IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 June 2025, accepted 4 July 2025, date of publication 21 July 2025, date of current version 30 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3590863

RESEARCH ARTICLE

Subliminal Channels in CRYSTALS-Kyber Key-Encapsulation Mechanism and Their Use in Quantum-Resistant TLS Protocols

ROBERTO ROMÁN, ROSARIO ARJONA, AND ILUMINADA BATURONE
Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC, Universidad de Sevilla, 41092 Seville, Spain

Corresponding author: Roberto Román (rrhajderek@us.es)

This work has been funded by Grants PDC2023–145873-I00, CPP2022–009796, and PID2023-150809OB-I00 funded by Ministerio de Ciencia, Innovación y Universidades (MICIU)/Agencia Estatal de Investigación (AEI)/10.13039/501100011033 and by the European Union (EU)–NextGenerationEU/Plan de Recuperación, Transformación y Resiliencia (PRTR); it has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101168311 (LICORICE Project), and it has been funded by grant USECHIP (TSI-069100-2023-001), project funded by the Secretary of State for Telecommunications and Digital Infrastructure, Ministry for Digital Transformation and Civil Service and by the European Union–Next GenerationEU/PRTR. The work of Roberto Romàn was supported by VI Plan Propio de Investigación y Transferencia, Universidad de Sevilla.

ABSTRACT Cryptographic protocols can be used to covertly exchange information without arousing suspicion. The covert channels created in this way are called subliminal channels. In this work, three different subliminal channels using CRYSTALS-Kyber are discovered. Kyber is employed in the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) standard published in FIPS 203. In the found subliminal channels, the covert message is embedded in the random data needed by the encapsulation or the key generation algorithms. Two settings are differentiated depending on if the covert receiver is an overt entity executing the key exchange protocol or a man-in-the-middle. An important feature achieved by the proposed subliminal channels is plausible deniability. Covert entities can convince a jury that they did not use a subverted version of the original Kyber algorithms by showing the random data used in the protocol, the values exchanged, and the outputs of the algorithms. The proposed subliminal channels can be used in quantum-resistant proposals of TLS (Transport Layer Security). Concretely, this work explores the use of the proposed subliminal channels in PQTLS and KEMTLS. Also, some countermeasures are proposed in the paper. Experimental results show that the overhead in execution times is not significant and that from 2 to 34 bytes of covered information can be transmitted per TLS handshake.

INDEX TERMS Subliminal channel, covert channel, TLS, post-quantum cryptography, CRYSTALS-Kyber, key encapsulation mechanism.

I. INTRODUCTION

Covert channels, which were introduced by Lampson in [1], are hidden channels that enable parties in a communication to exchange information in a way not intended to do so. The normal communication is exchanged between the Overt Sender (OS) and the Overt Receiver (OR). The Covert Sender (CS) and Covert Receiver (CR) are the parties that use this communication to exchange covert information. Overt and covert senders are typically assumed to be the same entity, but they can also be different depending on the situation. The same applies to overt and covert receivers. The entity that tries to uncover the hidden channel between the covert sender and the covert receiver is called the Warden (W) [2]. Covert channels have several applications, from the circumvention of censorship in restrictive communication scenarios to their use by malware to bypass Intrusion Detection Systems (IDSs) in controlled environments or data exfiltration [3].

Simmons discussed the need of covert channels through the prisoners' problem [4]. In this problem, two prisoners, Alice and Bob, want to establish a communication in an undetectable way. They can only exchange messages

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

through Walter, the Warden. Thus, Walter can read all the communications. Later, Simmons [5] introduced the notion of subliminal channels as a type of covert channel that exploits cryptographic primitives by modifying their original specifications to transmit covert information. Simmons showed that information can be embedded in the digital signatures without affecting their verification process.

A relevant feature of covert and subliminal channels is the achievement of *plausible deniability*. Deniability allows an entity to deny its participation in the execution of a particular protocol. The party to whom the entity denies its participation is called the Jury. In practice, deniability is a very subjective attribute [6]. A known way to transmit covert information is to subvert the algorithm used, i.e., to replace the honest implementation with another [7], [8], [9], [10], [11]. Thus, the jury may suspect that the parties used flawed algorithms to exchange covert information. Taking this into account, we estimate in this paper that plausible deniability is achieved if the jury can be convinced that the algorithm used to generate the sent data is the expected one and not a subverted version.

Subliminal channels are found very often in digital signatures [12], [13]. The work in [12] uses the ECDSA (Elliptic Curve Digital Signature Algorithm), and the work in [13] uses the high-speed digital signature EdDSA (Edwards-curve Digital Signature Algorithm). Since the field of quantum computing is evolving and quantum computers are a known threat to classical digital signatures that rely on the difficulty of factoring large integers and computing discrete logarithms, those primitives are being substituted by quantum-resistant ones [14], [15], [16].

The algorithms presented to the NIST post-quantum standardization process are divided into digital signature primitives on one side, and public-key encryption primitives and key-establishment algorithms on the other side. Among the latter, CRYSTALS-Kyber [17] was the only one selected for standardization [18], [19]. In the following, CRYSTALS-Kyber is named for simplicity as Kyber. Kyber is a Key-Encapsulation Mechanism (KEM) based on lattices and the hardness of the Module-Learning with Errors (MLWE) problem. It is employed in the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) standard published in FIPS 203 [20]. A KEM is a particular type of key-establishment scheme, which is used to establish a shared secret key between two parties communicating over a public channel. Once the shared key is established, it is used to encrypt the data communicated over the channel. While subliminal channels have been proposed in post-quantum digital signatures [21], no subliminal channels have been proposed in KEMs to communicate covert information, to the best of the authors' knowledge.

Digital signatures and key-establishment schemes are used in the Transport Layer Security (TLS) protocol, which is a cryptographic protocol extensively used to secure communications over the Internet, currently without using quantum-resistant primitives [22]. TLS provides confidentiality, integrity, and authentication, generally of a server to a client by means of a handshake at the beginning of the communication. Since it is a widely used protocol nowadays, several covert and subliminal channels have been proposed on it [23], [24]. As currently all versions of TLS use non-quantum-resistant primitives, candidate protocols are studied for their replacement using quantum-resistant ones. The proposals use a key-establishment scheme to agree on an ephemeral set of secret keys, but they differ in the way in which the server is authenticated to the client. In the variant called PQTLS, the server is authenticated through a digital signature [25]. The variant called KEMTLS authenticates the server through a unilateral authenticated key exchange that employs the encapsulation and decapsulation algorithms found in a KEM [26].

This paper proposes three subliminal channels in Kyber (selected for standardization) that achieve plausible deniability and analyzes their use in quantum-resistant TLS protocols. The main contributions are the following:

- Three different subliminal channels in Kyber, not reported in the existing literature, are presented and studied. Two of them are found in the encapsulation algorithm and the other in the key generation algorithm. A first subliminal channel assumes that the covert and overt senders and receivers are the same entities, a second one assumes that the covert receiver is an entity in a man-in-the-middle position, and the third one can be used in the two settings.
- The subliminal channels presented achieve plausible deniability because a jury can be convinced that no subverted algorithms have been used.
- This is the first work that investigates the use of a given subliminal channel in the context of a quantum-safe TLS protocol. PQTLS and KEMTLS, which are quantum-safe versions of TLS found in the literature, are analyzed.

The paper is organized as follows. Section II discusses the state of the art on subliminal channels. In Section III, some preliminaries regarding Kyber are shown. Section IV presents the assumptions made. The subliminal channels discovered in Kyber are presented in Section V, and in Section VI the security in the classical random oracle model is shown. Their use in quantum-resistant TLS protocols is described in Section VII. Various features of the subliminal channels discovered are shown in Section VIII. Although the proposed subliminal channels are difficult to detect, some countermeasures are presented in Section IX. Experimental results in terms of execution times and the amount of covert information that can be transferred per TLS handshake are presented in Section X. Finally, Section XI concludes the paper.

II. RELATED WORK

The field of subliminal channels began with their application to digital signature schemes. Typically, subliminal channels have been divided into broadband channels and narrowband

channels. The former allows the use of almost all the bits of the signature that are not needed for security against forgery protection, while the latter uses only a few bits of the signature. The latter results in very small subliminal bandwidths. In [5], one broadband subliminal channel is proposed for the DSA digital signature scheme. In this work, a secret should be known by both the sender and the receiver. In [27], a broadband subliminal channel is proposed for the ElGamal signature scheme, called the Newton channel. In [28], several signature schemes are targeted for subliminal channels. Specifically, ESIGN-D, SFLASHv3, and a concrete case of RSA-PSS are proposed. In [29], subliminal channels are reported for the Hess's ID-based digital signature scheme [30]. However, most of these digital signature schemes are roughly used in practice today.

In [12], a broadband subliminal channel and a narrowband subliminal channel are proposed for the elliptic-curve-based digital signature EdDSA. Also, the use of the broadband subliminal channel is proposed in the TLS 1.3 protocol. The narrowband subliminal channel was not clearly studied and analyzed, for example, in terms of the amount of subliminal information that a covert sender could send.

In [31], a different paradigm is proposed where the randomness of the secret key is used to create a subliminal channel. The advantage of the proposal is that it could be applied to different types of digital signatures. However, it is not possible to send more than one covert message per secret key, and this limits its use to very concrete scenarios.

