Fully Hybrid TLSv1.3 in WolfSSL on Cortex-M4

Mila Anastasova[1], Reza Azarderakhsh[1], and Mehran Mozaffari Kermani[2]

[1] Computer and Electrical Engineering and Computer Science Department and I-SENSE at Florida Atlantic University, Boca Raton, FL, USA
(manastasova2017, razarderakhsh)@fau.edu
[2] Computer Engineering and Science Department at University of South Florida, Tampa, FL, USA,
mehran2@usf.edu

Abstract. To provide safe communication across an unprotected medium such as the internet, network protocols are being established. These protocols employ public key techniques to perform key exchange and authentication. Transport Layer Security (TLS) is a widely used network protocol that enables secure communication between a server and a client. TLS is employed in billions of transactions per second. Contemporary protocols depend on traditional methods that utilize the computational complexity of factorization or (elliptic curve) logarithm mathematics problems. The ongoing advancement in the processing power of classical computers requires an ongoing increase in the security level of the underlying cryptographic algorithms. This study focuses on the analysis of Curve448 and Edwards curve Ed448, renowned for their superior security features that offer a 224-bit level of security as part of the TLSv1.3 protocol. The exponential advancement of quantum computers, however, presents a substantial threat to secure network communication that depends on classical crypto schemes, irrespective of their degree of security. Quantum computers have the capability to resolve these challenges within a feasible timeframe. In order to successfully transition to Post-Quantum secure network protocols, it is imperative to concurrently deploy both classical and post-quantum algorithms. This is done to fulfill the requirements of both enterprises and governments, while also instilling more assurance in the reliability of the post-quantum systems. This paper presents a detailed hybrid implementation architecture of the TLSv1.3 network protocol. We showcase the first deployment of Curve448 and Crystals-Kyber for the purpose of key exchanging, and Ed448 and Crystals-Dilithium for verifying the authenticity of entities and for X.509 Public Key Infrastructure (PKI). We rely upon the widely used OpenSSL library and the specific wolfSSL library for embedded devices to provide our results for server and client applications.

Keywords: Network Protocols, TLSv1.3, PKI, X.509, Elliptic Curve Cryptography (ECC), Post-Quantum Cryptography (PQC), Cortex-M4

1 Introduction

In the era of digital technology, where information is easily transmitted worldwide, the foundation of interconnected systems is formed by complex network

protocols ensuring reliability and interoperability of communication networks. Security protocols, such as the most world wide SSL/TLS, relays on cryptographical algorithms, to ensure data integrity, confidentiality, authentication and non-repudiation. Public Key Cryptography (PKC) allows the secure communication establishment between entities through insecure channel, such as the Internet, and is a fundamental component of security network protocols. Transport Security Layer (TLS), also knows by the name of its predecessor Socket Security Layer (SSL), integrates PKC algorithms for key exchange and digital signature to allow secure data exchange across communication parties. It enables them to transition their application data exchange to a Symmetric Key Cryptographic scheme, which guarantees significantly improved computation time. Throughout the years, the SSL/TLS protocol specification has been going through changes, in order to eliminate vulnerabilities, improve timing and increase security of the performed communication. In 2016, the latest version, TLSv1.3 [1], was released, included many revisions, along with the deprecation of weak algorithms that were supported in earlier versions of TLS, as well as the addition of new algorithms.

The Elliptic Curve Cryptography (ECC) family of algorithms is one of the most widely deployed cryptographic PKC schemes, owing to their small key and signature sizes, which allow them to be used in bandwidth-constrained scenarios, as well as their relatively low computational cost, which allows them to be used in both high- and low-end devices. Curve448, used for Elliptic Curve Diffie-Hellman (ECDH) based key derivation, and its birationally equivalent Ed448, forming Edwards curve Digital Signature Algorithm (EdDSA), have become of interest among multiple ECC primitive instantiations, based on eliminating several cryptographic security concerns inherent in NIST curves while offering high security level.

The progressive development of Quantum Computers, marked by the continuous increase in q-bit quantities, presents a significant challenge to conventional cryptographic methods that form the basis of network communication. Shor [2] demonstrates that classical cryptographic primitives are vulnerable to quantum computer attacks once a sufficiently powerful computer is constructed. This would enable the solving of Factorization and Discrete Logarithm problems, which form the core of classical cryptographic primitives.

In 2016, the National Institute of Standards and Technology [3] (NIST) began evaluating the efficacy and efficiency of a list of recently submitted cryptographic algorithms that are resistant to attacks by quantum computers. Following three rounds of evaluation and enhancements, NIST has announced four algorithms that will ultimately be standardized thereafter used in a broad variety of security protocols. Among six families of post-quantum robust cryptographic algorithms, lattice-based schemes show to be one of the most promising based on their relatively compact key sizes and the extremely efficient computational cost. Based on Module-Learning With Errors (M-LWE), Crystals-Kyber Key Encapsulation Mechanism (KEM), based on Public Key Encryption (PKE) along with a variation of the Fujisaki–Okamoto (FO) transform to ensure $IND-CCA2$-security,

is the only PQ key exchange finalists of the NIST PQ Standardization process. Similarly, M-LWE-based Crystals-Dilithium Digital Signature Algorithms (DSA) form part of the three PQ secure finalists for DSA along with Falcon and SPHINCS+, showing on average (key generation + sign + verify) performance advantage among its competitors and *reasonably* compact keys and signature sizes.

Given that the PQ algorithms are relatively new, they fail to fulfill the security criteria set by the government and industry. Therefore, a hybrid instantiation is necessary to provide a seamless transition to PQ network protocols. In this work, we present, to the best of our knowledge, the first entirely hybrid instantiation of the widely deployed TLSv1.3 protocols, integrating classical high security Curve448 ECDH and Crystals-Kyber1024 PQ KEM algorithms for key derivation among client and server and Edwards curve Ed448 traditional digital signature algorithm along with Crystals-Dilithium5, in order to ensure data privacy, integrity, authentication, and non-repudiation in the presence of classical and PQ adversary. We perform hybrid key derivation and enhance the Public Key Infrastructure (PKI) defined by the X.509 certificate standard by integrating hybrid keys, certificate (signature) generation and signature verification.

Ensuring the deployment of cryptographic algorithms and security protocols on resource-constrained devices and bandwidth-limited scenarios is crucial due to the increasing integration of small embedded systems in everyday life, driven by the Internet of Things (IoT) and the desire to enhance lifestyle and comfort. This study aims to assess the performance of the hybrid TLSv1.3 protocol using Curve448 with Kyber1024 and Ed448 with Dilithium5 algorithms. The evaluation is conducted on the NIST approved ARMv7-based Cortex-M4 processor, specifically on the WiFi enabled STM32F413 Discovery Board. We base our work on the widely deployed OpenSSL library in order to generate the hybrid X.509 keys and signatures and wolfSSL embedded-focused library for performance evaluation.

