

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

Ganyu Xu¹, Kalikinkar Mandal², and Guang Gong¹

¹Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, CANADA, Email: {g66xu,ggong}@uwaterloo.ca

²Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, E3B 5A3, CANADA, Email: kmandal@unb.ca

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 FO 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 revisit the “encrypt-then-MAC” paradigm as a generic KEM transformation: the PKE ciphertext is authenticated by a message authentication code (MAC) whose key is derived from the PKE plaintext. We prove that the transformation achieves IND-CCA security in the random oracle model when the underlying PKE is one-way under plaintext-checking attack (OW-PCA), and we introduce a fine-grained security notion, *one-wayness under q -bounded plaintext-checking attack* (OW- q PCA), under which the transformation achieves IND- q CCA security. In contrast to the tag-based IND-1CCA transformations of Huguenin-Dumittan and Vaudenay (Eurocrypt 2022), whose tag is a random-oracle hash of the plaintext and the entire ciphertext, our transformation applies the random oracle only to the short PKE plaintext, while the long ciphertext is processed by a standard-model MAC that needs only to be *one-time* existentially unforgeable. We present ML-KEM-EtM, an IND- q CCA secure KEM derived from applying the encrypt-then-MAC transformation to the PKE subroutines of ML-KEM, and McElieceEtM, an instantiation with Classic McEliece that achieves full IND-CCA security under the conjectured OW-PCA security of its Niederreiter PKE. We 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 the decapsulation CPU cycle count.

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 [50] and RSA signature in ISO 9796 [30], were badly broken

with sophisticated cryptanalysis [17, 36, 23]. 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 [32] into the widely adopted RSA-OAEP. The FO transformation [31, 42] 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, 52]. 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” [67].

1.1 Our contributions

Our contributions are as follows.

1. **Generic KEM transformation with a standard-model MAC.** We formalize an encrypt-then-MAC transformation that constructs an IND-CCA-secure KEM in the random oracle model from an OW-PCA-secure PKE and a one-time existentially unforgeable MAC. Unlike the Fujisaki–Okamoto transformation, our construction replaces de-randomization and re-encryption with MAC generation and verification. Compared with the tag-based transformations T_H and T_{CH} of Huguenin-Dumittan and Vaudenay [45], our construction applies the random oracle only to the short PKE plaintext, while the ciphertext is processed by a standard-model MAC. This design can substantially reduce the cost of processing long ciphertexts.
2. **Fine-grained chosen-ciphertext security.** The PKE subroutines of ML-KEM are not OW-PCA secure, but known plaintext-checking attacks require many adaptively chosen ciphertexts. To capture this setting, we introduce one-wayness under q -bounded plaintext-checking attack (OW- q PCA). We prove that the encrypt-then-MAC transformation converts an OW- q PCA-secure PKE into an IND- q CCA-secure KEM (Theorem 2), thereby giving a formal security guarantee as a function of the number of decapsulation queries.
3. **Instantiation with ML-KEM.** We present ML-KEM-EtM, obtained by applying our transformation to the PKE subroutines of ML-KEM. Compared with FIPS 203 [64], ML-KEM-EtM incurs modest encapsulation and ciphertext-size overhead while eliminating the re-encryption step in decapsulation, resulting in significant performance improvements (Table 1).
4. **Instantiation with Classic McEliece.** We present McElieceEtM, based on the Niederreiter PKE subroutines of Classic McEliece (Section 4). Under the conjectured OW-PCA security of the underlying PKE, McElieceEtM achieves full IND-CCA security. Our C implementation improves combined encapsulation-plus-decapsulation throughput by 9–12% over the reference implementation.

1.2 Limitations

We state the limitations of our results explicitly, since they determine where the encrypt-then-MAC transformation can and cannot be deployed.

