Energy Consumption Evaluation of Post-Quantum TLS 1.3 for Resource-Constrained Embedded Devices*

George Tasopoulos
Industrial Systems Institute, R.C. ATHENA
Patras, Greece
g.tasop@isi.gr

Charis Dimopoulos
Industrial Systems Institute, R.C. ATHENA &
Electrical and Computer Engineering Dpt, University of Patras
Patras, Greece
c.dimopoulos@upnet.gr

Apostolos P. Fournaris
Industrial Systems Institute, R.C. ATHENA
Patras, Greece
fournaris@isi.gr

Raymond K. Zhao
CSIRO's Data61
Sydney, Australia
raymond.zhao@data61.csiro.au

Amin Sakzad
Faculty of Information Technology, Monash University
Melbourne, Australia
amin.sakzad@monash.edu

Ron Steinfeld
Faculty of Information Technology, Monash University
Melbourne, Australia
ron.steinfeld@monash.edu

ABSTRACT

Post-Quantum cryptography (PQC), in the past few years, constitutes the main driving force of the quantum resistance transition for security primitives, protocols and tools. TLS is one of the widely used security protocols that needs to be made quantum safe. However, PQC algorithms integration into TLS introduce various implementation overheads compared to traditional TLS that in battery powered embedded devices with constrained resources, cannot be overlooked. While there exist several works, evaluating the PQ TLS execution time overhead in embedded systems there are only a few that explore the PQ TLS energy consumption cost. In this paper, a thorough power/energy consumption evaluation and analysis of PQ TLS 1.3 on embedded systems has been made. A WolfSSL PQ TLS 1.3 custom implementation is used that integrates all the NIST PQC algorithms selected for standardisation as well as 2 out of 3 of those evaluated in NIST Round 4. Also 1 out of 2 of the BSI recommendations have been included. The PQ TLS 1.3 with the various PQC algorithms is deployed in a STM Nucleo evaluation board under a mutual and a unilateral client-server authentication scenario. The power and energy consumption collected results are analyzed in detail. The performed comparisons and overall analysis provide very interesting results indicating that the choice of the PQC algorithms in TLS 1.3 to be deployed on an embedded system may be very different depending on the device use as an authenticated or not authenticated, client or server. Also, the results indicate that in some cases, PQ TLS 1.3 implementations can be equally or more energy consumption efficient compared to traditional TLS 1.3.

CCS CONCEPTS

• **Hardware** → **Power and energy**; • **Security and privacy** → *Embedded systems security*; • **Networks** → Network reliability.

KEYWORDS

post-quantum cryptography, energy consumption, resource-constrained systems, secure communication protocol

1 INTRODUCTION

The development of Post-Quantum Cryptography (PQC) has inevitably gained significant traction after the discovery that the integer factorization and discrete logarithm problems can be solved in polynomial time [20] by powerful and scalable Cryptographically Relevant Quantum Computers (CRQC) [10]. While the vast majority of the existing cryptographic schemes for public key cryptography are dependent on those problems for their security, PQC schemes rely on other arithmetic problems that are not vulnerable to these new quantum attacks.

In an effort to adapt to this new reality, the US National Institute of Standards and Technology (NIST) has currently completed a standardisation process for PQC algorithms that can mathematically resist quantum attacks, ready to be deployed in real-world applications for digital signature schemes or key exchange mechanisms [8]. For the time being, the 4 algorithms that have been selected for standardisation are 3 Digital Signature Schemes and 1 Key Encapsulation Mechanism (KEM), with some KEM candidates being reevaluated in Round 4 of this standardisation process. An additional Call for Proposals was issued, in order to further diversify the selected Digital Signature algorithms since the finalist schemes are mostly based on lattices.

An essential aspect of the PQC adoption process in real applications is not only the requirement that the PQC schemes should be robust, secure and efficient but also how compatible, efficient and flexible they prove to be when adapted into already existing security protocols and applications. One of the most widely used protocols in this category is the Transport Layer Security (TLS) protocol, responsible for the secure communication of the vast majority of today's Internet. It is, thus, of paramount importance that PQC algorithms are correctly integrated with TLS. This integration and its inevitable performance overhead that is being introduced, while trivial for powerful computing systems, is proving to be a

*Accepted for publication at the Malicious Software and Hardware in Internet of Things Workshop, 20th ACM International Conference on Computing Frontiers '23, May 9th-11th 2023, Bologna, Italy.

challenging issue when dealing with resource-constrained environments [23]. These devices must be holistically evaluated for a wide range of PQC performance metrics, like running time, memory, bandwidth, and network requirements.

However, the power/energy consumption of the device in use is one of the key metrics in the IoT domain. Especially, IoT end devices that operate on battery are significantly affected by the energy consumption of pivotal operations since they can be the main source of such consumption and battery usage. The PQC operations' increased computation overhead compared to traditional schemes as well as the significantly higher byte number of keys, digital signatures and certificates (to be exchanged/communicated between IoT server-clients) can leave a non-trivial footprint on an IoT device energy consumption and eventually drain such a device's battery. Thus the ability of a TLS-integrated PQC scheme to offer a small power/energy consumption footprint while being used within the TLS protocol is highly important in the overall design of an IoT device. This highlights the need for a thorough power/energy consumption profiling/evaluation of TLS 1.3 when various PQC schemes are used.

1.1 Related work

The scope of the current state of research for evaluating the performance of Post Quantum Cryptography (PQC) schemes for KEMs and digital signatures is still in its early stages, especially in regards to resource-constrained environments. A performance evaluation of PQ TLS 1.3 was realized in [23], focusing on the integration of the pqm4 and PQClean libraries on resource-constrained devices. In an effort to benchmark TLS for PQC, a framework was introduced in [12], where the network conditions can be customized and their impact on Post-Quantum TLS implementations can be evaluated, while Paul et al. [13] offer a migration strategy for authentication in a PQ era, while concurrently evaluating the performance of TLS 1.3 in various devices. A Raspberry Pi was used in [2] to perform an evaluation of PQ TLS with OQS [21] library. However, a Raspberry Pi is considered a higher-tiered embedded device and does not operate on the Cortex-M4 which is the reference Microcontroller Unit (MCU) for embedded devices proposed by NIST for the PQC standardisation. In [3], an integration of the CRYSTALS-Kyber KEM scheme and the SPHINCS+ digital signature was implemented into the MbedTLS library. Despite different platforms being tested from a performance perspective, no measurements for accurate energy consumption were provided. Various performance metrics were also assessed in [19], with only empirical tests employed for an energy consumption estimation.

