Faster generic IND-CCA secure KEM 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 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 is significantly slower than the underlying PKE decryption, ML-KEM's decapsulation performance is dominated by re-encryption. In this paper, we propose an alternative generic IND-CCA secure KEM transformation, called $\mathsf{KEM}_{\mathtt{EtM}}$ that applies the encrypt-then-MAC mechanism to an OW-PCA secure PKE and an existentially unforgeable MAC. Compared to the Fujisaki-Okamoto transformation, our encrypt-then-MAC transformation replaces de-randomization and re-encryption with computing a MAC tag. We instantiate our proposed KEM with the PKE sub-routines of ML-KEM and call it ML-KEM$^+$. We then implement ML-KEM$^+$ with a wide selection of MACs including Poly1305, GMAC, CMAC, and KMAC. At the cost of minimal increase in encapsulation CPU cycles (+1.8%) and ciphertext size (+2.1%), ML-KEM$^+$ achieves a massive reduction of decapsulation CPU cycles (-72.2%) compared to ML-KEM. Furthermore, we implement key exchange protocols and measure realistic network round trip times (RTTs), where ML-KEM$^+$ reduces RTTs by 23.9%-39.5% compared to ML-KEM.

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

1 Introduction

A key encapsulation mechanism (KEM) [Sho01] is a cryptographic primitive that allows two parties to establish a shared secret over an insecure channel. The desired security standard for a KEM is called indistinguishability under chosen-ciphertext attack (IND-CCA). Intuitively speaking, IND-CCA security requires that no efficient adversary can distinguish a pseudorandom shared secret from a uniformly random bit string of equal length, even with access to a decapsulation oracle throughout the attack. However, building a provably IND-CCA secure KEM is tremendously difficult. Early attempts without formal proofs, such as RSA encryption defined in PKCS#1 v1.5 [Kal98], were later shown to be vulnerable to practical chosen-ciphertext attacks [Ble98]. In recent decades, the most viable approach has been to start with cryptographic primitives possessing weaker security properties, such as a public-key encryption (PKE) scheme with one-way security under chosen-plaintext attack (OW-CPA), then add steps to achieve *ciphertext non-malleability* [BN00]. Some of the earliest proposals for generic IND-CCA secure constructions include OAEP [BR94], Fujisaki-Okamoto transformation [FO99][FO13], REACT [OP01b], and GEM [CHJ+02]. Some other notable examples for constructing IND-CCA secure public-key cryptosystems in the standard model include the constructions of Naor and Yung [NY90], Cramer and Shoup [CS98] and Canetti, Halevi and Katz [CHK04]. The construction of Naor and Yung uses an IND-CPA secure PKE and a non-interactive zero-knowledge (NIZK) proof system. Canetti, Halevi and Katz [CHK04] proposed a construction to build

an IND-CCA secure PKE scheme from an IND-CPA secure IBE scheme and a one-time signature scheme.

On the other hand, chosen-ciphertext security is a solved problem in symmetric cryptography. It is well understood that, by combining an IND-CPA secure symmetric encryption scheme with an existentially unforgeable message authentication code (MAC) in a pattern called encrypt-then-MAC [Kra01], one can build an authenticated encryption scheme [BN00] that achieves IND-CCA security. The encrypt-then-MAC (EtM) mechanism was standard in ISO 19772 [fS09]. While this technique cannot be directly applied in the context of public-key cryptography due to the lack of a shared symmetric key between the two communicating parties, the concept of authenticating ciphertext using a MAC still has strong merits. Abdalla, Rogaway, and Bellare proposed DHIES (also known as "Hashed ElGamal")[ABR99][ABR01], a hybrid public-key encryption (HPKE) scheme whose IND-CCA security reduces to the Gap Diffie-Hellman assumption [OP01a] under the random oracle model. The technique behind DHIES is to derive both the shared secret and a symmetric MAC key by hashing a random PKE plaintext, encrypting the PKE plaintext, then authenticating the PKE ciphertext using the previously derived MAC key. When the Gap Diffie-Hellman assumption holds and the MAC is existentially unforgeable, no efficient adversary can recover the decryption of an unknown ciphertext even with access to a decryption oracle because it cannot produce a valid tag for such unknown ciphertext.

1.1 Contributions

Our main contributions are threefold:

- **New IND-CCA secure KEM construction using encrypt-then-MAC.** We propose a generic IND-CCA secure KEM transformation called the encrypt-then-MAC transformation and prove that the IND-CCA security of the encrypt-then-MAC transformation reduces tightly to the OW-PCA security of the underlying PKE and the existential unforgeability of the underlying MAC. In addition, we argue that the encrypt-then-MAC transformation can be instantiated with one-time MAC such as Poly1305 for further performance improvements.

- **ML-KEM$^+$: Efficient CCA-secure KEM based on ML-KEM.** We present ML-KEM$^+$, an IND-CCA secure KEM constructed by applying the encrypt-then-MAC transformation to ML-KEM's underlying PKE sub-routines. Compared to ML-KEM, ML-KEM$^+$ adds a small amount of performance penalty to the encapsulation routine and a small increase in ciphertext size, but replaces the expensive re-encryption step in decapsulation with computing a MAC tag, which yields substantial performance improvements.

- **Performance evaluation and comparisons.** We implemented ML-KEM$^+$ in C and compared its performance with ML-KEM in a variety of scenarios. Compared to ML-KEM, ML-KEM$^+$ achieves 72.2%-79.1% reduction of CPU cycle count for decapsulation while only increasing encapsulation's CPU cycle count by 1.8%-7.3% and ciphertext size by 1.0%-2.0% (see Table 1). We also implemented the Kyber key exchange protocols [BDK$^+$18] with ML-KEM$^+$ and measured the round trip time (RTT) in realistic network settings. Compared to ML-KEM, ML-KEM$^+$ achieves 24%-28% reduction of round trip time in unauthenticated key exchange (KE), 29%-35% in unilaterally authenticated key exchange (UAKE), and 35%-48% reduction in mutually authenticated key exchange (AKE). See Table 2 for a summary of round trip times.

Table 1: ML-KEM$^+$ achieves large performance improvements in decapsulation at the cost of minimal penalty in encapsulation performance and ciphertext size

	ML-KEM$^+$-512 Poly1305	ML-KEM$^+$-768 KMAC 192-bit tag	ML-KEM$^+$-1024 KMAC 256-bit tag
Encap (ccl/tick)	93157 (+1.8%)	155219 (+13.8%)	228357 (+8.8%)
Decap (ccl/tick)	33733 (-72.2%)	52415 (-71.9%)	62269 (-74.7%)
CT size (bytes)	784 (+2.1%)	1112 (+2.2%)	1600 (+2.0%)

Table 2: ML-KEM$^+$ achieves substantial RTT savings despite increased encapsulation cost and ciphertext size

	ML-KEM$^+$-512 Poly1305	ML-KEM$^+$-768 KMAC 192-bit tag	ML-KEM$^+$-1024 KMAC 256-bit tag
KE RTT (μs)	70 (-23.9%)	103 (-23.7%)	144 (-25.4%)
UAKE RTT (μs)	103 (-29.0%)	154 (-28.4%)	213 (-32.3%)
AKE RTT (μs)	133 (-39.5%)	204 (-30.6%)	282 (-44.9%)

1.2 Related works

Optimal Asymmetric Encryption Padding (OAEP) [BR94][BDPR98] is a generic chosen-ciphertext secure PKE. Under the random oracle model, the chosen-ciphertext security of the OAEP encryption scheme reduces to the one-wayness of the input trapdoor permutation. Although it was discovered that there exist trapdoor permutations with which the OAEP encryption scheme does not achieve full IND-CCA security [Sho02], Fujisaki et al. later proved that the OAEP is IND-CCA secure when combined with the RSA trapdoor permutation [FOPS01][RSA78]. RSA-OAEP was standardized in PKCS#1 v2 [MKJR16] and is currently the most recommended of all RSA-based encryption schemes. Unfortunately, OAEP's requirement for a trapdoor permutation is immensely difficult to satisfy, and no other practical instantiation saw widespread adoption to this day.