Nearly all the subliminal channels commented on above will be limited due to the ongoing migration of the currently used primitives to quantum-resistant ones. In [13], subliminal channels in quantum-secure MQ signatures are investigated. However, they do not achieve plausible deniability. Furthermore, it is currently unclear whether any MQ signature will be standardized in the future. Various subliminal channels are proposed in [21] in quantum-resistant digital signatures submitted to the NIST post-quantum cryptography competition. They propose subliminal channels that could be used in the near future. However, subliminal channels in digital signatures have limitations when the covert receiver is in a man-in-the-middle position since digital signatures are typically encrypted when they are sent, as happens to TLS. This requires an additional mechanism to leak the encryption key, which is a problem because it compromises the confidentiality of the communication between the overt parties.

The works in the literature that use subliminal channels in lattice-based KEMs do not use them to communicate covert information but to exfiltrate the shared secret key established with the KEM. In [32] and [33], it is shown how to modify the key generation algorithm to send the shared secret key as subliminal information. In [34], another way to create a subliminal channel in Kyber to leak the key is proposed. These works have two major drawbacks. First, from our perspective, they cannot achieve plausible deniability since the parties involved cannot prove to a jury that they did not use a subverted algorithm to generate the exchanged public key. Later, their methods cannot be used in a man-in-the-middle scenario without compromising the confidentiality of the communication. Ways to take advantage of quantum-safe KEMs to exchange covert information with a receiver in a man-in-the-middle position while maintaining the confidentiality of the communication have not been studied in the literature.

III. PRELIMINARIES

In this section, some preliminaries related to Kyber are presented, which are necessary for the understanding of the following sections. The standardized version found in [20] is used as a reference. Kyber (*KYBER.CCAKEM*) is a key-encapsulation mechanism that is resistant to Chosen Ciphertext Attacks (CCA) performed by quantum computers [35]. The security of Kyber is based on the hardness of the Module-Learning with Errors (MLWE) problem in the classical and quantum random oracle models. It applies a Fujisaki-Okamoto transformation into a public-key encryption (PKE) scheme secure against Chosen Plaintext Attacks (CPA), named Kyber PKE (*KYBER.CPAPKE*). As a detailed explanation of Kyber PKE is not necessary to understand the proposed subliminal channels of this paper, it will not be explained in this section. We refer the reader to [17] and [20] for a detailed explanation of Kyber PKE if desired.

For a set S, we write $s \leftarrow S$ to denote that s is chosen uniformly at random from S. $\|$ is the concatenation operation. H and J are functions that take a variable-length input value and return a fixed-length 256-bit output value as $H : \{0,1\}^* \rightarrow \{0,1\}^{256}$ and $J : \{0,1\}^* \rightarrow \{0,1\}^{256}$. G is a function that takes a variable-length input value and returns a fixed-length 512-output value as $G : \{0,1\}^* \rightarrow \{0,1\}^{2\times 256}$. *KYBER.CPAPKE.KeyGen*, *KYBER.CPAPKE.Enc* and *KYBER.CPAPKE.Dec* are Kyber PKE key generation, encryption and decryption algorithms of Kyber PKE, respectively. *KYBER.CPAPKE.KeyGen* takes the uniform random number d as input and returns the encryption and decryption keys ek_{PKE} and dk_{PKE}. *KYBER.CPAPKE.Enc* takes as input the encryption key ek_{PKE}, a message m and a random number r and generates a ciphertext c. *KYBER.CPAPKE.Dec* takes as inputs the decryption key dk_{PKE} and a ciphertext c and returns a decrypted message m.

Kyber is defined by key generation (*KYBER.CCAKEM .KeyGen*), encapsulation (*KYBER.CCAKEM.Encaps*), and decapsulation (*KYBER.CCAKEM.Decaps*) algorithms.

The key generation algorithm is shown in Algorithm 1. It returns a pair (ek, dk) consisting of an encapsulation key ek and a decapsulation key dk. This is done by generating two 256-bit uniform random numbers d and z. d is used as a source of randomness for *KYBER.CPAPKE.KeyGen*. ek is directly the encryption key ek_{PKE} returned by *KYBER.CPAPKE.KeyGen*. dk is the concatenation of the decryption key dk_{PKE} returned by

Algorithm 1 Kyber Key Generation Algorithm

KYBER.CCAKEM.KeyGen()

Inputs: -
Outputs: *encapsulation key ek*, *decapsulation key dk*
1. $d \leftarrow \{0,1\}^{256}$
2. $z \leftarrow \{0,1\}^{256}$
3. $(ek_{PKE}, dk_{PKE}) := KYBER.CPAPKE.KeyGen(d)$
4. $ek := ek_{PKE}$
5. $dk := (dk_{PKE} \parallel ek \parallel H(ek) \parallel z)$
6. ***return***(ek, dk)

KYBER.CPAPKE.KeyGen, ek, the hash of the encapsulation key $H(ek)$, and z.

The encapsulation algorithm is shown in Algorithm 2. It takes as input an encapsulation key ek, which is the same as the encryption key ek_{PKE}, and returns a shared secret key K and a ciphertext c. First, a 256-bit uniform random number m is generated. H is applied to ek and the result is concatenated with m. The result of this operation is used as an input to G, which returns K and a random value r. Finally, m is encrypted using the *KYBER.CPAPKE.Enc* function, and ek_{PKE} and r to generate c.

Algorithm 2 Kyber Encapsulation Algorithm

KYBER.CCAKEM.Encaps$(ek = ek_{PKE})$

Inputs: *encapsulation key ek*
Outputs: *ciphertext c*, *shared secret key K*
1. $m \leftarrow \{0,1\}^{256}$
2. $(K, r) := G(m \parallel H(ek))$
3. $c := KYBER.CPAPKE.Enc(ek_{PKE}, m, r)$
4. ***return***(K, c)

The decapsulation algorithm is shown in Algorithm 3. It takes as input a decapsulation key dk and a ciphertext c and returns a shared secret key K. It is recalled that the decapsulation key dk is formed by concatenating the decryption key dk_{PKE}, the encapsulation key ek, the hash of the encapsulation key $H(ek)$ and a random value z, and that the encapsulation key ek is the same as the encryption key ek_{PKE}. The first step is to decrypt c using dk_{PKE} to produce the decrypted message m'. K and a random number r' are generated as in the *KYBER.CCAKEM.Encaps* algorithm. m' is re-encrypted using ek_{PKE} and r'. If the returned ciphertext c' is the same as c, the shared secret key K from the previous step is returned. If not, it returns the result of applying the function J to the concatenation of z and c.

Figure 1 shows how Kyber algorithms are used in a protocol by two parties, P_1 and P_2, to share a secret key K.

IV. ASSUMPTIONS

We assume a situation where only quantum-resistant primitives are available due to the advent of quantum computers. Note that the migration to post-quantum cryptographic primitives is underway today [15].

Algorithm 3 Kyber Decapsulation Algorithm

KYBER.CCAKEM.Decaps$(dk = (dk_{PKE} \parallel ek \parallel H(ek) \parallel z), c)$

Inputs: *decapsulation key dk*, *ciphertext c*
Outputs: *shared secret key K*
1. $m' := KYBER.CPAPKE.Dec(dk_{PKE}, c)$
2. $(K, r') := G(m' \parallel H(ek))$
3. $\bar{K} := J(z \parallel c)$
4. $c' := KYBER.CPAPKE.Enc(ek, m', r')$
5. ***if*** $c \neq c'$ ***then***
6. $\quad K := \bar{K}$
7. **end if**
8. ***return***K

There are four parties in the communication: the *Overt Sender* (OS), the *Overt Receiver* (OR), the *Covert Sender* (CS), and the *Covert Receiver* (CR). OS and OR make use of the overt channel to exchange information. CS and CR use the covert channel to transmit and receive information in a form not intended for the overt channel. In our proposals, OS and CS will be the same entity, while OR and CR can be different entities depending on the case. So, there will be cases where CR will be in a man-in-the-middle (MITM) position. As subliminal channels are a type of covert channels, we use the terms covert and subliminal interchangeably.

There is an entity called the *Jury* that can accuse the covert sender and receiver of exchanging covert information. Therefore, the covert sender and receiver should convince the Jury that this is unquestionably not true. The Warden can give the Jury the information exchanged that has raised the suspicion. The Jury asks the covert entities for the random data generated and used during the execution of the algorithms related to the key exchange. The Jury is convinced when it is assured that no algorithm-subverting attack is carried out by the covert entities. This is done by executing the Kyber algorithms using the expected random data and seeing if the output matches the data sent by the Warden. This situation is considered to prove plausible deniability.

V. PROPOSALS OF SUBLIMINAL CHANNELS IN KYBER KEM

In this section, three different subliminal channels are proposed for Kyber KEM. All of them exploit the fact that the key generation and encapsulation algorithms require the generation of random numbers.

A. SUBLIMINAL CHANNEL 1 (SC 1): DIRECT ENCRYPTION OF THE COVERT MESSAGE IN THE ENCAPSULATION

In this subliminal channel, it is assumed that the overt and covert entities are the same. In this scheme, the covert entities use a symmetric cryptosystem and a covert secret key SK_C to encrypt the original covert message m_C, resulting in $ENC(SK_C, m_C)$. This is common in the field of covert and subliminal channels to protect the confidentiality of the

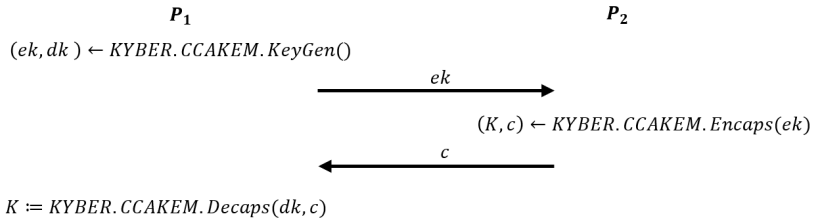

FIGURE 1. Kyber key exchange protocol.