1.1 Related Work

The widespread use of Internet of Things (IoT) devices, embedded systems, and various other low-end computer platforms has brought about an epoch of remarkable connectivity via the deployment of network protocols. Yet, the inherent characteristics of these devices, featuring restricted processing capabilities and limitations in power and energy supply, provide a significant obstacle in the implementation of resilient cryptographic protocols. Elliptic Curve Cryptography is considered a fundamental aspect of secure communication because of its simplicity and robust security promises. Nevertheless, the use of this technology on low-end devices requires a careful and sophisticated strategy to overcome the underlying constraints.

Curve448, introduced by *Hamburg* in [4], is meticulously designed to strike a balance between strength and computational efficiency, making it a compelling choice for secure key exchange and digital signatures. The high security level, in comparison to other NIST curves or Curve2551 and Ed25519 proposed in [5]

and [6], comes at the cost of computational overhead based on the larger length of the field arithmetics. This challenge is significantly important when it comes to IoT devices with limited computational resources and bounded battery life. This is the reason for exhaustive effort in the optimal implementation of the cryptographic schemes aiming at optimal execution on embedded devices. The nature of ECC allows optimizations on the field arithmetic layer, where different research teams have shown efficient implementation for Curve448 and Ed448 targeting 8-bit AVR and 16-bit MSP, and 32-bit Cortex-M4 devices [7], [8], [9], [10], [11], [12]. The higher-layer group operation may also introduce implementation optimizations based on applying optimal strategies for point addition and multiplication, the core of ECC schemes. Several works have been performed, applying different point multiplication architectures, such as low execution latency (Sliding Window [13] method, Signed Comb [14] method), compact code side (Double-and-Add [15]) or constant time performance (Double-and-Always-Add, Montgomery Ladder [16]). The optimal implementation of Curve448 and Ed448 is further being evaluated as part of network protocols [17].

Deploying post-quantum cryptography primitives on low-end devices is particularly problematic because to the increased resources needed for its implementation. Lattice-based post-quantum primitives, such as Kyber and Dilithium, rely on computationally easy problems and do not need intricate multi-precision field arithmetic due to the tiny modulus used in operations. Both methods rely on two computationally intensive algorithms: the Secure Hash Algorithm 3 (SHA-3) and the Number Theoretic Transform (NTT), which is essentially equivalent to the Fast Fourier Transform (FFT) function applied over finite fields. Various studies in the literature have offered distinct approaches, demonstrating the most effective techniques for executing certain procedures.

Several researchers focus on optimizing the design of ARMv8-based devices to demonstrate the effectiveness of NTT transform function and modular reduction in enabling the use of NEON-specific Single Instruction Multiple Data (SIMD) instructions. This optimization leads to significant improvements in the runtime performance of the lattice-based Kyber and Dilithium PQ primitives [18,19,20,21,22]. Targeting Advanced Vector Extension (AVX) ISA (AVX2 and AVX-512) was presented in [23,24]. Other writers focus on exploiting the computational capabilities of more advanced devices, such as GPUs, which are known for their high processing power [25,26,27]. *Botros et al.* have provided an implementation design for low-end IoT devices that focuses on optimizing the RAM use of Kyber while enhancing its speed performance [28]. *Alkim et al.* [29] demonstrate superior outcomes of NTT calculations by utilizing an improved modular reduction architecture, which enables the adoption of an efficient Instruction Set Architecture (ISA) in combination with lazy-reduction deployment. *Abdulrahman et al.* present several ways for implementing NTT, enhancing register use management, and achieving vector-matrix accumulation outcomes in their study [30].

The extensively improved classical and post-quantum public-key cryptography (PKC) primitives serve as the fundamental mathematical components of

security network protocols, particularly the SSL/TLS network protocol, which is the most extensively utilized. It employed several cryptographic techniques to provide safe and dependable communication between server and client entities within the Internet network. While extensive research has been conducted on the performance outcomes of cryptographic primitives, there has been a lack of sufficient research on the complete integration of classical and hybrid systems into network protocols, which is the focus of this study.

The incorporation of post-quantum (PQ) and hybrid operating modes into the TLSv1.3 network protocol necessitates substantial effort due to the utilization of numerous cryptographic methods. Several studies in the literature have explored the development of a PQ operational mode for the TLSv1.3 protocol and its predecessor, TLSv1.2, as well as other network protocols [31], [32]. *Kampanakis et al.* [33] examine the utilization of PQ signatures inside X.509 PKI certificates, specifically in relation to the package fragmentation mechanism. *Crockett et al.* [31] present a study of hybrid key exchange within the context of TLS and SSH network protocols. They examine several classical and post-quantum cryptographic primitives and explore ways for deploying hybrid authentication and X.509 PKI. *Campagna et al.* [34] describes the inclusion of hybrid key exchange in the TLSv1.2 network protocol, which incorporates SIKE and BIKE post-quantum key encapsulation mechanisms (PQ KEMs) in addition to elliptic curve Diffie-Hellman (ECDH) methods. *Sikeridis et al.* [35] improve the OQS and OpenSSL library by incorporating a PQ-standalone message signature and modifying the X.509 PKI with post-quantum capabilities. The authors also evaluate the impact of the PQ message signature `CertificateVerify` execution cost in TLSv1.3. [36]. *Marchsreiter et al.* [37] present an assessment of the post-quantum and hybrid operating mode of TLSv1.3, utilizing a hybrid key agreement and hybrid digital signature technique for server authentication via message signature `CertificateVerify`. Nevertheless, the authors provide their findings derived from the deployment of PQ-standalone X.509 PKI. Furthermore, the assessment solely relies on NIST curves, neglecting the assessment of TLSv1.3 integrated Curve448 and Ed448.

In our research, we focus on the deficiencies in the existing literature by conducting an assessment of hybrid TLSv1.3. Specifically, we provide the first evaluation of a fully hybrid TLSv1.3 implementation that utilizes Curve448 with Kyber1024 as classical and post-quantum key exchange methods, and Ed448 with Dilithium5 as classical and post-quantum digital signature algorithms, respectively.

1.2 Contributions

In this work, we present, to the best of our knowledge, the first fully hybrid TLSv1.3 based on Curve448 and Crystals-Kyber1024 for key exchange and Ed448 and Crystals-Dilithium5 for authentication and certificate verification. Our contributions include the following:

1. We provide the entirely hybrid version of the TLSv1.3 network protocol, including Curve448 and Ed448 to guarantee resilience against classical com-

puter adversaries, as well as Crystals-Kyber1024 and Crystals-Dilithium5 to provide protection against quantum computer attacks.
2. We enhance the widely deployed OpenSSL library by including the capability to generate the X.509 hybrid `Ed448_Dilithium5`-based keys and certificates in PEM file format, where the certificate hybrid key and signature are both based on classical Ed448 and PQ Dilithium5 algorithms.
3. We implement hybrid key exchange based on emerging high-security level Curve448 and the PQ Kyber1024 algorithms, where both communication parties issue a symmetric key value based on a classical and PQ shared secret derivation.
4. We upgrade the embedded-specific wolfSSL cryptographic library to sign a message using both classical and PQ signature algorithms and to verify hybrid signatures based on `Ed448_Dilithium5`.
5. We deploy functions to process hybrid certificates `Ed448_Dilithium5` certificates, including classical Ed448 and PQ Dilithium5 public key and signature values. We also enable verification of `Ed448_Dilithium5` hybrid certificate signatures.
6. We evaluate the proposed hybrid TLSv1.3 based on `Curve448_Kyber1024` for key agreement and `Ed448_Dilithium5` for authentication and certificate validation on the NIST recommended ARMv7 Cortex-M4 STM32F413 WiFi equipped microcontroller and report the execution timing for the entire TLSv1.3 transmitting a 15B short message both directions, and for the pure TLSv1.3 handshake, neglecting the AEAD scheme overhead.