The **Fujisaki-Okamoto transformation** [FO99][FO13] is another generic IND-CCA secure transformation. While Fujisaki and Okamoto originally proposed a hybrid public-key encryption scheme whose IND-CCA security reduces non-tightly to the OW-CPA security of the underlying PKE and the IND-CPA security of the symmetric encryption scheme. Later works [Den03][HHK17][DNR04][HHM22][BP18] tightened the security reduction, accounted for imperfect correctness, adapted the original proposal to build a KEM, and proved its security in the quantum random oracle model (QROM).

The modular Fujisaki-Okamoto KEM transformation is remarkably successful because of the simplicity of its construction, the tightness of the security bound, and the proven (though non-tight) security against quantum adversaries. It was adopted by many submissions to NIST's post-quantum cryptography competition, including Kyber [BDK$^+$18], Saber [DKRV18], FrodoKEM [BCD$^+$16], and classic McEliece [ABC$^+$20] among others. The $U_m^{\not\perp}$ variant of the Fujisaki-Okamoto transformation (see Figure 1) is adopted in ML-KEM.

However, the Fujisaki-Okamoto transformation is not perfect. Because it uses re-encryption for achieving ciphertext non-malleability, the Fujisaki-Okamoto transformation suffers from the following two problems:

- **Computational inefficiency**. The decapsulation routine needs to re-encrypt the decryption to ensure ciphertext has not been tempered with. For input PKE whose encryption routine carries substantial computational cost, such as most lattice-based cryptosystems, re-encryption slows down decapsulation significantly.

$\texttt{KEM}^{\not\perp}_m.\texttt{KeyGen}()$

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

$\texttt{KEM}^{\not\perp}_m.\texttt{Encap}(\texttt{pk})$

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

$\texttt{KEM}^{\not\perp}_m.\texttt{Decap}(\texttt{sk}, c)$

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

Figure 1: The $\texttt{KEM}^{\not\perp}_m$ variant of Fujisaki-Okamoto transformation is used in ML-KEM

- **Side-channel vulnerability**. Re-encryption also introduces side-channels that can leak information about the decrypted PKE plaintext. As demonstrated in [UXT+22][RRCB19][TUX+23], these side-channels can be converted into efficient plaintext-checking attacks that can fully recover the secret key.

The rest of the paper is organized as follows. In Section 2, we define and discuss the preliminary concepts and theorems. In Section 3, we present our encrypt-then-MAC transformation in details, prove its security results, and discusses its relationship to the ElGamal cryptosystem. In Section 4, we present ML-KEM$^+$, an instantiation of the encrypt-then-MAC transformation using ML-KEM's PKE sub-routines, then discuss some implementation rationales. In Section 5, we benchmark and compare the performance of individual KEM routines and key exchange round trip times between ML-KEM$^+$ and ML-KEM.

2 Preliminaries

2.1 Public-key encryption scheme

Syntax. A public-key encryption scheme $\texttt{PKE}(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ is a collection of three routines defined over some plaintext space $\mathcal{M}$ and some ciphertext space $\mathcal{C}$. Key generation $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}()$ is a randomized routine that returns a keypair. The encryption routine $\texttt{Enc} : (\texttt{pk}, m) \mapsto c$ encrypts the input plaintext under the input public key. The decryption routine $\texttt{Dec} : (\texttt{sk}, c) \mapsto m$ decrypts the input ciphertext under the input secret key. 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. The decryption routine is assumed to always be deterministic. Some decryption routines can detect malformed ciphertext and output the rejection symbol $\perp$ accordingly.

Correctness. Following the definition in [DNR04], a $\texttt{PKE}$ 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 $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{PKE.KeyGen()}$. For many lattice-based cryptosystems, including ML-KEM, 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 problem.

Security. The security of public-key encryption is conventionally discussed within the context of adversarial games played between a challenger and an adversary [GM82]. There are two main types of games: in the one-wayness game, the adversary is given a random encryption, then asked to produce the correct decryption; in the indistinguishability game, the adversary is given the encryption of one of two adversary-chosen plaintexts, then asked to decide which of the plaintexts corresopnds with the given encryption. Depending on the attack model, the adversary may have access to various oracles. Within the context of public-key cryptography, adversaries are always assumed to have the public key with which they can mount chosen-plaintext attack (CPA). If the adversary has access to a plaintext-checking oracle (PCO) [OP01b] then it can mount plaintext-checking attack (PCA). Where the adversary has access to a decryption oracle, it can mount chosen-ciphertext attacks (CCA).

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: $\hat{m} \xleftarrow{\$} A^{\mathcal{O}_{\mathtt{ATK}}}(1^\lambda, \mathtt{pk}, c^*)$
5: **return** $[\![\hat{m} = m^*]\!]$

IND-ATK Game

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

$\mathcal{O}_{\mathtt{PCO}}(m, c)$

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

$\mathcal{O}_{\mathtt{Dec}}(c)$

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

Figure 2: The one-way game, indistinguishability game, plaintext-checking oracle (PCO), and decryption oracle. $\mathtt{ATK} \in \{\mathtt{CPA}, \mathtt{PCA}, \mathtt{CCA}\}$

The advantage of an adversary in the `OW-ATK` game is the probability that it ouputs the correct decryption. The advantage of an adversary in the `IND-ATK` game is defined below. A PKE is `OW-ATK`/`IND-ATK` secure if no efficient adversary has non-negligible advantage in the corresponding security game.

$$\mathtt{Adv}_{\mathtt{IND\text{-}ATK}}(A) = \left| P\left[\hat{b} = b\right] - \frac{1}{2} \right|$$

2.2 Key encapsulation mechanism (KEM)

Syntax A key encapsulation mechanism $\mathtt{KEM}(\mathtt{KeyGen}, \mathtt{Encap}, \mathtt{Decap})$ is a collection of three routines defined over some ciphertext space $\mathcal{C}$ and some key space $\mathcal{K}$. The key generation routine takes the security parameter 1^λ and outputs a keypair $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda)$. $\mathtt{Encap}(\mathtt{pk})$ is a probabilistic routine that takes a public key `pk` and outputs a pair of values (c, K) where $c \in \mathcal{C}$ is the ciphertext (also called encapsulation) and $K \in \mathcal{K}$ is the shared secret (also called session key). $\mathtt{Decap}(\mathtt{sk}, c)$ is a deterministic routine that takes the secret key `sk` and the encapsulation c and returns the shared secret K if the ciphertext is valid. Some KEM constructions use explicit rejection, where if c is invalid then `Decap` will return a rejection symbol $\perp$; other KEM constructions use implicit rejection, where if c is invalid then `Decap` will return a fake session key that depends on the ciphertext and some other secret values.

Security The security of KEMs is similarly discussed in adversarial games (Figure 3), although the win conditions differ slightly from the win conditions of a PKE's indistinguishability game. In a KEM's indistinguishability game, an adversary is given the public key and a challenge ciphertext, then asked to distinguish a pseudorandom shared secret K_0 associated with the challenge ciphertext from a truly random bit string of equal length.

