Faster generic CCA secure KEM transformation using encrypt-then-MAC

Anonymous submission

Abstract. `ML-KEM` is a lattice-based IND-CCA secure key encapsulation mechanism (KEM) standardized by NIST in FIPS 203. `ML-KEM` achieves chosen-ciphertext security by applying the Fujisaki-Okamoto (FO) transformation to an IND-CPA secure public key encryption (PKE) scheme. The Fujisaki-Okamoto transformation uses de-randomization and re-encryption to ensure ciphertext non-malleability, but because `ML-KEM`'s underlying PKE encryption subroutine is slower than the decryption subroutine, `ML-KEM`'s decapsulation performance is dominated by re-encryption. In this paper, we propose a new generic IND-CCA secure KEM transformation that achieves chosen-ciphertext security by applying the "encrypt-then-authenticate" pattern to a PKE with one-wayness under plaintext-checking attack (OW-PCA) and an existentially unforgeable message authentication code (MAC). Compared to the FO transformation, our transformation replaces re-encryption with computing a MAC tag. We present `ML-KEM-EtM`, an IND-1CCA secure KEM derived from applying the encrypt-then-MAC transformation to the PKE subroutines of `ML-KEM`. We then implement `ML-KEM-EtM` with a wide selection of MACs including `Poly1305`, `GMAC`, `CMAC`, and `KMAC256`. At the cost of minimal increase in encapsulation CPU cycle count and ciphertext size, `ML-KEM-EtM` achieves a substantial reduction of decapsulation CPU cycle count.

Keywords: Key encapsulation mechanism · Post-quantum cryptography · Lattice cryptography · Message authentication code · Fujisaki-Okamoto transformation

1 Introduction

Key encapsulation mechanism (KEM) is a public-key cryptographic primitive that allows two parties to establish a shared secret over an insecure communication channel. The accepted security requirement of a KEM is *indistinguishability under adaptive chosen ciphertext attack (IND-CCA)*. Intuitively speaking, IND-CCA security implies that no efficient adversary (usually defined as probabilistic polynomial time Turing machine) can distinguish a pseudorandom shared secret from a uniformly random bit string of identical length even with access to a decapsulation oracle. On the other hand, IND-1CCA is a restricted form of IND-CCA security where an adversary can make only one query to the decapsulation oracle. Unfortunately, CCA security is difficult to achieve from scratch. Early attempts at constructing CCA secure public-key cryptosystems using only

heuristics argument and without using formal proof, such as RSA encryption in PKCS #1 [46] and RSA signature in ISO 9796 [72], were badly broken with sophisticated cryptanalysis [14,32,20]. Afterwards, provable chosen ciphertext security became a necessity for new cryptographic protocols. There have been many provable CCA secure constructions since then. Notable examples include Optimal Asymmetric Encryption Padding (OAEP) [7], which is combined with RSA [28] into the widely adopted RSA-OAEP. The Fujisaki-Okamoto transformation [27,38] is another generic CCA secure transformation that was thoroughly studied and widely adopted, particularly by many KEM candidates in NIST's Post Quantum Cryptography (PQC) standardization project.

Chosen ciphertext security is a solved problem within the context of symmetric cryptography. It is well understood that authenticated encryption can be achieved by combining a semantically secure symmetric encryption scheme with an existentially unforgeable message authentication code (MAC) using either the "encrypt-then-MAC" (AES-GCM, ChaCha20-Poly1305) or "MAC-then-encrypt" pattern (AES-CCM)[6,48]. However, adapting this technique for public-key cryptosystems is challenging, since the two communicating parties do not have a pre-shared symmetric key. One attempt at such adaption is the Diffie-Hellman integrated encryption scheme (DHIES) [1,2] proposed by Abdalla, Bellare, and Rogaway, who proved its chosen ciphertext security under a non-standard but well studied assumption called "Gap Diffie-Hellman problem" [60].

1.1 Our contributions

Our contributions are as follows:

New generic CCA secure KEM transformation. We propose the encrypt-then-MAC KEM transformation. Our transformation constructs a KEM with provable CCA security under the random oracle model using a public-key encryption scheme (PKE) with one-wayness under plaintext-checking attack and a message authentication code with existential unforgeability. Compared to the Fujisaki-Okamoto transformation, which is widely adopted by many KEM candidates in NIST's Post Quantum Cryptography (PQC) standardization project, our transformation replaces *de-randomization* and *re-encryption* with computing and comparing a MAC tag. We formally prove the security of our KEM transformation.

Instantiation with `ML-KEM`. We present `ML-KEM-EtM`, an IND-1CCA secure KEM derived from applying the encrypt-then-MAC transformation to `ML-KEM`'s PKE subroutines. Compared to `ML-KEM` in FIPS 203 [59], `ML-KEM-EtM` adds a small amount of performance penalty to the encapsulation routine and a small increase in ciphertext size, but removes the expensive re-encryption step in decapsulation, which yields impressive performance improvements (Table 1).

	ML-KEM-EtM-512 + Poly1305	ML-KEM-EtM-768 + Poly1305	ML-KEM-EtM-1024 + Poly1305
Encap (ccl/tick)	93,157 (+1.8%)	146,405 (+7.3%)	205,763 (+3.3%)
Decap (ccl/tick)	33,733 (-72.2%)	43,315 (-76.8%)	51,375 (-79.1%)
CT size (bytes)	784 (+2.1%)	1104 (+1.4%)	1584 (+1.0%)

Table 1: ML-KEM-EtM achieves impressive reduction of decapsulation CPU cycles count compared to ML-KEM while incurring minimal penalty in encapsulation performance and ciphertext size.

1.2 Related works

Fujisaki-Okamoto transformation Fujisaki and Okamoto proposed to construct CCA secure hybrid PKE by combining a OW-CPA secure PKE and a semantically secure symmetric-key encryption (SKE) scheme [27]. The main techniques are *de-randomization* (deriving the randomness in encryption from the plaintext, thus making the encryption deterministic) and *re-encryption* (re-running encryption in decryption and compare the re-encrypted ciphertext with the input ciphertext). Under the random oracle model, Fujisaki and Okamoto reduced the CCA security of the hybrid PKE tightly to the semantic security of the input SKE and *non-tightly* to the OW-CPA security of the input PKE (with loss factor q, the number of hash oracle queries). Later works extended the original proposal to build CCA secure KEM: KEM's security model makes building secure KEM simpler than building secure PKE, and it is well-known that combining a CCA secure KEM with a CCA secure data encapsulation mechanism (DEM), such as some authenticated encryption scheme (e.g. AES-GCM, AES-CCM, ChaCha20-Poly1305), results in a CCA secure hybrid PKE [71,69]. Further studies [23,38,12,39,78,45] gave tighter security bounds, accounted for decryption failures in the underlying PKE, and analyzed the security under quantum random oracle model (QROM). To this day, the Fujisaki-Okamoto transformation is the only known generic CCA secure transformation that can convert OW-CPA/IND-CPA PKE into a CCA secure KEM. Because of the minimal input requirement and the simple construction, the Fujisaki-Okamoto transformation was widely adopted among post-quantum KEM candidates submitted to the PQC standardization project, including Kyber [16], Saber [22], FrodoKEM [15], and Classic McEliece [11].

Despite its widespread adoption, the Fujisaki-Okamoto transformation has many flaws:

– **Computational inefficiency.** In all variants of Fujisaki-Okamoto transformation, decapsulation routine needs to re-encrypt the decryption to ensure ciphertext non-malleability. For input PKE whose encryption routine carries

significant computational cost, such as most lattice-based cryptosystems, re-encryption substantially slows down decapsulation.
- **Side-channel vulnerability.** Re-encryption introduces side channels that can leak information about the decrypted PKE plaintext. As demonstrated in [74,73,40,36,31,50,49,55,25,75], these side channels can be converted into efficient key-recovery and/or message-recovery attacks.
- **Security degradation.** *de-randomization* can degrade the security of a randomized PKE. Where the security parameters did not account for this loss, the security of the KEM can fall below the expected level. Consequently, larger parameters are necessary to account for the security loss, which slows down the cryptosystem [9,10].

Other generic CCA transformations Mihir Bellare and Phillip Rogaway proposed *Optimal Asymmetric Encryption Padding (OAEP)* in 1994 [7] and reduced its CCA security to the one-wayness of the underlying trapdoor permutation under the random oracle model. However, Victor Shoup identified a non-trivial gap in OAEP's security proof that cannot be filled under ROM [70], although Fujisaki et al. later proved that RSA-OAEP is secure under the RSA assumption [28]. RSA-OAEP is widely used in secure communication protocols such as TLS 1.2. The main drawback of OAEP is that it requires its input to be an one-way trapdoor permutation, which is difficult to find: to this day, RSA remains the only viable candidate to apply OAEP to.

Okamoto and Pointchval proposed REACT [61] in 2001, followed by GEM [19] in 2002. Both are generic CCA transformation with security proved under ROM. Okamoto and Pointcheval first defined the security notion of one-wayness under plaintext checking attack (OW-PCA) and reduced the CCA security of the transformation to the OW-PCA security of the input public-key cryptosystem. One can also draw similarity between the our scheme and the Diffie-Hellman Integrated Encryption Scheme (DHIES) [1,2]: both schemes protects the integrity of the ciphertext using a MAC whose key is derived from some PKE plaintext. However, our scheme generalizes the security requirement of the underlying PKE and offers a number of performance optimizations, which we will discuss in Section 3.4.

1.3 Paper organization

In Section 2, we review the preliminary definitions and theorems. In Section 3, we present the encrypt-then-MAC KEM transformation, prove its CCA security, and discuss practical attacks. In Section 4, we present `ML-KEM-EtM`, an instantiation of the encrypt-then-MAC transformation using subroutines from `ML-KEM`.

2 Preliminaries

Notations. For finite set S, we use $x \xleftarrow{\$} S$ to denote uniformly random sample from the set. For deterministic routine f, we use $x \leftarrow f()$ to denote assigning the

output of f to x. For randomized routine g, we use $x \xleftarrow{\$} g()$ to denote assigning the randomized output of g to x. For boolean statement B, we denote $[\![B]\!]$ to be 1 if B is true and 0 otherwise. For probabilistic Turing machine A, we denote access to oracle $\mathcal{O}$ by $A^{\mathcal{O}}$. We sometimes model hash function H as a random oracle, in which case we will use $\mathcal{L}^H$ to denote the record of input-output queries $(x, H(x))$ made to the oracle.