For the works focusing in energy requirements, the authors of [17] conducted an evaluation of the energy consumption of PQC candidates on an Intel Core i7-6700 CPU and ranked them based on the results. These algorithms' versions however are based on the Round 2 of the NIST standardisation competition. The work in [18] evaluates the overhead in energy and latency demand that the transition to PQC algorithms will cause, focusing on the TLS handshake. However, the TLS 1.2 version is utilized and the energy consumption comparisons do not include digital signature schemes, while some of them have currently been disqualified from the final results of the NIST competition. Additionally, another energy efficiency comparison for PQC algorithms was carried out in [1]. The differentiating factor in this work is the effort to use pre-quantum hardware accelerators and compare them to a RISC-V software implementation. Despite the speedup, the algorithms tested are also based on the Round 2 of the NIST competition and no full TLS handshake was executed. Lastly, in [11], an extensive energy requirement analysis for the new PQC candidates of the NIST standardisation competition was realized using an STM X-NUCLEO_LPM01A "PowerShield", a programmable power supply for NUCLEO devices that can also measure power consumption. The PQC algorithms were executed on a NUCLEO-F411RE development board and the collected measurements were incorporated into guidelines for correct PQC algorithm usage in IoT. These measurements, however, do not incorporate TLS 1.3 evaluation, but only analysis of PQC primitives. To the authors' knowledge, there is no complete work on real-time energy consumption evaluation for TLS 1.3 using PQC in resource-constrained devices.

1.2 Contribution

In this paper, extending the work in [23], an exhaustive power/energy consumption profiling of TLS 1.3 protocol when integrating various NIST PQC standardisation candidates is made and a comparison between them is performed in order to determine the energy consumption overhead each PQC algorithm combination introduces to the TLS 1.3 protocol. [1] The setup used includes an oscilloscope for accurate power measurements, a Cortex-M4 MCU board, the reference platform for embedded devices for the NIST PQC standardisation process and a laptop that acts as a server. More specifically, the power and energy consumption cost of a PQ TLS 1.3 handshake was evaluated in the two most typical scenarios in the IoT world:

- When a resource-constrained embedded system (an end-node) is connected to another device (an end-node or a more powerful sever) and both are mutually authenticated.
- When a resource-constrained embedded system (an end-node) is connected to another device (usually a more powerful server) and only the latter is authenticated.

Several PQC algorithms were integrated into TLS 1.3: All NIST PQC algorithms selected for standardisation; 2 out of 3 NIST PQC Round 4 KEMs; the KEM that the BSI (Germany's Federal Office for Information Security) recommends in its technical references for quantum resistance. Specifically, CRYSTALS-Dilithium, Falcon, SPHINCS+, CRYSTALS-Kyber, BIKE, HQC and FrodoKEM.

After integrating these PQC algorithms into TLS 1.3, a series of experiments were conducted in the two aforementioned scenarios and power and energy consumption measurements were taken. Additionally, the primitive operations of the PQC algorithms mentioned, were evaluated. Even though the primitive operations measurements does not introduce any novelty over [11], they were included for the sake of completeness, to support the PQ TLS 1.3 results as well as due to the different experimental setup and devices used, which led to different absolute values.

It should also be noted that even though the PowerShield from ST is the most common way to measure the power consumption of

[1]The source code is publicly available as two repositories at
https://gitlab.com/g_tasop/pq-wolfssl-for-embedded
https://gitlab.com/g_tasop/pq-wolfssl-for-pc

a Nucleo board, it seemed inadequate to measure the overall power consumption of a Nucleo board when running a full PQ TLS 1.3 handshake (it exceeded the 50mA current threshold for dynamic measurements [22]). Thus, we decided to use instruments that offer higher accuracy and higher current or voltage limits. The PicoScope 5444B [24] is a flexible, high-performance PC oscilloscope that was used in pair with PicoScope 7 software [2] to take measurements of the overall power consumption of a Nucleo board when running a PQ TLS 1.3 handshake.

The rest of the paper is structured as follows: In Section 2, the necessary background information on TLS 1.3 is described, while in Section 3 the experimental setup used is presented. In Section 4 the results of this evaluation process are presented and lastly, Section 5 concludes the paper.

2 BACKGROUND

2.1 TLS Protocol

The potential existence of a Cryptographically Relevant Quantum Computer (CRQC) [10] poses a threat to real-world security protocols that are utilising public key cryptography as their foundation. Transport Layer Security (TLS) protocol is among the most affected, due to its heavy usage of public key cryptography and its wide adoption. It is used by the majority of the applications wanting to communicate securely over the Internet (email servers [5], website transfer [15], smartphone apps [14]) making it the *de facto* standard. The latest version (TLS 1.3) [16] is currently at an adequate level of adoption, mainly due to the centralisation of the Internet [6]. In TLS 1.3, a whole round-trip is eliminated, in comparison to the previous version (TLS 1.2) and some potentially insecure primitives are deprecated, making it faster and more secure [16].

TLS consists of two main components: the *handshake protocol* and the *record protocol*. The *handshake protocol* is in charge of negotiating cryptographic parameters, establishing shared key material and authenticating communicating parties and to do this it is utilising public key cryptography (Diffie Hellman, Elliptic Curve cryptography, RSA). *Record protocol* is using symmetric key ciphers (AES, ChaCha) to secure the messages that the two parties exchange. While a Quantum Computer could potentially utilize Grover's algorithm [4] to speed up an attack on symmetric ciphers, it is not considered a serious threat. NIST is considering AES 128, which offers 128-bits of security, to be secure against quantum computers for the foreseeable future and a conservative approach would be to double the key length, thus doubling the 'bits' of security [9]. On the other hand, the public key cryptographic primitives of the protocol are seriously vulnerable against a CRQC running Shor's algorithm and need a replacement that would provide security against such attacks. The integration of post-quantum public key algorithms in such a protocol is mitigating this threat.

2.2 TLS implementations

In this paper, the WolfSSL implementation of the TLS protocol is chosen, because it is licensed under a GPLv2 open-source license and it is specifically targeting resource-constrained embedded devices. Also, it has implemented TLS version 1.3, in contrast with other popular open-source libraries (eg. MbedTLS).