	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. **ML-KEM-EtM is not a drop-in replacement for ML-KEM.** The PKE subroutines of ML-KEM are *not* OW-PCA secure: efficient key-recovery plaintext-checking attacks are known (Section 3.3). Consequently, Theorem 1 does *not* apply to ML-KEM-EtM, and ML-KEM-EtM must not be used in applications where a key pair serves many decapsulation queries, such as static-key servers. The security claim we make for ML-KEM-EtM is IND- q CCA (Theorem 2) for small q , which covers ephemeral key exchange ($q = 1$) and bounded key reuse.
2. **The IND- q CCA claim rests on a conjecture.** Theorem 2 is unconditional as a reduction, but its application to ML-KEM-EtM requires the conjectured OW- q PCA security of the PKE subroutines (Section 3.3). The conjecture is supported only by the query complexity of the best *known* plaintext-checking attacks, which is an upper bound; no lower bound is known.
3. **Full IND-CCA security requires an OW-PCA secure PKE, which is rare among post-quantum candidates.** Among the NIST PQC KEM candidates, only Classic McEliece has no known plaintext-checking attack, and its OW-PCA security is itself conjectural. Our fully IND-CCA secure instantiation (McElieceEtM, Section 4) inherits this conjecture.

1.3 Related works

1.3.1 Fujisaki-Okamoto transformation

Fujisaki and Okamoto proposed to construct CCA secure hybrid PKE by combining an OW-CPA secure PKE and a semantically secure symmetric-key encryption (SKE) scheme [31]. 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 [81, 79]. Further studies [26, 42, 14, 43, 87, 49] 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 FO transformation is the only known generic transformation that converts an OW-CPA/IND-CPA PKE into a *fully* IND-CCA secure KEM without additional assumptions on the PKE. Because of the minimal input requirement and the simple construction, the FO transformation was widely adopted among post-quantum KEM candidates submitted to the PQC standardization project, including Kyber [19], Saber [25], FrodoKEM [18], and Classic

McEliece [13].

Despite its widespread adoption, the FO transformation has many drawbacks:

- **Computational inefficiency.** In all variants of FO 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 [83, 82, 44, 40, 35, 54, 53, 60, 28, 84], 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 [10, 11].

1.3.2 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 [80], although Fujisaki et al. later proved that RSA-OAEP is secure under the RSA assumption [32]. 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 Pointcheval proposed REACT [68] in 2001, followed by GEM [22] in 2002. Both are generic CCA transformations 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 our scheme and the Diffie-Hellman Integrated Encryption Scheme (DHIES) [1, 2]: both schemes protect 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.

1.3.3 Tag-based IND-1CCA transformations

Closest to our work are the transformations T_H , T_{CH} , and T_{RH} of Huguenin-Dumittan and Vaudenay [45], which are themselves inspired by REACT [68]. In T_{CH} , the encapsulator appends the tag $tag \leftarrow H(m, c')$ to the PKE ciphertext c' , where H is a random oracle; the decapsulator recomputes the tag from the decrypted plaintext instead of re-encrypting. Under the ROM, T_H and T_{CH} convert an OW-CPA/IND-CPA PKE into an IND-1CCA secure KEM, which suffices for ephemeral key exchange; subsequent works analyzed these transformations in the QROM [48, 89]. Our transformation replaces the random-oracle tag $H(m, c')$ with the MAC tag $MAC(G(m), c')$, and this difference is precisely what our contributions exploit: (i) the random oracle is applied only to the short plaintext m , while the long ciphertext c' is processed by a standard-model MAC, which is structurally lighter as an assumption and free to be instantiated with fast constructions, including ones built from encryption algorithms: universal-hash MACs (Poly1305, GMAC) and block-cipher MACs (CMAC with AES-NI) run at one to two CPU cycles per byte, several times faster than the SHA3 evaluation that a hash tag requires over the same input (see Section 5.2 for the full discussion, including the sponge-based KMAC256 case, where

Transform	PKE requirement	Integrity mechanism	KEM security	Decap. cost
EtM (This work)	OW-PCA OW- q PCA	MAC tag $\text{MAC}(G(m), c')$, RO on short input only	IND-CCA IND- q CCA (ROM)	Dec + MAC
FO [31, 42]	OW-CPA/IND-CPA	re-encryption	IND-CCA (ROM, QROM)	Dec + Enc
OAEP [7]	OW trapdoor permutation	redundancy in padding	IND-CCA (ROM)	Dec
REACT [68]	OW-PCA	hash tag $H(m, c')$	IND-CCA (ROM)	Dec+ hash
DHIES [2]	Gap-DH (ElGamal only)	MAC tag	IND-CCA	Dec + MAC
$U^\mathcal{I}$ [42]	OW-PCA	none (implicit rejection only)	IND-CCA (ROM, QROM)	Dec+ hash
T_H/T_{CH} [45]	OW-CPA/IND-CPA	hash tag $H(m, c')$	IND-1CCA (ROM; QROM [48, 89])	Dec+ hash