Algorithm 4 Modified Kyber Encapsulation Algorithm for Subliminal Channel 1.

$KYBER.CCAKEM.Encaps.SC1(ek = ek_{PKE}, SK_C^i, m_C)$

Inputs: *encapsulation key* ek, *covert secret key* SK_C^i, *covert message* m_C
Outputs: *ciphertext* c, *shared secret key* K, *updated covert secret key* SK_C^{i+1}
1. $m := ENC(SK_C^i, m_C)$
2. $(K, r) := G(m \parallel H(ek))$
3. $c := KYBER.CPAPKE.Enc(ek_{PKE}, m, r)$
4. $SK_C^{i+1} := KDF(SK_C^i, inf)$
5. ***return***(K, c, SK_C^{i+1})

Algorithm 5 Modified Kyber Decapsulation Algorithm for Subliminal Channel 1.

$KYBER.CCAKEM.Decaps.SC1(dk = (dk_{PKE} \parallel ek \parallel H(ek) \parallel z), c, SK_C^i)$

Inputs: *decapsulation key* dk, *ciphertext* c, *covert secret key* SK_C^i
Outputs: *shared secret key* K, *covert message* m_C, *updated covert secret key* SK_C^{i+1}
1. $m' := KYBER.CPAPKE.Dec(dk_{PKE}, c)$
2. $(K, r') := G(m' \parallel H(ek))$
3. $\bar{K} := J(z \parallel c)$
4. $c' := KYBER.CPAPKE.Enc(ek, m', r')$
5. ***if*** $c \neq c'$ ***then***
6. $\quad K := \bar{K}$
7. **end if**
8. $m_C := DEC(SK_C^i, m')$
9. $SK_C^{i+1} := KDF(SK_C^i, inf)$
10. ***return***(K, m_C, SK_C^{i+1})

original covert message in case the covert or subliminal channel is detected [13], [21], [36]. This also makes m pseudo-random, which is important for maintaining the security of the scheme, as will be shown in the next section.

In this approach, the Kyber Encapsulation algorithm shown in Algorithm 2 is modified by replacing the uniformly generated random number in step 1 with the encrypted covert message m_C. This value is encrypted by the covert sender using the *KYBER.CPAPKE.Enc* function and, since the covert receiver has the decryption key dk_{PKE}, s/he can directly decrypt the received ciphertext c using the *KYBER.CPAPKE.Dec* function and the decryption function $DEC(SK_C^i, m')$ to recover m_C. Algorithm 4 and Algorithm 5 show $KYBER.CCAKEM.Encaps.SC1$ and $KYBER.CCAKEM.Decaps.SC1$, which are the modifications of the original *KYBER.CCAKEM.Encaps* and *KYBER.CCAKEM.Decaps* algorithms. Also, a new secret key is generated for each subsequent execution of the modified KEM. Here, *KDF* is a key derivation function that uses a key *SK* and some additional information *inf* to generate a new key.

In this subliminal channel, the overt and covert receivers should be the same entity since the decryption key dk_{PKE} is required to decrypt the covert message. If this key is shared with other parties, the confidentiality of the overt communication could be compromised.

Algorithm 6 Modified Kyber Encapsulation Algorithm for Subliminal Channel 2.

$KYBER.CCAKEM.Encaps.SC2(ek = ek_{PKE}, SK_C^i, m_C)$

Inputs: *encapsulation key* ek, *covert secret key* SK_C^i, *covert message* m_C
Outputs: *shared secret key* K, *ciphertext* c, *updated covert secret key* SK_C^{i+1}
1. $m := PRF(SK_C^i, m_C)$
2. $(K, r) := G(m \parallel H(ek))$
3. $c := KYBER.CPAPKE.Enc(ek_{PKE}, m, r)$
4. $SK_C^{i+1} := KDF(SK_C^i, inf)$
5. ***return***(K, c, SK_C^{i+1})

Figure 2 shows the modified Kyber protocol to allow the transmission of the covert message m_C.

B. SUBLIMINAL CHANNEL 2 (SC 2): INJECTION OF THE COVERT MESSAGE IN THE RANDOM MESSAGE OF THE ENCAPSULATION

In this subliminal channel, shown in Figure 3, the covert receiver is in a man-in-the-middle position rather than a covert receiver. This approach does not assume that the covert message m_C is encrypted as in subliminal channel 1.

As this subliminal channel is stateful, the procedure is explained for the ith execution of the protocol. Let *PRF* be

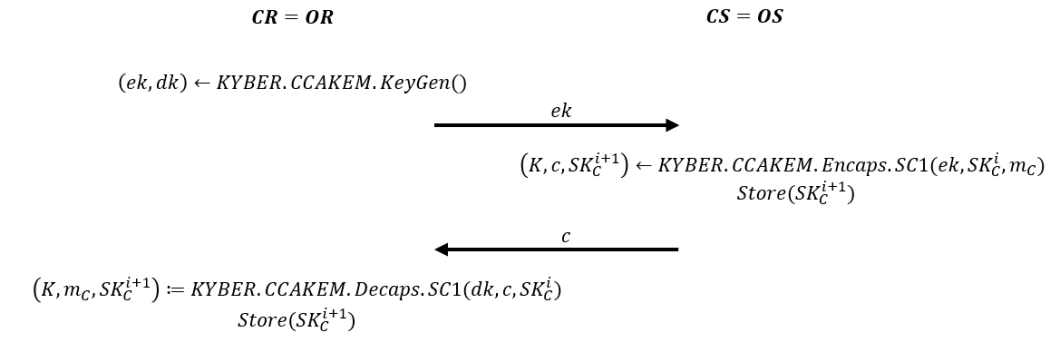

FIGURE 2. Kyber protocol for subliminal channel 1.

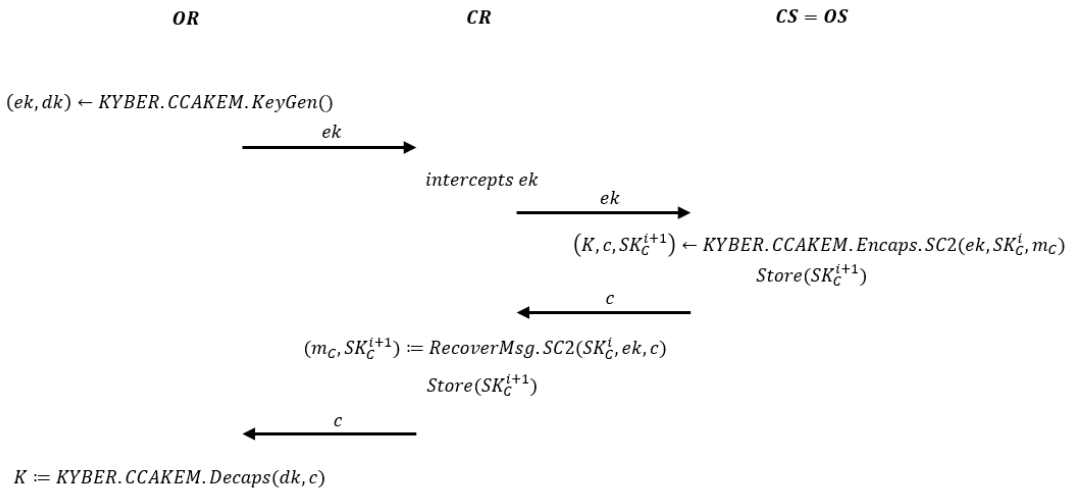

FIGURE 3. Kyber protocol for subliminal channel 2.

a pseudo-random function as $PRF : \{0,1\}^K x \{0,1\}^M \rightarrow \{0,1\}^{256}$ where K is the size of the secret key, and M is the size of m_C. KDF is a key derivation function that uses a key SK and some additional information inf to generate a new key. It is assumed that the covert entities have already pre-shared a covert secret key SK_C^i. Let N be the number of possible covert messages that can be sent on this subliminal channel.

Algorithm 6 shows $KYBER.CCAKEM.Encaps.SC2$, the modified Key Encapsulation algorithm for injecting the covert message using subliminal channel 2. Instead of a uniformly generated random number, the message m used in the modified version is the output of the PRF, which has as inputs the covert message m_c and the covert secret key SK_C^i. The covert secret key SK_C^i is updated to a new one SK_C^{i+1} by applying the KDF and is stored in memory for the next execution of the protocol. To recover the covert message m_C, the covert receiver uses the $RecoverMsg.SC2$ algorithm shown in Algorithm 7 to test all the N possible covert messages by computing a test ciphertext c_T and comparing it with the received ciphertext c.

C. SUBLIMINAL CHANNEL 3 (SC 3): INJECTION OF THE COVERT MESSAGE IN THE RANDOM SEED OF THE KEY GENERATION

In this subliminal channel, shown in Figure 4, the covert receiver can be a man-in-the-middle or directly the overt receiver. As with Subliminal Channel 2, this approach does not require the covert message m_C to be encrypted.