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: $\hat{b} \xleftarrow{\$} A^{\mathcal{O}_{\mathtt{ATK}}}(1^\lambda, \mathtt{pk}, c^*, K_b)$
6: **return** $[\![\hat{b} = b]\!]$

$\mathcal{O}_{\mathtt{Decap}}(c)$

1: **return** $\mathtt{Decap}(\mathtt{sk}, c)$

Figure 3: IND-ATK game for KEM and decapsulation oracle $\mathcal{O}_{\mathtt{Decap}}$

The decapsulation oracle $\mathcal{O}^{\mathtt{Decap}}$ takes a ciphertext c and returns the output of the Decap routine using the secret key. The advantage $\epsilon_{\mathtt{IND\text{-}CCA}}$ of an IND-CCA adversary $\mathcal{A}_{\mathtt{IND\text{-}CCA}}$ is defined by the adversary's ability to correctly distinguish the two cases beyond blind guess:

$$\mathtt{Adv}_{\mathtt{IND\text{-}CCA}}(A) = \left| P[A^{\mathcal{O}_{\mathtt{Decap}}}(a^\lambda, \mathtt{pk}, c^*, K_b) = b] - \frac{1}{2} \right|$$

A KEM is IND-ATK secure if no efficient adversary has non-negligible advantage in the corresponding security game.

2.3 Message authentication code (MAC)

A message authentication code $\mathtt{MAC}(\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$ is a collection of routines defined over some key space $\mathcal{K}$, some message space $\mathcal{M}$, and some tag space $\mathcal{T}$. The signing routine $\mathtt{Sign}(k, m)$ authenticates the message m under the secret key k by producing a tag t (also called digest). The verification routine $\mathtt{Verify}(k, m, t)$ takes the triplet of secret key, message, and tag, and outputs 1 if the message-tag pair is valid under the secret key, or 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 perform verification using a simple comparison.

The security of a MAC is defined in an adversarial game in which an adversary, with access to some signing oracle $\mathcal{O}_{\mathtt{Sign}}(m)$, tries to forge a new valid message-tag pair that has never been queried before. The existential unforgeability under chosen message attack (EUF-CMA) game is shown below:

EUF-CMA game

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

Figure 4: The existential forgery game

The advantage $\mathtt{Adv}_{\mathtt{EUF\text{-}CMA}}$ of the existential forgery adversary is the probability that it wins the EUF-CMA game. Some MACs are one-time existentially unforgeable, meaning that each secret key can be used to authenticate only a single message. The corresponding security game is modified such that the signing oracle will only answer a single signing query.

3 The encrypt-then-MAC transformation

In this section we introduce the encrypt-then-MAC transformation that transforms an OW-PCA secure PKE and an one-time existentially unforgeable MAC into an IND-CCA secure KEM. Our scheme mainly differs from DHIES in its versatility and input requirement. Whereas the IND-CCA security of DHIES reduces specifically to the Gap Diffie-Hellman assumption, the chosen-ciphertext security of the encrypt-then-MAC KEM reduces more generally to the OW-PCA security [OP01b] of the input scheme (we will show in Section 3.2 that the Gap Diffie-Hellman assumption implies OW-PCA security). In addition, we propose that because each call to encapsulation samples a fresh PKE plaintext, the encrypt-then-MAC KEM can be instantiated with one-time secure MAC such as Poly1305 for further performance improvements (M. Abdalla et al. originally proposed to use HMAC and CBC-MAC, which are many-time secure MAC but less efficient than one-time MAC, see Section 4.2). The encapsulation data flow is illustrated in Figure 5.

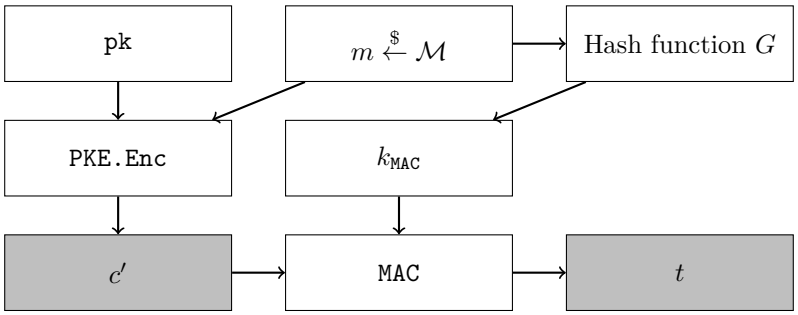

Figure 5: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

In Section 3.1 we reduce 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 3.2, we show that DHIES is a special case of the encrypt-then-MAC transformation by reducing the OW-PCA security of the ElGamal cryptosystem to the Gap Diffie-Hellman assumption.

Let $\mathcal{B}^*$ denote the set of finite bit strings. Let $\mathtt{PKE}(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be a public-key encryption scheme defined over message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathtt{MAC} : \mathcal{K}_{\mathtt{MAC}} \times \mathcal{B}^* \to \mathcal{T}$ be a deterministic message authentication code that takes a key $k \in \mathcal{K}_{\mathtt{MAC}}$, some message $m \in \mathcal{B}^*$, and outputs a digest $t \in \mathcal{T}$. Let $G : \mathcal{M} \to \mathcal{K}_{\mathtt{MAC}}$ be a hash function that maps from $\mathtt{PKE}$'s plaintext space to $\mathtt{MAC}$'s key space. Let $H : \mathcal{B}^* \to \mathcal{K}_{\mathtt{KEM}}$ be a hash function that maps bit strings into the set of possible shared secrets. The encrypt-then-MAC transformation $\mathtt{EtM}[\mathtt{PKE}, \mathtt{MAC}, G, H]$ constructs a key encapsulation mechanism $\mathtt{KEM}_{\mathtt{EtM}}(\mathtt{KeyGen}, \mathtt{Encap}, \mathtt{Decap})$, whose routines are described in Figure 6.

Since the encrypt-then-MAC transformation removes re-encryption in decapsulation, there is no longer the need for fixing the pseudorandom coin r in the PKE's encryption routine. If the input PKE is already rigid, then the shared secret may be derived from hashing the PKE plaintext alone. However, if the input PKE is not rigid, then the shared secret must be derived from hashing both the PKE plaintext and the PKE ciphertext.

The chosen-ciphertext security of the encrypt-then-MAC scheme 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 t for some unauthenticated ciphertext c' under the symmetric key $k \leftarrow G(\mathtt{Dec}(\mathtt{sk}', c'))$ implies that A must either know the symmetric key k or produce a forgery. Under the random oracle model, A also cannot know k without knowing its preimage $\mathtt{Dec}(\mathtt{sk}', c')$, so A must either have produced

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

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

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

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

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

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

Figure 6: The encrypt-then-MAC transformation builds a KEM $\texttt{KEM}_{\texttt{EtM}}$ using a $\texttt{PKE}(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$, a $\texttt{MAC}$, and two hash functions G, H

c' honestly, or have broken the one-way security of PKE. This means that the decapsulation oracle will not give out information on decryptions that the adversary does not already know.

However, a decapsulation oracle can still give out some information: for a known plaintext m, all possible encryptions $c' \xleftarrow{\$} \texttt{Enc}(\texttt{pk}, m)$ can be correctly signed, while ciphertexts that don't decrypt back to m cannot be correctly signed. This means that a decapsulation oracle can be converted into a plaintext-checking oracle, so every chosen-ciphertext attack against the KEM can be converted into a plaintext-checking attack against the underlying PKE.

