded systems and Internet of Things technologies.