2.1 Public-key encryption scheme

Syntax A public-key encryption scheme (PKE) is a collection of three routines $(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ defined over some plaintext space $\mathcal{M}$ and some ciphertext space $\mathcal{C}$. Key generation $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$ is a randomized routine that returns a keypair consisting of a public encryption key and a secret decryption key. The encryption routine $\texttt{Enc} : (\texttt{pk}, m) \mapsto c$ encrypts the input plaintext m under the input public key $\texttt{pk}$ and produces a ciphertext c. The decryption routine $\texttt{Dec} : (\texttt{sk}, c) \mapsto m$ decrypts the input ciphertext c under the input secret key and produces the corresponding plaintext. Where the encryption routine is randomized, we denote the randomness by a coin $r \in \mathcal{R}$ where $\mathcal{R}$ is called the coin space. Decryption routines are assumed to always be deterministic.

Correctness A PKE is δ-correct if

$$E\left[\max_{m \in \mathcal{M}} P\left[\texttt{Dec}(\texttt{sk}, c) \neq m \mid c \xleftarrow{\$} \texttt{Enc}(\texttt{pk}, m)\right]\right] \leq \delta$$

Where the expectation is taken with respect to the probability distribution of all possible keypairs. For many lattice-based cryptosystems, decryption failures could leak information about the secret key, although the probability of a decryption failure is low enough that classical adversaries cannot exploit decryption failure more than they can defeat the underlying lattice problems.

Security The security of PKE's is conventionally discussed using adversarial games played between a challenger and an adversary [30]. In the OW-ATK game (Figure 1), the challenger samples a random keypair and a random encryption. The adversary is given the public key, the random encryption (also called the challenge ciphertext), and access to ATK, then asked to decrypt the challenge ciphertext.

The advantage of an adversary is its probability of producing the correct decryption: $\text{Adv}^{\text{OW-ATK}}_{\texttt{PKE}}(A) = P[\hat{m} = m^*]$. A PKE is said to be OW-ATK secure if no efficient adversary can win the OW-ATK game with non-negligible probability.

In the IND-ATK game (Figure 2), the adversary chooses two distinct messages and receives the encryption of one of them, randomly selected by the challenger. The advantage of an adversary is its probability of correctly distinguishing the ciphertext of one message from the other beyond blind guess:

OW-ATK game

1: $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda)$
2: $m^* \xleftarrow{\$} \mathcal{M}$
3: $c^* \xleftarrow{\$} \mathtt{Enc}(\mathtt{pk}, m)$
4: $\hat{m} \xleftarrow{\$} A^{\mathrm{ATK}}(1^\lambda, \mathtt{pk}, c^*)$
5: **return** $[\![\hat{m} = m^*]\!]$

Fig. 1: The one-wayness game: challenger samples a random keypair and a random encryption, and the adversary wins if it correctly produces the decryption

IND-ATK game

1: $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda)$
2: $(m_0, m_1) \xleftarrow{\$} A^{\mathrm{ATK}}(1^\lambda, \mathtt{pk})$
3: $b \xleftarrow{\$} \{0, 1\}$
4: $c^* \xleftarrow{\$} \mathtt{Enc}(\mathtt{pk}, m_b)$
5: $\hat{b} \xleftarrow{\$} A^{\mathrm{ATK}}(1^\lambda, \mathtt{pk}, c^*)$
6: **return** $[\![\hat{b} = b]\!]$

Fig. 2: IND-ATK game: adversary is asked to distinguish the encryption of one message from another

$\mathrm{Adv}^{\text{IND-ATK}}_{\mathtt{PKE}}(A) = |P[\hat{b} = b] - \frac{1}{2}|$. A PKE is said to be IND-ATK secure if no efficient adversary can win the IND-ATK game with non-negligible advantage.

In public-key cryptography, all adversaries are assumed to have access to the public key (ATK = CPA). If the adversary has access to a decryption oracle, it is said to mount chosen-ciphertext attack (ATK = CCA). If the adversary has access to a plaintext-checking oracle (PCO) , then it is said to mount plaintext-checking attack (ATK = PCA). See Figure 3 for algorithmic description of the oracles.

$$\mathrm{ATK} = \begin{cases} \mathrm{CPA} & \mathcal{O}^{\mathrm{ATK}} = \cdot \\ \mathrm{PCA} & \mathcal{O}^{\mathrm{ATK}} = \mathcal{O}^{\mathtt{PCO}} \\ \mathrm{CCA} & \mathcal{O}^{\mathrm{ATK}} = \mathcal{O}^{\mathtt{Dec}} \end{cases}$$

2.2 Key encapsulation mechanism (KEM)

Syntax A key encapsulation mechanism (KEM) is a collection of three routines $(\mathtt{KeyGen}, \mathtt{Encap}, \mathtt{Decap})$ defined over some ciphertext space $\mathcal{C}$ and some key space

$\mathcal{O}^{\mathtt{Dec}}(c)$
1: **return** $\mathtt{Dec}(\mathtt{sk}, c)$

$\mathcal{O}^{\mathtt{PCO}}(m, c)$
1: **return** $[\![\mathtt{Dec}(\mathtt{sk}, c) = m]\!]$

Fig. 3: Decryption oracle $\mathcal{O}^{\mathtt{Dec}}$ (left) answers decryption queries by returning the decryption of the queried ciphertext. Plaintext-checking oracle $\mathcal{O}^{\mathtt{PCO}}$ (right) answers whether the queried plaintext is the decryption of the queried ciphertext.

$\mathcal{K}$. Key generation $\mathtt{KeyGen} : 1^\lambda \mapsto (\mathtt{pk}, \mathtt{sk})$ is a randomized routine that returns a keypair. Encapsulation $\mathtt{Encap} : \mathtt{pk} \mapsto (c, K)$ is a randomized routine that takes a public encapsulation key and returns a pair of ciphertext c and shared secret K (also commonly referred to as session key). Decapsulation $\mathtt{Decap} : (\mathtt{sk}, c) \mapsto K$ is a deterministic routine that uses the secret key $\mathtt{sk}$ to recover the shared secret K from the input ciphertext c. Where the KEM chooses to reject invalid ciphertext explicitly, the decapsulation routine can also output the rejection symbol $\perp$. We assume a KEM to be perfectly correct:

$$P\left[\mathtt{Decap}(\mathtt{sk}, c) = K \mid (\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda); (c, K) \xleftarrow{\$} \mathtt{Encap}(\mathtt{pk})\right] = 1$$

Security Similar to PKE security, the security of KEM is discussed using adversarial games. In the IND-ATK game (Figure 4), the challenger generates a random keypair and encapsulates a random secret; the adversary is given the public key and the ciphertext, then asked to distinguish the shared secret from a random bit string.

KEM IND-ATK Game
1: $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}((1^\lambda))$
2: $(c^*, K_0) \xleftarrow{\$} \mathtt{Encap}(\mathtt{pk})$
3: $K_1 \xleftarrow{\$} \mathcal{K}$
4: $b \xleftarrow{\$} \{0, 1\}$
5: $\hat{b} \xleftarrow{\$} A^{\mathtt{ATK}}(1^\lambda, \mathtt{pk}, c^*, K_b)$
6: **return** $[\![\hat{b} = b]\!]$

Fig. 4: The IND-ATK game for KEM

The advantage of an adversary is its probability of winning beyong blind guess. A KEM is said to be IND-ATK secure if no efficient adversary can win the IND-ATK game with non-negligible advantage.

$$\mathtt{Adv}^{\mathtt{IND\text{-}ATK}}(A) = \left| P\left[A^{\mathtt{ATK}}(1^\lambda, c^*, K_b) = b \;\middle|\; \begin{array}{r} (\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda); \\ (c^*, K_0) \xleftarrow{\$} \mathtt{Encap}(\mathtt{pk}); \\ K_1 \xleftarrow{\$} \mathcal{K}; b \xleftarrow{\$} \{0,1\} \end{array} \right] - \frac{1}{2} \right|$$

By default, all adversaries are assumed to have the public key, with which they can mount chosen plaintext attacks (ATK = CPA). If the adversary as access to a decapsulation oracle $\mathcal{O}^{\mathtt{Decap}} : c \mapsto \mathtt{Decap}(\mathtt{sk}, c)$, it is said to mount a chosen-ciphertext attack (ATK = CCA).

Bounded CCA (qCCA) security Where the number of decapsulation queries q is fixed, the security notion is called *IND-qCCA* [21,41]. In many practical scenarios such as ephemeral key exchange (e.g. TLS 1.3 handshake [66]), each key pair is used at most once, which translates to a fixed limit on the number of decapsulation queries allowed in the adversarial games. In these scenarios we only require the KEM to be IND-1CCA secure [68,26].

2.3 Message authentication code (MAC)

Syntax A message authentication code (MAC) is a collection of two routines $(\mathtt{Sign}, \mathtt{Verify})$ defined over some key space $\mathcal{K}$, some message space $\mathcal{M}$, and some tag space $\mathcal{T}$. The signing routine $\mathtt{Sign} : (k, m) \mapsto t$ authenticates the message m under the symmetric key k by producing a tag t. The verification routine $\mathtt{Verify}(k, m, t)$ outputs 1 if the message-tag pair (m, t) is authentic under the symmetric key k and 0 otherwise. Many MAC constructions are deterministic: for these constructions it is simpler to denote the signing routine by $t \leftarrow \mathtt{MAC}(k, m)$, and verification done using a simple comparison. Some MAC constructions require a distinct or randomized nonce $r \xleftarrow{\$} \mathcal{R}$, and the signing routine will take this additional argument $t \leftarrow \mathtt{MAC}(k, m; r)$.

Security The standard security notion for a MAC is *existential unforgeability under chosen message attack (EUF-CMA)*. We define it using an adversarial game in which an adversary has access to a signing oracle $\mathcal{O}^{\mathtt{Sign}} : m \mapsto \mathtt{Sign}(k, m)$ and tries to produce a valid message-tag pair that has not been queried from the signing oracle (Figure 5).

The advantage of the adversary is the probability that it successfully produces a valid message-tag pair. A MAC is said to be EUF-CMA secure if no efficient adversary has non-negligible advantage. Some MACs are *one-time existentially unforgeable* (we call them one-time MAC), meaning that each secret key can be used to authenticate exactly one message. The corresponding security game is identical to the EUF-CMA game except for that the signing oracle will only answer up to one query.