On the other hand, if the underlying PKE is OW-PCA secure and the underlying MAC is one-time existentially unforgeable, then the encrypt-then-MAC KEM is IND-CCA secure:

Theorem 1. *For every IND-CCA adversary A against $\texttt{KEM}_{\texttt{EtM}}$ that makes q decapsulation queries, there exists an OW-PCA adversary B who makes at least q plaintext-checking queries against the underlying PKE, and an one-time existential forgery adversary C against the underlying MAC such that*

$$\textit{Adv}_{\textit{IND-CCA}}(A) \leq q \cdot \textit{Adv}_{\textit{OT-MAC}}(C) + 2 \cdot \textit{Adv}_{\textit{OW-PCA}}(B)$$

Theorem 1 naturally flows into an equivalence relationship between the security of the KEM and the security of the PKE:

Lemma 1. *$\texttt{KEM}_{\texttt{EtM}}$ is IND-CCA secure if and only if the input PKE is OW-PCA secure*

3.1 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 7 and 8. From a high level we made three incremental modifications to the IND-CCA game for $\texttt{KEM}_{\texttt{EtM}}$: replace true decapsulation with simulated

decapsulation, replace the pseudorandom MAC key $k^* \leftarrow G(\mathtt{pk}, m^*)$ with a truly random MAC key $k^* \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$, and finally replace pseudorandom shared secret $K_0 \leftarrow H(m^*, t)$ with a truly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$. 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.

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

1: $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KEM}_{\mathtt{EtM}}.\mathtt{KeyGen}()$
2: $m^* \xleftarrow{\$} \mathcal{M}$
3: $c' \xleftarrow{\$} \mathtt{PKE.Enc}(\mathtt{pk}, m^*)$
4: $k^* \leftarrow G(m^*)$ ▷ Game 0-1
5: $k^* \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$ ▷ Game 2-3
6: $t \leftarrow \mathtt{MAC}(k^*, c')$
7: $c^* \leftarrow (c', t)$
8: $K_0 \leftarrow H(m^*, c^*)$ ▷ Game 0-2
9: $K_0 \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$ ▷ Game 3
10: $K_1 \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$
11: $b \xleftarrow{\$} \{0, 1\}$
12: $\hat{b} \leftarrow A^{\mathcal{O}^{\mathtt{Decap}}}(\mathtt{pk}, c^*, K_b)$ ▷ Game 0
13: $\hat{b} \leftarrow A^{\mathcal{O}_1^{\mathtt{Decap}}}(\mathtt{pk}, c^*, K_b)$ ▷ Game 1-3
14: **return** $[\![\hat{b} = b]\!]$

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

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

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

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

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

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

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

1: **if** $\exists(\tilde{m}, \tilde{c}, \tilde{K}) \in \mathcal{L}^H : \tilde{m} = m \wedge \tilde{c} = c$ **then**
2: **return** $\tilde{K}$
3: **end if**
4: $K \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$
5: $\mathcal{L}^H \leftarrow \mathcal{L}^H \cup \{(m, c, K)\}$
6: **return** K

Figure 7: Sequence of games

Proof. Game 0 is the standard IND-CCA game for KEMs. 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 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 MAC under an unknown and uniformly random key. Furthermore, the security game does not include a signing oracle, so this is a zero-time forgery. While zero-time forgery is not a standard security definition for a MAC, we can bound it by the advantage of a one-time forgery adversary C:

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

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

$$\texttt{Adv}_{G_0}(A) - \texttt{Adv}_{G_1}(A) \leq q \cdot \texttt{Adv}_{\texttt{OT-MAC}}(C)$$

Game 2 is identical to game 1, except that the challenger samples a uniformly random MAC key $k^* \xleftarrow{\$} \mathcal{K}_{\texttt{MAC}}$ 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 $P[\texttt{QUERY G}]$, then:

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

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 8.

B simulates game 3 for A: receiving the public key 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:

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

Combining all equations above produce the desired security bound. □

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

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**

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

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

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

if $\mathtt{PCO}(m, c'^*) = 1$ **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

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

1: **if** $\mathtt{PCO}(m, c'^*) = 1$ **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

Figure 8: OW-PCA adversary B simulates game 3 for IND-CCA adversary A

3.2 ElGamal is OW-PCA secure

We show that the DHAES/DHIES hybrid encryption scheme is a special case of the encrypt-then-MAC transformation. Specifically, we will sketch a proof of the following lemma:

Lemma 2. *For every OW-PCA adversary A against the ElGamal cryptosystem, there exists a Gap Diffie-Hellman problem solver B such that:*

$$Adv_{GapDH}(B) = Adv_{OW\text{-}PCA}(A)$$

In other words, ElGamal is OW-PCA secure under the Gap Diffie-Hellman assumption.

Each ElGamal cryptosystem [Gam85] is parameterized by a cyclic group $G = \langle g \rangle$ of prime order $q > 2$. A summary of the routine is shown in Figure 9:

`KeyGen()`

1: $x \xleftarrow{\$} \mathbb{Z}_q$
2: $\mathtt{sk} \leftarrow x$
3: $\mathtt{pk} \leftarrow g^x$
4: **return** $(\mathtt{pk}, \mathtt{sk})$

$\mathtt{Enc}(\mathtt{pk} = g^x, m \in G)$

Require: $m \in G$
1: $y \xleftarrow{\$} \mathbb{Z}_q$
2: $w \leftarrow g^y$
3: $v \leftarrow m \cdot (g^x)^y$
4: **return** $c = (w, v)$

$\mathtt{Dec}(\mathtt{sk} = x, c = (w, v) \in G^2)$

1: $\hat{m} \leftarrow (w^x)^{-1} \cdot v$
2: **return** $\hat{m}$

Figure 9: ElGamal cryptosystem

The security of ElGamal cryptosystem reduces to the conjectured intractability of the computational Diffie-Hellman problem and the decisional Diffie-Hellman problem:

Definition 1 (**computational Diffie-Hellman problem (CDH)**)**.** Let $x, y \xleftarrow{\$} \mathbb{Z}_q$ be uniformly random samples. Given (g, g^x, g^y), compute g^{xy}.

Definition 2 (**decisional Diffie-Hellman problem (DDH)**)**.** Let $x, y, z \xleftarrow{\$} \mathbb{Z}_q$ be uniformly random samples. Let $h \xleftarrow{\$} \{g^z, g^{xy}\}$ be randomly chosen between g^z and g^{xy}. Given (g, g^x, g^y, h), determine whether h is g^{xy} or g^z

It is also conjectured in [ABR01] that for certain choice of cyclic group G, the computational Diffie-Hellman problem remains intractable even if the adversary as access to a restricted decisional Diffie-Hellman oracle. This assumption is captured in the Gap Diffie-Hellman problem:

Definition 3 (**Gap Diffie-Hellman problem**)**.** Let $G = \langle g \rangle$ be a cyclic group of prime order $q > 2$. Let $x, y \xleftarrow{\$} \mathbb{Z}_q$ be uniformly random samples. Given (g, g^x, g^y) and a restricted DDH oracle $\mathcal{O}^{\mathrm{DDH}} : (u, v) \mapsto [\![u^x = v]\!]$, compute g^{xy}.