The subsequent sections of the paper are structured in the following manner. In Section 2 we provide a comprehensive explanation of the mathematical principles that form the foundation of Curve448 and Ed448 ECC algorithms. Additionally, we analyze the distinctive features of Crystals-Kyber and Crystals-Dilithium PQ algorithms. Section 3 provides an introduction to the TLS1.3 network protocol and the X.509 PKI architecture. It also emphasizes the improvements made in the design to achieve a completely hybrid TLSv1.3. In Section 4 we present an evaluation of the execution latency of the TLSv1.3 protocol, as part of the wolfSSL embedded cryptographic library. Ultimately, we bring our effort to a close in Section 5.

2 Preliminaries

This section offers a concise explanation of the mathematical concepts that form the foundation of the classical Curve448 and Ed448 key exchange and digital signature algorithms. We discuss the mathematical base of PQ Crystals-Kyber and Crystals-Dilithium algorithms. Finally, we present the X.509 PKI structure denoting the required changes for hybrid PQ transition.

2.1 ECC Mathematical Background

Elliptic Curve cryptography stands as one of the most optimal asymmetric key encryption scheme due to the compact key sizes and minimal computation la-

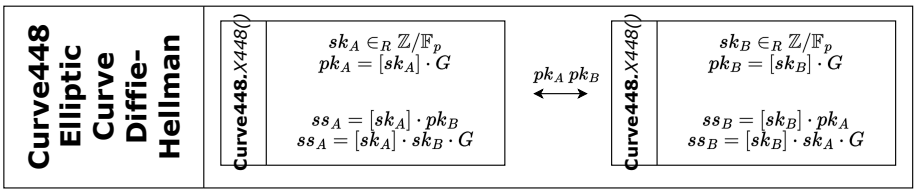

Fig. 1. X448 algorithm. G represents the value of the base point

tency, converting it in suitable scheme in scenarios of limited bandwidth or processing power, which is often the case of low-end embedded devices.

Ed448-Goldilocks Edwards curve is denoted by the equation:

$$E_{Ed}/\mathbb{F}_p : ax^2 + y^2 = 1 + dx^2y^2$$

Given that $d = -39081$ and $a = 1$ the curve operations for cryptographic purposes are defined over a finite field denoted as $\mathbb{F}_p$ where p is equal to $2^{448} - 2^{224} - 1$. The curve elements of Curve448 are represented by coordinate pairs $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$. A birational map exists to project a point from Edwards curve representation to the Montgomery curve representation.

$$(x, y) = (sqrt(156324) * u/v, (1 + u)/(1 - u))$$

The use of Montgomery curves representation often guarantees an ideal design for implementation, since it allows for efficient execution of group operations. The fundamental operation of ECC involves performing integer-point multiplications such that $P = k \cdot Q$ results in point Q being added to itself k times. Among the various point multiplication techniques is the so-called Montgomery Ladder, which ensures execution latency benefits based on the unified point doubling and addition formula, in addition to the constant time execution ensuring Simple Power Analysis (SPA) resistance, and the well-defined Differential Power Analysis countermeasure integration techniques. Furthermore, Montgomery ladder provides $X-$only coordinate operations when curve elements are presented in projective representation with three coordinates (X, Y, Z). The affine coordinates are retrieved at the end the of Montgomery Ladder execution as $(x, y) = (\frac{X}{Z}, \frac{Y}{Z})$.

ECC provides resilience against classic computer adversaries because to the challenging nature of solving the Elliptic Curve Discrete Logarithm problem. It is employed for both key exchange and authentication, making it a desirable choice in cryptographic network protocols like TLSv1.3, which is the main subject of this study.

2.2 X448

The implementation of key agreement with Curve448 is achieved by the Elliptic Curve Diffie-Hellman-like algorithm (ECDH). Similar to other techniques based

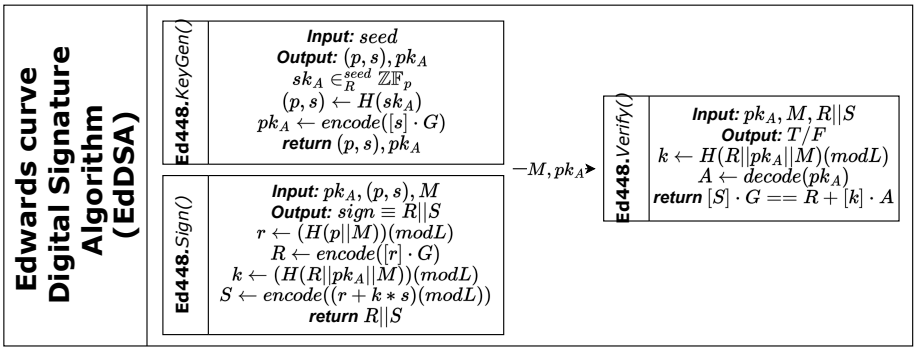

Fig. 2. Ed448 algorithm [38]. H denotes $SHAKE256$. L represents the order of Ed448 curve. G represents the value of the base point

on elliptic curve cryptography (ECC), Curve448 depends on performing scalar-point multiplication.

During the execution of the Elliptic Curve Diffie-Hellman (ECDH) protocol, both parties involved generate a scalar secret key sk. They then perform a scalar-point multiplication operation, known as `X448` for Curve448, using a parameter base point G. This operation results in a new point on the curve, which is used as their public key information pk as $pk = sk \cdot G$. The parties exchange the calculated public keys. Both parties currently possess their own private keys as well as the public key of the other party. By utilizing the obtained public key and their own secret key, each participant engages in an additional scalar-point multiplication operation `X448`. As a consequence, a shared secret value ss is obtained, which is denoted by a point on the elliptic curve. The ECDH technique based on Curve448 is visually represented in Figure 1, with the subindex value A, B indicating the computation parties as Alice and Bob.

Alice and Bob apply a Key Derivation Function (KDF) to extract a symmetric key value from their shared secret. This enables them to transition to a computationally efficient symmetric encryption method, guaranteeing that their data flow is secured and safeguarded from eavesdropping.

2.3 Ed448

The Digital Signature Algorithm (DSA) enables the recipient of a message to verify the sender's identity (authentication), guarantees the integrity of the data by preventing any unauthorized modifications during transmission (data integrity), and eliminates the sender's ability to deny delivering the message (non-repudiation). In order to uphold these cryptographic principles, one can employ ECC techniques, which rely on either EC Digital Signature Algorithms (ECDSA) or Edwards Curve Digital Signature Algorithms (EdDSA), depending on the specific elliptic curve being deployed.