In recent versions of WolfSSL, support for CRYSTALS-Dilithium, Falcon, and SPHINCS+ has been added using code from *liboqs* [25]. Also, CRYSTALS-Kyber has been present in the WolfSSL library but the code is available only after contacting WolfSSL [3]. However, these implementations are not targeting resource-constrained systems and are unoptimized for the Cortex-M4, with the exception of a limited experimental setup using Kyber512 directly from *pqm4* [26].

In the work of [23], the WolfSSL library has been extended in order to support all the ARM Cortex-M4 optimised versions of the PQC algorithms. Any implementation that was not present in the pqm4 repository was complemented with the respective implementation from PQClean [7]. More specifically, the authors in [23] used the *pqm4* implementation for CRYSTALS-Dilithium, Falcon, CRYSTALS-Kyber, BIKE, FrodoKEM and the *PQClean* implementation for SPHINCS+ and HQC. Adopting this paper's approach of this paper, we used the paper's code libraries and used them for our evaluation board, acting as the starting point of our research. The security levels of each of these deployed PQC algorithms are noted in Table 1 and are discussed further in Section 4.

2.3 PQC Modifications

In order to make TLS quantum-safe, architectural changes were made to the WolfSSL implementation and changes were introduced to the protocol itself. Two TLS message fields were altered, called "Extension Fields": "Supported Groups" and "Signature Algorithms". Additionally, changes were made in order to use KEMs instead of the classic Diffie-Hellman Key Exchange methods, as well as for the protocol to use post-quantum authentication both in Digital Signatures and in the Certificates employed. These changes are discussed in detail in [23] and are displayed in Figure 1.

3 EXPERIMENTAL SETUP

In our work, the power/energy consumption of the PQC TLS 1.3 was evaluated in the two most typical IoT scenarios:

Scenario 1 (mutual authentication): When a low-end, resource-constrained embedded device (the end-node) is connected to another device, another embedded device or maybe a more powerful device, and they are mutually authenticated, e.g. in an MQTT configuration where a number of devices need to communicate in a secure way to exchange data, before transmitting them to a server.

Scenario 2 (unilateral authentication): When our end-node is connected to a more powerful device (the server) and only the latter is authenticated, e.g when a sensor connects to a cloud server to transmit the data it has collected. In this scenario, the end-node, which acts as a TLS client, performs a TLS handshake with the server and only the server is authenticated.

In our experimental setup, the following devices were used: A Nucleo-F439ZI, which is equipped with a 32-bit ARM Cortex-M4 running at 180 MHz, with 192 KB of usable SRAM (plus 64 KB of CCM RAM that is not utilised) and 2 MB of Flash memory. It also provides Ethernet connectivity and a series of GPIO pins available

[2] Available at https://www.picotech.com/downloads

[3] https://github.com/wolfSSL/wolfssl/blob/master/wolfcrypt/src/wc_kyber.c#L25

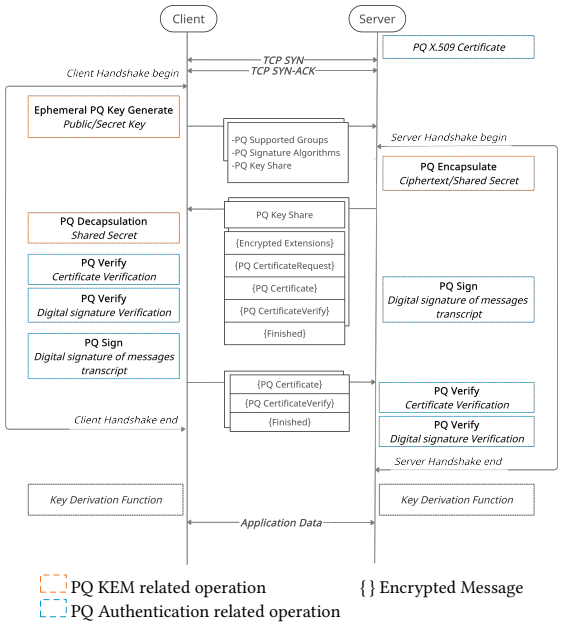

Figure 1: Post-quantum TLS 1.3 handshake messages.

to the user. This is a typical device for IoT end-nodes in such configurations and it additionally incorporates a Cortex-M4 core that NIST has selected as their reference platform for embedded devices in the PQC process. The second device is a laptop that is running Ubuntu 20.04 in x86_64 architecture and it is equipped with an Intel i7-1165G7 with 8 cores running at 2.8 GHz. This might not be a server-grade machine, but it is adequately more powerful than the embedded device and can act as a cloud server.

Both machines are connected to the same access point through the Ethernet interface, with a mean Round-Trip Time (RTT) of 0.493 ms, without any intermediate gateways (note that the Nucleo board is equipped with an Ethernet connector).

For the power consumption measurements, we used the PicoScope 5444B from the PicoScope 5000 Series [24], able to capture traces with up to 1 GS/s real-time sampling. In our experiment, we used a sampling rate of 5 MS/s, which is adequate to capture power consumption measurements and at the same time provides a large enough time window (5 seconds) to execute the operations under evaluation many times. In some exceptions, like a TLS handshake with SPHINCS+, where the total capture time should be a lot more than 5 seconds, a lower sampling rate was used, 833 kS/s, and a larger time window, 100 seconds. Two probes were used in a differential setup, to capture the Voltage trace across a small “shunt" resistor (1.5 Ohm) that was inserted in the IDD jumper of the board. This jumper (IDD) is specifically used to measure the current, with an Ammeter, that the 3.3V voltage regulator supplies to the whole circuit of the board, meaning the MCU, the peripherals, etc. In other words, we measured the realistic energy consumption of the whole resource-constrained embedded system in the two most typical scenarios in an IoT network, while being Quantum-Safe.

In Fig. 2, the power/energy consumption measurement setup can be seen. Two probes are connected to the “shunt" resistor, which

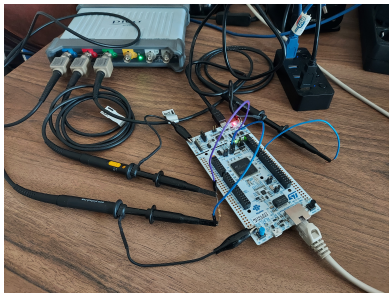

Figure 2: The Nucleo board and the PicoScope in the energy consumption configuration.

is inserted in the IDD jumper through some jumper cables and a third probe to a GPIO pin, through a plain wire. This is the D7 pin of the microcontroller and it is used as the "trigger" signal for the PicoScope. All the probes have their ground connected to the extra ground pins in the perimeter of the Nucleo board, an Ethernet cable is connected to the local access point and a USB cable is connected to a PC to power, program and control the board via a serial port.