We now present the proof for Lemma 2

Proof. We will prove by sequence of game. A summary can be found in Figure 10

$G_0 - G_2$

1: $x \xleftarrow{\$} \mathbb{Z}_q$
2: $m^* \xleftarrow{\$} G$
3: $y \xleftarrow{\$} \mathbb{Z}_q, w \leftarrow g^y$
4: $v \leftarrow m^* \cdot (g^x)^y$ ▷ G_0 - G_1
5: $v \xleftarrow{\$} G$ ▷ G_2
6: $c^* \leftarrow (w, v)$
7: $\hat{m} \xleftarrow{\$} A^{\mathtt{PCO}}(g^x, c^*)$ ▷ G_0
8: $\hat{m} \xleftarrow{\$} A^{\mathtt{PCO}_1}(g^x, c^*)$ ▷ G_1 - G_2
9: **return** $[\![\hat{m} = m^*]\!]$ ▷ G_0 - G_1
10: **return** $[\![\hat{m} = w^{-x} \cdot v]\!]$ ▷ G_2

$\mathtt{PCO}(m, c = (w, v))$

1: **return** $[\![m = (w^x)^{-1} \cdot v]\!]$

$\mathtt{PCO}_1(m, c = (w, v))$

1: **return** $[\![(w^x) = m^{-1} \cdot v]\!]$

Figure 10: Lemma 2 sequence of games

Game 0 is the OW-PCA game. Adversary A has access to the plaintext-checking oracle $\mathtt{PCO}$ and wins the game if it can correctly recover the challenge plaintext m^*.

Game 1 is identical to game 0, except that the formulation of the $\mathtt{PCO}$ is changed. When servicing the plaintext-checking query $(m, c = (w, v))$, $\mathtt{PCO}_1$ checks whether w^x is equal to $m^{-1} \cdot v$. Observe that in the cyclic group G, the algebraic expressions in $\mathtt{PCO}$ and $\mathtt{PCO}_1$ are equivalent, which means that $\mathtt{PCO}_1$ behaves identically to $\mathtt{PCO}$.

Game 2 is identical to game 1 except for two modifications: first, when computing the challenge ciphertext, v is no longer computed from m^* but is randomly sampled; second, the win condition changed from $\hat{m} = m^*$ to $\hat{m} = w^{-x} \cdot v$. Notice that in game 0-1, $v \xleftarrow{\$} m^* \cdot (g^x)^y$ follows uniform random distribution on the cyclic group G because m^* is uniformly random in the cyclic group, so adversary A retains its advantage in "recovering the decryption" when v becomes truly uniformly random in the cyclic group. It is easy to verify that the two win conditions are equivalent. Up to this point, we have simply moved things around, and game 0 through game 2 are algebraically equivalent:

$$\mathtt{Adv}_0(A) = \mathtt{Adv}_1(A) = \mathtt{Adv}_2(A)$$

The Gap Diffie-Hellman adversary B can perfectly simulate game 2 for A (see Figure 11): B receives as the Gap Diffie-Hellman problem inputs g^x and g^y. g^x simulates an ElGamal public key, where as g^y simulates the first component of the challenge ciphertext. As in game 2, the second component of the challenge ciphertext can be randomly sampled. Finally, the $\mathtt{PCO}_1$ from game 2 can be perfectly simulated using the restricted DDH oracle $\mathcal{O}^{\mathrm{DDH}}$.

$B^{\mathcal{O}^{\mathrm{DDH}}}(g, g^x, g^y)$

1: $w \leftarrow g^y$
2: $v \xleftarrow{\$} G$
3: $c^* \leftarrow (w, v)$
4: $\hat{m} \xleftarrow{\$} A^{\mathtt{PCO}_2}(g^x, c^*)$
5: **return** $\hat{m}^{-1} \cdot v$

$\mathcal{O}^{\mathrm{DDH}}(u, v)$

1: **return** $[\![u^x = v]\!]$

$\mathtt{PCO}_2(m, c = (w, v))$

1: **return** $\mathcal{O}^{\mathrm{DDH}}(w, m^{-1} \cdot v)$

Figure 11: Gap Diffie-Hellman adversary B simulates game 2 for A

If A wins game 2, then its output is $\hat{m} = w^{-x} \cdot v = g^{-xy} \cdot v$, so $m^{-1} \cdot v$ is g^{xy}, the correct answer to the Gap Diffie-Hellman problem. In other words, B solves its Gap Diffie-Hellman problem if and only if A wins the simulated game 2.

$$\mathtt{Adv}_2(A) = \mathtt{Adv}_{\mathrm{GapDH}}(B)$$

□

4 Practical instantiation with ML-KEM

ML-KEM is an IND-CCA secure key encapsulation mechanism standardized by NIST in FIPS 203 [oST24]. The chosen-ciphertext security of ML-KEM is achieved in two steps. First, ML-KEM constructs a PKE (called K-PKE, see Figure 16 in Appendix) whose IND-CPA security reduces to the conjectured intractability of the decisional Module Learning with Error (MLWE) problem [Reg05, LPR10, Reg09, LPR10, Pei09]. Then, the $\mathtt{KEM}_m^{\not\perp}$ variant (see Figure 1) of the Fujisaki-Okamoto transformation is applied to convert the IND-CPA secure PKE into an IND-CCA secure KEM (See Figure 17 in Appendix).

4.1 ML-KEM+

We apply the encrypt-then-MAC transformation to the K-PKE routines of ML-KEM. The resulting KEM is called ML-KEM$^+$ (See Figure 12).

For performance and practical security, ML-KEM$^+$ makes two main modifications to the encrypt-then-MAC transformation:

- **Hashing $\mathtt{pk}$ into MAC key and shared secret.** The public key goes into deriving both the MAC key and the shared secret for two main reasons. First, in a realistic key exchange, the public key and the random PKE plaintext are generated by opposite parties. Hashing both the public key and the PKE plaintext ensures that both parties have inputs into the shared secret. The second reason is to prevent multitarget attacks described in [BDK+18]: if the MAC key is derived from the PKE plaintext alone, then an adversary can pre-compute a large dictionary mapping MAC keys to the pre-image PKE plaintext. Upon receiving some ciphertext, an adversary can

ML-KEM⁺.KeyGen()

$z \xleftarrow{\$} \{0,1\}^{256}$
$(\mathtt{pk}, \mathtt{sk}') \xleftarrow{\$} \mathtt{K\text{-}PKE.KeyGen}()$
$h \leftarrow H(\mathtt{pk})$
$\mathtt{sk} \leftarrow (sk' \| \mathtt{pk} \| h \| z)$
return $(\mathtt{pk}, \mathtt{sk})$

ML-KEM⁺.Encap(pk)

$m \xleftarrow{\$} \{0,1\}^{256}$
$(\overline{K}, k) \leftarrow G(m \| H(\mathtt{pk}))$
$c' \xleftarrow{\$} \mathtt{K\text{-}PKE.Enc}(\mathtt{pk}, m)$
$t \leftarrow \mathtt{MAC}(k, c')$
$K \leftarrow \mathtt{KDF}(\overline{K} \| t)$
$c \leftarrow (c', t)$
return (c, K)

ML-KEM⁺.Decap(sk, c)

Require: Secret key $\mathtt{sk} = (\mathtt{sk}' \| \mathtt{pk} \| h \| z)$
Require: Ciphertext $c = (c' \| t)$
$(\mathtt{sk}', \mathtt{pk}, h, z) \leftarrow \mathtt{sk}$
$(c', t) \leftarrow c$
$\hat{m} \leftarrow \mathtt{K\text{-}PKE.Dec}(\mathtt{sk}', c')$
$(\overline{K}, \hat{k}) \leftarrow G(\hat{m} \| h)$
$\hat{t} \leftarrow \mathtt{MAC}(\hat{k}, c')$
if $\hat{t} = t$ **then**
    $K \leftarrow \mathtt{KDF}(\overline{K} \| t)$
else
    $K \leftarrow \mathtt{KDF}(z \| c)$
end if
return K

Figure 12: ML-KEM$^+$ applies our encrypt-then-MAC transformation to K-PKE. G is SHA3-512, H is SHA3-256, KDF is Shake256

search through this key-plaintext dictionary and recover the decryption. Adding the public key into the derivation of the MAC key will dramatically increase the space and time requirement of such dictionary attack and render it infeasible in practice.