Ed448 is an EdDSA method that utilizes the scalar-point multiplication operation `X448`, similar to the ECDH algorithm based on Curve448. In order to deal with the arbitrary length of messages conveyed across the Internet, extra hashing methods are employed to create a fixed and concise message digest. Ed448 utilizes the recently developed Secure Hash Algorithms 3 (SHA-3) hashing algorithm, specifically employing the eXtendable Output algorithm (XOF) SHAKE256 instantiation.

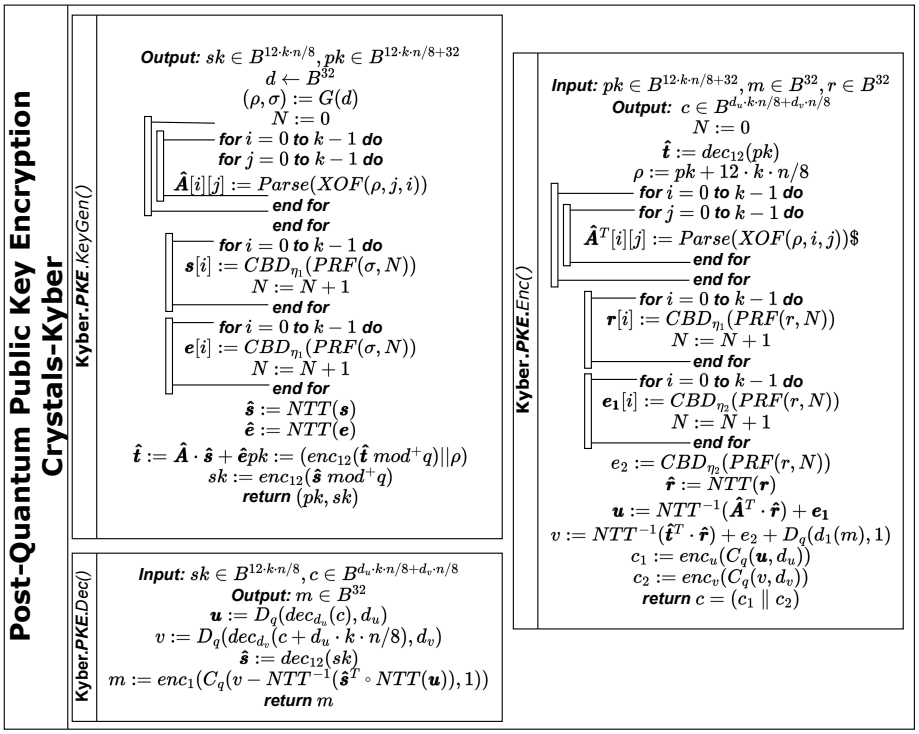

Fig. 3. Crystals-Kyber algorithm [39]. Each variable represents (the coefficients of) a polynomial, bold text style denotes vector of polynomials, capital letter notation denotes a matrix. *enc* and *dec* represents encode/decode, C and D present Compress/Decompress, respectively

The signature entity performs key generation and signing functions using the two fundamental operations of point multiplication and hashing, as illustrated in Figure 2. Like Curve448 ECDH, a key pair (sk, pk) is created, where the secret key value is then utilized to acquire the signature of the message M, $R||S$. After receiving the message and the signature, the recipient can authenticate the sender's identity by utilizing the public key value. The verification function determines whether to accept or reject the signature on the message value.

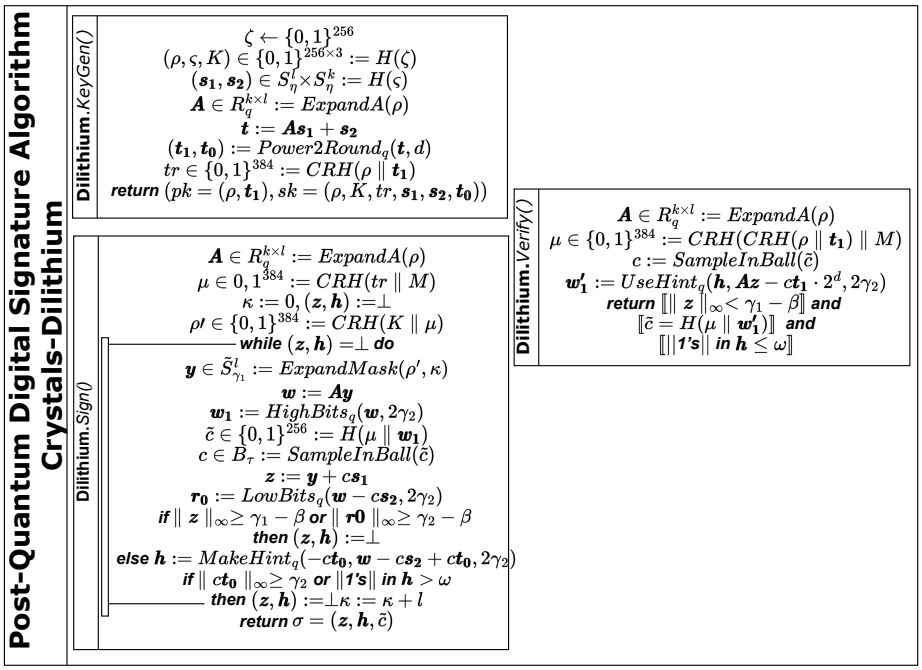

Fig. 4. CRYSTALS-Dilithium algorithm [40]. Each variable represents a polynomial, bold text style denotes vector of polynomials, capital letter notation denotes a matrix

Within the realm of network protocols, digital signature algorithms play a vital role in verifying the identity of communication parties and are an integral component of the Public Key Infrastructure, namely in certificate verification procedures. The integration of elliptic curve algorithms into TLSv1.3 is gaining popularity, although it is vulnerable to attacks in the age of quantum computing. This necessitates the switch to post-quantum key agreement and digital signature techniques.

2.4 Lattices Mathematical Background

Post Quantum Cryptographic Algorithms promise resistance against an adversary with a quantum computing power. Based on complex mathematical problems, PQ schemes promise to upgrade the cryptographic strength of the security network protocols. Among different families of PQ primitives, such as Code-, Hash-, Multivariate-, and Isogeny-based schemes, Lattice-based cryptographic primitive ensure *relatively* compact key sizes, compared to other PQ schemes, with the main advantage the limited computational requirements. Lattices are used to define both - public key encryption schemes and digital signature algorithms, relying on the Shortest Vector Problem (SVP), Closest Vector Problem (CVP) and Learning With Errors (LWE) (and it variations such as

Ring-Learning With Errors (RLWE), Module-Learning With Errors (MLWE), Learning with Rounding (LWR), etc.), believed to be resistant against quantum adversary.