A measurement capture is completed with the following steps:

- The first device (either the end-node or the server) runs as the TLS 1.3 server.
- The second device (either the server or the end-node respectively) runs as the TLS 1.3 client and connects to the TLS 1.3 server.
- When the two devices are connected at the TCP level and before the TLS 1.3 handshake begins, the end-node triggers the D7 pin and the PicoScope begins to measure the voltage of the probes in the differential format.
- The measurement continues as the two devices continuously perform the following: a complete TLS 1.3 handshake and an exchange of 2048 bytes of data in a loop until the 5 seconds window has passed.
- When the measurements are complete, the PicoScope software utility is used to compute the average power consumption, knowing the voltage trace across the "shunt" resistor and the value of the resistor.
- The average energy consumption can be computed by multiplying the average power consumption that we computed, with the average time measurement of the TLS 1.3 handshake that it was captured as well.

The process is the same for both experimental Scenarios, either when the board acts as a server or as a client. The only exception is TLS 1.3 with SPHINCS+ where a lower sampling rate is used and thus a larger time window to be able to fit its timely operations.

4 MEASUREMENTS, EVALUATION AND RESULTS

In this section, the power and energy consumption measurements regarding the PQC algorithm primitive operations and the PQ TLS 1.3 are presented and analyzed in depth. It should be noted that in all collected measurements, in order to evaluate the PQC algorithms and TLS 1.3 in real conditions, we have included the average power

Table 1: Time, Power and Energy consumption of Traditional and Post-quantum Primitives.

		Time (ms)			*Power (mW)*			*Energy (mJ)*		
KEMs	***NIST Security Level***	***Key Gen.***	***Enc.***	***Dec.***	***Key Gen.***	***Enc.***	***Dec.***	***Key Gen.***	***Enc.***	***Dec.***
ECDHE (secp256r)	*T 128-bit* [1]	8.428	17.687[2]	—	188.400	190.000[2]	—	1.588	3.369[2]	—
Kyber512	*1*	8.133	6.239	3.419	151.200	156.100	154.200	1.230	0.974	0.527
Bike-1	*1*	200.620	25.969	411.308	167.300	154.800	156.100	33.564	4.020	64.205
HQC-128	*1*	30.202	50.682	72.775	170.600	170.400	167.500	5.152	8.636	12.190
FrodoKEM1-AES	*1*	360.353	355.235	350.556	161.300	158.100	157.900	58.125	56.163	55.353
FrodoKEM1-SHAKE	*1*	522.917	520.583	516.083	166.700	166.100	165.200	87.170	86.469	85.257
Kyber768	*3*	12.224	11.412	7.924	154.700	157.900	156.500	1.891	1.802	1.240
Kyber1024	*5*	12.918	11.623	8.539	157.200	160.400	162.000	2.031	1.864	1.383
Digital Sig.	***NIST Security Level***	***Key Gen.***	***Sign***	***Verify***	***Key Gen.***	***Sign***	***Verify***	***Key Gen.***	***Sign***	***Verify***
RSA 2048-bit	*T 112-bit* [1]	462.853[3]	448.250	12.500	144.500[3]	139.700	143.800	66.882[3]	62.621	1.798
ECDSA (secp256r)	*T 128-bit* [1]	8.428	12.305	25.193	188.400	185.500	189.800	1.588	2.283	4.782
Sphincs128-small-simple	*1*	8674.000	66 239.000	61.588	194.100	194.900	195.300	1683.623	12 909.981	12.028
Sphincs128-fast-simple	*1*	137.750	3361.000	190.167	194.700	197.000	199.000	26.820	662.117	37.843
Falcon512	*1*	1266.667	243.881	3.275	132.700	124.800	142.100	168.087	30.436	0.465
Dilithium2	*2*	12.063	25.404[4]	9.569	160.000	161.400	161.000	1.930	4.100[6]	1.541
Dilithium3	*3*	19.438	39.309[5]	16.244	158.000	155.700	159.200	3.071	6.120[7]	2.586
Falcon1024	*5*	4802.667	527.789	6.852	137.100	128.100	144.800	658.446	67.610	0.992

[1] 'T' means bits of traditional security and 0-bit of quantum security, [2] key agreement, [3] public key + private key generation, [4] (11/114) (min/max) (ms) over the execution of 1000 signatures, [5] (15/138) (min/max) (ms) over the execution of 1000 signatures, [6] (1.775/18.399) (min/max) (mJ), [7] (2.335/21.487) (min/max) (mJ).

Table 2: Time, Power and Energy consumption of Traditional and PQ TLS 1.3 Handshake.

	Time (ms)			*Power (mW)*			*Energy (mJ)*		
Notation	***Client (mut)***	***Server (mut)***	***Client (uni)***	***Client (mut)***	***Server (mut)***	***Client (uni)***	***Client (mut)***	***Server (mut)***	***Client (uni)***
Dil1+Kyb1	96.318	91.062	69.639	155.800	161.300	176.300	15.006	14.688	12.277
Falc1+Kyb1	288.305	285.951	44.548	139.500	136.800	165.500	40.219	39.118	7.373
Dil3+Kyb3	157.126	153.492	98.481	161.600	164.200	178.400	25.392	25.203	17.569
Falc5+Kyb3	594.495	589.058	66.254	137.100	133.900	168.900	81.505	78.875	11.190
Dil3+Kyb5	165.590	152.537	105.865	161.200	162.900	180.100	26.693	24.848	19.066
Falc5+Kyb5	601.827	592.302	73.967	138.000	133.700	172.500	83.052	79.191	12.759
Sph1s+Kyb1	66 977.000	66 776.000	911.000	175.700	175.500	163.400	11 767.859	11 719.188	148.857
Dil2+Bike1	878.818	143.28	768.000	154.200	160.900	175.200	135.514	23.054	134.554
Dil2+Hqc1	198.603	145.989	183.622	156.000	158.800	174.800	30.982	23.183	32.097
Dil2+Frodo1AES	—	—	936.294	—	—	180.400	—	—	168.907
Dil2+Frodo1SHAKE	—	—	1252.250	—	—	199.700	—	—	250.074
RSA+ECDHE	540.220	538.158	77.237	144.500	150.400	168.200	78.062	80.939	12.991
ECDSA+ECDHE	109.171	106.927	102.109	167.100	175.500	181.500	18.242	18.766	18.533

consumption cost of 100 mW for idle operations in the evaluation board.