MAC EUF-CMA game

1: $k^* \xleftarrow{\$} \mathcal{K}$
2: $(\hat{m}, \hat{t}) \xleftarrow{\$} A^{\mathtt{CMA}}()$
3: **return** $[\![\mathtt{Verify}(k^*, \hat{m}, \hat{t}) \wedge (\hat{m}, \hat{t}) \notin \mathcal{O}^{\mathtt{Sign}}]\!]$

Fig. 5: The signing oracle signs the queried message with the secret key. The adversary must produce a message-tag pair that has never been queried before

3 The encrypt-then-MAC transformation

In this section we present the encrypt-then-MAC KEM transformation. The transformation constructs an IND-CCA secure KEM using a OW-PCA secure PKE and an existentially unforgeable MAC. Our scheme is inspired by DHIES, but differs from it in two key aspects: whereas DHIES reduces its CCA security specifically to the Gap Diffie-Hellman assumption [60], our construction's CCA security reduces generically to the PCA security of the input PKE; in addition, we argue that if the PKE's plaintext space is large and the sampling method has sufficient entropy, then the MAC only needs to be one-time existentially unforgeable (Abdalla, Rogaway, and Bellare originally proposed to use HMAC and CBC-MAC, which are many-time secure MAC but less efficient than one-time MAC). The data flow of the encapsulation is illustrated in Figure 6.

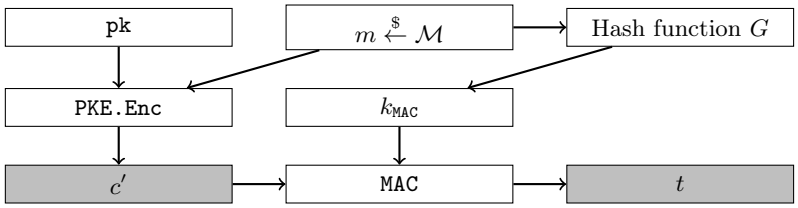

Fig. 6: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

In Section 3.1 we will describe the encrypt-then-MAC KEM routines and state the security reduction. In Section 3.2 we present the proof reducing the IND-CCA security of the KEM tightly to the OW-PCA security of the underlying PKE and non-tightly to the unforgeability of the MAC. In Section 3.3 we discuss how the OW-PCA security of the PKE relates to the CCA security of the encrypt-then-MAC KEM. In Section 3.4 we discuss some additional implementation details.

3.1 The construction

Let $\mathcal{B}^*$ denote the set of finite bit strings. Let $\mathcal{K}_{\texttt{KEM}}$ denote the set of all possible shared secrets. Let $(\texttt{KeyGen}_{\texttt{PKE}}, \texttt{Enc}_{\texttt{PKE}}, \texttt{Dec}_{\texttt{PKE}})$ be a PKE defined over message space $\mathcal{M}_{\texttt{PKE}}$ and ciphertext space $\mathcal{C}_{\texttt{PKE}}$. Let $\texttt{MAC} : \mathcal{K}_{\texttt{MAC}} \times \mathcal{B}^* \to \mathcal{T}$ be a MAC over key space $\mathcal{K}_{\texttt{MAC}}$ and tag space $\mathcal{T}$. Let $G : \mathcal{B}^* \to \mathcal{K}_{\texttt{MAC}}, H : \mathcal{B}^* \to \mathcal{K}_{\texttt{KEM}}$ be hash functions. The encrypt-then-MAC transformation $\texttt{EtM}[\texttt{PKE}, \texttt{MAC}, G, H]$ constructs a KEM $(\texttt{KeyGen}_{\texttt{EtM}}, \texttt{Encap}_{\texttt{EtM}}, \texttt{Decap}_{\texttt{EtM}})$ (Figure 7).

$\texttt{KeyGen}_{\texttt{EtM}}()$

1: $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}_{\texttt{PKE}}()$
2: $s \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$
3: $\texttt{sk} \leftarrow (\texttt{sk}, s)$
4: **return** $(\texttt{pk}, \texttt{sk})$

$\texttt{Encap}_{\texttt{EtM}}(\texttt{pk})$

1: $m \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$
2: $k \leftarrow G(m)$
3: $c' \xleftarrow{\$} \texttt{Enc}_{\texttt{PKE}}(\texttt{pk}, m)$
4: $t \leftarrow \texttt{MAC}(k, c')$
5: $c \leftarrow (c', t)$
6: $K \leftarrow H(m, c)$
7: **return** (c, K)

$\texttt{Decap}_{\texttt{EtM}}(\texttt{sk}, c)$

Require: $c = (c', t)$
Require: $\texttt{sk} = (\texttt{sk}', s)$
1: $\hat{m} \leftarrow \texttt{Dec}_{\texttt{PKE}}(\texttt{sk}', c')$
2: $\hat{k} \leftarrow G(\hat{m})$
3: **if** $\texttt{MAC}(\hat{k}, c') = t$ **then**
4: $\quad K \leftarrow H(\hat{m}, c)$
5: **else**
6: $\quad K \leftarrow H(s, c)$
7: **end if**
8: **return** K

Fig. 7: The encrypt-then-MAC KEM routines

We chose to construct $\texttt{KEM}_{\texttt{EtM}}$ using implicit rejection $K \leftarrow H(s, c)$: on invalid ciphertexts, the decapsulation routine returns a fake shared secret that depends on the ciphertext and some secret values, though choosing to use explicit rejection should not impact the security of the KEM. In addition, because the underlying PKE can be randomized, the shared secret $K \leftarrow H(m, c)$ must depend on both the plaintext and the ciphertext. According to [23,38], if the input PKE is *rigid* (i.e. $m = \texttt{Dec}(\texttt{sk}, c)$ if and only if $c = \texttt{Enc}(\texttt{pk}, m)$), such as with RSA, then the shared secret may be derived from the plaintext alone $K \leftarrow H(m)$.

The CCA security of $\texttt{KEM}_{\texttt{EtM}}$ can be intuitively argued through an adversary's inability to learn additional information from the decapsulation oracle. For an adversary A to produce a valid tag for some unauthenticated ciphertext c', it must either know the correct symmetric key or produce a forgery. Under the Random Oracle Model (ROM), A cannot know the symmetric key without knowing its pre-image under the hash function G, so A must either produced c' honestly, or have broken the one-wayness of the underlying PKE. This means that the decapsulation oracle will not leak information on decryption that the adversary does not already know. We formalize the security in Theorem 1

Theorem 1. *For every IND-CCA adversary A against $\texttt{KEM}_{\texttt{EtM}}$ that makes q decapsulation queries, there exists a OW-PCA adversary B against the underlying*

PKE making at least q decapsulation queries, and an existential forgery adversary C against the underlying MAC such that:

$$\mathit{Adv}^{\mathit{IND\text{-}CCA}}_{\mathit{KEM}_{\mathit{EtM}}}(A) \leq q \cdot \mathit{Adv}_{\mathit{MAC}}(C) + 2 \cdot \mathit{Adv}^{\mathit{OW\text{-}PCA}}_{\mathit{PKE}}(B)$$

3.2 Proof of Theorem 1

We will prove Theorem 1 using a sequence of game. A summary of the the sequence of games can be found in Figure 8 and 9. From a high level we made three incremental modifications to the IND-CCA game for $\mathtt{KEM}_{\mathtt{EtM}}$:

1. Replace the true decapsulation oracle with a simulated decapsulation oracle. The simulated decapsulation oracle does not directly decrypt the queried ciphertext. Instead it searches through the hash oracle and looks for matching queries using the plaintext-checking oracle. The true decapsulation oracle and the simulated decapsulation oracle disagree if and only if the adversary queries with a ciphertext that contains a forged MAC tag, so the adversary cannot distinguish the two games more than it can perform existential forgery against the underlying MAC.
2. Replace the pseudorandom MAC key $k^* \leftarrow G(m^*)$ with a uniformly random MAC key $k^* \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$. Under ROM, the adversary cannot distinguish this game from the previous one unless it queries the hash oracle with m^*, in which case a second adversary with access to a plaintext-checking oracle can win the OW-PCA game against the underlying PKE.
3. Replace the pseudorandom shared secret $K_0 \leftarrow H(m^*, c)$ with a truly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$. Similar to the point above, the adversary cannot distinguish the change more than it can break the one-wayness of the underlying PKE. Furthermore, since both K_0 and K_1 are uniformly random, no adversary can have any advantage.

A OW-PCA adversary can then simulate the modified IND-CCA game for the KEM adversary, and the advantage of the OW-PCA adversary is associated with the probability of certain behaviors of the KEM adversary.

Proof. Game 0 is the standard KEM IND-CCA game. The decapsulation oracle $\mathcal{O}^{\mathtt{Decap}}$ executes the decapsulation routine using the challenge keypair and return the results faithfully. The queries made to the hash oracles $\mathcal{O}^G, \mathcal{O}^H$ are recorded to their respective tapes $\mathcal{L}^G, \mathcal{L}^H$.

Game 1 is identical to game 0 except that the true decapsulation oracle $\mathcal{O}^{\mathtt{Decap}}$ is replaced with a simulated oracle $\mathcal{O}_1^{\mathtt{Decap}}$. Instead of directly decrypting c' as in the decapsulation routine, the simulated oracle searches through the tape $\mathcal{L}^G$ to find a matching query $(\tilde{m}, \tilde{k})$ such that $\tilde{m}$ is the decryption of c'. The simulated oracle then uses $\tilde{k}$ to validate the tag t against c'.