This subliminal channel is also stateful, so the procedure is explained for the ith execution of the protocol. As in Subliminal Channel 2, let PRF be a pseudo-random function as $PRF : \{0,1\}^K x \{0,1\}^M \rightarrow \{0,1\}^{256}$ where K is the size of the secret key, and M is the size of the covert message m_C. KDF is a key derivation function that uses a key SK and some additional information inf to generate a new key. The covert entities have already pre-shared a covert secret key SK_C^i. N is the number of possible covert messages that can be sent on this subliminal channel.

This subliminal channel uses a modification of the Kyber Key Generation algorithm $KYBER.CCAKEM.KeyGen$, that is shown in Algorithm 8 as $KYBER.CCAKEM.KeyGen.SC3$. As in Subliminal Channel 2, the output of a pseudo-random

Algorithm 7 Algorithm Used to Recover the Covert Message in Subliminal Channel 2

$RecoverMsg.SC2(SK_C^i, ek = ek_{PKE}, c)$

Inputs: *covert secret key* SK_C^i, *encapsulation key* ek, *ciphertext* c
Outputs: *covert message* m_c, *updated covert secret key* SK_C^{i+1}

1. $j := 0$
2. ***for*** j ***from*** 0 ***to*** $N-1$ ***do***
3. $\quad m := PRF(SK_C^i, m_{C,j})$
4. $\quad (K, r) := G(m \parallel H(ek))$
5. $\quad c_T := KYBER.CPAPKE.Enc(ek_{PKE}, m, r)$
6. $\quad$ ***if*** $c_T = c$ ***then***
7. $\quad\quad m_C := m_{C,j}$
8. $\quad\quad$ ***break***
9. $\quad$ ***end if***
10. ***end for***
11. $SK_C^{i+1} := KDF(SK_C^i, inf)$
12. ***return*** (m_c, SK_C^{i+1})

Algorithm 8 Modified Kyber Key Generation Algorithm for Subliminal Channel 3.

$KYBER.CCAKEM.KeyGen.SC3(SK_C^i, m_C)$

Inputs: *covert secret key* SK_C^i, *covert message* m_C
Outputs: *encapsulation key* ek, *decapsulation key* dk, *updated covert secret key* SK_C^{i+1}

1. $d := PRF(SK_C^i, m_C)$
2. $z \leftarrow \{0,1\}^{256}$
3. $(ek_{PKE}, dk_{PKE}) := KYBER.CPAPKE.KeyGen(d)$
4. $ek := ek_{PKE}$
5. $dk := (dk_{PKE} \parallel ek \parallel H(ek) \parallel z)$
6. $SK_C^{i+1} := KDF(SK_C^i, inf)$
7. ***return*** (ek, dk, SK_C^{i+1})

function is used as the random seed d instead of a uniformly random generated value. The PRF has as input the covert message m_C and the covert secret key SK_C^i. Also, SK_C^i is updated by using the *KDF* and is stored for the next execution $i+1$. The covert receiver uses the $RecoverMsg.SC3$ algorithm shown in Algorithm 9 to test all the N possible covert messages by computing a test encapsulation key ek_T and comparing it with the received encapsulation key ek.

VI. SECURITY ANALYSIS

This section discusses the security of the three schemes resulting from the creation of subliminal channels. This is because we are interested in maintaining the security of the underlying key encapsulation mechanism (KEM).

First, we will recall the security of the Kyber KEM (*Kyber.CCAKEM*) which is in the random oracle model (ROM). The security is proven when G, H, J (seen in

Algorithm 9 Algorithm Used to Recover the Covert Message in Subliminal Channel 3

$RecoverMsg.SC3(SK_C^i, ek)$

Inputs: *covert secret key* SK_C^i, *encapsulation key* ek
Outputs: *recovered covert message* m_c, *updated covert secret key* SK_C^{i+1}

1. $j := 0$
2. ***for*** j ***from*** 0 ***to*** $N-1$ ***do***
3. $\quad d := PRF(SK_C^i, m_{C,j})$
4. $\quad (ek_T, dk) := KYBER.CPAPKE.KeyGen(d)$
5. $\quad$ ***if*** $ek_T = ek$ ***then***
6. $\quad\quad m_c := m_{C,j}$
7. $\quad\quad$ ***break***
8. $\quad$ ***end if***
9. ***end for***
10. $SK_C^{i+1} := KDF(SK_C^i, inf)$
11. ***return*** (m_c, SK_C^{i+1})

Section III), and the function *XOF* employed to generate the random matrix in Kyber PKE algorithms are modeled as random oracles. Regarding the IND-CPA (Indistinguishability against Chosen-Plaintext Attacks) security of the Kyber PKE scheme (*Kyber.CPAPKE*), it has been shown in [17] that, for any adversary A with advantage $\boldsymbol{Adv}^{CPA}_{Kyber.CPAPKE}(A)$, there exists an adversary B for the MLWE problem with advantage $\boldsymbol{Adv}^{MLWE}_{k+1,k,\eta}(B)$ such that:

$$\boldsymbol{Adv}^{CPA}_{Kyber.CPAPKE}(A) \leq 2 \cdot \boldsymbol{Adv}^{MLWE}_{k+1,k,\eta}(B)$$

For simplicity, we are ignoring the advantage an adversary could have over the pseudorandom function used to sample from the central binomial distribution in the Kyber PKE algorithms in [20]. Then, regarding the IND-CCA (Indistinguishability against Chosen-Cyphertext Attacks) security of the scheme in the classical random oracle model, it is shown in [37] that, for any adversary A with advantage $\boldsymbol{Adv}^{CCA}_{Kyber.CCAKEM}(A)$, there exists an adversary B with advantage $\boldsymbol{Adv}^{CPA}_{Kyber.CPAPKE}(B)$ such that:

$$\boldsymbol{Adv}^{CCA}_{Kyber.CCAKEM}(A) \leq 3 \cdot \boldsymbol{Adv}^{CPA}_{Kyber.CPAPKE}(B) + q_{RO} \cdot \delta + \frac{3 \cdot q_{RO}}{2^{256}}$$

where the term δ reflects the failure probability δ of Kyber PKE and q_{RO} is the maximum number of queries to the random oracles allowed to A.

The security of the modified KEMs is based on the security of the original Kyber KEM, also using the random oracle model. First, we note that the block cipher used in the symmetric encryption *ENC* can be viewed as a pseudo-random permutation (PRP). Then, we observe that, according to the *PRP-PRF switching lemma* in [38], a PRP behaves as a PRF for any adversary when the term $q_O \cdot (q_O - 1)/2 \cdot N$, also referred to as the birthday bound, is negligible. Here, q_O is the number of oracle queries and N is the size of the block

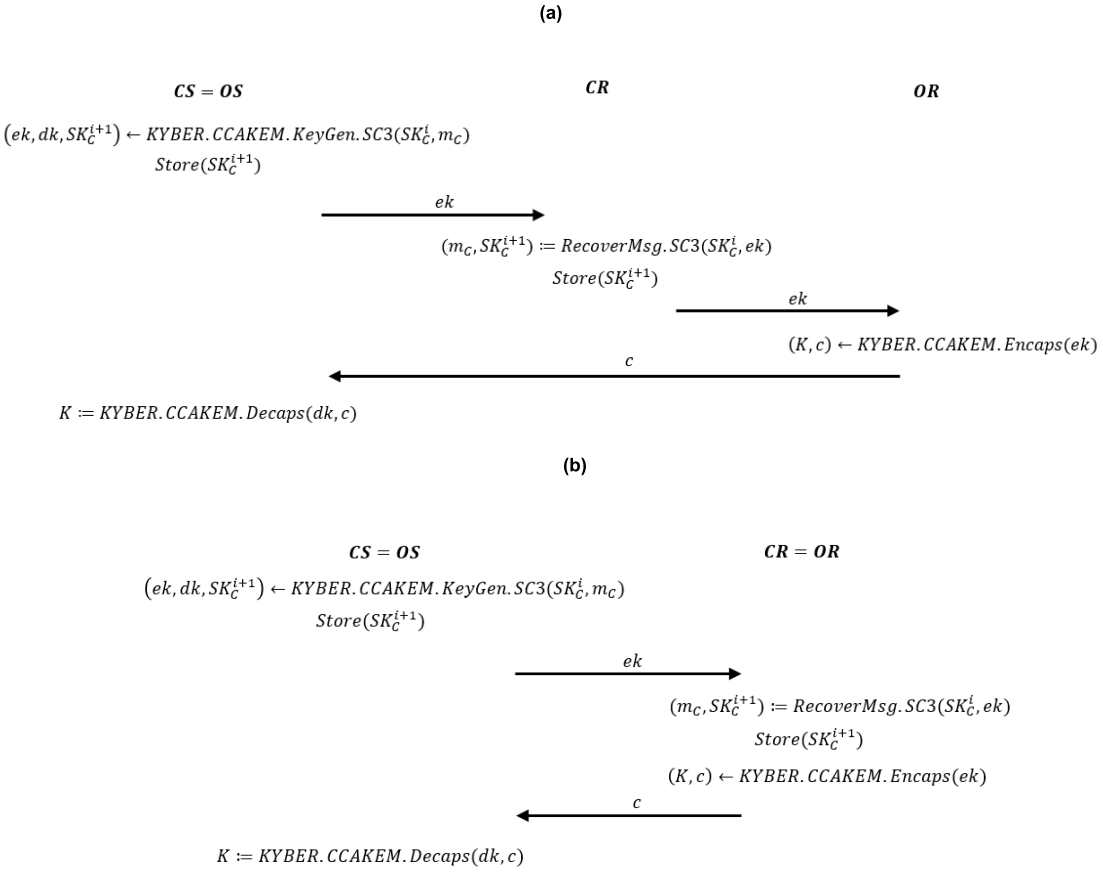

FIGURE 4. Protocol execution for the proposal of subliminal channel 3: (a) when CR is a MITM, and (b) when CR is the OR.

cipher output space. Since the secret key SK_c^i is updated each time, we can assume that an adversary can only make a very limited number of oracle queries. Also, a block cipher with a relatively high N can be chosen. Examples include AES with $N = 2^{128}$ and Saturnin [39] with $N = 2^{256}$. Therefore, we conclude that *ENC* behaves as a pseudo-random function in this case. Since the message size is 256 bits in Kyber KEM, two blocks will be used with two different keys if AES is used in *ENC*. Because we assume that the adversary does not know the covert secret keys involved in the protocol, we will model the pseudo-random functions introduced in Section V as random oracles.