Table 2: Comparison of generic CCA transformations. “Hash tag” mechanisms evaluate a random oracle over the plaintext and the entire ciphertext; the MAC tag in **EtM** evaluates the random oracle only over the short plaintext.

the speed advantage vanishes but the assumption advantage remains); (ii) assuming the stronger OW-PCA (respectively OW- q PCA) security of the PKE, our transformation achieves full IND-CCA (respectively IND- q CCA) security, interpolating between the IND-1CCA guarantee of T_{CH} and the IND-CCA guarantee of REACT. Table 2 summarizes the comparison.

1.3.4 Relation to the modular Fujisaki-Okamoto framework

Hofheinz, Hövelmanns, and Kiltz [42] (HHK17) decompose FO into modular transformations, including U -transformations that construct IND-CCA-secure KEMs from OW-PCA- or OW-PCVA-secure PKEs without re-encryption. Our construction is closely related but uses a MAC to provide explicit ciphertext integrity while processing the long ciphertext outside the random oracle. This permits explicit rejection from OW-PCA without requiring the additional γ -spreadness conditions associated with the corresponding explicit-rejection FO constructions.

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 $\llbracket B \rrbracket$ 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 (PKE) scheme is a collection of three routines (KeyGen , Enc , Dec) defined over some plaintext space $\mathcal{M}$ and some ciphertext space $\mathcal{C}$. Key generation $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$ is a randomized routine that returns a keypair consisting of a public encryption key and a secret decryption key. The encryption routine $\text{Enc} : (\text{pk}, m) \mapsto c$ encrypts the input plaintext m under the input public key pk and produces a ciphertext c . The decryption routine $\text{Dec} : (\text{sk}, c) \mapsto m$ 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

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

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.

Rigidity. Following Bernstein and Persichetti [14], a correct deterministic PKE is called *rigid* if for every keypair (pk, sk) and every ciphertext c in the ciphertext space, $\text{Dec}(\text{sk}, c) = m \neq \perp$ implies $\text{Enc}(\text{pk}, m) = c$. In other words, decryption followed by encryption is the identity map on valid ciphertexts, so every valid ciphertext is the honest encryption of its own decryption. RSA is rigid; randomized PKEs are in general not rigid, since many ciphertexts may decrypt to the same plaintext.

Security. The security of PKE's is conventionally discussed using adversarial games played between a challenger and an adversary [34]. In the OW-ATK game (Figure 6), 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 the probability of producing the correct decryption: $\text{Adv}_{\text{PKE}}^{\text{OW-ATK}}(A) = P[\hat{m} = m^*]$. 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 1), 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: $\text{Adv}_{\text{PKE}}^{\text{IND-ATK}}(A) = |P[\hat{b} = b] - \frac{1}{2}|$. 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 1 for algorithmic description of the oracles. Throughout the paper, we call an adversary playing the OW-ATK (resp. IND-ATK) game an *OW-ATK* (resp. *IND-ATK*) *adversary*; for example, an OW-PCA adversary is an adversary with access to the plaintext-checking oracle whose goal is to invert the challenge ciphertext, and an IND-CCA adversary against a KEM is an adversary with access to the decapsulation oracle whose goal is to distinguish the encapsulated shared secret from random (Section 2.2).

The advantage of an adversary in the OW-ATK game is the probability that it outputs the correct decryption. The advantage of an adversary in the IND-ATK game is defined below. A