If the simulated oracle accepts the queried ciphertext as valid, then there is a matching query that also validates the tag, which means that the queried ciphertext is honestly generated. Therefore, the true oracle must also accept

$\texttt{IND-CCA}$ game for $\texttt{KEM}_{\texttt{EtM}}$

$(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}_{\texttt{EtM}}()$
$m^* \xleftarrow{\$} \mathcal{M}$
$c' \xleftarrow{\$} \texttt{Enc}_{\texttt{PKE}}(\texttt{pk}, m^*)$
$k^* \leftarrow G(m^*)$ ▷ Game 0-1
$k^* \xleftarrow{\$} \mathcal{K}_{\texttt{MAC}}$ ▷ Game 2-3
$t \leftarrow \texttt{MAC}(k^*, c')$
$c^* \leftarrow (c', t)$
$K_0 \leftarrow H(m^*, c^*)$ ▷ Game 0-2
$K_0 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$ ▷ Game 3
$K_1 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$
$b \xleftarrow{\$} \{0, 1\}$
$\hat{b} \leftarrow A^{\mathcal{O}^{\texttt{Decap}}}(\texttt{pk}, c^*, K_b)$ ▷ Game 0
$\hat{b} \leftarrow A^{\mathcal{O}_1^{\texttt{Decap}}}(\texttt{pk}, c^*, K_b)$ ▷ Game 1-3
return $[\![\hat{b} = b]\!]$

Hash oracle $\mathcal{O}^G(m)$

if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \tilde{m} = m$ **then**
  return $\tilde{k}$
end if
$k \xleftarrow{\$} \mathcal{K}_{\texttt{MAC}}$
$\mathcal{L}^G \leftarrow \mathcal{L}^G \cup \{(m, k)\}$
return k

Decap oracle $\mathcal{O}^{\texttt{Decap}}(c)$

$(c', t) \leftarrow c$
$\hat{m} = \texttt{Dec}_{\texttt{PKE}}(\texttt{sk}', c')$
$\hat{k} \leftarrow G(\hat{m})$
if $\texttt{MAC}(\hat{k}, c') = t$ **then**
  $K \leftarrow H(\hat{m}, c)$
else
  $K \leftarrow H(z, c)$
end if
return K

$\mathcal{O}_1^{\texttt{Decap}}(c)$

$(c', t) \leftarrow c$
if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \tilde{m} = \texttt{Dec}_{\texttt{PKE}}(\texttt{sk}', c') \wedge \texttt{MAC}(\tilde{k}, c') = t$ **then**
  $K \leftarrow H(\tilde{m}, c)$
else
  $K \leftarrow H(z, c)$
end if
return K

$\mathcal{O}^H(m, c)$

if $\exists(\tilde{m}, \tilde{c}, \tilde{K}) \in \mathcal{L}^H : \tilde{m} = m \wedge \tilde{c} = c$ **then**
  return $\tilde{K}$
end if
$K \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$
$\mathcal{L}^H \leftarrow \mathcal{L}^H \cup \{(m, c, K)\}$
return K

Fig. 8: Sequence of games in the proof of Theorem 1

the queried ciphertext. On the other hand, if the true oracle rejects the queried ciphertext, then the tag is simply invalid under the MAC key $k = G(\texttt{Dec}(\texttt{sk}', c'))$. Therefore, there could not have been a matching query that also validates the tag, and the simulated oracle must also rejects the queried ciphertext.

This means that from the adversary A's perspective, game 1 and game 0 differ only when the true oracle accepts while the simulated oracle rejects, which means that t is a valid tag for c' under $k = G(\texttt{Dec}(\texttt{sk}', c'))$, but k has never been queried. Under the random oracle model, such k is a uniformly random sample of $\mathcal{K}_{\texttt{MAC}}$ that the adversary does not know, so for A to produce a valid tag is to produce a forgery against the $\texttt{MAC}$ under an unknown and uniformly random key. Therefore, we can bound the probability that the true decapsulation oracle

$B(\mathtt{pk}, c'^*)$

1: $z \xleftarrow{\$} \mathcal{M}$
2: $k \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$
3: $t \leftarrow \mathtt{MAC}(k, c'^*)$
4: $c^* \leftarrow (c'^*, t)$
5: $K \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$
6: $\hat{b} \leftarrow A^{\mathcal{O}_B^{\mathtt{Decap}}, \mathcal{O}_B^G, \mathcal{O}_B^H}(\mathtt{pk}, c^*, K)$
7: **if** $\mathtt{ABORT}(m)$ **then**
8: **return** m
9: **end if**

$\mathcal{O}_B^H(m, c)$

if $\mathcal{O}^{\mathtt{PCO}}(m, c'^*) = 1$ **then**
$\mathtt{ABORT}(m)$
end if
if $\exists(\tilde{m}, \tilde{c}, \tilde{K}) \in \mathcal{L}^H : \tilde{m} = m \wedge \tilde{c} = c$ **then**
return $\tilde{K}$
end if
$K \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$
$\mathcal{L}^H \leftarrow \mathcal{L}^H \cup \{(m, c, K)\}$
return K

$\mathcal{O}_B^{\mathtt{Decap}}(c)$

1: $(c', t) \leftarrow c$
2: **if** $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \mathcal{O}^{\mathtt{PCO}}(\tilde{m}, c') = 1 \wedge \mathtt{MAC}(\tilde{k}, c') = t$ **then**
3: $K \leftarrow H(\tilde{m}, c)$
4: **else**
5: $K \leftarrow H(z, c)$
6: **end if**
7: **return** K

$\mathcal{O}_B^G(m)$

1: **if** $\mathcal{O}^{\mathtt{PCO}}(m, c'^*) = 1$ **then**
2: $\mathtt{ABORT}(m)$
3: **end if**
4: **if** $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \tilde{m} = m$ **then**
5: **return** $\tilde{k}$
6: **end if**
7: $k \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$
8: $\mathcal{L}^G \leftarrow \mathcal{L}^G \cup \{(m, k)\}$
9: **return** k

Fig. 9: OW-PCA adversary B simulates game 3 for IND-CCA adversary A in the proof for Theorem 1

disagrees with the simulated oracle by the probability that some MAC adversary produces a forgery:

$$P\left[\mathcal{O}^{\mathtt{Decap}}(c) \neq \mathcal{O}_1^{\mathtt{Decap}}(c)\right] \leq \mathtt{Adv}_{\mathtt{MAC}}(C).$$

Across all q decapsulation queries, the probability that at least one query is a forgery is thus at most $q \cdot P\left[\mathcal{O}^{\mathtt{Decap}}(c) \neq \mathcal{O}_1^{\mathtt{Decap}}(c)\right]$. By the difference lemma:

$$|\mathtt{Adv}_{G_0}(A) - \mathtt{Adv}_{G_1}(A)| \leq q \cdot \mathtt{Adv}_{\mathtt{MAC}}(C).$$

Game 2 is identical to game 1, except that the challenger samples a uniformly random MAC key $k^* \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$ instead of deriving it from m^*. From A's perspective the two games are indistinguishable, unless A queries G with the value of m^*. Denote the probability that A queries G with m^* by $P[\mathtt{QUERY\ G}]$, then:

$$|\mathtt{Adv}_{G_1}(A) - \mathtt{Adv}_{G_2}(A)| \leq P\left[\mathtt{QUERY\ G}\right].$$

Game 3 is identical to game 2, except that the challenger samples a uniformly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$ instead of deriving it from m^* and t. From A's perspective the two games are indistinguishable, unless A queries H with $(m^*, \cdot)$. Denote the probability that A queries H with $(m^*, \cdot)$ by $P[\texttt{QUERY H}]$, then:

$$|\texttt{Adv}_{G_2}(A) - \texttt{Adv}_{G_3}(A)| \leq P\left[\texttt{QUERY H}\right].$$

Since in game 3, both K_0 and K_1 are uniformly random and independent of all other variables, no adversary can have any advantage: $\texttt{Adv}_{G_3}(A) = 0$.

We will bound $P[\texttt{QUERY G}]$ and $P[\texttt{QUERY H}]$ by constructing a OW-PCA adversary B against the underlying PKE that uses A as a sub-routine. B's behaviors are summarized in Figure 9.

B simulates game 3 for A: upon receiving the public key $\texttt{pk}$ and challenge encryption c'^*, B samples random MAC key and session key to produce the challenge encapsulation, then feeds it to A. When simulating the decapsulation oracle, B uses the plaintext-checking oracle to look for matching queries in $\mathcal{L}^G$. When simulating the hash oracles, B uses the plaintext-checking oracle to detect when $m^* = \texttt{Dec}(\texttt{sk}', c'^\star)$ has been queried. When m^* is queried, B terminates A and returns m^* to win the OW-PCA game. In other words:

$$P\left[\texttt{QUERY G}\right] \leq \texttt{Adv}^{\texttt{OW-PCA}}_{\texttt{PKE}}(B),$$
$$P\left[\texttt{QUERY H}\right] \leq \texttt{Adv}^{\texttt{OW-PCA}}_{\texttt{PKE}}(B).$$

Combining all equations above produce the desired security bound.

3.3 OW-PCA security

In this section, we consider the OW-PCA security of various well-known PKE schemes and discuss the suitability of applying the encrypt-then-MAC KEM transformation.

The security notion of *one-wayness under plaintext-checking attack* (OW-PCA) was introduced by Okamoto and Pointcheval in [61], where the authors reduced the security of a generic CCA secure transformation (REACT) to the OW-PCA security of the input public-key cryptosystem. Following REACT, Pointcheval et al. proposed GEM [19], another generic CCA secure transformation whose security reduces to the OW-PCA security of the underlying PKE. Around the time REACT and GEM were published, the best known CCA secure transformation is Optimal Asymmetric Encryption Padding (OAEP)[7]. Compared to OAEP's requirement for one-way trapdoor permutation, OW-PCA security is easier to achieve: any one-way trapdoor permutation (such as RSA) is automatically OW-PCA secure, while there are cryptosystems that are OW-PCA secure but not one-way trapdoor permutation (Table 2). More recently, the modular Fujisaki-Okamoto transformation [38] proposed CCA secure KEM whose security reduces to the OW-PCA security of the input PKE under both ROM and QROM.

Theorem 1 stated that if the underlying PKE is OW-PCA secure, then the encrypt-then-MAC KEM is IND-CCA secure. Conversely, if the underlying PKE is not OW-PCA secure, then the encrypt-then-MAC KEM is not IND-CCA secure. This is captured in Lemma 1

Lemma 1. *For every q-query OW-PCA adversary A against the underlying PKE, there exists a q-query IND-CCA adversary B against the encrypt-then-MAC KEM such that:*

$$\mathbf{Adv}^{\mathit{IND\text{-}CCA}}_{\mathit{KEM}_{\mathit{Etm}}}(B) = \mathbf{Adv}^{\mathit{OW\text{-}PCA}}_{\mathit{PKE}}(A)$$

For a sketch of proof, we observe that the IND-CCA adversary B can perfectly simulate the plaintext-checking oracle for the OW-PCA adversary A (Figure 10), and if A succeeds in breaking the one-wayness of the underlying PKE, then B can compute the shared secret associated with the challenge ciphertext, which allows B to distinguish true shared secret from random bit strings.