- **Hashing the MAC tag instead of the entire ciphertext.** Because we allow the PKE to be randomized, the shared secret must have inputs from both the plaintext and the ciphertext. However, we find it unnecessary to hash the entire ciphertext. Instead, since the MAC tag functions as a keyed hash of the ciphertext, we can use the MAC tag as a substitute ciphertext hash, which yields meaningful performance improvements for shared secret derivation. On the other hand, not hashing the entire ciphertext will not give an adversary additional advantage because asssuming that the adversary does not already know the underlying decryption, it cannot tamper with any part of the challenge ciphertext without making the ciphertext-tag invalid, so the decapsulation oracle will always return the implicit rejection.

Compared to ML-KEM, ML-KEM$^+$ makes the following performance trade-offs:

- ML-KEM$^+$ encapsulation contains two additional steps: computing the MAC tag and hashing the MAC tag into the shared secret.
- ML-KEM$^+$ ciphertext size increases by the size of the MAC tag
- ML-KEM$^+$ decapsulation replaces re-encryption with computing the MAC tag

4.2 Choosing a message authentication code (MAC)

For implementation, we instantiated the MAC with a selection that covered a wide range of MAC designs, including Poly1305 [Ber05], GMAC [MV04], CMAC [IK03][BR05], and KMAC [Gro13]. All MACs are parameterized with a 256-bit key to ensure that MAC keys are at least as hard to guess as the underlying PKE plaintext. On the other hand, all MACs use 128-bit tag except for KMAC, which can have variable tag length. We suspect

that 128-bit tag might be insufficient for the 192-bit and 256-bit security level, so for instantiation with the higher security levels, we only use KMAC with 192-bit and 256-bit tag length output.

Poly1305 and GMAC are both Carter-Wegman style authenticators [WC81] that compute the tag using finite field arithmetic. Generically speaking, Carter-Wegman MAC operates by breaking the message into message blocks that can then be parsed into finite field elements. The tag is computed by evaluating a polynomial whose coefficients are the message blocks and whose interdeterminate is the secret key, which is also called a *hash key*. Specifically, Poly1035 operates in the prime field $\mathbb{F}_q$ where $q = 2^{130} - 5$ whereas GMAC operates in the binary field $\mathbb{F}_{2^{128}}$.

CMAC is based on the CBC-MAC with the block cipher instantiated from AES-256. To compute a CMAC tag, the message is first broke into 128-bit blocks with appropriate padding. Each block is first XOR'd with the previous block's output, then encrypted under AES using the symmetric key. The final output is XOR'd with a sub key derived from the symmetric key, before being encrypted for one last time.

KMAC is based on the SHA-3 family of sponge functions [oST15]. We chose KMAC-256, which uses Shake256 as the underlying extendable output functions. KMAC is the only MAC to support variable key and tag length, though we fixed the key length at 256 bits. On the other hand, we chose the appropriate tag length for the corresponding security level: when instantiated with ML-KEM-512, the tag length is 128 bits; with ML-KEM-768 the tag length is 192 bits; with ML-KEM-1024 the tag length is 256 bits.

5 Performance comparison

We measured the real-world performance of the individual routines of our ML-KEM$^+$ construction, as well as the round trip time of performing key exchange over a network in a variety of authentication modes. Our implementation extended from the reference implementation by the PQCrystals team [BDK+24]. All C 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.

5.1 MAC performance

To isolate the performance characteristics of each authenticator in our instantiation of ML-KEM$^+$ we measured the CPU cycles needed for each authenticator to compute a tag on random inputs whose sizes correspond to the ciphertext sizes of ML-KEM. The measurements are summarized in Table 3.

Table 3: CPU cycles needed to compute tag on various input sizes

Input size: 768 bytes	
MAC	Median (ccl)
Poly1305	909
GMAC	3899
CMAC	6291
KMAC 128-bit tag	6373

Input size: 1088 bytes	
MAC	Median (ccl)
KMAC 192-bit tag	9697

Input size: 1568 bytes	
MAC	Median (ccl)
KMAC 256-bit tag	11647

5.2 Individual routine performance

Compared to the $U_m^{\not\perp}$ variant of Fujisaki-Okamoto transformed used in ML-KEM, the encrypt-then-MAC transformation achieves 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 $U_m^{\not\perp}$ transformation is dominated by the runtime saving gained from replacing *re-encryption* with MAC. A comparison between ML-KEM and variations of the ML-KEM$^+$ can be found in Table 4.

Remark. We also included the performance of Kyber as it is submitted to the third round of the NIST PQC standardization project, which differs from ML-KEM by deriving the shared secret from hashing the public key, the PKE plaintext, and the PKE ciphertext, while ML-KEM derives its shared secret by hashing only the public key and the PKE plaintext. This results in ML-KEM having meaningful performance improvement in encapsulation over Kyber, though the performance difference in decapsulation is negligible.

Table 4: CPU cycles of each KEM routine

128-bit security	
size parameters (bytes)	
`pk` size	800
`sk` size	1632
ct size	768
`KeyGen` cycles/tick	
Median	75945
Average	76171

KEM variant	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
ML-KEM-512	91467	92065	121185	121650
Kyber512	97811	98090	119937	120299
ML-KEM$^+$-512 w/ Poly1305	93157	93626	33733	33908
ML-KEM$^+$-512 w/ GMAC	97369	97766	37725	37831
ML-KEM$^+$-512 w/ CMAC	99739	99959	40117	39943
ML-KEM$^+$-512 w/ KMAC	101009	101313	40741	40916

192-bit security	
size parameters (bytes)	
`pk` size	1184
`sk` size	2400
ct size	1088
`KeyGen` cycles/tick	
Median	129895
Average	130650

KEM variant	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
ML-KEM-768	136405	147400	186445	187529
Kyber768	153061	153670	182129	182755
ML-KEM$^+$-768 w/ KMAC	155219	155848	52415	52611

256-bit security	
size parameters (bytes)	
`pk` size	1568
`sk` size	3168
ct size	1568
`KeyGen` cycles/tick	
Median	194921
Average	195465

KEM variant	`Encap` cycles/tick		`Decap` cycles/tick	
	Median	Average	Median	Average
ML-KEM-1024	199185	199903	246245	247320
Kyber1024	222351	223260	258231	259067
ML-KEM$^+$-1024 w/ KMAC	216761	217468	62269	62516

5.3 Key exchange protocols

The CRYSTALS-Kyber team [BDK+18] proposed three key exchange protocols: `Kyber.KE` for a passively secure ephemeral key exchange, `Kyber.UAKE` for unilaterally authenticated key exchange, and `Kyber.AKE` for mutually authenticated key exchange. First formally proposed in [BCK98] and later analyzed in [CK01], 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 Kyber key exchange protocols using ML-KEM$^+$ 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) recorded.