OW-ATK Game	IND-ATK Game	$\mathcal{O}_{\text{PCO}}(m, c)$
1: $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$	1: $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$	1: return $\llbracket m = \text{Dec}(\text{sk}, c) \rrbracket$
2: $m^* \xleftarrow{\$} \mathcal{M}$	2: $(m_0, m_1) \xleftarrow{\$} A^{\mathcal{O}_{\text{ATK}}}(1^\lambda, \text{pk})$	
3: $c^* \xleftarrow{\$} \text{Enc}(\text{pk}, m^*)$	3: $b \xleftarrow{\$} \{0, 1\}$	
4: $\hat{m} \xleftarrow{\$} A^{\mathcal{O}_{\text{ATK}}}(1^\lambda, \text{pk}, c^*)$	4: $c^* \xleftarrow{\$} \text{Enc}(\text{pk}, m_b)$	$\mathcal{O}_{\text{Dec}}(c)$
5: return $\llbracket \hat{m} = m^* \rrbracket$	5: $\hat{b} \xleftarrow{\$} A^{\mathcal{O}_{\text{ATK}}}(1^\lambda, \text{pk}, c^*)$	1: return $\text{Dec}(\text{sk}, c)$
	6: return $\llbracket \hat{b} = b \rrbracket$	

Figure 1: The one-way game, indistinguishability game, plaintext-checking oracle (PCO), and decryption oracle. $\text{ATK} \in \{\text{CPA}, \text{PCA}, \text{CCA}\}$

PKE is OW-ATK/IND-ATK secure if no efficient adversary has non-negligible advantage in the corresponding security game:

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

Bounded plaintext-checking attack ($q\text{PCA}$). We introduce a fine-grained restriction of the plaintext-checking attack in which the adversary’s plaintext-checking capability is bounded. This notion is motivated by the fact that, for many post-quantum PKEs that are not OW-PCA secure, all known plaintext-checking attacks require a large number of *adaptively crafted* ciphertexts (see Section 3.3); a plaintext-checking oracle that answers queries on only a few ciphertexts is therefore conjectured to be of limited use to the adversary.

Definition 1 (OW- $q\text{PCA}$ security). Let $q \geq 0$ be an integer. The OW- $q\text{PCA}$ game is identical to the OW-PCA game, except that the plaintext-checking oracle $\mathcal{O}^{\text{PCO}}(m, c)$ answers only queries in which the ciphertext c is the challenge ciphertext c^* or one of at most q distinct other ciphertexts; once the adversary has queried q distinct ciphertexts $c \neq c^*$, further queries involving new ciphertexts are answered with $\perp$. A PKE is said to be OW- $q\text{PCA}$ secure if no efficient adversary can win the OW- $q\text{PCA}$ game with non-negligible probability.

We make two remarks on Definition 1. First, the bound counts *distinct ciphertexts* rather than oracle queries: this is the quantity that governs known key-recovery plaintext-checking attacks, each of which must submit many *different* perturbed ciphertexts, whereas repeated queries $(m_1, c), (m_2, c), \dots$ against a fixed ciphertext c reveal at most the single decryption of c . Second, queries against the challenge ciphertext c^* are unrestricted: such queries only *confirm* candidate decryptions of an honestly generated ciphertext and correspond to a plaintext-confirmation capability that the adversary in our reductions obtains for free. By definition, OW-PCA security implies OW- $q\text{PCA}$ security for every q , and OW- $q\text{PCA}$ security implies OW- $q'\text{PCA}$ security for every $q' \leq q$.

2.2 Key encapsulation mechanism (KEM)

Syntax. A key encapsulation mechanism (KEM) is a collection of three routines (**KeyGen**, **Encap**, **Decap**) defined over some ciphertext space $\mathcal{C}$ and some key space $\mathcal{K}$. Key generation **KeyGen** : $1^\lambda \mapsto (\text{pk}, \text{sk})$ is a randomized routine that returns a keypair. Encapsulation **Encap** : $\text{pk} \mapsto (c, K)$ 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 **Decap** : $(\text{sk}, c) \mapsto K$ is a deterministic routine that uses the secret key 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:

$$P \left[\text{Decap}(\text{sk}, c) = K \mid \begin{array}{l} (\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda); \\ (c, K) \xleftarrow{\$} \text{Encap}(\text{pk}) \end{array} \right] = 1.$$

Security. Similar to PKE security, the security of KEM is discussed using adversarial games. In the IND-ATK game (Figure 9), 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.

The advantage of an adversary is its probability of winning beyond 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.

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

where the probability is taken over the randomness of the game in Figure 9. By default, all adversaries are assumed to have the public key, with which they can mount chosen plaintext attacks (ATK = CPA). If the adversary has access to a decapsulation oracle $\mathcal{O}^{\text{Decap}} : c \mapsto \text{Decap}(\text{sk}, c)$, 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* [24, 45]. In many practical scenarios such as ephemeral key exchange (e.g. TLS 1.3 handshake [74]), 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 [76, 29].