In 2022 NIST has announced the finalists of the PQ standardization process, where the only Key Encapsulation Mechanics *to-be-standardized* is the lattice-based Crystals-Kyber and among three PQ DSA finalists, two (Crystals-Dilithium and Falcon) are lattice-based. CRYSTALS-{Kyber, Dilithium} CRYptographic SuiTe for Algebraic LatticeS (CRYSTALS) schemes rely on the difficulty (MLWE problem) differentiating $(A, As_1 + s_2)$ with $A \in Z_q^{n \times l}$, $s_1 \in Z_q^l$, and $s_2 \in Z_q^n$ from (A, b) with uniformly chosen value b. Along with the Shortest Integer Solution (SIS) that lattices pose, consisting of finding a non-trivial value x such that $A \cdot x = 0$, the PQ KEM Kyber and PQ DSA Dilithium are created.

Government and industry security standards do not support the inclusion of standalone PQ primitives into network protocols, considering their widespread usage by billions of users every day. Therefore, a hybrid mode of operation is necessary for a smooth transition to post-quantum robustness.

2.5 Crystals-Kyber

Crystals-Kyber Key Encapsulation Mechanism was presented in 2018 in [39]. As the rest of PQ KEMs Kyber relies on $INC - CPA$ Public Key Encryption (PKE) scheme wrapped by the (variant of the) Fujisaki-Okamoto (FO) transform. The Kyber.PKE method consists of key generation, encryption and decryption, represented as Kyber.PKE.KeyGen(), Kyber.PKE.En(), and Kyber.PKE.Dec() in Figure 3, respectively. The Key Encapsulation Mechanism (KEM) algorithm wraps this functions in key generation, encapsulation and decapsulation, via some additional hash and XOF functions, in order to provide $IND - CCA2$ security of the underlying scheme.

Crystals-Kyber is instantiated with different set of parameters to offer distinct levels of security. Specifically, Kyber512, Kyber768, and Kyber1024 correspond to NIST Security Level 2, 3, and 5, respectively. Based on the security level of Curve448 and Ed448, providing 224-bit security, and the nature of the PQ primitives lacking trust, we consider the highest security level, in particular, Kyber1024, to integrate in the TLSv1.3 protocol in the scope of this work. We should note that Crystals-Kyber768 is the recommended security level to be used. The least recommended security level to be utilized is Crystals-Kyber768. Despite its high security level, Kyber1024 remains appealing due to its low latency, which is equivalent to that provided by traditional cryptographic public key methods.

2.6 Crystals-Dilithium

The Crystals-Dilithium lattice-based DSA method is derived from the mathematical issue of Module Learning With Error (MLWE), similar to the Crystals-Kyber algorithm. It was suggested in the publication by *Ducas et al.* [40] and has been selected as one of the three DSA finalists for the NIST PQ standardization process. Like Kyber, this system provides many levels of security. In our study, we specifically concentrate on Dilithium5, which guarantees a high level

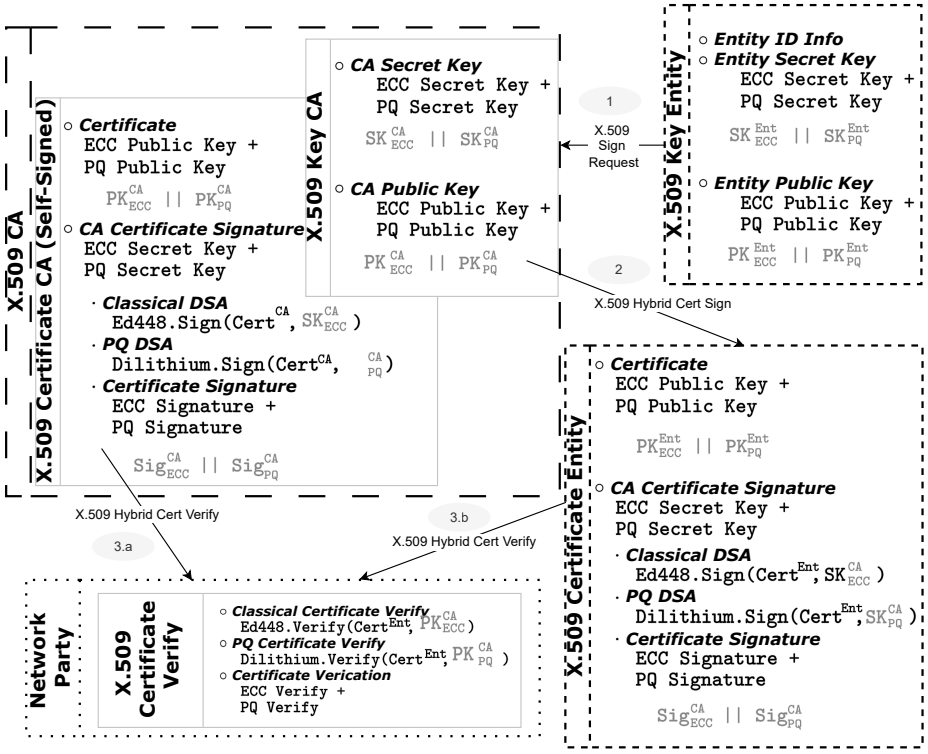

Fig. 5. The Public Key Infrastructure (PKI) built using classical (Ed448) and post-quantum (Dilithium5) Digital Signature Algorithm (DSA) techniques. The gray data refers to the information fields found in the X.509 files. Superscript indicates the owner of the data, while subscript indicates the type of information

of security comparable to Ed448. The specifics of the PQ signature algorithm are presented in Figure 4, where the details about the three underlying functions Dilithium.KeyGen(), Dilithium.Sign(), and Dilithium.Verify() are presented.

The application of DSA in the scope of TLSv1.3 involves the signing of a TLSv1.3 handshake message for server authentication and its incorporation into the PKI framework, where trusted authorities issue signatures to verify the authenticity of information for a specific entity. This work specifically addresses both aspects.

3 Hybrid Network Protocol Deployment

This section provides a comprehensive explanation of the TLSv1.3 network protocols and the underlying X.509 Public Key Infrastructure architecture. We

represent the primary execution points using graphical representation, highlighting the changes that were made to enhance the protocol's execution mode. These modifications involve incorporating Curve448 and Kyber1024 for key exchange, as well as Ed448 and Dilithium5 for message signature and PKI certificate validation.

3.1 PQ X.509

The Public Key Infrastructure (PKI) architecture is employed to guarantee the authentication of communication participants in the network through a reliable organization known as a Certificate Authority (CA). Public certificates are issued and confirmed using X.509 standards via digital signature algorithms. Typically, CA signatures rely on classical DSA algorithms. In the era of quantum computing, the verification of an entity's identification by a trusted third party can be easily forged in the presence of a quantum adversary.