Then, we argue that the security of the modified KEMs shown in the previous section is equivalent to the security of Kyber KEM. Note that the message m is uniformly sampled at random in the proof of the IND-CCA security of the KEM (see Theorem 3.4 of [37]). In our case, the message comes from a random oracle. Therefore, the proof remains unaffected, and we can conclude that the security is also unaffected. The same applies to the obtention of the value d in the *KYBER.CCAKEM.KeyGen.SC*3 algorithm and, hence, the security is also unaffected. Note that the covert secret key should be kept secret. One advantage of the modified KEMs is that they cannot be distinguished from normal Kyber KEMs. Therefore, the adversary does not know whether a secret key is in use.

VII. USE OF THE PROPOSED SUBLIMINAL CHANNELS IN QUANTUM-RESISTANT TLS PROTOCOLS

With the TLS (Transport Layer Security) protocol, a client wants to establish a secure connection with a server. The server is authenticated to the client, and secret session keys are established to encrypt the data. In the typical TLS handshake protocol, the server is authenticated by a digital signature.

In order to resist attacks from quantum computers, a post-quantum TLS that includes a quantum-resistant KEM protocol and a post-quantum digital signature is shown in [25]. This protocol, named as PQTLS, is shown in Figure 5(a). After a TCP (Transmission Control Protocol) synchronization message from the client (*TCP SYN*) and the acknowledgment of the TCP synchronization message from the server (*TCP SYN ACK*), the KEM starts. The client generates a pair of public and secret keys ek and dk with the key generation algorithm *KEM.CCAKEM.KeyGen* and sends ek to the server. The server has a pair of public and secret keys pk_s and sk_s and the certificate of its public key, $cert[pk_s]$. The server applies the

encapsulation algorithm *KEM.CCAKEM.Encaps* to generate the shared key K_e and the ciphertext c. From K_e, a set of ephemeral keys K, K', K'', K''' are generated by using a Key Derivation Function *KDF*. Then the server sends to the client the ciphertext c, and an *AEAD* (Authenticated Encryption with Associated Data) message signed and encrypted. The message contains the certificate of the server's public key $cert[pk_s]$, a transcript (tr.) signed with a post-quantum signature *PQ.Sig*, and a key confirmation (*keyconf*). The client employs the decapsulation algorithm *KEM.CCAKEM.Decaps* to obtain the shared key K_e, and generates from it the set of ephemeral keys K, K', K'', K'''. The ephemeral keys are used to encrypt the key confirmation from the client to the server, and the application data (*appdata*) from the server to the client and vice versa.

In the PQTLS protocol, there are four possible scenarios to apply the proposed subliminal channels: 1) The server can act as the covert sender and the client as the covert receiver with the proposed subliminal channel 1 (SC 1); 2) the server can be the covert sender and an entity in a MITM position can be the covert receiver with the proposed subliminal channel 2 (SC 2); 3) the client can act as the covert sender and the server as the covert receiver with the proposed subliminal channel 3 (SC 3); 4) the server can act as the covert sender and an entity in a MITM position can act as the covert receiver with the proposed subliminal channel 3 (SC 3). These scenarios are shown in Table 1.

Another quantum-resistant TLS protocol presented in [26] is named KEMTLS. It is an alternative TLS handshake protocol that avoids the authentication of the server through signatures. Instead, KEMTLS authenticates the server through a unilateral authenticated key exchange that employs the encapsulation and decapsulation primitives found in the KEM, as shown in Figure 5(b). The server has a pair of encapsulation and decapsulation keys ek_s and dk_s. After a TCP (Transmission Control Protocol) synchronization message from the client (*TCP SYN*) and the acknowledgment of the TCP synchronization message from the server (*TCP SYN ACK*), the KEM starts. The client generates a pair of encapsulation and decapsulation keys ek_e and dk_e with the key generation algorithm *KEM.CCAKEM.KeyGen* and sends ek_e to the server. The server applies the encapsulation algorithm *KEM.CCAKEM.Encaps* to generate the shared key K_e and the ciphertext c_e. From K_e, a set of ephemeral keys K_1, K_1' are generated by using a Key Derivation Function *KDF*. Then the server sends to the client the ciphertext c_e, and an *AEAD* message signed and encrypted with the ephemeral key K_1 and including the certificate of the server's encapsulation key $cert[ek_s]$. The client employs the decapsulation algorithm *KEM.CCAKEM.Decaps* to obtain the shared key K_e and generates from it the set of ephemeral keys K_1, K_1'. The client applies the encapsulation algorithm to generate another shared key K_s and the ciphertext c_s from ek_s, and sends an *AEAD* message signed and encrypted with the ephemeral key K_1' and including the ciphertext c_s. The server employs the decapsulation algorithm to obtain the shared key K_s. Another set of ephemeral keys K_2, K_2', K_2'', K_2''' is generated at the client and at the server from the shared keys K_e and K_s. These ephemeral keys are used to encrypt the key confirmation (*key conf*.) and the application data (*app data*) from the client to the server, and vice versa.

The same subliminal channels used in PQTLS can be used in KEMTLS. In addition, when the client is the covert sender and the server is the covert receiver, the subliminal channel 1 (SC 1) can be used together with the subliminal channel 3 (SC 3). Also, when the client is the covert sender and the covert receiver is an entity in the MITM position, the subliminal channel 2 (SC 2) can be used together with the subliminal channel 3 (SC 3). These scenarios are shown in Table 1.

VIII. FEATURES OF THE PROPOSED SUBLIMINAL CHANNELS

Our proposed subliminal channels accomplish the following features:

- *Long-term usability*. Since Kyber is quantum-resistant and is the basis of the standard FIPS 203 by NIST (as ML-KEM), it is expected to be widely used in TLS and have a long lifetime. This is not the case for other subliminal channels that have short-term usability because they use cryptographic primitives that are not post-quantum.
- *Well-suited for man-in-the-middle covert receivers*. Key-establishment schemes are used to set an ephemeral secret key (or various keys) for encrypting communications between two parties. Hence, the covert sender is not encrypting the messages sent to the overt receiver at the beginning of the communication, whenever the shared key has not been established yet. In that case, a covert receiver in the MITM does not need the shared key. Note that this is not the case in the typical subliminal channels based on digital signatures, because in them, the signature can be encrypted to be transmitted to the overt receiver, as shown in [12] for the subliminal channel in the TLS protocol version 1.3.
- *The secret key of the covert sender remains private.* Other subliminal channels, such as the broadband subliminal channel in [12], require the covert receiver to have the secret key of the covert sender. This can compromise the authenticity of the covert sender, as the covert receiver can forge their signatures. In our subliminal channels, it is not necessary for the covert sender and receiver to exchange the secret key of the covert sender, even in SC 1, which is the broader subliminal channel.
- *Plausible deniability*. The covert sender and the covert receiver can plausibly deny the existence of the hidden communication between them. The covert entities can show to an observer that there is an overt communication between them in which there is a key that is exchanged and that the steps of the original algorithms are followed step by step, i.e., they can demonstrate that they have not executed a tampered version of the original

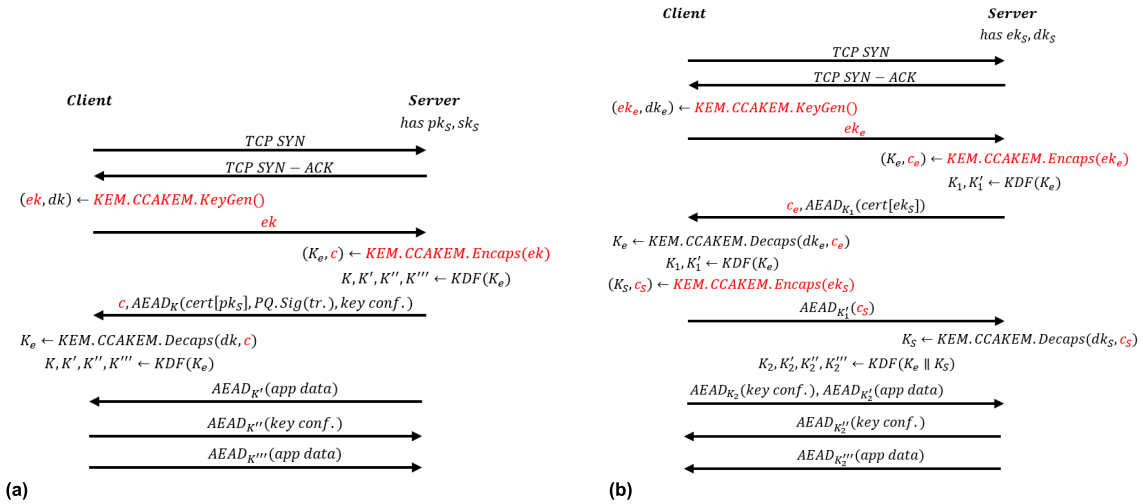

FIGURE 5. (a) Post-Quantum TLS (PQTLS) protocol. (b) KEMTLS protocol. The algorithms that can be manipulated are depicted in red.