2.3 Message authentication code (MAC)

Syntax. A message authentication code (MAC) is a collection of two routines (**Sign**, **Verify**) defined over some key space $\mathcal{K}$, some message space $\mathcal{M}$, and some tag space $\mathcal{T}$. The signing routine **Sign** : $(k, m) \mapsto t$ authenticates the message m under the symmetric key k by producing a tag t . The verification routine **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 $t \leftarrow \text{MAC}(k, m)$, 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 $t \leftarrow \text{MAC}(k, m; r)$.

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 $\mathcal{O}^{\text{Sign}} : m \mapsto \text{Sign}(k, m)$ and tries to produce a valid message-tag pair that has not been queried from the signing oracle (Figure 10).

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.

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 an OW-PCA secure PKE and an existentially unforgeable MAC, and an IND-qCCA secure KEM when the PKE is only OW-qPCA secure. Our

$\text{KeyGen}_{\text{EtM}}()$	$\text{Encap}_{\text{EtM}}(\text{pk})$	$\text{Decap}_{\text{EtM}}(\text{sk}, c)$
1: $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}_{\text{PKE}}()$ 2: $s \xleftarrow{\$} \mathcal{M}_{\text{PKE}}$ 3: $\text{sk} \leftarrow (\text{sk}, s)$ 4: return (pk, sk)	1: $m \xleftarrow{\$} \mathcal{M}_{\text{PKE}}$ 2: $k \leftarrow G(m)$ 3: $c' \xleftarrow{\$} \text{Enc}_{\text{PKE}}(\text{pk}, m)$ 4: $t \leftarrow \text{MAC}(k, c')$ 5: $c \leftarrow (c', t)$ 6: $K \leftarrow H(m, c)$ 7: return (c, K)	Require: $c = (c', t)$ Require: $\text{sk} = (\text{sk}', s)$ 1: $\hat{m} \leftarrow \text{Dec}_{\text{PKE}}(\text{sk}', c')$ 2: $\hat{k} \leftarrow G(\hat{m})$ 3: if $\text{MAC}(\hat{k}, c') = t$ then 4: $K \leftarrow H(\hat{m}, c)$ 5: else 6: $K \leftarrow H(s, c)$ 7: end if 8: return K

Figure 2: The encrypt-then-MAC KEM routines builds a KEM, denoted by KEM_{EtM} , using a $\text{PKE}(\text{KeyGen}, \text{Enc}, \text{Dec})$, a MAC, and two hash functions G, H .

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 [67], our construction’s CCA security reduces generically to the PCA security of the input PKE; in addition, we prove (Lemma 2) 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, Bellare, and Rogaway originally proposed to use HMAC and CBC-MAC, which are many-time secure MACs but less efficient than one-time MACs). Compared with the hash-tag transformations T_H/T_{CH} [45], the random oracles G and H in our construction are evaluated only on short inputs (the PKE plaintext, and the plaintext concatenated with a short tag, respectively), while the long PKE ciphertext is consumed exclusively by the standard-model MAC. The data flow of the encapsulation is illustrated in Figure 4.

In Section 3.1 we will describe the encrypt-then-MAC KEM routines and state the security reductions. 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, then derive the fine-grained IND- q CCA reduction (Theorem 2) by counting the plaintext-checking queries of the reduction. 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 $\mathcal{K}_{\text{KEM}}$ denote the set of all possible shared secrets. Let $(\text{KeyGen}_{\text{PKE}}, \text{Enc}_{\text{PKE}}, \text{Dec}_{\text{PKE}})$ be a PKE defined over message space $\mathcal{M}_{\text{PKE}}$ and ciphertext space $\mathcal{C}_{\text{PKE}}$. Let $\text{MAC} : \mathcal{K}_{\text{MAC}} \times \mathcal{B}^* \rightarrow \mathcal{T}$ be a MAC over key space $\mathcal{K}_{\text{MAC}}$ and tag space $\mathcal{T}$. Let $G : \mathcal{B}^* \rightarrow \mathcal{K}_{\text{MAC}}, H : \mathcal{B}^* \rightarrow \mathcal{K}_{\text{KEM}}$ be hash functions. The encrypt-then-MAC transformation $\text{EtM}[\text{PKE}, \text{MAC}, G, H]$ constructs a KEM $(\text{KeyGen}_{\text{EtM}}, \text{Encap}_{\text{EtM}}, \text{Decap}_{\text{EtM}})$ (Figure 2).