$\mathcal{O}^{\mathsf{PCO}}_{\mathsf{Decap}}(m, c')$

1: $k \leftarrow G(m)$
2: $t \leftarrow \mathtt{MAC}(k, c')$
3: $c \leftarrow (c', t)$
4: $K \leftarrow H(m, c)$
5: **return** $[\![\mathcal{O}^{\mathsf{Decap}}(c) = K]\!]$

Fig. 10: Plaintext-checking oracle of the underlying PKE can be simulated using decapsulation oracle of the encrypt-then-MAC KEM. Let c' denote some PKE ciphertext. If m is the decryption of c', then m will hash into the correct MAC key and produce the correct tag t, and the decapsulation oracle will accept the KEM ciphertext $c = (c', t)$ and return the true shared secret $K = H(m, c)$. If m is not the decryption of c', then the probability of producing the correct tag is negligible, and the decapsulation oracle will reject (c', t).

For the remainder of this section, we will review the OW-PCA security of some well-known cryptosystems. A summary can be found in Table 2.

Number-theoretic cryptosystems The RSA cryptosystem as it was originally proposed in [67] is OW-PCA secure. This is because the RSA cryptosystem is a one-way trapdoor permutation [28], meaning that encryption and decryption are both injective. Given some public key $(N = pq, e)$ and a plaintext-ciphertext pair (m, c), one can check that m is the decryption of c by checking that $c \equiv m^e \mod N$, which renders a plaintext-checking oracle completely useless.

The ElGamal cryptosystem [29] is OW-PCA secure if the Gap Diffie-Hellman assumption [60] holds for the underlying group (finite field or elliptic curve). This is the basis on which the CCA security of DHIES [2] is proved.

PKE	Category	Is it OW-PCA?
RSA	Number theoretic	Yes [23,71]
ElGamal	Number theoretic or elliptic curve	If Gap Diffie-Hellman assumption holds [2,60]
FrodoKEM	Lattice-based	No, full key recovery [33,4]
Kyber/Saber	Lattice-based	No, full key recovery [40,77,35]
NTRU	Lattice-based	No, full key recovery [37,43,74]
HQC	Code-based	No, full key recovery [40,4]
BIKE	Code-based	No, partial key recovery [34]
Classic McEliece	Code-based	Unknown, but there is no known attack [74]

Table 2: The landscape of OW-PCA security

Lattice-based cryptosystems In many prominent cryptosystems based on the Learning with Error (LWE) problems, the plaintext is encrypted by adding noise. If an adversary adds additional noise to the ciphertext, the modified ciphertext may or may not decrypt back to the same plaintext. Querying the plaintext-checking oracle with modified ciphertexts containing varying amount of noise thus allows an adversary to recover the secret key. Such key-recovery plaintext-checking attacks (KR-PCA) were described for FrodoKEM in [33,4]. Similar attacks for Kyber and Saber were described in [40,77,35]. As Peikert pointed out in [63], because of the search-decision equivalence of the (Ring) Learning With Error problem, lattice-based cryptosystems are unlikely to have inherent OW-PCA security.

NTRU is another family of lattice-based cryptosystems. Hoffstein and Silverman [37] proposed KR-PCA for the original NTRU cryptosystem, which Ueno et al. [74] adapted into a KR-PCA against modern instantiations such as NTRU-HPS and NTRU-HRSS (also see [79]). Ueno et al. also adapted [43] into a KR-PCA against NTRU-Prime.

Code-based cryptosystems HQC [53], despite being based on hard coding problems, is structurally similar to lattice-based cryptosystems Kyber, Saber, etc. [74]. Consequently, KR-PCA for lattice cryptosystems can be easily adapted to work on HQC [40,4]. BIKE [3] is based on the Niederreiter cryptosystem instantiated with quasi-cyclic moderate density parity check (QC-MDPC) code [54]. [34] described KR-PCA against QC-MDPC code, which can partially recover BIKE secret keys [77]. There is no known adaptive attack against Classic McEliece [74], although the decoding subroutine of Classic McEliece is significantly slower than the encoding subroutine, so applying the encrypt-then-MAC transformation will not yield meaningful performance improvements.

3.4 Additional implementation notes

Securely deriving MAC key Because the MAC key is pseudorandomly derived from the PKE plaintext (instead of uniformly and independently sampled), it is possible to construct a large lookup table mapping each PKE plaintext to a MAC key, then check the KEM ciphertext (c, t) against each MAC key to recover the pre-image. As was pointed out in [10,9], this relationship between the MAC tag and the PKE plaintext could damage the security of the scheme. This can be mitigated by including the PKE public key when deriving the MAC key $k_{\texttt{MAC}} \leftarrow G(\texttt{pk}, m)$ or even including random salt at each encapsulation $k_{\texttt{MAC}} \leftarrow G(\texttt{pk}, m, \text{salt})$, which can increase the cost of such dictionary attacks.

One-time MAC When the encrypt-then-MAC KEM is instantiated with a PKE with a sufficiently large plaintext space, we expect the probability of two encapsulations sampling the same PKE plaintext to be negligible within some reasonable keypair lifetime, and if the hash function is collision resistant, then the probability of two encapsulations deriving the same MAC key is also negligible. Therefore, the encrypt-then-MAC KEM can be instantiated with one-time MACs [18] with no security impact. On the other hand, one-time MACs can be computationally more efficient than many-time secure MACs.

Deriving shared secret If the decryption routine of the underlying PKE is not injective, then the shared secret must be derived from both the PKE plaintext and the ciphertext: if the shared secret is derived from the plaintext alone, and the KEM adversary A can find a modified ciphertext that decrypts back to the same PKE plaintext, then A can query the decapsulation oracle with the modified ciphertext and obtain the true decapsulation. However, the shared secret does not have to be derived from hashing the entire ciphertext. Instead, we propose to derive the shared secret from the MAC tag, which is functionally equivalent to a keyed hash of the ciphertext. Since the MAC tag is usually much smaller than the ciphertext, hashing the tag instead of the entire ciphertext can lead to a noticeable speedup.

4 Applying encrypt-then-MAC to `ML-KEM`

`ML-KEM` is an IND-CCA secure post-quantum KEM standardized by NIST in FIPS 203 [59]. The chosen-ciphertext security of `ML-KEM` is achieved in two steps. First, `ML-KEM` constructs a PKE whose IND-CPA security reduces to the conjectured intractability of the decisional Module Learning with Error (MLWE) problem [64,52,65,62] (see Figure 12 in Appendix). Then, `ML-KEM` applies the Fujisaki-Okamoto transformation to convert the IND-CPA secure PKE into an IND-CCA secure KEM (see Figure 13 in Appendix).

4.1 `ML-KEM` under encrypt-then-MAC

We apply the encrypt-then-MAC transformation to the PKE routines of `ML-KEM` [59]. The resulting KEM is denoted as `ML-KEM-EtM` (Figure 11). The key gener-

ation routines are identical between ML-KEM-EtM and ML-KEM. The ML-KEM-EtM encapsulation and decapsulation algorithms make use of two additional algorithms, namely a MAC and a KDF.

In Section 3.3, we mentioned that ML-KEM's PKE subroutines are not OW-PCA secure. Consequently, applying the encrypt-then-MAC KEM transformation to the PKE subroutines will not achieve full IND-CCA security. However, the plaintext-checking attack against ML-KEM requires several thousands of oracle queries, which translates to equivalent complexity of any chosen-ciphertext attacks. Therefore, ML-KEM-EtM achieves IND-1CCA security and is suitable for use in ephemeral key exchange such as in post-quantum TLS 1.3 key exchange [68,41,80,44]. In ML-KEM, the encryption subroutine is significantly slower than the decryption subroutine, so the encrypt-then-MAC transformation achieves significant performance improvements over the Fujisaki-Okamoto transformation.

$\texttt{KeyGen}_{\texttt{ML-KEM-EtM}}()$

1: $z \xleftarrow{\$} \{0,1\}^{256}$
2: $(\texttt{pk}, \texttt{sk}') \xleftarrow{\$} \texttt{KeyGen}_{\texttt{K-PKE}}()$
3: $h \leftarrow H(\texttt{pk})$
4: $\texttt{sk} \leftarrow (sk' \| \texttt{pk} \| h \| z)$
5: **return** $(\texttt{pk}, \texttt{sk})$

$\texttt{Encap}_{\texttt{ML-KEM-EtM}}(\texttt{pk})$

1: $m \xleftarrow{\$} \{0,1\}^{256}$
2: $h \leftarrow H(\texttt{pk})$
3: $(\overline{K}, k) \leftarrow G(m \| h)$
4: $c' \xleftarrow{\$} \texttt{Enc}_{\texttt{K-PKE}}(\texttt{pk}, m)$
5: $t \leftarrow \texttt{MAC}(k, c')$
6: $K \leftarrow \texttt{KDF}(\overline{K} \| t)$
7: $c \leftarrow (c', t)$
8: **return** (c, K)

$\texttt{Decap}_{\texttt{ML-KEM-EtM}}(\texttt{sk}, c)$

1: $(\texttt{sk}', \texttt{pk}, h, z) \leftarrow \texttt{sk}$
2: $(c', t) \leftarrow c$
3: $\hat{m} \leftarrow \texttt{Dec}_{\texttt{K-PKE}}(\texttt{sk}', c')$
4: $(\overline{K}, \hat{k}) \leftarrow G(\hat{m} \| h)$
5: $\hat{t} \leftarrow \texttt{MAC}(\hat{k}, c')$
6: **if** $\hat{t} = t$ **then**
7: $\quad K \leftarrow \texttt{KDF}(\overline{K} \| t)$
8: **else**
9: $\quad K \leftarrow \texttt{KDF}(z \| c)$
10: **end if**
11: **return** K

Fig. 11: ML-KEM-EtM applies the encrypt-then-MAC KEM transformation to the PKE subroutines of ML-KEM.

4.2 Choosing MAC

For a concrete instantiation of ML-KEM-EtM, we chose four MACs covering a variety of design architectures. All MACs are parameterized with a 256-bit key and 128-bit tag.

Poly1305 [8] and GMAC [56] are both Carter-Wegman style MACs [18,76], which compute the tag using finite field arithmetic. It first parses the message into a sequence of finite field elements, then evalutes a polynomial whose coefficients are the message blocks and whose indeterminate is the secret key. Specifically,

Poly1305 operates in the prime field $\mathbb{F}_q$ where $q = 2^{130} - 5$. GMAC operates in the binary extension field $\mathbb{F}_{2^{128}}$.

Remark. In our implementation, we used OpenSSL 3.3.1's EVP_MAC interface. Within this interface, the Poly1305 implementation does not include a nonce and is thus only one-time secure. On the other hand, GMAC is implemented by passing all data into the associated data field of the authenticated encryption scheme AES-256-GCM. Assuming that nonce does not repeat within the lifetime of the symmetric key, GMAC is many-time secure.