TABLE 1. Application scenarios of the proposed subliminal channels in PQTLS and KEMTLS protocols.

Role		Covert Receiver (CR)		
		Client	MITM	Server
Covert Sender (CS)	Client	-	SC 3 (in PQTLS) SC 2 + SC 3 (in KEMTLS)	SC 3 (in PQTLS) SC 1 + SC 3 (in KEMTLS)
	Server	SC 1 (in PQTLS) SC 1 (in KEMTLS)	SC 2 (in PQTLS) SC 2 (in KEMTLS)	-

algorithms. The intermediate results can be stored for demonstration, from the random seeds or messages to *ek* and *c*. This is achieved when ML-KEM is used because the hash of the ciphertext is not employed in the derivation of the shared key.

- *They can be used in other KEMs*. We argue that the same idea can be applied to create subliminal channels in other KEMs where some random information is encrypted or encoded and later decrypted or decoded by the other party, such as Saber and Classic McEliece (which are other algorithms considered by the post-quantum cryptography standardization process of the NIST).

Table 2 shows the comparison of the subliminal channels proposed in this paper with the subliminal channels found in other proposals. Since the subliminal channels of other proposals are found in digital signatures, they can only be used in situations where a party wants to be authenticated. On the other hand, the subliminal channels presented in this paper are found in KEMs, which can be used for both key exchange and authentication. Note that the subliminal channels found in Kyber can be used with other subliminal channels in Table 2. For example, if the PQTLS scheme is used with Falcon as the signature scheme and CRYSTALS-Kyber as the KEM, the subliminal channels associated can be combined to transmit more covert information while achieving plausible deniability. Note that we argue that with digital signatures, the covert receiver cannot be a man-in-the-middle, because in TLS the communication between the overt sender and receiver is encrypted.

Finally, if it is disclosed that there is a hidden communication between the entities, past communications cannot be inspected to find the covert messages in SC 2 and SC 3, because the shared covert key K_c is refreshed at each execution of the protocol. Thus, past covert keys cannot be disclosed, which means that these channels accomplish *forward secrecy of covert messages*. The disadvantage is that the covert entities must be well synchronized for proper operation.

IX. COUNTERMEASURES

This section proposes two countermeasures to mitigate the above subliminal channels.

A. INCLUSION OF A PROXY SERVER

In a controlled environment, such as a company with workstations, a simple way to mitigate the subliminal channel SC 1 is to place a proxy server between an entity inside the company and the outside world. This denies a direct connection between entities inside and outside the controlled environment. In this case, the client establishes a TLS connection with the proxy server, and the proxy server establishes a TLS connection with the server outside the

TABLE 2. Comparison of the subliminal channels found in this work with subliminal channels found in other works.

Scheme	Long-term usability?	Suitable for MITM covert receivers?	Plausible deniability?
Dilithium [21]	Yes (only authentication)	No	No
Falcon [21]	Yes (only authentication)	No	Yes
SPHINCS+ [21]	Yes (only authentication)	No	No
EdDSA [12]	No	No	No
Kyber KEM (Ours)	Yes (authentication and key exchange)	Yes*	Yes

* When SC 2 and SC 3 are used.

controlled environment. The application data received from the server is then passed through the proxy server to the client.

Note that this countermeasure resembles the practice of TLS inspection, in which TLS traffic is decrypted, analyzed, and re-encrypted for the purpose of detecting incoming malware, anomaly detection, or/and application detection [40], and that it can have privacy concerns depending on the context of use [41].

B. TRUSTED MODULE FOR EXECUTION OF KEY GENERATION AND ENCAPSULATION ALGORITHMS

In this countermeasure, one or both entities executing the key-encapsulation mechanism are forced to use a trusted module to execute the key generation and encapsulation algorithms. These are the algorithms that have been found to be susceptible to tampering for subliminal channel creation. The trusted module with the highest level of assurance contains a hardware component that cannot be tampered with, for example, Physical Unclonable Functions (PUFs), whose private keys are not accessible from the outside, not even by the owner of the device with the trusted module, and can be attested remotely (so that there is a way to verify that the module is doing exactly what it claims). An example of how to achieve this functionality can be seen in [42]. The module executes the algorithms and signs the generated encapsulation key or ciphertext with its protected private key. It is assumed that the Warden has the corresponding public keys to verify the signature of the public key or ciphertext.

A hardware-based trusted module contains a certified True Random Number Generator (TRNG). An example of how to design and use a TRNG and protect by hardware the private keys in a secure device can be seen in [43]. Thus, the required random seed or message is truly generated randomly and cannot be used to establish subliminal channels. Although we propose the use of a specific hardware module for the highest security, other trusted solutions can be applied for lower security requirements, such as the one proposed in [44].

The disadvantages of this countermeasure are the increase of the bandwidth, as post-quantum digital signatures typically have considerable size, and the increase of cost, as this countermeasure requires the addition of a hardware component.

X. EXPERIMENTAL RESULTS

A study was conducted to find the bandwidths and the execution times of the proposed subliminal channels as well as the bandwidths of the quantum-resistant TLS protocols in the different scenarios where they are applied. The codes were developed in C by using the Kyber library [45]. Codes were executed on a laptop equipped with an Intel 11th Gen i7-11800H processor running at a CPU frequency of 2.30 GHz with a RAM memory of 16 GB. We used the same laptop to simulate the steps performed by the client and the server. That is, we implemented the routines in the laptop with the algorithms that the covert sender and the covert receiver should execute, and we executed them sequentially according to the schemes presented in the previous section.

A. BANDWIDTHS AND EXECUTION TIMES OF THE PROPOSED SUBLIMINAL CHANNELS

The bandwidth of the subliminal channel SC 1 is 32 bytes, as the covert message is directly encrypted with the Kyber PKE algorithm. We take into account the bandwidth after encryption of the covert message, as commonly done in literature. Regarding execution times for SC 1, the original encapsulation algorithm and its tampered version provided quite similar values. For subliminal channels SC 2 and SC 3, the bandwidth is limited by the number of possible combinations that the covert receiver has to check to recover

TABLE 3. Bandwidth and execution times for the subliminal channel SC 2.

Bandwidth (bits)	Security Level	Avg. Execution Time of Encaps. (ms)	Avg. Execution Time of Modified Encaps. (ms)	Avg. Execution Time to Recover a Covert Message (ms)	Worst-Case Execution Time to Recover a Covert Message (ms)
4	I	0.14	0.11	0.70	1.66
	III	0.21	0.17	1.39	2.60
	V	0.28	0.25	2.23	3.75
8	I	0.14	0.11	13.64	25.26
	III	0.21	0.16	21.41	39.73
	V	0.28	0.25	37.64	57.96
12	I	0.13	0.10	261.38	401.26
	III	0.19	0.16	336.11	642.92
	V	0.26	0.23	643.76	939.35
16	I	0.14	0.10	3,679.61	7,120.01
	III	0.19	0.16	5,281.49	11,336.92
	V	0.25	0.23	9,172.02	16,773.44

the covert message. This is because s/he has to use a trial-and-error (or brute-force) approach, as was shown in the algorithms of Section V.

Table 3 shows the bandwidths and execution times for the subliminal channel SC 2. The average times for the original and tampered encapsulation algorithms, as well as the average and worst-case times to recover a covert message, are shown. The results were obtained for different security levels of Kyber. It can be seen that the tampered version of the encapsulation algorithm has slightly lower execution times. This is because generating a random number using the Wincrypt library was slightly slower than performing the pseudo-random function. The table also shows that a bandwidth of 16 bits (2 bytes) can be used since it requires average times to recover the covert message of, approximately, 4, 5 or 9 seconds with worst-case times of, approximately, 7, 11 or 17 seconds (depending on the security levels of Kyber). More than 16 bits were studied. However, a higher number of bits led to very high times to recover the messages. For the subliminal channel SC 3, the results are shown in Table 4. The execution times of the tampered key generation algorithm are slightly slower, and proper execution times to recover the cover message were achieved with a bandwidth of 16 bits.

Table 5 compares the bandwidth of the proposed SC 1 with the other proposal from the literature that achieves plausible deniability and is quantum secure, namely the subliminal channel of the Falcon digital signature [21]. Also, for a better comparison, we have chosen SC 1 since it is the one non intended to be used in a man-in-the-middle scenario. The results are given in bytes and as a percentage of the signature size for Falcon and of the ciphertext for Kyber. The table shows that the bandwidths are similar. Table 5 shows the number of cycles required to perform a signature with Falcon or an encapsulation with Kyber, allowing for a comparison of the computational cost. This comparison allows for AVX2 vector instructions. The Falcon values are taken from [46], and the Kyber values are taken from [47]. As can be seen, Kyber encapsulation takes less time than a Falcon signature. Therefore, if the covert entities have the freedom to choose the cryptographic primitive, Kyber will require less computational cost.