We chose to construct KEM_{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 [26, 42], if the input PKE is *rigid* (i.e. $m = \text{Dec}(\text{sk}, c)$ if and only if $c = \text{Enc}(\text{pk}, m)$), such as with RSA, then the shared secret may be derived from the plaintext alone $K \leftarrow H(m)$.

The CCA security of KEM_{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 KEM_{EtM} that makes at most q_D decapsulation queries, q_G queries to G , and q_H queries to H , there exist an OW-PCA adversary B against the underlying PKE and a (no-message) existential forgery adversary C against the underlying MAC, both running in about the time of A , such that:*

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

where B makes at most $q_G + q_H + q_D \cdot q_G$ plaintext-checking queries, and the queried ciphertexts comprise the challenge ciphertext together with at most q_D distinct other ciphertexts.

3.2 Proof of Theorem 1

We will prove Theorem 1 using a sequence of games. A summary of the sequence of games can be found in Figures 11 and 12. From a high level we made three incremental modifications to the IND-CCA game for KEM_{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 $k^* \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$. 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 $K_0 \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$. 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.

The sequence-of-games figures and the complete proof are deferred to Appendix C.

The query counting at the end of the proof shows that the plaintext-checking capability that B requires is driven by the number of *decapsulation* queries of A : every ciphertext that B submits to the plaintext-checking oracle, other than the challenge ciphertext itself, originates from one of A 's decapsulation queries. Since the OW-qPCA game (Definition 1) restricts exactly this quantity, the proof of Theorem 1 carries over verbatim to the bounded setting, yielding our fine-grained reduction:

Theorem 2. *Let $q \geq 0$ be an integer. For every IND-qCCA adversary A against KEM_{EtM} that makes at most $q_D \leq q$ decapsulation queries, q_G queries to G , and q_H queries to H , there exist an OW-qPCA adversary B against the underlying PKE and a (no-message) existential forgery adversary C against the underlying MAC, both running in about the time of A , such that:*

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

Proof. Identical to the proof of Theorem 1. The adversary B queries the plaintext-checking oracle on the challenge ciphertext (unrestricted in the OW- q PCA game) and on at most $q_D \leq q$ distinct other ciphertexts, so B is a valid OW- q PCA adversary. $\square$

Theorem 2 interpolates between two extremes. When the underlying PKE is fully OW-PCA secure (e.g. conjecturally, the Niederreiter PKE of Classic McEliece; see Section 3.3), taking $q \rightarrow \infty$ recovers full IND-CCA security. When the PKE admits key-recovery plaintext-checking attacks using N crafted ciphertexts but is conjectured OW- q PCA secure for $q \ll N$ (e.g. the PKE subroutines of ML-KEM), the KEM remains IND- q CCA secure, which covers ephemeral key exchange ($q = 1$) as well as bounded key reuse.

Concrete bounds We illustrate the concrete security of the bound in Theorem 2 with the ephemeral key exchange setting ($q_D = 1$) and the instantiation ML-KEM-EtM-768 with Poly1305. The one-time forgery advantage of Poly1305 against a message of L bytes is at most $8\lceil L/16 \rceil / 2^{106}$ [9]; for the 1088-byte PKE ciphertext of ML-KEM-768, this is at most 2^{-96} . The bound of Theorem 2 therefore reads

$$\text{Adv}_{\text{KEM}_{\text{EtM}}}^{\text{IND-1CCA}}(A) \leq 2^{-96} + 2 \cdot \text{Adv}_{\text{PKE}}^{\text{OW-1PCA}}(B),$$

i.e., the security of the KEM is, up to a factor of two and a negligible additive term, exactly the conjectured one-wayness of the underlying PKE against an adversary holding a single-ciphertext plaintext-checking capability. For the block-cipher and sponge-based MACs (CMAC, KMAC256) with 128-bit tags, the forgery term is bounded by standard PRF switching bounds and is likewise negligible for q_D far exceeding any realistic key reuse. In all cases the reduction to the PKE is tight, so no re-parameterization of the underlying PKE is necessary; this is in contrast to the original FO transformation, whose reduction to OW-CPA loses a factor of the number of hash queries [31].