CMAC [58] is based on the CBC-MAC [13]. To compute a CMAC tag, the message is first padded and parsed into blocks. Each block is first XOR'ed with the previous block's output, then encrypted under a block cipher using the secret key. The final output is XOR'ed with a sub-key derived from the secret key before being encrypted for one last time. In our implementation, the block cipher is instantiated with AES-256, which makes it particularly suitable for embedded devices with constrainted computing capacity but hardware support for AES. CMAC is many-time secure.

KMAC256 [47] is a pseudorandom function and keyed hash function based on KECCAK [57], although unlike the extendable output function (XOF) SHAKE, altering the requested output length generates a new unrelated output. KMAC256 is slower than other MACs, but it is the only construction with a flexible key and tag length.

Remark There are new lightweight MACs such as EliMAC [24], LightMAC [51], and ZMAC [42], which may offer a better performance than the MACs listed above. However, the performance of the encrypt-then-MAC KEM is dominated by the public-key cryptographic operations and the impact of the MAC is insignificant.

4.3 Experimental results

Our implementation of ML-KEM-EtM extended from Kyber's reference implementation [1]. Source code is compiled with GCC 11.4.1 and OpenSSL 3.0.8. All binaries are executed on an AWS c7a.medium instance (AMD EPYC 9R14 CPU at 3.7 GHz and 1 GB of RAM) in the us-west-2 region. We measured the CPU cycles count of the encapsulation/decapsulation routines of ML-KEM-EtM using the RDTSC instruction.

Compared to the FO transformation used in ML-KEM, the encrypt-then-MAC transformation achieves a massive CPU cycle count reduction in decapsulation while incurring only a minimal increase of encapsulation cycle count and ciphertext size. Since K-PKE.Enc carries significantly more computational complexity than K-PKE.Dec or any MAC we chose, the performance advantage of the encrypt-then-MAC transformation over the Fujisaki-Okamoto transformation is

[1] https://github.com/pq-crystals/kyber

Parameters	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
`ML-KEM-512`	91,467	92,065	121,185	121,650
`ML-KEM-EtM-512` + `Poly1305`	93,157	93,626	33,733	33,908
`ML-KEM-EtM-512` + `GMAC`	97,369	97,766	37,725	37,831
`ML-KEM-EtM-512` + `CMAC`	99,739	99,959	40,117	39,943
`ML-KEM-EtM-512` + `KMAC256`	101,009	101,313	40,741	40,916

Parameters	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
`ML-KEM-768`	136,405	147,400	186,445	187,529
`ML-KEM-EtM-768` + `Poly1305`	146,405	146,860	43,315	43,463
`ML-KEM-EtM-768` + `GMAC`	149,525	150,128	46,513	46,706
`ML-KEM-EtM-768` + `CMAC`	153,139	153,735	49,841	50,074
`ML-KEM-EtM-768` + `KMAC256`	155,219	155,848	52,415	52,611

Parameters	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
`ML-KEM-1024`	199,185	199,903	246,245	247,320
`ML-KEM-EtM-1024` + `Poly1305`	205,763	206,499	51,375	51,562
`ML-KEM-EtM-1024` + `GMAC`	208,805	209,681	54,573	54,780
`ML-KEM-EtM-1024` + `CMAC`	213,667	214,483	59,175	59,408
`ML-KEM-EtM-1024` + `KMAC256`	216,761	217,468	62,269	62,516

Table 3: CPU cycles of each KEM routine

dominated by the runtime saving gained from replacing *re-encryption* with MAC. A timing comparison between the `ML-KEM` and the variations of the `ML-KEM-EtM` is provided in Table 3. Because the CPU cycle count reduction in decapsulation is significantly more than the overhead in encapsulation and the increase in ciphertext size, `ML-KEM-EtM` can be used to implement efficient ephemeral key exchange. Compared to key exchange using `ML-KEM`, `ML-KEM-EtM` key exchange saves on average 29%-48% network round trip time (see Appendix B).

5 Concluding remarks

In this paper we presented "encrypt-then-MAC", a KEM constructed from a PKE and a MAC. We reduced the IND-CCA security of the KEM tightly to the OW-PCA security of the underlying PKE and non-tightly to the security of the underlying MAC. We also analyzed generic attacks on the KEM and proposed countermeasures. We then proposed `ML-KEM-EtM`, an IND-1CCA secure KEM derived from applying the encrypt-then-MAC transformation to the PKE subroutines of `ML-KEM`. Compared to `ML-KEM`, `ML-KEM-EtM` replaces the expensive re-encryption with computing a MAC tag. At the cost of minimal increase in encapsulation cost and ciphertext size, `ML-KEM-EtM` achieves substantially faster

decapsulation, which makes it a great candidate for constructing efficient post-quantum key exchange protocols.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHAES: an encryption scheme based on the Diffie-Hellman problem. IACR Cryptol. ePrint Arch. p. 7 (1999), http://eprint.iacr.org/1999/007
2. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2020, pp. 143–158. Springer (2001). https://doi.org/10.1007/3-540-45353-9_12, https://doi.org/10.1007/3-540-45353-9_12
3. Aragon, N., Barreto, P.L., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., Melchor, C.A., Misoczki, R., Persichetti, E., Richter-Brockmann, J., Sendrier, N., Tillich, J.P., Vasseur, V., Zémor, G.: NIST post-quantum cryptography standardization round 4 submission: BIKE: Bit flipping key encapsulation. Tech. rep., National Institute of Standards and Technology (NIST) (2024), https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf
4. Baetu, C., Durak, F.B., Huguenin-Dumittan, L., Talayhan, A., Vaudenay, S.: Misuse attacks on post-quantum cryptosystems. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11477, pp. 747–776. Springer (2019). https://doi.org/10.1007/978-3-030-17656-3_26, https://doi.org/10.1007/978-3-030-17656-3_26
5. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: Vitter, J.S. (ed.) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998. pp. 419–428. ACM (1998). https://doi.org/10.1145/276698.276854, https://doi.org/10.1145/276698.276854
6. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000). https://doi.org/10.1007/3-540-44448-3_41, https://doi.org/10.1007/3-540-44448-3_41
7. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Lecture Notes in Computer Science, vol. 950, pp. 92–111. Springer (1994). https://doi.org/10.1007/BFB0053428, https://doi.org/10.1007/BFb0053428
8. Bernstein, D.J.: The Poly1305-AES message authentication code. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. Lecture Notes

in Computer Science, vol. 3557, pp. 32–49. Springer (2005). https://doi.org/10.1007/11502760_3, https://doi.org/10.1007/11502760_3
9. Bernstein, D.J.: FO derandomization sometimes damages security. Cryptology ePrint Archive, Paper 2021/912 (2021), https://eprint.iacr.org/2021/912
10. Bernstein, D.J.: On the looseness of FO derandomization. IACR Cryptol. ePrint Arch. p. 912 (2021), https://eprint.iacr.org/2021/912
11. Bernstein, D.J., Heninger, N., Lange, T., van Beirendonck, M., et al.: Classic mceliece: Specification (October 2022), https://classic.mceliece.org/mceliece-spec-20221023.pdf, accessed: 2025-01-29
12. Bernstein, D.J., Persichetti, E.: Towards KEM unification. IACR Cryptol. ePrint Arch. p. 526 (2018), https://eprint.iacr.org/2018/526
13. Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: The three-key constructions. In: Bellare, M. (ed.) Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 197–215. Springer (2000). https://doi.org/10.1007/3-540-44598-6_12, https://doi.org/10.1007/3-540-44598-6_12
14. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1462, pp. 1–12. Springer (1998). https://doi.org/10.1007/BFb0055716, https://doi.org/10.1007/BFb0055716
15. Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 1006–1018. ACM (2016). https://doi.org/10.1145/2976749.2978425, https://doi.org/10.1145/2976749.2978425
16. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018. pp. 353–367. IEEE (2018). https://doi.org/10.1109/EuroSP.2018.00032, https://doi.org/10.1109/EuroSP.2018.00032
17. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 453–474. Springer (2001). https://doi.org/10.1007/3-540-44987-6_28, https://doi.org/10.1007/3-540-44987-6_28
18. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. **18**(2), 143–154 (1979). https://doi.org/10.1016/0022-0000(79)90044-8, https://doi.org/10.1016/0022-0000(79)90044-8
19. Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2271, pp. 263–276. Springer (2002). https://doi.org/10.1007/3-540-45760-7_18, https://doi.org/10.1007/3-540-45760-7_18

20. Coron, J., Naccache, D., Stern, J.P.: On the security of RSA padding. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 1–18. Springer (1999). https://doi.org/10.1007/3-540-48405-1_1, https://doi.org/10.1007/3-540-48405-1_1
21. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4833, pp. 502–518. Springer (2007). https://doi.org/10.1007/978-3-540-76900-2_31, https://doi.org/10.1007/978-3-540-76900-2_31
22. D'Anvers, J., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 282–305. Springer (2018). https://doi.org/10.1007/978-3-319-89339-6_16, https://doi.org/10.1007/978-3-319-89339-6_16
23. Dent, A.W.: A designer's guide to KEMs. In: Paterson, K.G. (ed.) Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2898, pp. 133–151. Springer (2003). https://doi.org/10.1007/978-3-540-40974-8_12, https://doi.org/10.1007/978-3-540-40974-8_12
24. Dobraunig, C., Mennink, B., Neves, S.: EliMAC: Speeding up LightMAC by around 20%. IACR Trans. Symmetric Cryptol. **2023**(2), 69–93 (2023). https://doi.org/10.46586/TOSC.V2023.I2.69-93, https://doi.org/10.46586/tosc.v2023.i2.69-93
25. Dong, H., Guo, Q.: OT-PCA: new key-recovery plaintext-checking oracle based side-channel attacks on HQC with offline templates. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2025**(1), 251–274 (2025). https://doi.org/10.46586/TCHES.V2025.I1.251-274, https://doi.org/10.46586/tches.v2025.i1.251-274
26. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015. pp. 1197–1210. ACM (2015). https://doi.org/10.1145/2810103.2813653, https://doi.org/10.1145/2810103.2813653
27. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer (1999). https://doi.org/10.1007/3-540-48405-1_34, https://doi.org/10.1007/3-540-48405-1_34
28. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139,