B. BANDWIDTHS OF THE PROPOSED SUBLIMINAL CHANNELS IN QUANTUM-RESISTANT TLS PROTOCOLS

Based on the above-mentioned bandwidths, the amount of data that can be sent using the proposed subliminal channels in the quantum-resistant TLS protocols described in Section VII was calculated. Table 6 shows the bandwidth in each TLS handshake execution using PQTLS and KEMTLS, respectively, considering the scenarios that were shown in Table 1 and the analysis done in the previous subsection.

Based on these results, the amount of covered information that can be sent in a given period of time was estimated. For that, we assumed that the number of TLS handshakes that a client and a server can execute in a day without being too suspicious depending on the level of activity (either low, medium, or high) are those shown in Table 7. Obviously, this can vary depending on the context. We assume that the number of TLS handshakes allowed is different if the covert receiver is directly the client/server or an entity in the MITM position. This is because we consider that too many TLS handshakes with the same party are more suspicious than different TLS handshakes with different parties.

TABLE 4. Bandwidth and execution times for the subliminal channel SC 3.

Bandwidth (bits)	Security Level (bits)	Avg. Execution Time of KeyGen (ms)	Avg. Execution time of Modified KeyGen (ms)	Avg. Execution Time to Recover a Covert Message (ms)	Worst-Case Execution Time to Recover a Covert Message (ms)
4	I	0.15	0.12	0.74	1.79
	III	0.20	0.17	1.36	2.75
	V	0.28	0.23	2.05	3.82
8	I	0.14	0.11	14.92	27.76
	III	0.19	0.16	18.80	40.44
	V	0.29	0.25	27.50	58.20
12	I	0.14	0.11	279.98	437.48
	III	0.19	0.16	232.63	642.86
	V	0.26	0.23	550.97	973.06
16	I	0.14	0.11	3,674.08	7,809.45
	III	0.19	0.16	5,625.55	11,012.65
	V	0.28	0.25	8,592.47	16,138.29

TABLE 5. A comparison of subliminal channel 1 of Kyber with the subliminal channel presented in Falcon.

Scheme	Instance	Bandwidth (bytes)	Bandwidth (%)	Computational cost (# of cycles)
Falcon [21]	512	40	6.48	881,136 *
	768	40	4.02	-
	1024	40	3.24	1,784,268
Kyber (SC1)	512	32	4.17	45,200 **
	768	32	2.94	67,624
	1024	32	2.04	97,324

* Intel Core i7-4770K (Haswell) processor clocked at 3,492 MHz
** Intel i7-6567U (Skylake core), nominally running at 3.3 GHz

TABLE 6. Bandwidths (in bytes) using the proposed subliminal channels in the PQTLS and KEMTLS protocols.

Role		Covert Receiver (CR)		
		Client	MITM	Server
Covert Sender (CS)	Client	-	2 (in PQTLS) 2 + 2 (in KEMTLS)	2 (in PQTLS) 32 + 2 (in KEMTLS)
	Server	32 (in PQTLS) 32 (in KEMTLS)	2 (in PQTLS) 2 (in KEMTLS)	-

TABLE 7. Number of TLS handshakes per day considering different levels of activity (C = Client, S = Server, M = entity in MITM position).

Level of activity	$C \rightarrow S$, $S \rightarrow C$	$C \rightarrow M$, $S \rightarrow M$
Low	3	10
Medium	10	30
High	20	60

Figure 6 shows the amount of covered information that could be transmitted in 1 day depending on which entity is the covert sender and the covert receiver, using PQTLS and KEMTLS protocols, respectively. When the server acts as the covert sender and the client as the covert receiver, the amount of covert information that could be transmitted is the highest for the PQTLS and very high for the KEMTLS. When the covert sender is the client and the server is the covert receiver, the highest amount of covert information can be transmitted when using KEMTLS. When PQTLS is used, the amount of information is considerably lower. The same amount of information can be transmitted using PQTLS when the covert sender is either the client or the server and the covert receiver is an entity in a MITM position. This amount is the double

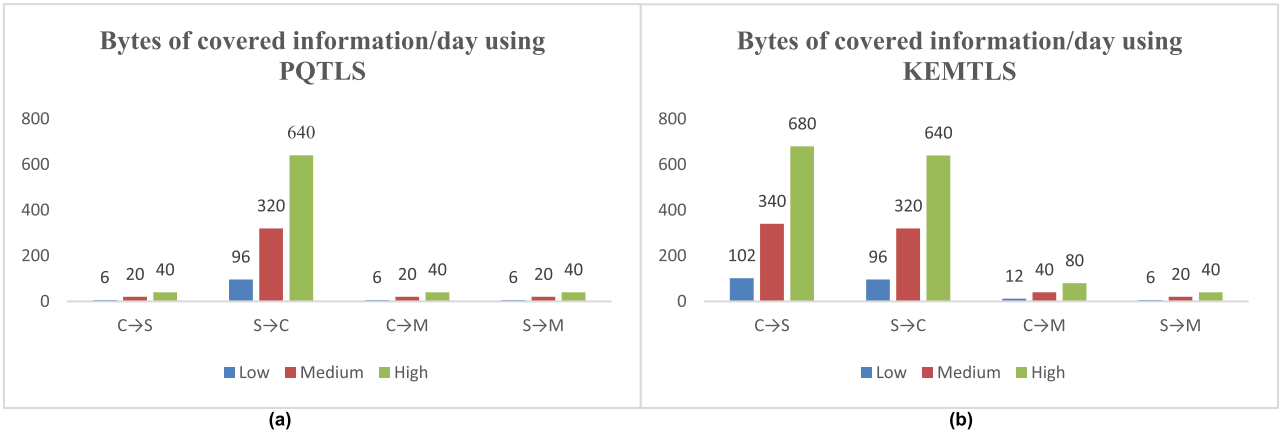

FIGURE 6. Amount of covered information transmitted per day considering different levels of activity and different cases of covert senders and covert receivers (S = Server, C = Client or M = MITM): (a) using PQTLS, (b) using KEMTLS.

when the client is the covert sender, the covert receiver is an entity in the MITM position, and KEMTLS is used.

XI. CONCLUSION

In this work, we explored three subliminal channels found in the Kyber Key-Encapsulation Mechanism. Although in this paper we were focused on Kyber KEM, the proposed subliminal channels can be used in other quantum-secure KEMs, such as Saber and Classic McEliece. Two different scenarios were explored depending on if the covert receiver is one of the entities that executes the KEM mechanism or if it is an entity in a man-in-the-middle position. One subliminal channel exploits the fact that the encapsulation algorithm of the KEM directly encrypts a random message that is directly decrypted by the other entity. Another subliminal channel is based on manipulating the way in which the random message is created in the encapsulation algorithm, specially intended for an entity in a man-in-the-middle position that does not have the secret key to decrypt the random message, as in the first subliminal channel. The last subliminal channel is based on manipulating the way in which the random seed is generated in the key generation algorithm of the KEM. This subliminal channel is intended for both an entity that is executing the KEM and an entity in a man-in-the-middle position. We argued that plausible deniability of covert exchange of information is achieved with the given subliminal channels. Additionally, the KEM remains secure in the classical setting as long as the adversary does not know the covert secret key. Security analysis in the quantum random oracle model (QROM) is left as future work.

Also, this work explored how to use the proposed subliminal channels in two quantum-resistant TLS protocols found in the literature, PQTLS and KEMTLS. Some countermeasures were proposed to mitigate the covert exchange of information, including a proxy server and a trusted module for execution of key generation and encapsulation algorithms.

Finally, experimental results concerning execution times and bandwidths were shown using a commercial laptop. Original and tampered algorithms where covert messages are injected provided similar execution times, making the subliminal channels stealthier. With the first subliminal channel, up to 32 bytes of covert information can be sent for one TLS handshake execution. As the other two need a recover algorithm based on a trial-and-error approach, their bandwidth was smaller, with 2 bytes of covert information for one TLS handshake execution.

REFERENCES

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, Oct. 1973.

[2] L. Caviglione, "Trends and challenges in network covert channels countermeasures," *Appl. Sci.*, vol. 11, no. 4, p. 1641, Feb. 2021.

[3] I. Miketic, K. Dhananjay, and E. Salman, "Covert channel communication as an emerging security threat in 2.5D/3D integrated systems," *Sensors*, vol. 23, no. 4, p. 2081, Feb. 2023.

[4] G. J. Simmons, "The Prisoners' problem and the subliminal channel," in *Proc. Advances in Cryptology*. Boston, MA, USA: Springer, 1984, pp. 51–67.

[5] G. J. Simmons, "Subliminal communication is easy using the DSA," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 2007, pp. 218–232.

[6] D. Collins, S. Colombo, and L. Huguenin-Dumittan, "Real-world deniability in messaging," *Proc. Privacy Enhancing Technol.*, vol. 2025, no. 1, pp. 320–340, Jan. 2025.

[7] M. Armour and B. Poettering, "Algorithm substitution attacks against receivers," *Int. J. Inf. Secur.*, vol. 21, no. 5, pp. 1027–1050, Oct. 2022.

[8] S. Berndt and M. Liśkiewicz, "Algorithm substitution attacks from a steganographic perspective," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1649–1660.

[9] C. Jiang, C. Xu, Z. Zhang, and K. Chen, "SR-PEKS: Subversion-resistant public key encryption with keyword search," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 3168–3183, Jul. 2023.