The entirely hybrid TLSv1.3 model offers security based on both classical and post quantum algorithms. Making an abrupt move to PQ-only protocol implementation would jeopardize network security due to the relatively recent application of the PQ algorithms in technology. The continued employment of traditional cryptographic techniques poses a concern due to the rapid advancements in quantum computing technology and the expanding computational capabilities of these machines. Thus, a hybrid model, secures network traffic relaying on two independent categories of cryptographic algorithms, offering robustness against different adversaries. Among other works, [31], go into the details of the security and performance implications of the hybrid execution model of widely deployed network protocols, and define the motivation behind hybrid operation mode as guaranteeing the security of the system as long as one of the underlying cryptographic algorithms remains uncompromised.

As shown in Figure 5, a Certificate Authority (CA) , denoted as X.509 CA, owns a key file that contains both secret and public key values. By employing the confidential key data, the CA generates a signature, first, for its own certificate. The certificate includes the public key information of the CA for the purpose of validating issued signatures. It is important to note that real-world scenarios frequently involve a chain of CA certificates, which is not addressed in this study. After obtaining its own key and certificate files, the CA proceeds to distribute the certificate to third-parties for further verification reasons.

We outline, in Figure 5, the sequential steps involved in acquiring a validated entity certificate. After an entity creates a key file, it sends identifying information and the public key as a Signature Request to the CA for validation (1. X.509 Sign Request in Figure 5) and verification of its identity. The CA verifies the data of the entity and affixes its signature to the information, therefore generating a certificate for the specified entity (2. X.509 (Hybrid) Cert Sign in Figure 5).

Finally, when any communication network party initializes a connection with the given entity, the certificate information is used (3.a X.509 (Hybrid) Cert

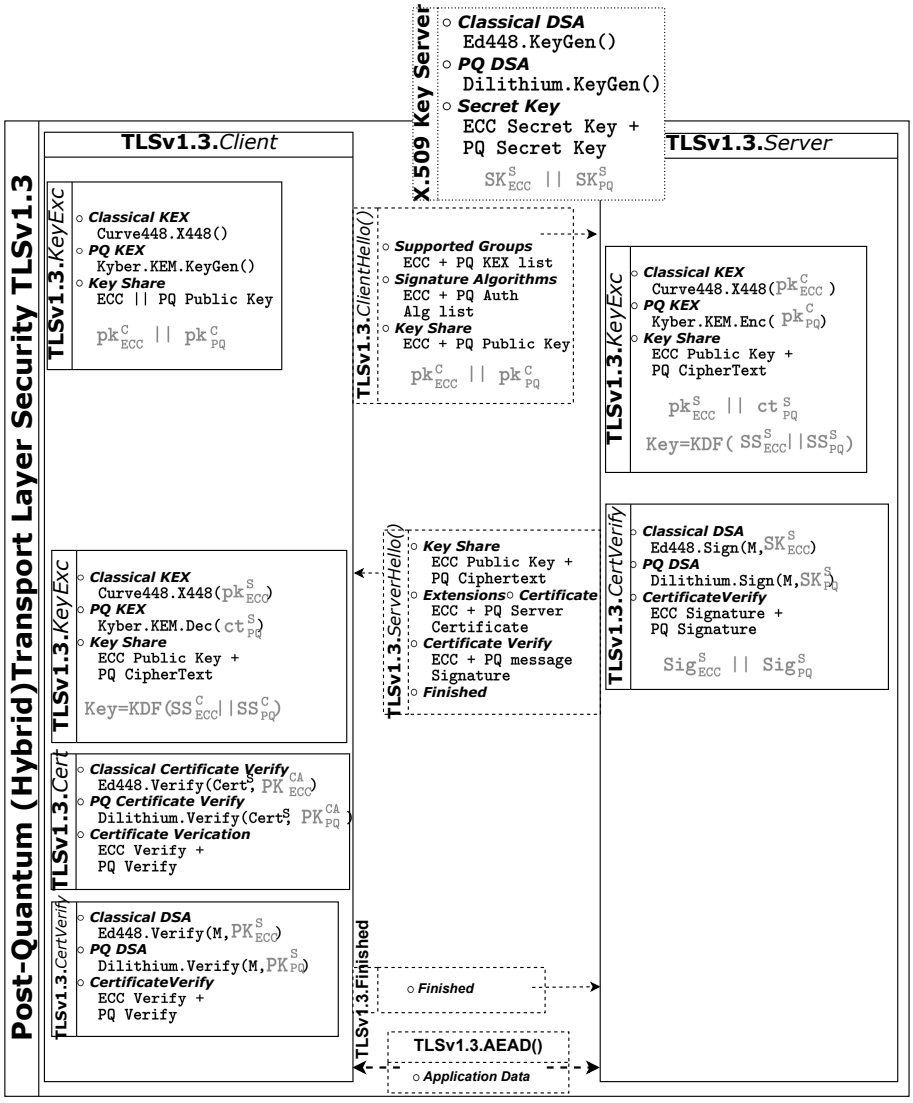

Fig. 6. TLSv1.3 execution flow graphical representation. Gray data refers to the information fields included in X.509 files, where superscript indicates the owner of the data and subscript indicates the type of information. The compute stages are represented by solid box lines, the message flow is represented by discontinuous lines, and the certificate file is represented by scattered box lines

Verify in Figure 5). To verify the identity of the entity, the public key, part of the CA certificate, is being used (3.b X.509 (Hybrid) Cert Verify in Figure 5).

The implementation of Public Key Infrastructure (PKI) includes the use of Digital Signature Algorithm (DSA) processes, which are susceptible to attacks

from adversaries with quantum computing capabilities. This study proposes a way of applying a PQ DSA algorithm, in addition to the conventional approach, to guarantee the system's reliability.

This paper introduces a novel hybrid Public Key Infrastructure (PKI) architecture that combines the use of Ed448 with Dilithium5 DSA. We utilized the OpenSSL cryptography library to generate hybrid PEM key files. We combined the secret and private key values by concatenating them. While several methods exist for representing data in the information fields, our primary focus was on the functional aspects and effectiveness of the system. Implementing more modifications to the data field placements may be effortlessly done, yet, it falls outside the scope of this project.

The entity owner of a key PEM file, which consists of a classical and PQ secret keys followed by a classical and a PQ key public keys, uses its information fields to generate digital signatures. In the context of a Certification Authority (CA), the confidential key value is utilized to authorize the issuance of certificates for other participants within the network. In the event of another entity, the secret key values are utilized to generate hybrid signatures. This includes both classical and PQ signatures of, for instance, the TLSv1.3 message `CertificateVerify`, as explained in the subsequent section.

The public key information is stored within the certificate PEM files. In this context, a certificate includes both classical and PQ public key values. These values are then utilized by a recipient party that is interested in verifying the authenticity of the transmitting entity. The verification procedure relies on both classical and PQ signature systems, analogous to the signature generation. Both signatures of the message are being transmitted simultaneously, with the lower bytes representing the classical signature and the upper bytes representing the PQ signature value.

Lastly, in the certificate signature field, two separate signatures are generated and saved by a trusted third party (CA). The CA utilizes classic secret key data to produce a classical signature value. Subsequently, the PQ key value is employed to generate a PQ signature. By utilizing the CA certificate's keys, which are often integrated into the communication parties' systems, the authenticity of both signatures on any certificate, issued by the specified CA, is confirmed, therefore confirming the identities of the communication parties.