pp. 260–274. Springer (2001). https://doi.org/10.1007/3-540-44647-8_16, https://doi.org/10.1007/3-540-44647-8_16
29. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074, https://doi.org/10.1109/TIT.1985.1057074
30. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA. pp. 365–377. ACM (1982). https://doi.org/10.1145/800070.802212, https://doi.org/10.1145/800070.802212
31. Goy, G., Maillard, J., Gaborit, P., Loiseau, A.: Single trace HQC shared key recovery with SASCA. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 64–87 (2024). https://doi.org/10.46586/TCHES.V2024.I2.64-87, https://doi.org/10.46586/tches.v2024.i2.64-87
32. Grieu, F.: A chosen messages attack on the ISO/IEC 9796-1 signature scheme. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 70–80. Springer (2000). https://doi.org/10.1007/3-540-45539-6_5, https://doi.org/10.1007/3-540-45539-6_5
33. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM. Cryptology ePrint Archive, Paper 2020/743 (2020), https://eprint.iacr.org/2020/743
34. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 789–815 (2016). https://doi.org/10.1007/978-3-662-53887-6_29, https://doi.org/10.1007/978-3-662-53887-6_29
35. Guo, Q., Mårtensson, E.: Do not bound to a single position: Near-optimal multi-positional mismatch attacks against Kyber and Saber. In: Johansson, T., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14154, pp. 291–320. Springer (2023). https://doi.org/10.1007/978-3-031-40003-2_11, https://doi.org/10.1007/978-3-031-40003-2_11
36. Hermelink, J., Ning, K., Petri, R., Strieder, E.: The insecurity of masked comparisons: Scas on ml-kem's fo-transform. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024. pp. 2430–2444. ACM (2024). https://doi.org/10.1145/3658644.3690339, https://doi.org/10.1145/3658644.3690339
37. Hoffstein, J., Silverman, J.H.: Reaction attacks against the NTRU public key cryptosystem. Tech. rep., Technical Report 15, NTRU Cryptosystems (1999)
38. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-

15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer (2017). https://doi.org/10.1007/978-3-319-70500-2_12, https://doi.org/10.1007/978-3-319-70500-2_12

39. Hövelmanns, K., Hülsing, A., Majenz, C.: Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13794, pp. 414–443. Springer (2022). https://doi.org/10.1007/978-3-031-22972-5_15, https://doi.org/10.1007/978-3-031-22972-5_15
40. Huguenin-Dumittan, L., Vaudenay, S.: Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12146, pp. 208–227. Springer (2020). https://doi.org/10.1007/978-3-030-57808-4_11, https://doi.org/10.1007/978-3-030-57808-4_11
41. Huguenin-Dumittan, L., Vaudenay, S.: On IND-qCCA security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13277, pp. 613–642. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_22, https://doi.org/10.1007/978-3-031-07082-2_22
42. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A fast tweakable block cipher mode for highly secure message authentication. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 34–65. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_2, https://doi.org/10.1007/978-3-319-63697-9_2
43. Jaulmes, É., Joux, A.: A chosen-ciphertext attack against NTRU. In: Bellare, M. (ed.) Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 20–35. Springer (2000). https://doi.org/10.1007/3-540-44598-6_2, https://doi.org/10.1007/3-540-44598-6_2
44. Jiang, H., Ma, Z., Zhang, Z.: Post-quantum security of key encapsulation mechanism against CCA attacks with a single decapsulation query. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14441, pp. 434–468. Springer (2023). https://doi.org/10.1007/978-981-99-8730-6_14, https://doi.org/10.1007/978-981-99-8730-6_14
45. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018,

Proceedings, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 96–125. Springer (2018). https://doi.org/10.1007/978-3-319-96878-0_4, https://doi.org/10.1007/978-3-319-96878-0_4

46. Kaliski, B.: PKCS #1: RSA encryption version 1.5. RFC **2313**, 1–19 (1998). https://doi.org/10.17487/RFC2313, https://doi.org/10.17487/RFC2313
47. Kelsey, J., jen Chang, S., Perlner, R.: SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Tech. Rep. NIST Special Publication 800-185, National Institute of Standards and Technology, U.S. Department of Commerce (2016), https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-185.pdf
48. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 310–331. Springer (2001). https://doi.org/10.1007/3-540-44647-8_19, https://doi.org/10.1007/3-540-44647-8_19
49. Kundu, S., Chowdhury, S., Saha, S., Karmakar, A., Mukhopadhyay, D., Verbauwhede, I.: Carry your fault: A fault propagation attack on side-channel protected lwe-based KEM. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(2), 844–869 (2024). https://doi.org/10.46586/TCHES.V2024.I2.844-869, https://doi.org/10.46586/tches.v2024.i2.844-869
50. Li, J., Cheng, C., Shen, M., Chen, P., Guo, Q., Liu, D., Wu, L., Weng, J.: Grafted trees bear better fruit: An improved multiple-valued plaintext-checking side-channel attack against kyber. IACR Cryptol. ePrint Arch. p. 1174 (2024), https://eprint.iacr.org/2024/1174
51. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC mode for lightweight block ciphers. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 43–59. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_3, https://doi.org/10.1007/978-3-662-52993-5_3
52. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_1, https://doi.org/10.1007/978-3-642-13190-5_1
53. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: NIST post-quantum cryptography standardization round 4 submission: Hamming quasi-cyclic (HQC). Tech. rep., National Institute of Standards and Technology (NIST) (2024), https://pqc-hqc.org/doc/hqc-specification_2024-10-30.pdf
54. Misoczki, R., Tillich, J., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In: Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013. pp. 2069–2073. IEEE (2013). https://doi.org/10.1109/ISIT.2013.6620590, https://doi.org/10.1109/ISIT.2013.6620590

55. Mondal, P., Kundu, S., Bhattacharya, S., Karmakar, A., Verbauwhede, I.: A practical key-recovery attack on lwe-based key-encapsulation mechanism schemes using rowhammer. In: Pöpper, C., Batina, L. (eds.) Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14585, pp. 271–300. Springer (2024). https://doi.org/10.1007/978-3-031-54776-8_11, https://doi.org/10.1007/978-3-031-54776-8_11
56. National Institute of Standards and Technology: Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. Tech. Rep. NIST Special Publication 800-38D, U.S. Department of Commerce (2007), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
57. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Tech. rep., Federal Information Processing Standards (FIPS) Publication 202 (August 2015). https://doi.org/10.6028/NIST.FIPS.202, http://dx.doi.org/10.6028/NIST.FIPS.202
58. National Institute of Standards and Technology: Recommendation for block cipher modes of operation: The CMAC mode for authentication. Tech. Rep. NIST Special Publication 800-38B, U.S. Department of Commerce (2016), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38B.pdf
59. National Institute of Standards and Technology: Module-lattice-based key-encapsulation mechanism standard. Federal Information Processing Standards Publication NIST FIPS 203, Department of Commerce, Washington, D.C. (2024). https://doi.org/10.6028/NIST.FIPS.203, https://doi.org/10.6028/NIST.FIPS.203
60. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings. Lecture Notes in Computer Science, vol. 1992, pp. 104–118. Springer (2001). https://doi.org/10.1007/3-540-44586-2_8, https://doi.org/10.1007/3-540-44586-2_8
61. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2020, pp. 159–175. Springer (2001). https://doi.org/10.1007/3-540-45353-9_13, https://doi.org/10.1007/3-540-45353-9_13
62. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 333–342. ACM (2009). https://doi.org/10.1145/1536414.1536461, https://doi.org/10.1145/1536414.1536461
63. Peikert, C.: Lattice cryptography for the internet. Cryptology ePrint Archive, Paper 2014/070 (2014), https://eprint.iacr.org/2014/070
64. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603, https://doi.org/10.1145/1060590.1060603

65. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009). https://doi.org/10.1145/1568318.1568324, https://doi.org/10.1145/1568318.1568324
66. Rescorla, E.: The transport layer security (tls) protocol version 1.3. RFC 8446, RFC Editor (August 2018)
67. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342, https://doi.org/10.1145/359340.359342
68. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 1461–1480. ACM (2020). https://doi.org/10.1145/3372297.3423350, https://doi.org/10.1145/3372297.3423350
69. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 275–288. Springer (2000). https://doi.org/10.1007/3-540-45539-6_19, https://doi.org/10.1007/3-540-45539-6_19
70. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 239–259. Springer (2001). https://doi.org/10.1007/3-540-44647-8_15, https://doi.org/10.1007/3-540-44647-8_15
71. Shoup, V.: A proposal for an ISO standard for public key encryption. IACR Cryptol. ePrint Arch. p. 112 (2001), http://eprint.iacr.org/2001/112
72. for Standardization, I.O.: ISO/IEC 9796: Information Technology — Security Techniques — Digital Signature Scheme Giving Message Recovery, Part 1: Mechanisms Using Redundancy (1999), part 1 of the ISO/IEC 9796 standard
73. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(3), 473–503 (2023). https://doi.org/10.46586/TCHES.V2023.I3.473-503, https://doi.org/10.46586/tches.v2023.i3.473-503
74. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum KEMs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 296–322 (2022). https://doi.org/10.46586/TCHES.V2022.I1.296-322, https://doi.org/10.46586/tches.v2022.i1.296-322
75. Wang, R., Brisfors, M., Dubrova, E.: A side-channel attack on a higher-order masked crystals-kyber implementation. In: Pöpper, C., Batina, L. (eds.) Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14585, pp. 301–324. Springer (2024). https://doi.org/10.1007/978-3-031-54776-8_12, https://doi.org/10.1007/978-3-031-54776-8_12
76. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. **22**(3), 265–279 (1981). https://doi.org/10.1016/0022-0000(81)90033-7, https://doi.org/10.1016/0022-0000(81)90033-7

77. Xagawa, K., Ito, A., Ueno, R., Takahashi, J., Homma, N.: Fault-injection attacks against NIST's post-quantum cryptography round 3 KEM candidates. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 33–61. Springer (2021). https://doi.org/10.1007/978-3-030-92075-3_2, https://doi.org/10.1007/978-3-030-92075-3_2
78. Xagawa, K., Yamakawa, T.: (tightly) QCCA-secure key-encapsulation mechanism in the quantum random oracle model. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers. Lecture Notes in Computer Science, vol. 11505, pp. 249–268. Springer (2019). https://doi.org/10.1007/978-3-030-25510-7_14, https://doi.org/10.1007/978-3-030-25510-7_14
79. Zhang, X., Cheng, C., Ding, R.: Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. Cryptology ePrint Archive, Paper 2021/168 (2021), https://eprint.iacr.org/2021/168
80. Zhou, B., Jiang, H., Zhao, Y.: CPA-secure KEMs are also sufficient for post-quantum TLS 1.3. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 15486, pp. 433–464. Springer (2024). https://doi.org/10.1007/978-981-96-0891-1_14, https://doi.org/10.1007/978-981-96-0891-1_14

A ML-KEM subroutines

Below are high-level summaries of ML-KEM's subroutines.