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 [68], 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 [22], 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 3). More recently, the modular FO transformation [42] 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:*

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

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 5), 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.

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 3.

Number-theoretic cryptosystems The RSA cryptosystem as it was originally proposed in [75] is OW-PCA secure. This is because the RSA cryptosystem is a one-way trapdoor permutation [32], 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 $c \equiv m^e \pmod{N}$, which renders a plaintext-checking oracle completely useless.

The ElGamal cryptosystem [33] is OW-PCA secure if the Gap Diffie-Hellman assumption [67] holds for the underlying group (finite field or elliptic curve). This is the basis on which the CCA security of DHIES [2] is proved. We do not benchmark DHIES against our post-quantum instantiations: DHIES relies on the discrete logarithm problem and is broken by Shor’s algorithm [77, 78], so it is not a candidate for the post-quantum deployments targeted in this paper, and a performance comparison across the classical/post-quantum boundary (where elliptic-curve DHIES enjoys far smaller keys and ciphertexts but different hardness assumptions) would not be meaningful.

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 [37, 4]. Similar attacks for Kyber and Saber were described in [44, 86, 39]. As Peikert pointed out in [70], 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 [41] proposed KR-PCA for the original NTRU cryptosystem, which Ueno et al. [83] adapted into a KR-PCA against modern instantiations such as NTRU-HPS and NTRU-HRSS (also see [88]). Ueno et al. also adapted [47] into a KR-PCA against NTRU-Prime.

Code-based cryptosystems HQC [58], despite being based on hard coding problems, is structurally similar to lattice-based cryptosystems Kyber, Saber, etc. [83]. Consequently, KR-PCA for lattice cryptosystems can be easily adapted to work on HQC [44, 4]. BIKE [3] is based on the Niederreiter cryptosystem instantiated with quasi-cyclic moderate density parity check (QC-MDPC) code [59]. [38] described KR-PCA against QC-MDPC code, which can partially recover BIKE secret keys [86]. There is no known adaptive attack against Classic McEliece [83]. While the decoding subroutine of Classic McEliece is significantly slower than the encoding subroutine, so removing re-encryption yields a smaller relative speedup than for ML-KEM, our implementation still measures a 9–12% throughput improvement (Section 4).

Query complexity and OW- q PCA security All of the key-recovery plaintext-checking attacks cited above are *adaptive* and require the plaintext-checking oracle to answer queries on a large number of distinct, adversarially crafted ciphertexts: each query perturbs the ciphertext with a different noise offset, and the attack aggregates the answers to reconstruct the secret

key coefficient by coefficient. For the parameter sets of Kyber and Saber, the reported attacks require on the order of 10^3 oracle queries [44, 86, 39]. In contrast, a plaintext-checking oracle restricted to a handful of distinct ciphertexts is not known to give the adversary any advantage in recovering the secret key or inverting an honest ciphertext. We therefore conjecture that the PKE subroutines of ML-KEM are OW- q PCA secure (Definition 1) for q substantially below the query complexity of the best known plaintext-checking attack. We emphasize the epistemic status of this conjecture: known attacks provide only an *upper* bound on the query complexity of plaintext-checking attacks, and any improvement in attack query complexity directly erodes the margin of the conjecture. Deriving a lower bound, e.g. from the search-decision equivalence of the underlying lattice problem, is an interesting open problem.

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 [11, 10], 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 $k_{\text{MAC}} \leftarrow G(\text{pk}, m)$ or even including random salt at each encapsulation $k_{\text{MAC}} \leftarrow G(\text{pk}, m, \text{salt})$, which can increase the cost of such dictionary attacks.

One-time MAC Within the security proof of Theorem 1, the forgery adversary C requires no signing oracle, so one-time (indeed no-message) existential unforgeability of the MAC already suffices for the reduction. In deployment, a MAC key repeats across two encapsulations under the same public key only if the two encapsulations sample the same PKE plaintext or the hash function G collides. The following lemma makes the folklore argument precise.