4.1 PQC algorithms measurements

Using the experimental setup described in 3, initially, for the sake of completeness, power/energy consumption evaluation data on individual PQC primitive operations were collected. The collected measurements complement and validate the measurements taken in [11]. The basic concept that was observed in our work and [11] regarding PQC algorithms is that the relationship between cycle count (or wall-time in our experiments) and the energy consumption is NOT linear and NOT algorithm-independent, in Cortex-M4 devices. In our measurements, variations in the power consumption up to 47% between PQC algorithms are observed, as is presented in Table 1 where each PQC algorithm's execution time, power and energy consumption is reported.

4.2 PQ TLS measurements

The main contribution of this work is the power and energy consumption evaluation of a full PQ TLS 1.3 handshake for various PQC algorithm combinations from the perspective of a typical IoT end-node in the two most common scenarios described in 3, a mutual authentication when the board acts as a client and as a server, and a server-only authentication when the board acts only as a client. The collected power/energy consumption measurements are presented in Table 2 along with the average time execution for a PQ TLS 1.3 handshake. Furthermore, a visual representation of the collected energy consumption measurements is provided in Fig. 1. Using these measurements, in the following subsections we perform an energy efficiency evaluation/comparison of the PQ TLS 1.3 when various PQC algorithm combinations are used. Furthermore, we evaluate/compare PQ TLS 1.3 with the various PQC algorithm combinations against the traditional TLS 1.3 and display the findings of this comparison in Table 3.

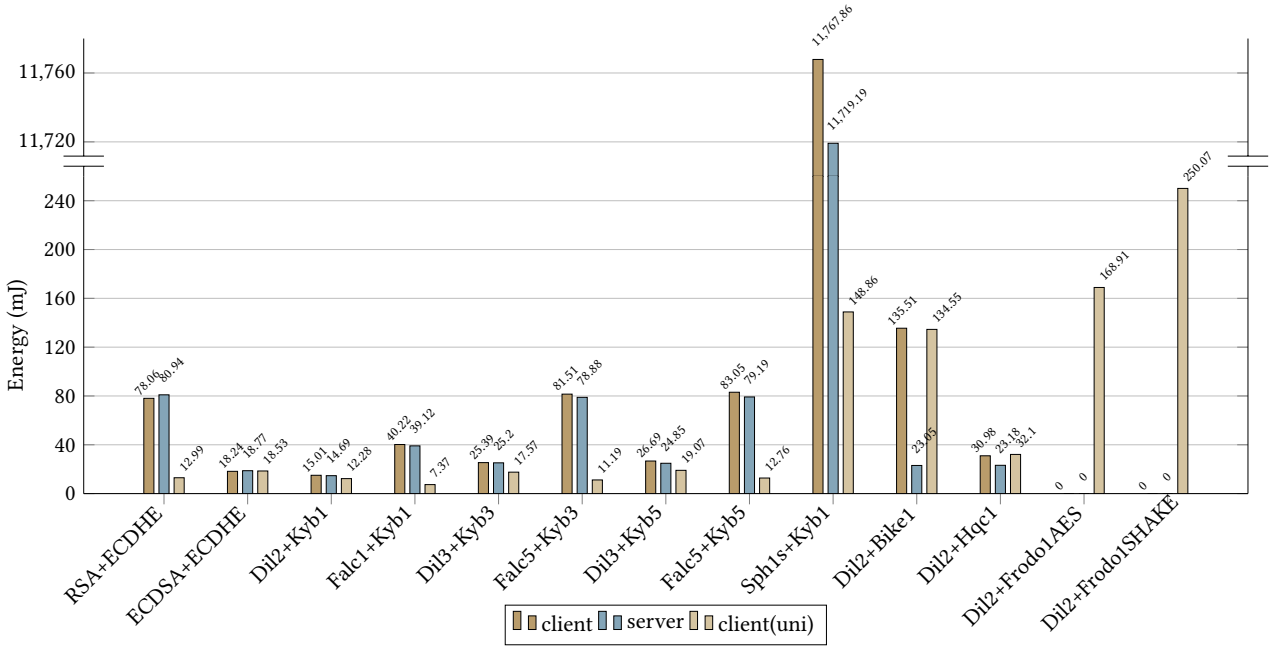

Figure 3: Average Energy Consumption (mJ) of Traditional and PQ TLS 1.3 handshake

In Table 2, the various PQC algorithm combinations when integrated into TLS 1.3 are presented in the first column. For example, the Notation *Dil2+Kyb1* means that Dilithium 2 is used for Digital Signatures and that Kyber 512 (NIST Security Level 1) is used for establishing shared keys. Note that when, for example, Dilithium 2 is used in Digital Signatures, it is assumed that employed Certificates and Digital Signatures in the TLS 1.3 handshake are using this algorithm. The rest of the Table is split vertically into 3 parts, average time delay for a single TLS 1.3 Handshake (denoted as *Time* in Table 2), average power consumption for a single TLS 1.3 Handshake (denoted as *Power* in Table 2) and average energy consumption for a single TLS 1.3 Handshake (denoted as *Energy* in Table 2). For each part of the Table, we provide 3 types of measurement, client and server TLS 1.3 handshake measurements in the mutual authentication scenario (*Client(mut) and Server (mut) respectively*) and client TLS 1.3 Handshake measurements in the unilateral authentication, when only the server is authenticated (*Client(uni)*).

In Fig. 3, the average energy consumption of a full PQ TLS 1.3 is presented, when using the respective PQC algorithm combinations, in the scenarios that have been identified. Note that the y-axis is "broken" to be able to include the SPHINCS+ combination's extremely high energy consumption measurements while being able to see the rest of the plot's measurements clearly. The first two bars in each PQ algorithm combination correspond to the mutual authentication scenario, when the board acts as a client (dark brown bars) and a server (dark blue bars). The third bar is the unilateral scenario of when the board acts as a client (light brown bars).