[10] G. Ateniese, B. Magri, and D. Venturi, "Subversion-resilient signatures: Definitions, constructions and applications," *Theor. Comput. Sci.*, vol. 820, pp. 91–122, Jun. 2020.

[11] M. Bellare, J. Jaeger, and D. M. Kane, "Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks," *Cryptol. ePrint Arch.*, vol. 2015, p. 808, Jan. 2015.

[12] A. Hartl, R. Annessi, and T. Zseby, "A subliminal channel in EdDSA: Information leakage with high-speed signatures," in *Proc. Int. Workshop Manag. Insider Secur. Threats*, Oct. 2017, pp. 67–78.

[13] A. Härtl, R. Annessi, and T. Zseby, "Subliminal channels in high-speed signatures," *J. Wireless Mobile Netw.*, vol. 9, pp. 30–53, Mar. 2018.

[14] E. Zeydan, Y. Turk, B. Aksoy, and S. B. Ozturk, "Recent advances in post-quantum cryptography for networks: A survey," in *Proc. 7th Int. Conf. Mobile Secure Services (MobiSecServ)*, Gainesville, FL, USA, Feb. 2022, pp. 1–8.

[15] K. F. Hasan, L. Simpson, M. A. R. Baee, C. Islam, Z. Rahman, W. Armstrong, P. Gauravaram, and M. McKague, "A framework for migrating to post-quantum cryptography: Security dependency analysis and case studies," *IEEE Access*, vol. 12, pp. 23427–23450, 2024.
[16] J. P. Mattsson, B. Smeets, and E. Thormarker, "Quantum technology and its impact on security in mobile networks," *Ericsson Technol. Rev.*, vol. 2021, no. 12, pp. 2–12, Dec. 2021.
[17] J. Bos, "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Apr. 2018, pp. 353–367.
[18] *NIST, Post-Quantum Cryptography*. Accessed: Jul. 13, 2025. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography
[19] *NIST, Post-Quantum Cryptography, Selected Algorithms*. Accessed: Jul. 13, 2025. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms
[20] *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, Standard FIPS 203, National Institute of Standards and Technology, Washington, DC, USA, Aug. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdfH
[21] Galteland and K. Gjøsteen, "Subliminal channels in post-quantum digital signature schemes," *Cryptol. ePrint Arch.*, vol. 2019, p. 574, Jan. 2019.
[22] S. Turner, "Transport layer security," *IEEE Internet Comput.*, vol. 18, no. 6, pp. 60–63, Nov. 2014.
[23] J. Merrill and D. Johnson, "Covert channels in SSL session negotiation headers," in *Proc. Int. Conf. Secur. Manage. (SAM'15)*, 2015, pp. 70–72.
[24] C. Heinz, W. Mazurczyk, and L. Caviglione, "Covert channels in transport layer security," in *Proc. Eur. Interdiscipl. Cybersecurity Conf.*, Nov. 2020, pp. 1–6.
[25] S. Celi, A. Faz-Hernández, N. Sullivan, G. Tamvada, L. Valenta, T. Wiggers, B. Westerbaan, and C. A. Wood, "Implementing and measuring KEMTLS," *Cryptol. ePrint Arch.*, vol. 2021, pp. 88–107, Jan. 2021.
[26] P. Schwabe, D. Stebila, and T. Wiggers, "Post-quantum TLS without handshake signatures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1461–1480.
[27] R. Anderson, S. Vaudenay, B. Preneel, and K. Nyberg, "The Newton channel," in *Proc. Inf. Hiding, 1st Int. Workshop*, 1996, pp. 151–156.
[28] J.-M. Bohli and R. Steinwandt, "On subliminal channels in deterministic signature schemes," in *Proc. ICISC*, 2005, pp. 182–194.
[29] F. Zhang, B. Lee, and K. Kim, "Exploring signature schemes with subliminal channel," in *Proc. Symp. Cryptogr. Inf. Secur. (SCIS)*, 2003, pp. 1–5.
[30] F. Heß, "Efficient identity based signature schemes based on pairings," in *Proc. Int. Workshop Sel. Areas Cryptograph.*, 2003, pp. 310–324.
[31] Q. Hu, C. Xu, and W. Li, "A broadband subliminal channel in signatures without sharing the signing key," in *Proc. IEEE 22nd Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Exeter, U.K., Nov. 2023, pp. 814–822.
[32] Z. Yang, T. Xie, and Y. Pan, "Lattice klepto revisited," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 867–873.
[33] T. Hemmert, "How to backdoor LWE-like cryptosystems," *Cryptol. ePrint Arch.*, vol. 2022, p. 1381, Jan. 2022.
[34] P. Ravi, S. Bhasin, A. Chattopadhyay, A. Aikata, and S. S. Roy, "Backdooring post-quantum cryptography: Kleptographic attacks on lattice-based KEMs," *Cryptol. ePrint Arch.*, vol. 2022, pp. 1–7, 2024.
[35] J. B. Almeida, "Formally verifying Kyber Episode V: Machine-checked IND-CCA security and correctness of ML-KEM in easyCrypt," *Cryptol. ePrint Arch.*, vol. 2024, p. 843, Jan. 2024.
[36] J. Keller and S. Wendzel, "Reversible and plausibly deniable covert channels in one-time passwords based on hash chains," *Appl. Sci.*, vol. 11, no. 2, p. 731, Jan. 2021.
[37] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki–Okamoto transformation," *Cryptol. ePrint Arch.*, vol. 2017, pp. 341–371, Jan. 2017.
[38] D. Chang and M. Nandi, "A short proof of the PRP/PRF switching lemma," *Cryptol. ePrint Arch.*, vol. 2008, p. 78, Jan. 2008.
[39] A. Canteaut, S. Duval, G. Leurent, M. Naya-Plasencia, L. Perrin, T. Pornin, and A. Schrottenloher, "Saturnin: A suite of lightweight symmetric algorithms for post-quantum security," *IACR Trans. Symmetric Cryptol.*, vol. 2020, pp. 160–207, Jun. 2020.
[40] C. Oh, J. Ha, and H. Roh, "A survey on TLS-encrypted malware network traffic analysis applicable to security operations centers," *Appl. Sci.*, vol. 12, no. 1, p. 155, Dec. 2021.
[41] M. O'Neill, S. Ruoti, K. Seamons, and D. Zappala, "TLS inspection: How often and who cares?" *IEEE Internet Comput.*, vol. 21, no. 3, pp. 22–29, May 2017.
[42] R. Román, R. Arjona, and I. Baturone, "A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices," *Future Gener. Comput. Syst.*, vol. 148, pp. 425–435, Nov. 2023.
[43] I. Baturone, R. Román, and Á. Corbacho, "A unified multibit PUF and TRNG based on ring oscillators for secure IoT devices," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 6182–6192, Apr. 2023.
[44] M. Morbitzer, B. Kopf, and P. Zieris, "GuaranTEE: Introducing control-flow attestation for trusted execution environments," in *Proc. IEEE 16th Int. Conf. Cloud Comput. (CLOUD)*, Chicago, IL, USA, Jul. 2023, pp. 547–553.
[45] *Github, Pq-crystals*. Accessed: Jul. 13, 2025. [Online]. Available: https://github.com/pq-crystals/kyber
[46] T. Pornin, "New efficient, constant-time implementations of falcon," *Cryptol. ePrint Arch.*, vol. 2019, p. 893, Jan. 2019.
[47] *CRYSTALS-Cryptographic Suite for Algebraic Lattices, Kyber*. Accessed: Jul. 13, 2025. [Online]. Available: https://pq-crystals.org/kyber/

ROBERTO ROMÁN received the bachelor's degree in electronic engineering of telecommunications from the University of Barcelona, in 2018, and the Master of Science degree in microelectronics from the University of Seville, in 2020, where he is currently pursuing the Ph.D. degree in microelectronics with the Microelectronics Institute of Seville (IMSE-CNM), CSIC, University of Seville. In 2019, he won a JAEIntro grant of CSIC and now his research is supported by the VI Plan Propio de Investigación y Transferencia, University of Seville. His current research interests include post-quantum cryptography, blockchain technologies, self-sovereign identities, and biometrics.

ROSARIO ARJONA received the degree in computer science engineering and the Ph.D. degree (Hons.) in microelectronics from the University of Seville, Spain, in 2009 and 2014, respectively. Since September 2009, she has been with the Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC, University of Seville. She is currently an Associate Professor with the Department of Electronics and Electromagnetism, University of Seville. She is the author of 46 scientific articles and two patents. She has collaborated in 26 national and international research, industrial, and teaching innovation projects (leading four of them). Her research interests include crypto-biometrics, post-quantum cryptography, hardware security, physical unclonable functions (PUFs), blockchain, the Internet of Things (IoT), and neuro-fuzzy systems.

ILUMINADA BATURONE received the 5-year B.S. degree (Hons.) in physics and the Ph.D. degree. (Hons.) in physics-electronics from the University of Seville, Seville, Spain, in 1991 and 1996, respectively. She is currently a Full Professor with the Department of Electronics and Electromagnetism, University of Seville, and a Senior Researcher with the Microelectronics Institute of Seville (IMSE-CNM), CSIC, University of Seville. She has co-authored two books and more than 200 articles. She has filed four patents and participated in more than 45 projects, leading 15 of them. Her current research interests include hardware security, post-quantum cryptography, biometrics, blockchain technologies, and neuro-fuzzy systems.