$\texttt{KeyGen}_{\texttt{K-PKE}}$

1: $A \xleftarrow{\$} R_q^{k \times k}$
2: $\mathbf{s} \xleftarrow{\$} \mathcal{X}_{\eta_1}^k$
3: $\mathbf{e} \xleftarrow{\$} \mathcal{X}_{\eta_1}^k$
4: $\mathbf{t} \leftarrow A\mathbf{s} + \mathbf{e}$
5: $\texttt{pk} \leftarrow (A, \mathbf{t})$
6: $\texttt{sk} \leftarrow \mathbf{s}$
7: **return** $(\texttt{pk}, \texttt{sk})$

$\texttt{Enc}_{\texttt{K-PKE}}(\texttt{pk}, m)$

Ensure: $\texttt{pk} = (A, \mathbf{t})$
Ensure: $m \in R_2$
1: $\mathbf{r} \xleftarrow{\$} \mathcal{X}_{\eta_1}^k$
2: $\mathbf{e}_1 \xleftarrow{\$} \mathcal{X}_{\eta_2}^k$
3: $e_2 \xleftarrow{\$} \mathcal{X}_{\eta_2}$
4: $\mathbf{c}_1 \leftarrow A\mathbf{r} + \mathbf{e}_1$
5: $c_2 \leftarrow \mathbf{t}^\intercal \mathbf{r} + e_2 + m \cdot \lfloor \frac{q}{2} \rceil$
6: **return** $(\mathbf{c}_1, c_2)$

$\texttt{Dec}_{\texttt{K-PKE}}(\texttt{sk}, c)$

Ensure: $c = (\mathbf{c}_1, c_2)$
Ensure: $\texttt{sk} = \mathbf{s}$
1: $\hat{m} \leftarrow c_2 - \mathbf{c}_1^\intercal \cdot \mathbf{s}$
2: $\hat{m} \leftarrow \texttt{Round}(\hat{m})$
3: **return** $\hat{m}$

Fig. 12: K-PKE is an IND-CPA secure PKE based on the conjectured intractability of the Module Learning with Error (MWLE) problem

$\texttt{KeyGen}_{\texttt{ML-KEM}}()$

1: $(\texttt{pk}, \texttt{sk}') \xleftarrow{\$} \texttt{KeyGen}_{\texttt{K-PKE}}()$
2: $h \leftarrow G(\texttt{pk})$
3: $z \xleftarrow{\$} \mathcal{M}$
4: $\texttt{sk} \leftarrow (\texttt{sk}', \texttt{pk}, h, z)$
5: **return** $(\texttt{pk}, \texttt{sk})$

$\texttt{Encap}_{\texttt{ML-KEM}}(\texttt{pk})$

1: $m \xleftarrow{\$} \mathcal{M}$
2: $h \leftarrow G(\texttt{pk})$
3: $(K, r) \leftarrow H(m, h)$
4: $c \leftarrow \texttt{Enc}_{\texttt{K-PKE}}(\texttt{pk}, m, r)$
5: **return** (c, K)

$\texttt{Decap}_{\texttt{ML-KEM}}(\texttt{sk}, c)$

1: $\hat{m} \leftarrow \texttt{Dec}_{\texttt{K-PKE}}(\texttt{sk}', c)$
2: $(\overline{K}, \hat{r}) \leftarrow H(\hat{m}, h)$
3: $\hat{c} \leftarrow \texttt{Enc}_{\texttt{K-PKE}}(\texttt{pk}, m, \hat{r})$
4: **if** $\hat{c} = c$ **then**
5: $\quad K \leftarrow \overline{K}$
6: **else**
7: $\quad K \leftarrow G(c, z)$
8: **end if**
9: **return** K

Fig. 13: `ML-KEM` uses the Fujisaki-Okamoto transformation to convert CPA-secure PKE into CCA-secure KEM. Hash function G is `SHA3-256`, H is `SHA3-512`

B Key exchange protocols

The CRYSTALS-Kyber team [16] proposed three key exchange protocols:

- `Kyber.KE` for a passively secure ephemeral key exchange.
- `Kyber.UAKE` for unilaterally authenticated key exchange.
- `Kyber.AKE` for mutually authenticated key exchange

First formally proposed in [5] and later analyzed in [17], these key exchange protocols all achieve weak forward secrecy, and the authenticated key exchange protocols are resistant to certain categories of active adversaries (e.g. Man-in-the-Middle attacks).

We implemented each of the three key exchange protocols using `ML-KEM-EtM` and `ML-KEM`, then measured the round trip time of each handshake. We denote the party who sends the first message to be the client and the other party to be the server. Round trip time (RTT) is defined to be the time interval between the moment before the client starts generating ephemeral keypairs and the moment after the client derives the final session key. All experiments are run on a pair of AWS c7a.medium instances both located in the `us-west-2` region. For each experiment, a total of 10,000 rounds of key exchange are performed, with the median and average round trip time (measured in microsecond μs) recorded.

Unauthenticated key exchange In unauthenticated key exchange (KE), a single pair of ephemeral keypair is generated by the client. The client transmits the ephemeral public key $\texttt{pk}_e$ to the server, who runs the encapsulation routine and transmits the ciphertext c_e back to the client. The client finally decapsulates the ciphertext to recover the shared secret. The key derivation function is instantiated using `Shake256`, and the final session key is 256 bits in length. `ML-KEM-EtM-512` with `Poly1305` reduces RTT by 23.9% compared to `ML-KEM-512`; `ML-KEM-EtM-768` reduces RTT by up to 26.7%; `ML-KEM-EtM-1024` with reduces RTT by up to 28.5% (See Table 4).

Table 4: KE RTT comparison

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	800	768	92	97
ML-KEM-EtM-512 + Poly1305	800	784	70	72
ML-KEM-EtM-512 + GMAC	800	784	73	76
ML-KEM-EtM-512 + CMAC	800	784	75	79
ML-KEM-EtM-512 + KMAC256	800	784	76	78

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	1184	1088	135	140
ML-KEM-EtM-768 + Poly1305	1184	1120	99	104
ML-KEM-EtM-768 + GMAC	1184	1120	101	105
ML-KEM-EtM-768 + CMAC	1184	1120	103	109
ML-KEM-EtM-768 + KMAC256	1184	1120	103	107

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	1568	1568	193	199
ML-KEM-EtM-1024 + Poly1305	1568	1600	138	141
ML-KEM-EtM-1024 + GMAC	1568	1600	140	145
ML-KEM-EtM-1024 + CMAC	1568	1600	143	148
ML-KEM-EtM-1024 + KMAC256	1568	1600	144	149

Unilaterally authenticated key exchange In unilaterally authenticated key exchange (UAKE), the authenticating party proves its identity to the other party by demonstrating possession of a secret key that corresponds to a published long-term public key. In this implementation, the client possesses the long-term public key $\mathtt{pk}_S$ of the server, and the server authenticates itself by correctly decrypting the challenge encryption sent by the client using its long-term secret key $\mathtt{sk}_S$. ML-KEM-EtM-512 with Poly1305 reduces RTT by 29.0% compared to ML-KEM-512; ML-KEM-EtM-768 reduces RTT by up to 33.0%; ML-KEM-EtM-1024 reduces RTT by up to 34.8% (See Table 5).

Mutually authenticated key exchange (AKE) Mutually authenticated key exchange is largely identical to unilaterally authenticated key exchange, except for that client authentication is required. This means that client possesses server's long-term public key $\mathtt{pk}_S$ and its own long-term secret key $\mathtt{sk}_C$, while the server possesses client's long-term public key $\mathtt{pk}_C$ and its own long-term secret key $\mathtt{sk}_S$. The session key is derived by applying KDF onto the concatenation of shared secrets produced under the ephemeral keypair, server's long-term keypair, and client's long-term keypair, in this order. ML-KEM-EtM-512 with Poly1305 reduces RTT by 39.5% compared to ML-KEM-512; ML-KEM-EtM-768 reduces RTT by up to 35.3%; ML-KEM-EtM-1024 reduces RTT by up to 48.0% (See Table 6).

Table 5: UAKE RTT comparison

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	1568	768	145	151
ML-KEM-EtM-512 + Poly1305	1584	784	103	106
ML-KEM-EtM-512 + GMAC	1584	784	106	110
ML-KEM-EtM-512 + CMAC	1584	784	108	112
ML-KEM-EtM-512 + KMAC256	1584	784	109	113

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	2272	1088	215	222
ML-KEM-EtM-768 + Poly1305	2288	1104	144	150
ML-KEM-EtM-768 + GMAC	2288	1104	149	156
ML-KEM-EtM-768 + CMAC	2288	1104	153	160
ML-KEM-EtM-768 + KMAC256	2288	1104	154	159

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	3136	1568	310	318
ML-KEM-EtM-1024 + Poly1305	3152	1584	202	209
ML-KEM-EtM-1024 + GMAC	3152	1584	212	228
ML-KEM-EtM-1024 + CMAC	3152	1584	212	218
ML-KEM-EtM-1024 + KMAC256	3152	1584	213	220

Table 6: AKE RTT comparison

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	1568	1536	220	213
ML-KEM-EtM-512 + Poly1305	1584	1568	133	138
ML-KEM-EtM-512 + GMAC	1584	1568	139	143
ML-KEM-EtM-512 + CMAC	1584	1568	143	148
ML-KEM-EtM-512 + KMAC256	1584	1568	145	151

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	2272	2176	294	301
ML-KEM-EtM-768 + Poly1305	2288	2208	190	196
ML-KEM-EtM-768 + GMAC	2288	2208	197	210
ML-KEM-EtM-768 + CMAC	2288	2208	202	208
ML-KEM-EtM-768 + KMAC256	2288	2208	204	210

Parameter set	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	3136	3136	512	511
ML-KEM-EtM-1024 + Poly1305	3152	3168	266	273
ML-KEM-EtM-1024 + GMAC	3152	3168	273	282
ML-KEM-EtM-1024 + CMAC	3152	3168	280	287
ML-KEM-EtM-1024 + KMAC256	3152	3168	282	288