4.2.1 Comparison between PQ TLS combinations. Regarding our first scenario (mutual authentication), we can see that the difference in the energy consumption when the board acts as a client and as a server, is determined by the KEM (since the Authentication operations are symmetric). As seen in Fig. 1, the client executes a "Key Generate" and a "Decapsulate" and the server just an "Encapsulate" operation. The client operations consume more energy than the server operations, but the difference is minimal, on average between 2% and 5%. An exception is *Dil2+Bike1*, that due to the high energy consumption of its "Key Generate" and "Decapsulate" operations (as seen in Table 1, the client needs more than 5 times the energy of the server. *Dil2+Hqc1* also has a larger difference between client and server: the client consumes almost 33% more energy than the server. Taking into account that, apart from the above-mentioned exceptions, small energy consumption differences exist between client and server, in the rest of this subsection we consider the case when the board acts as a *client* and aim to compare PQ TLS 1.3 combinations for the same security level when possible.

From Table 2, it can easily be observed that *Dil2+Kyb1* offer the best energy consumption performance. *Falc1+Kyb1*, with its costly "Sign" operation falls behind needing ~2.68 times more energy.

Dil2+Bike1 as a client consumes ~7 times more energy than *Dil2+Kyb1* and ~2.6 times more energy than *Falc1+Kyb1*. As a server, though, *Dil2+Bike1* consumes just 33% more energy than *Dil2+Kyb1* but needs only half the energy of *Falc1+Kyb1*. *Dil2+Hqc1* has competitive energy requirements, needing twice the energy of *Dil2+Kyb1* but only 3/4 the energy of *Falc1+Kyb1*.

Sph1s+Kyb1, in the mutual authentication scenario, is almost unusable, as the energy requirements are orders of magnitude higher than the rest of the combinations. Just for reference, we report that *Dil2+Bike1* needs 111 times the energy of the most costly combination of the other combinations, i.e. *Dil2+Bike1*. It should be noted

Table 3: Comparison of PQC with Traditional TLS 1.3 Handshake (Energy Consumption ratio).

	PQ TLS 1.3 versus RSA+ECDHE TLS 1.3 (×)			*PQ TLS 1.3 versus ECDSA+ECDHE TLS 1.3 (×)*		
Notation	***Client (mut)***	***Server (mut)***	***Client (uni)***	***Client (mut)***	***Server (mut)***	***Client (uni)***
Dil1+Kyb1	0.192	0.181	0.945	0.823	0.783	0.662
Falc1+Kyb1	0.515	0.483	0.568	2.205	2.085	0.398
Dil3+Kyb3	0.325	0.311	1.352	1.392	1.343	0.948
Falc5+Kyb3	1.044	0.974	0.861	4.468	4.203	0.604
Dil3+Kyb5	0.342	0.307	1.468	1.463	1.324	1.029
Falc5+Kyb5	1.064	0.978	0.982	4.553	4.220	0.688
Sph1s+Kyb1	150.751	144.790	11.458	645.080	624.501	8.032
Dil2+Bike1	1.363	0.242	10.357	5.832	1.044	7.260
Dil2+Hqc1	0.397	0.286	2.471	1.698	1.235	1.732
Dil2+Frodo1AES	—	—	13.002	—	—	9.114
Dil2+Frodo1SHAKE	—	—	19.249	—	—	13.494

that SPHINCS+ offers 2 versions: A small version, with small artifacts but slow execution time and a fast version with big artifacts and fast execution time. The latter, offers such big artifacts that exceed the *Max_record_size* of the TLS protocol itself. The small version can be used in PQ TLS 1.3 but the high execution delay, especially in a resource-constrained system like ours, lead to a very high energy consumption cost.

Regarding our second scenario (unilateral authentication of the server when the board acts as a client), the picture is different. It can be seen that *Dil2+Kyb1* is now 66% more costly than *Falc1+Kyb1*, which is now the most energy-efficient combination, even on higher security levels. For example, *Falc5+Kyb5* is still 50% more efficient than *Dil3+Kyb5*, even though Dilithium is NIST security level 3 compared to Falcon which is 5.

Dil2+Bike1 energy consumption is still high as the energy consumption of costly KEM operations is still needed in the scenario where the board acts as a client, even without the need to be authenticated. *Dil2+Hqc1* now requires ∼2.6 times more energy than *Dil2+Kyb1* and ∼4.3 times more than *Falc1+Kyb1*.

It is interesting, that in this scenario, *Sph1s+Kyb1* becomes usable. The energy consumption is still high, but it is now comparable to the other combinations. It requires x12 times more energy than *Dil2+Kyb1* and x20 times more than *Falc1+Kyb1*. As SPHINCS+ is the conservative choice of the NIST PQC standardisation process regarding PQ Authentication (it provides robust security based on the security of the underlying hash function) the high energy cost can be out-weighted by the robustness of its security.

It should be noted that in the second experimental scenario (board acts as a client without the need to be authenticated), FrodoKEM (AES and SHAKE variants) was able to be evaluated, since it is the recommended PQC KEM by the BSI standardisation body. In fact, the BSI recommends FrodoKEM-976 and FrodoKEM-1344 which are NIST Security Level 3 and 5 respectively. Unfortunately, they are unusable in our selected board because of their high memory requirements and instead, FrodoKEM-678 (NIST security level 1) was used, to give a general view of its performance.

Even though the energy consumption is high for these combinations, their cost remain comparable to the other combinations. *Dil2+Frodo1AES* needs ∼13.7 times more energy than *Dil2+Kyb1* and ∼23 times more than *Falc1+Kyb1*. Compared to the NIST Round 4 combinations, it needs 25% more energy than *Dil2+Bike1* and ∼5.26 times the energy of *Dil2+Hqc1*. Finally, compared to *Sph1s+Kyb1* we can see that their energy requirements are close, needing just 13% more energy. *Dil2+Frodo1SHAKE* comparison in energy consumption follows *Dil2+Frodo1AES* in the same spirit. We report only that it needs 48% more energy than *Dil2+Frodo1AES*. Nevertheless, it must be noted that the BSI recommendation on FrodoKEM is prioritising performance, but gives an emphasis on claimed security. As FrodoKEM is a lattice-based KEM that does not depend on any underlying lattice structure (that may be exploited in the future to craft an attack), like for example Kyber, the security assumption is minimal and includes a significant security margin.