Implementing a hybrid architecture mode for the PKI is a complex task due to the large number of files being created and the diverse functionalities required to process these files and extract their value. This work introduces the first version of hybrid Ed448 and Dilithium5 PEM keys and certificates. The creation and processing of these keys and certificates are performed using the OpenSSL general crypto library and the wolfSSL embedded device-specific library.

3.2 PQ TLSv1.3

TLSv1.3 guarantees a secure connection setup with a single roundtrip communications. The client and server establish a shared secret using key agreement

Table 1. Performance of the entirely hybrid TLSv1.3 handshake and the overall TLSv1.3 protocol when a short 15B message is delivered between communication parties. The values are expressed in terms of clock cycles [CC]

Work	KEX	Auth	Cert Verify	TLS1.3 handshake	TLS1.3 with AEAD
wolfSSL [41]	X448	Ed448	Ed448	-	44,358,855
Anastasova et al. [17]	X448	Ed448	Ed448	-	46,310,749
This work	X448 & Kyber1024	Ed448 & Dil5	-	97,624,103	106,735,300
	X448 & Kyber1024	Ed448 & Dil5	Ed448 & Dil5	114,017,313	123,139,034

cryptographic mechanisms like ECDH or PQ KEMs. Both communication parties utilize a key derivation function (KDF) to create a symmetric key value. This key value is then utilized to encrypt their application data traffic using an Authenticated Encryption with Additional Data (AEAD) cipher.

The simplified graphical representation of TLSv1.3 is depicted in Figure 6. The server and client exchange their respective certificate files, which contain a signature created by a trusted third party (CA) to authenticate the certificate's legitimacy. After obtaining the certificate, the client confirms its genuineness by verifying the signature using the public key information of the CA, which is incorporated on the client's side. The server's authentication step entails creating a signature of the message. After receiving the message, the user authenticates the signature by utilizing the server's public key value acquired from the server's certificate. Ultimately, the server sends an HMAC (Hash-based Message Authentication Code) of the entire message using the predetermined symmetric key value. Upon the successful completion of the TLSv1.3 protocol handshake, both communicating entities have the ability to securely transmit data across the established channel.

Within the present work, we enhance the design of the TLSv1.3 network protocol by integrating fully hybrid version of the protocol. For key exchange, we use the Curve448 ECDH method in conjunction with the Kyber1024 PQ scheme. Specifically, as indicated in Figure 6, the cipher key for TLSv1.3 is obtained by utilizing the shared secret information from both Curve448 and Kyber1024. The session data is produced by combining the classical and PQ values and utilizing them as input to a Key Derivation Function (KDF).

In order to carry out message signature, we utilize a hybrid technique by using both Ed448 and Dilithium5. Like the Key Exchange (KEX) methods, both the traditional and PQ DSA are performed simultaneously. The server's PEM key file has a classical secret key value, which is stored at the most significant bytes of the secret key field. This value is utilized to generate a classical signature. The PQ signature is generated using the least significant bytes from the secret key field, which contains the PQ secret key data. After generating both the classical and PQ signatures, they are combined and sent as a component of the Certificate Verify message.

The hybrid Public Key Infrastructure (PKI), which is a component of TLSv1.3, is also included in this project. In this work, the PEM key and certificate fields have been altered to include the classical and Post-Quantum (PQ) values, as explained in the previous section. Our hybrid TLSv1.3 architecture now includes the complete integration of the hybrid PKI.

4 Performance Evaluation

The next section examines the obtained results in relation to performance. We provide information on the latency of our design when it is run on the STM32F413 discovery board. This board is equipped with a Cortex-M4 CPU and is built on the ARMv7 architecture, which has been selected by NIST for evaluating post-quantum primitives on low-end embedded devices. We execute our experiments at a frequency of 76.6MHz, simulating a real-world scenario. The findings are presented in terms of clock cycles. Multiple scenarios are considered in relation to verification processes. We provide the complete implementation of the TLSv1.3 protocol, including the exchange of brief messages between the client and server. Additionally, we demonstrate the independent execution of the TLSv1.3 handshake, showcasing the modifications made throughout this project.

The generation of the X.509 key and certificate files is based on modification deployed on the OpenSSL cryptographic library. Since keys and certificates are being generated outside the scope of the TLSv1.3 protocol, we do not report performance results. However, it is important to note, that for the X.509 key generation and certificate verification (signature), again a hybrid approach involving Ed448 and Dilithium5 was used.

We report the performance of TLSv1.3 protocol after integrating Curve448 and Crystal-Kyber, and Ed448 and Crystals-Dilithium for key generation, entity authentication and certificate verification. The client computes ECC key generation and PQ KEM key generation and decapsulation routines in order to derive a session key with the server. On the server side, the X.509 hybrid certificate is being transmitted to the client along with a signature over the entire footprint of the TLSv1.3 message value. The client uses the CA public key value to verify the validity of the server certificate, thus executes hybrid signature verification.

The communication parties transmit data through a UART serial connection based on 115200bps transmission speed. It is important to note that, based on the large sizes of the transmitted certificate values, the performance results show significant drop. However, the communication latency forms a large part of the protocol execution time in a real-world scenario where data is transmitted all over the worlds and should not be neglected when evaluating the impact of PQ protocol transition.

We report around 114 million clock cycles for the execution of the fully hybrid TLSv1.3 handshake based on Curve448 and Kyber1024 and Ed4448 and Dilithium5 cryptographic primitives. The implementation of the whole TLSv1.3 protocol, including the transmission of a brief message encrypted using an AEAD cipher, leads to an additional computational cost of around 20.3 million clock cycles. By excluding the verification of the server certificate, which is based

on the CA signature, we observe an approximate improvement of 17.5% and 16% for the TLSv1.3 handshake and the fully hybrid TLSv1.3 handshake with AEAD encrypted message transmission, respectively. However, it is important to note that this scenario is not typical in real-world network communication. Therefore, our focus is on the statistics related to the complete execution of the TLSv1.3 protocol on the STM32F413 board. Enabling TLSv1.3 in entirely hybrid mode leads to a $\times 2.77$ the execution of the original wolfSSL classical-only implementation and $\times 2.67$ the latency of the optimal and side-channel robust Curve448 and Ed448 design as part of wolfSSL [17].

5 Conclusion

This work introduces the initial fully hybrid operating mode of the widely used TLSv1.3 network security protocol. The mode is based on Curve448 and Crystals-Kyber for key exchange, and Ed448 and Crystals-Dilithium for the digital signature method. Within our approach, the client and server engage in the exchange of information, encompassing public key data for both classical and post-quantum (PQ) primitives. Our solution offers an architectural framework where the involved parties engage in message signing and verification using a combination of traditional and post-quantum methods. In addition, we offer hybrid Public Key Infrastructure (PKI) by making changes to the commonly used OpenSSL software. This allows us to create hybrid keys and certificates that comply with the X.509 standard. We add the ability to process the hybrid data within these keys and certificates, transforming the Certification Authority (CA) into a hybrid entity that possesses both classical and Post-Quantum (PQ) key values. To present performance results on the full hybrid TLSv1.3 protocol, we utilize the wolfSSL cryptography library specifically designed for embedded devices.