5.3.1 Unauthenticated key exchange

In unauthenticated key exchange (KE), a single pair of ephemeral keypair $(\texttt{pk}_e, \texttt{sk}_e) \xleftarrow{\$} \texttt{KeyGen}()$ is generated by the client. The client transmits the ephemeral public key $\texttt{pk}_e$ to the server, who runs the encapsulation routine $(c_e, K_e) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_e)$ and transmits the ciphertext c_e back to the client. The client finally decapsulates the ciphertext to recover the shared secret $K_e \leftarrow \texttt{Decap}(\texttt{sk}_e, c_e)$. The key exchange routines are summarized in Figure 13. The key derivation function is instantiated using Shake256, and the final session key is 256 bits in length. The RTT comparison is summarized in Table 5

$\texttt{KE}_\texttt{C}()$

1: $(\texttt{pk}_e, \texttt{sk}_e) \xleftarrow{\$} \texttt{KeyGen}()$
2: $\texttt{send}(\texttt{pk}_e)$
3: $c_e \leftarrow \texttt{read}()$
4: $K_e \leftarrow \texttt{Decap}(\texttt{sk}_e, c_e)$
5: $K \leftarrow \texttt{KDF}(K)$
6: **return** K

$\texttt{KE}_\texttt{S}()$

1: $\texttt{pk}_e \leftarrow \texttt{read}()$
2: $(c_e, K_e) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_e)$
3: $\texttt{send}(c_e)$
4: $K \leftarrow \texttt{KDF}(K_e)$
5: **return** K

Figure 13: Unauthenticated key exchange (KE) routines

Table 5: KE RTT comparison

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	800	768	92	97
ML-KEM$^+$-512 w/ Poly1305	800	784	70	72
ML-KEM$^+$-512 w/ GMAC	800	784	73	76
ML-KEM$^+$-512 w/ CMAC	800	784	75	79
ML-KEM$^+$-512 w/ KMAC	800	784	76	78

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	1184	1088	135	140
ML-KEM$^+$-768 w/ KMAC	1184	1120	103	107

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	1568	1568	193	199
ML-KEM$^+$-1024 w/ KMAC	1568	1600	144	149

5.3.2 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 $\texttt{pk}_S$ of the server, and the server authenticates itself by correctly decrypting the challenge encryption sent by the client. UAKE routines are summarized in Figure 14. Performance data is summaried in Table 6.

5.3.3 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

$\texttt{UAKE}_\texttt{C}(\texttt{pk}_S)$

Require: Server's long-term $\texttt{pk}_S$

1: $(\texttt{pk}_e, \texttt{sk}_e) \xleftarrow{\$} \texttt{KeyGen}()$
2: $(c_S, K_S) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_S)$
3: $\texttt{send}(\texttt{pk}_e, c_S)$
4: $c_e \leftarrow \texttt{read}()$
5: $K_e \leftarrow \texttt{Decap}(\texttt{sk}_e, c_e)$
6: $K \leftarrow \texttt{KDF}(K_e \| K_S)$
7: **return** K

$\texttt{UAKE}_\texttt{S}(\texttt{sk}_S)$

Require: Server's long-term $\texttt{sk}_S$

1: $(\texttt{pk}_e, c_S) \leftarrow \texttt{read}()$
2: $K_S \leftarrow \texttt{Decap}(\texttt{sk}_S, c_S)$
3: $(c_e, K_e) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_e)$
4: $\texttt{send}(c_e)$
5: $K \leftarrow \texttt{KDF}(K_e \| K_S)$
6: **return** K

Figure 14: Unilaterally authenticated key exchange (UAKE) routines

Table 6: UAKE RTT comparison

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	1568	768	145	151
ML-KEM$^+$-512 w/ Poly1305	1584	784	103	106
ML-KEM$^+$-512 w/ GMAC	1584	784	106	110
ML-KEM$^+$-512 w/ CMAC	1584	784	108	112
ML-KEM$^+$-512 w/ KMAC	1584	784	109	113

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	2272	1088	215	222
ML-KEM$^+$-768 w/ KMAC	2288	1104	154	159

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	3136	1568	310	318
ML-KEM$^+$-1024 w/ KMAC	3152	1584	213	220

server's long-term public key and its own long-term secret key, while the server possesses client's long-term public key and its own long-term secret key. 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. the key exchange routines are described in Figure 15 and the performance data summarized in Table 7.

6 Conclusions and future works

The encrypt-then-MAC transformation is a generic KEM construction whose IND-CCA security reduces to the OW-PCA security of the input PKE and the one-time existential unforgeability of the input MAC in the random oracle model. Compared to the Fujisaki-Okamoto transformation, the encrypt-then-MAC replaces the computationally expensive re-encryption with computing a MAC tag. At the cost of minimal increase in encapsulation cost and ciphertext size, the encrypt-then-MAC substantially improves the efficiency of the decapsulation routine. Where the input PKE's encryption is slower than decryption, the encrypt-then-MAC KEM achieves meaningful time savings in practical key exchange protocols.

In our future work, we would like to look at applying encryption-then-MAC transformation to other post-quantum cryptosystems for improving their practical implementation efficiency.

$\texttt{AKE}_\texttt{C}(\texttt{pk}_S, \texttt{sk}_C)$

Require: Server's long-term $\texttt{pk}_S$
Require: Client's long-term $\texttt{sk}_C$
1: $(\texttt{pk}_e, \texttt{sk}_e) \xleftarrow{\$} \texttt{KeyGen}()$
2: $(c_S, K_S) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_S)$
3: $\texttt{send}(\texttt{pk}_e, c_S)$
4: $(c_e, c_C) \leftarrow \texttt{read}()$
5: $K_e \leftarrow \texttt{Decap}(\texttt{sk}_e, c_e)$
6: $K_C \leftarrow \texttt{Decap}(\texttt{sk}_e, c_C)$
7: $K \leftarrow \texttt{KDF}(K_e \| K_S \| K_C)$
8: **return** K

$\texttt{AKE}_\texttt{S}(\texttt{sk}_S, \texttt{pk}_C)$

Require: Server's long-term $\texttt{sk}_S$
Require: Client's long-term $\texttt{pk}_C$
1: $(\texttt{pk}_e, c_S) \leftarrow \texttt{read}()$
2: $K_S \leftarrow \texttt{Decap}(\texttt{sk}_S, c_S)$
3: $(c_e, K_e) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_e)$
4: $(c_C, K_C) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}_C)$
5: $\texttt{send}(c_e, c_C)$
6: $K \leftarrow \texttt{KDF}(K_e \| K_S \| K_C)$
7: **return** K

Figure 15: Mutually authenticated key exchange (AKE) routines

Table 7: AKE RTT comparison

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-512	1568	1536	220	213
ML-KEM$^+$-512 w/ Poly1305	1584	1568	133	138
ML-KEM$^+$-512 w/ GMAC	1584	1568	139	143
ML-KEM$^+$-512 w/ CMAC	1584	1568	143	148
ML-KEM$^+$-512 w/ KMAC	1584	1568	145	151

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-768	2272	2176	294	301
ML-KEM$^+$-768 w/ KMAC	2288	2208	204	210

KEM variant	Client TX bytes	Server TX bytes	RTT (μs)	
			Median	Average
ML-KEM-1024	3136	3136	512	511
ML-KEM$^+$-1024 w/ KMAC	3152	3168	282	288

Code-based cryptosystems. Because the general problem of decoding a linear code is proven NP hard [BMvT78], code-based cryptosystems does not suffer from the inherent search-decision equivalence of lattice problems and thus be viable candidates with OW-PCA security. Unfortunately, among the code-based submissions to NIST PQC, HQC [MAB+18] and BIKE [ABB+22] are known to be vulnerable to key-recovery plaintext-checking attacks (KR-PCA) [TUX+23]. On the other hand, classic McEliece [ABC+20] seems to be PCA secure and thus a viable candidate, although in classic McEliece, the decoding routine is more expensive than the encryption routine, so applying encrypt-then-MAC may yield less impressive performance gains.