4.2.2 Comparison with Traditional TLS. To provide a more complete and detailed analysis, a comparison was made between the energy consumption cost of PQ TLS 1.3 and traditional TLS 1.3 (using traditional Public Key Cryptography algorithms as offered by the TLS 1.3 standard). Even though traditional algorithms provide no resistance to potential future quantum attacks, they can be used as a point of reference in understanding the energy consumption overhead that the new PQC algorithms will introduce to protocols like TLS and the new challenges that need to be addressed in real conditions. In Table 3 the energy consumption overhead of all the PQ TLS 1.3 combinations when compared to two traditional TLS 1.3 combinations (i.e *RSA+ECDHE* and *ECDSA+ECDHE*) is presented. In the first column, the quotient of the energy consumption of a PQ TLS combination divided by the energy consumption of *RSA+ECDHE* is displayed (PQC vs *RSA+ECDHE* ratio) while in the second column, the quotient of the energy consumption of a PQ TLS combination divided by the energy consumption of *ECDSA+ECDHE* is presented (PQC vs *ECDSA+ECDHE* ratio). For example, *Dil1+Kyb1* requires 0.192 times the energy of *RSA+ECDHE* when the board acts as a TLS client, in a mutual authentication scenario.

Regarding the PQC combinations NIST has selected for standardisation, i.e. *Dil1+Kyb1* and *Falc1+Kyb1*, it can be observed that compared to *RSA+ECDHE* TLS 1.3, they require *less* energy in both scenarios. It should be noted that RSA's energy efficient "Sign" operation, almost challenges *Dil1+Kyb1* energy consumption. When compared to *ECDSA+ECDHE*, the *Dil1+Kyb1* TLS 1.3 performs better in all scenarios but *Falc1+Kyb1* has worse energy consumption in the mutual authentication scenario. Of course, in a unilateral authentication, when the board acts as an unauthenticated client, *Falc1+Kyb1* is the most energy-efficient approach.

The *Dil2+Bike1* setup is more energy costly than both traditional TLS 1.3 combinations in both scenarios, except when the evaluation board acts as a server (in the mutual authentication scenario). It also challenges the energy consumption of RSA+ECDHE as a client in the mutual authentication scenario, by being just 1.363 times more costly.

Dil2+Hqc1 outperforms *RSA+ECDHE* in the mutual authentication scenario but has worse energy consumption than *RSA+ECDHE* in the unilateral scenario. It is still more costly than *ECDSA+ECDHE*, but in a comparable manner, consuming 1.698, 1.235 and 1.732 times more energy than the *ECDSA+ECDHE* TLS 1.3 in the mutual authentication (client and server) and the unilateral authentication (client) scenarios respectively.

Sph1s+Kyb1 has a significant energy consumption difference in comparison to all traditional TLS 1.3 combinations in the mutual authentication scenario (it is many thousands of times more energy consumption hungry compared to traditional TLS 1.3 combinations) but in the unilateral scenario, as shown in Table 3, it is just 11.458 times more energy costly than RSA+ECDHE and 8.032 times more costly than ECDSA+ECDHE TLS 1.3.

Regarding *Dil2+Frodo1AES* and *Dil2+Frodo1SHAKE*, they are 13 and 19.249 times more costly than *RSA+ECDHE* TLS 1.3 respectively and 9.114 and 13.494 times than *ECDSA+ECDHE* TLS 1.3. This has an energy consumption cost that many systems could afford in order to be conservatively secure against potential quantum attacks.

Some remarks can also be made on the higher NIST security levels of the PQC algorithms that are selected for standardisation. It can be observed from Table 3 that when compared with *RSA+ECDHE* TLS 1.3, on the mutual authentication scenario, all such PQ TLS 1.3 combinations outperform it in the energy consumption cost. This means that embedded systems using these PQC algorithms (i.e Dilithium or Falcon and Kyber) to make a quantum resistance transition from traditional *RSA+ECDHE* TLS 1.3 could benefit from such a security upgrade with marginally no energy consumption overhead. It should be noted that *RSA+ECDHE* and *ECDSA+ECDHE* offer approximately 128-bits of traditional security while higher PQC algorithms combinations offer 192-bit and 256-bit of traditional security. This benefit is however lost when replacing *ECDSA+ECDHE* TLS 1.3 with PQC high-security level TLS 1.3. In such case, for the mutual authentication scenario, all NIST standardised PQC combinations require more energy than *ECDSA+ECDHE* TLS 1.3, with *Dil3+Kyb3* and *Dil3+Kyb5* requiring 1.392 and 1.463 more energy respectively and with *Falc5+Kyb3* and *Falc5+Kyb5* increasing this energy consumption overhead by requiring 4.468 and 4.553 times more energy respectively. The above results refer to a mutually authenticated TLS *client* however the TLS *server* measurements are very similar. On the unilateral authentication scenario though, all PQC combinations require less energy than *ECDSA+ECDHE* TLS 1.3, thus offering an upgrade in security (traditional and quantum) without any energy consumption overhead.

5 CONCLUSIONS

In this paper, the goal was to provide a detailed energy consumption profiling of the PQ TLS 1.3 in line with the latest outcomes of the NIST standardisation effort and map in detail the energy consumption overhead of the various PQC algorithms within the TLS 1.3 protocol handshake. This process resulted in power/energy consumption measurements and comparisons between various PQC algorithm combinations (KEMs combined with Digital Signatures) within the PQ TLS 1.3 but also power/energy consumption measurements and comparisons between PQ TLS 1.3 combinations and traditional TLS 1.3 ones. Considering the need for making a Quantum resistant transition in IoT within the next few years (as NIST will provide PQC standards), this paper's work provides valuable insight into the challenges to be faced when trying to deploy PQC in resource-constrained battery-powered embedded systems. From the performed comparisons and analysis, it can firstly be concluded that when dealing with resource-constrained systems, like the ARM Cortex-M4, each PQC algorithm introduces a different amount of power consumption with up to 47% variation. Secondly, when these PQC algorithms are integrated into protocols, like TLS, an impact on its power dissipation can be observed too. The variation is generally smaller in a TLS 1.3 implementation since the protocol includes operations that are common in all TLS 1.3 variations, but the variation in some cases can reach up to 49%. This power consumption along with the PQC algorithms' execution time influences the PQC algorithms' energy consumption and eventually the TLS 1.3 implementation that uses it. From the paper's analysis of the energy consumption results, it can be concluded that some PQ TLS 1.3 combinations perform better in the mutual scenario while some others perform better in the unilateral scenario. In the mutual authentication scenario, combinations with Dilithium consume the least energy while in the unilateral, combinations with Falcon offer the least energy consumption cost. It can also be concluded that the NIST conservative security choice of SPHINCS+ as a Digital Signature (to be standardised) can only be used in a unilateral authentication since SPHINCS+ needs tremendous amounts of energy and time just for a single handshake. Furthermore, PQ TLS 1.3 combinations with the NIST Round 4 KEMs have very unbalanced energy consumption requirements leading to the odd conclusion that the most energy-efficient setting is the device to act as a server. Despite the fact that this is not a common scenario, there might exist some settings where this could be beneficial. Lastly, FrodoKEM, which is the conservative recommendation of BSI, seems to require a substantial amount of energy but still in a comparable manner with the rest of the PQC algorithms in a TLS 1.3 implementation.