6 Acknowledgements

The authors would like to thank the reviewers for their comments. This work is supported by NSF 214796 grant.

References

1. E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, Aug. 2018.
2. 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.
3. T. N. I. of Standards and T. (NIST)., "Post-quantum cryptography standardization, 2017-2018.." Last accessed on May 20, 2021.
4. M. Hamburg, "Ed448-Goldilocks, a new elliptic curve," *Cryptology ePrint Archive*, 2015.
5. D. J. Bernstein, "Curve25519: New Diffie-Hellman speed records," in *International Workshop on Public Key Cryptography*, pp. 207–228, Springer, 2006.

6. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 124–142, Springer, 2011.
7. H. Seo, "Compact implementations of Curve Ed448 on low-end IoT platforms," *ETRI Journal*, vol. 41, no. 6, pp. 863–872, 2019.
8. A. Faz-Hernández, J. López, and R. Dahab, "High-performance implementation of elliptic curve cryptography using vector instructions," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 3, pp. 1–35, 2019.
9. H. Seo and R. Azarderakhsh, "Curve448 on 32-bit ARM Cortex-M4," in *International Conference on Information Security and Cryptology*, pp. 125–139, Springer, 2020.
10. M. Anastasova, M. Bisheh-Niasar, H. Seo, R. Azarderakhsh, and M. M. Kermani, "Efficient and Side-Channel Resistant Design of High-Security Ed448 on ARM Cortex-M4," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 93–96, IEEE, 2022.
11. M. Anastasova, R. Azarderakhsh, M. M. Kermani, and L. Beshaj, "Time-Efficient Finite Field Microarchitecture Design for Curve448 and Ed448 on Cortex-M4," in *International Conference on Information Security and Cryptology*, pp. 292–314, Springer, 2022.
12. M. Bisheh-Niasar, M. Anastasova, A. Abdulgadir, H. Seo, and R. Azarderakhsh, "Side-Channel Analysis and Countermeasure Design for Implementation of Curve448 on Cortex-M4," in *Proceedings of the 11th International Workshop on Hardware and Architectural Support for Security and Privacy*, pp. 10–17, 2022.
13. I. Blake, G. Seroussi, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, vol. 265. Cambridge university press, 1999.
14. M. Hamburg, "Fast and compact elliptic-curve cryptography," *Cryptology ePrint Archive*, 2012.
15. N. Meloni, "New point addition formulae for ECC applications," in *International Workshop on the Arithmetic of Finite Fields*, pp. 189–201, Springer, 2007.
16. P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of computation*, vol. 48, no. 177, pp. 243–264, 1987.
17. M. Anastasova, R. El Khatib, A. Laclaustra, R. Azarderakhsh, and M. M. Kermani, "Highly Optimized Curve448 and Ed448 design in wolfSSL and Side-Channel Evaluation on Cortex-M4," in *2023 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–8, IEEE, 2023.
18. H. Becker, V. Hwang, M. J. Kannwischer, B.-Y. Yang, and S.-Y. Yang, "Neon ntt: Faster dilithium, kyber, and saber on cortex-a72 and apple m1," *Cryptology ePrint Archive*, 2021.
19. D. T. Nguyen and K. Gaj, "Optimized software implementations of CRYSTALS-Kyber, NTRU, and Saber using NEON-based special instructions of ARMv8," in *Proceedings of the NIST 3rd PQC Standardization Conference (NIST PQC 2021)*, 2021.
20. L. Zhao, J. Zhang, J. Huang, Z. Liu, and G. Hancke, "Efficient implementation of kyber on mobile devices," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 506–513, IEEE, 2021.
21. Y. Kim, J. Song, T.-Y. Youn, S. C. Seo, *et al.*, "Crystals-dilithium on armv8," *Security and Communication Networks*, vol. 2022, 2022.
22. J. Zheng, F. He, S. Shen, C. Xue, and Y. Zhao, "Parallel Small Polynomial Multiplication for Dilithium: A Faster Design and Implementation," in *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 304–317, 2022.

23. G. Seiler, "Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography," *Cryptology ePrint Archive*, 2018.
24. J. Zheng, H. Zhu, Z. Song, Z. Wang, and Y. Zhao, "Optimized Vectorization Implementation of CRYSTALS-Dilithium," *arXiv preprint arXiv:2306.01989*, 2023.
25. J. Wright, M. Gowanlock, C. Philabaum, and B. Cambou, "A crystals-dilithium response-based cryptography engine using gpgpu," in *Proceedings of the Future Technologies Conference*, pp. 32–45, Springer, 2021.
26. X. Zhao, B. Wang, Z. Zhao, Q. Qu, and L. Wang, "Highly efficient parallel design of Dilithium on GPUs," 2022.
27. S. Shen, H. Yang, W. Dai, H. Zhang, Z. Liu, and Y. Zhao, "High-throughput gpu implementation of dilithium post-quantum digital signature," *arXiv preprint arXiv:2211.12265*, 2022.
28. L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of Kyber on Cortex-M4," in *Progress in Cryptology–AFRICACRYPT 2019: 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9–11, 2019, Proceedings 11*, pp. 209–228, Springer, 2019.
29. E. Alkim, Y. A. Bilgin, M. Cenk, and F. Gérard, "Cortex-M4 optimizations for {R, M} LWE schemes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 336–357, 2020.
30. A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster kyber and dilithium on the cortex-M4," in *International Conference on Applied Cryptography and Network Security*, pp. 853–871, Springer, 2022.
31. E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH," *Cryptology ePrint Archive*, 2019.
32. M. Anastasova, P. Kampanakis, and J. Massimo, "PQ-HPKE: post-quantum hybrid public key encryption," *Cryptology ePrint Archive*, 2022.
33. P. Kampanakis, P. Panburana, E. Daw, and D. Van Geest, "The viability of post-quantum X. 509 certificates," *Cryptology ePrint Archive*, 2018.
34. M. Campagna and E. Crockett, "Hybrid post-quantum key encapsulation methods (PQ KEM) for transport layer security 1.2 (TLS)," *Internet Engineering Task Force, Internet-Draft draft-campagna-tls-bike-sike-hybrid*, vol. 1, 2019.
35. D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-quantum authentication in TLS 1.3: a performance study," *Cryptology ePrint Archive*, 2020.
36. D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pp. 149–156, 2020.
37. D. Marchsreiter and J. Sepúlveda, "Hybrid Post-Quantum Enhanced TLS 1.3 on Embedded Devices," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, pp. 905–912, IEEE, 2022.
38. S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)." RFC 8032, Jan. 2017.
39. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–367, IEEE, 2018.
40. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018.
41. wolfSSL, "wolfSSL." Last accessed on Jan 23, 2023 from https://www.wolfssl.com/.