Isogeny-based cryptosystems. The intractability assumptions of isogeny-based cryptography resemble the classical Diffie-Hellman assumptions, and it seems possible to formulate a "Gap Diffie-Hellman assumption" in supersingular isogeny [FTTY18]. In fact, SIKE [ACC+17] also uses the Fujisaki-Okamoto transformation, and given the heavy computational cost of isogeny computation, replacing re-encryption with MAC will result in substantial performance improvements. While SIKE and SIDH were found to be insecure [CD23], other isogeny-based cryptosystem such as CSIDH [CLM+18] remains unaffected by the aforementioned attack and might be suitable candidates.

7 Appendix

Below are high-level summaries of the PKE sub-routines and the KEM routines of ML-KEM:

K-PKE.KeyGen()

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: $\mathtt{pk} \leftarrow (A, \mathbf{t})$
6: $\mathtt{sk} \leftarrow \mathbf{s}$
7: **return** $(\mathtt{pk}, \mathtt{sk})$

K-PKE.Enc(pk, m)

Ensure: $\mathtt{pk} = (A, \mathbf{t})$
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** $(\mathbf{c}_1, c_2)$

K-PKE.Dec(sk, c)

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

Figure 16: K-PKE is an IND-CPA secure PKE based on the conjectured intractability of the Module Learning with Error (MWLE) problem

ML-KEM.KeyGen()

1: $(\mathtt{pk}, \mathtt{sk}') \xleftarrow{\$} \mathtt{K\text{-}PKE.KeyGen()}$
2: $h \leftarrow \mathtt{SHA3\text{-}256}(\mathtt{pk})$
3: $z \xleftarrow{\$} \mathcal{M}$
4: $\mathtt{sk} \leftarrow (\mathtt{sk}', \mathtt{pk}, h, z)$
5: **return** $(\mathtt{pk}, \mathtt{sk})$

ML-KEM.Encap(pk)

1: $m \xleftarrow{\$} \mathcal{M}$
2: $h \leftarrow \mathtt{SHA3\text{-}256}(\mathtt{pk})$
3: $(K, r) \leftarrow \mathtt{SHA3\text{-}512}(m, h)$
4: $c \leftarrow \mathtt{K\text{-}PKE.Enc}(\mathtt{pk}, m, r)$
5: **return** (c, K)

ML-KEM.Decap(sk, c)

1: $\hat{m} \leftarrow \mathtt{K\text{-}PKE.Dec}(\mathtt{sk}', c)$
2: $(\overline{K}, \hat{r}) \leftarrow \mathtt{SHA3\text{-}512}(\hat{m}, h)$
3: $\hat{c} \leftarrow \mathtt{K\text{-}PKE.Enc}(\mathtt{pk}, m, \hat{r})$
4: **if** $\hat{c} = c$ **then**
5: $\quad K \leftarrow \overline{K}$
6: **else**
7: $\quad K \leftarrow \mathtt{SHA3\text{-}256}(c, z)$
8: **end if**
9: **return** K

Figure 17: ML-KEM uses the $U_m^{\not\perp}$ variant of the modular Fujisaki-Okamoto KEM transformation

References

[ABB+22] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, et al. Bike: bit flipping key encapsulation. *NIST PQC Round 4*, 2022.

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, RubenNiederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter

Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic mceliece. Technical report, National Institute of Standards and Technology, 2020.

[ABR99] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHAES: an encryption scheme based on the diffie-hellman problem. *IACR Cryptol. ePrint Arch.*, page 7, 1999.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[ACC+17] Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, et al. Supersingular isogeny key encapsulation. *Submission to the NIST Post-Quantum Standardization project*, 152:154–155, 2017.

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1006–1018. ACM, 2016.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 419–428. ACM, 1998.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 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*, pages 353–367. IEEE, 2018.

[BDK+24] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. official reference implementation of the kyber key encapsulation mechanism, 2024.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[Ber05] Daniel J. Bernstein. The poly1305-aes message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1998.

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory*, 24(3):384–386, 1978.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *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*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[BP18] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. *IACR Cryptol. ePrint Arch.*, page 526, 2018.

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[BR05] John Black and Phillip Rogaway. CBC macs for arbitrary-length messages: The three-key constructions. *J. Cryptol.*, 18(2):111–131, 2005.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.

[CHJ+02] Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. GEM: A generic chosen-ciphertext secure encryption method. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2002.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 207–222. Springer, 2004.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, pages 13–25. Springer, 1998.

[Den03] Alexander W. Dent. A designer's guide to kems. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2003.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, volume 10831 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2018.

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, 2004.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptol.*, 26(1):80–101, 2013.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2001.

[fS09] International Organization for Standardization. *ISO/IEC 27004: Information technology-security techniques-information security management-measurement.* ISO, 2009.

[FTTY18] Atsushi Fujioka, Katsuyuki Takashima, Shintaro Terada, and Kazuki Yoneyama. Supersingular isogeny diffie-hellman authenticated key exchange.

In Kwangsu Lee, editor, *Information Security and Cryptology - ICISC 2018 - 21st International Conference, Seoul, South Korea, November 28-30, 2018, Revised Selected Papers*, volume 11396 of *Lecture Notes in Computer Science*, pages 177–195. Springer, 2018.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[Gro13] Joint Task Force Transformation Initiative Interagency Working Group. Security and privacy controls for federal information systems and organizations. Technical Report NIST Special Publication (SP) 800-53, Rev. 4, Includes updates as of January 22, 2015, National Institute of Standards and Technology, Gaithersburg, MD, 2013.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.

[HHM22] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Failing gracefully: Decryption failures and the fujisaki-okamoto transform. In Shweta Agrawal and Dongdai Lin, editors, *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*, volume 13794 of *Lecture Notes in Computer Science*, pages 414–443. Springer, 2022.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: one-key CBC MAC. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.

[Kal98] Burt Kaliski. PKCS #1: RSA encryption version 1.5. *RFC*, 2313:1–19, 1998.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *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*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[MAB+18] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti,

Gilles Zémor, and IC Bourges. Hamming quasi-cyclic (hqc). *NIST PQC Round*, 2(4):13, 2018.

[MKJR16] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, November 2016.

[MV04] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[OP01a] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.

[OP01b] Tatsuaki Okamoto and David Pointcheval. REACT: rapid enhanced-security asymmetric cryptosystem transform. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer, 2001.

[oST15] National Institute of Standards and Technology. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical Report Federal Information Processing Standards Publication (FIPS) NIST FIPS 202, U.S. Department of Commerce, Washington, D.C., 2015.

[oST24] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report Federal Information Processing Standards Publication (FIPS) NIST FIPS 203, U.S. Department of Commerce, Washington, D.C., 2024.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[RRCB19] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and KEM schemes. *IACR Cryptol. ePrint Arch.*, page 948, 2019.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. *IACR Cryptol. ePrint Arch.*, page 112, 2001.

[Sho02] Victor Shoup. OAEP reconsidered. *J. Cryptol.*, 15(4):223–249, 2002.

[TUX+23] Yutaro Tanaka, Rei Ueno, Keita Xagawa, Akira Ito, Junko Takahashi, and Naofumi Homma. Multiple-valued plaintext-checking side-channel attacks on post-quantum kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):473–503, 2023.

[UXT+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/em analysis on post-quantum kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, 2022.

[WC81] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.