ACKNOWLEDGMENTS

The work in this paper has received funding from the European Union's Horizon 2020 programme ENERMAN under grant agreement No 958478.

REFERENCES

[1] Utsav Banerjee, Siddharth Das, and Anantha P. Chandrakasan. 2020. Accelerating Post-Quantum Cryptography using an Energy-Efficient TLS Crypto-Processor. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. https://doi.org/10.1109/ISCAS45731.2020.9180550

[2] Jon Barton, William J. Buchanan, Nikolaos Pitropakis, Sarwar Sayeed, and Will Abramson. 2022. Post Quantum Cryptography Analysis of TLS Tunneling on a Constrained Device. In *ICISSP*. SCITEPRESS, 551–561.

[3] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. 2020. 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* (Taipei, Taiwan) *(ASIA CCS '20)*. Association for Computing Machinery, New York, NY, USA, 841–852. https://doi.org/10.1145/3320269.3384725

[4] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *STOC*. ACM, 212–219.

[5] Paul E. Hoffman. 2002. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207. https://doi.org/10.17487/RFC3207

[6] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. 2020. Tracking the deployment of TLS 1.3 on the web: a story of experimentation and centralization. *Comput. Commun. Rev.* 50, 3 (2020), 3–15.

[7] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. 2022. Improving Software Quality in Cryptography Standardization Projects. In *EuroS&P Workshops*. IEEE, 19–30.

[8] NIST. 2022. NIST PQC Standardisation. Retrieved 28-02-2023 from https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4

[9] NIST-FAQ. 2018. To protect against the threat of quantum computers, should we double the key length for AES now? Retrieved 28-02-2023 from https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs#xisl

[10] NSA. 2021. Quantum Computing and Post-Quantum Cryptography FAQ. Retrieved 28-02-2023 from https://media.defense.gov/2021/Aug/04/2002821837/-1/-1/1/Quantum_FAQs_20210804.PDF 2nd question: What is a "Cryptographically Relevant Quantum Computer (CRQC)?.

[11] Markku-Juhani O. Saarinen. 2020. 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)*. 23–30. https://doi.org/10.1109/MobileCloud48802.2020.00012

[12] Christian Paquin, Douglas Stebila, and Goutam Tamvada. 2020. Benchmarking Post-quantum Cryptography in TLS. In *Post-Quantum Cryptography*, Jintai Ding and Jean-Pierre Tillich (Eds.). Springer International Publishing, Cham, 72–91.

[13] Sebastian Paul, Yulia Kuzovkova, Norman Lahr, and Ruben Niederhagen. 2022. Mixed Certificate Chains for the Transition to Post-Quantum Authentication in TLS 1.3. In *AsiaCCS*. ACM, 727–740.

[14] Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Johanna Amann, and Phillipa Gill. 2018. Studying TLS Usage in Android Apps. In *ANRW*. ACM, 5.

[15] Eric Rescorla. 2000. HTTP Over TLS. RFC 2818. https://doi.org/10.17487/RFC2818

[16] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://doi.org/10.17487/RFC8446

[17] Crystal Andrea Roma, Chi-En Amy Tai, and M. Anwar Hasan. 2021. Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms. *IEEE Access* 9 (2021), 71295–71317. https://doi.org/10.1109/ACCESS.2021.3077843

[18] Maximilian Schöffel, Frederik Lauer, Carl C. Rheinländer, and Norbert Wehn. 2021. 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* (Charlottesvle, VA, USA) *(IoTDI '21)*. Association for Computing Machinery, New York, NY, USA, 158–168. https://doi.org/10.1145/3450268.3453528

[19] Jose-Antonio Septien-Hernandez, Magali Arellano-Vazquez, Marco Antonio Contreras-Cruz, and Juan-Pablo Ramirez-Paredes. 2022. A Comparative Study of Post-Quantum Cryptosystems for Internet-of-Things Applications. *Sensors* 22, 2 (2022). https://doi.org/10.3390/s22020489

[20] Peter W. Shor. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *FOCS*. IEEE Computer Society, 124–134.

[21] Douglas Stebila and Michele Mosca. 2016. Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project. In *SAC (Lecture Notes in Computer Science, Vol. 10532)*. Springer, 14–37.

[22] STMicroelectronics. 2023. X-Nucleo-LPM01A Overview. Retrieved 28-02-2023 from https://www.st.com/en/evaluation-tools/x-nucleo-lpm01a.html#st_all-features_sec-nav-tab All features.

[23] George Tasopoulos, Jinhui Li, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. 2022. Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems. In *Information Security Practice and Experience*, Chunhua Su, Dimitris Gritzalis, and Vincenzo Piuri (Eds.). Springer International Publishing, Cham, 432–451.

[24] Pico Technologies. 2023. PicoScope 5000 Series Datasheet. Retrieved 28-02-2023 from https://www.picotech.com/download/datasheets/MM040.en-8.pdf PicoScope 5444B.

[25] WolfSSL. 2022. WolfSSL Changelog. Retrieved 28-02-2023 from https://github.com/wolfSSL/wolfssl/blob/master/ChangeLog.md#post-quantum-3

[26] WolfSSL. 2022. WolfSSL PQ Key Establishment in Cortex-M4. Retrieved 28-02-2023 from https://www.wolfssl.com/post-quantum-tls-1-3-key-establishment-comes-stm32-cortex-m4