Lemma 2. *Suppose at most q_E encapsulations are performed under a single public key, with PKE plaintexts sampled uniformly from $\mathcal{M}_{\text{PKE}}$ and G modeled as a random oracle. Then the probability that any MAC key is used to authenticate more than one message is at most*

$$\frac{q_E^2}{2} \left(\frac{1}{|\mathcal{M}_{\text{PKE}}|} + \frac{1}{|\mathcal{K}_{\text{MAC}}|} \right).$$

Consequently, replacing a many-time EUF-CMA secure MAC with a one-time existentially unforgeable MAC [21] degrades the concrete security of KEM_{ETM} by at most this additive term.

Proof. A MAC key repeats only if two encapsulations sample the same plaintext m , which by a union bound over all $\binom{q_E}{2}$ pairs happens with probability at most $q_E^2/(2|\mathcal{M}_{\text{PKE}}|)$, or if two distinct plaintexts collide under G , which for a random oracle with range $\mathcal{K}_{\text{MAC}}$ happens with probability at most $q_E^2/(2|\mathcal{K}_{\text{MAC}}|)$. $\square$

For ML-KEM, $|\mathcal{M}_{\text{PKE}}| = 2^{256}$ and $|\mathcal{K}_{\text{MAC}}| = 2^{256}$, so the additive term is negligible for any realistic keypair lifetime. 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. The security justification is as follows: substituting $K \leftarrow H(m, t)$ for $K \leftarrow H(m, c)$ is dangerous only if an adversary can find a second ciphertext $c_2 \neq c'$ that carries the same plaintext m and the same tag t , in which case the decapsulations of (c', t) and (c_2, t) would collide. But producing such a c_2 is producing a second message with a valid tag under the MAC key $G(m)$, which is unknown to the adversary; by the argument of game 1 in the proof of Theorem 1, this event is bounded by the forgery advantage against the MAC. The same design principle appears in Kyber’s original NIST submission, which derives the shared secret from a *hash* of the ciphertext rather than the full ciphertext [19]; our tag plays the role of that hash, keyed and precomputed.

Side-channel and fault attack considerations A substantial body of work exploits the re-encryption step of the FO transformation as a side-channel and fault target: leakage or fault behavior during re-encryption acts as a plaintext-checking oracle, enabling key recovery [83, 82, 40, 71, 37]. By removing re-encryption entirely, the encrypt-then-MAC transformation eliminates this attack surface. However, the transformation introduces two new surfaces that implementations must protect. First, the tag comparison $\text{MAC}(\hat{k}, c') \stackrel{?}{=} t$ must be implemented in constant time, and, in the gray-box fault model, an adversary who can skip or force the comparison obtains the “accept” branch $K \leftarrow H(\hat{m}, c)$ for an invalid ciphertext, which converts the decapsulation into a plaintext-dependent oracle. Standard countermeasures apply, e.g. performing the comparison twice, comparing over a randomized encoding, or deriving both branches and selecting in constant time. Second, the MAC key derivation $\hat{k} \leftarrow G(\hat{m})$ processes the decrypted plaintext, which is the same sensitive intermediate targeted by plaintext-checking side-channel attacks; a protected implementation must mask this hash call as ML-KEM implementations already must for their $G(\hat{m}||h)$ call. We note that the computation that must be masked in our decapsulation (one hash of a short input and one MAC) is substantially cheaper than masked re-encryption, which dominates the cost of protected ML-KEM implementations; quantifying this advantage with a masked implementation and a leakage evaluation (e.g. TVLA) is left for future work.

4 Applying encrypt-then-MAC to code-based cryptosystem

Background. Code-based cryptography was introduced by Robert J. McEliece in 1978 [57]. The McEliece cryptosystem samples a random (n, k) -linear code over some finite field $\mathbb{F}$ with generator matrix $G \in \mathbb{F}^{n \times k}$, then generates the public key by scrambling the generator matrix $\mathbf{pk} = G' \leftarrow P \cdot G \cdot S$ using a secret permutation matrix $P \in \mathbb{F}^{n \times n}$ and a secret invertible matrix $S \in \mathbb{F}^{k \times k}$. Because P and S are both invertible, G' is also a generator matrix for an (n, k) -linear code. The secret key consists of the decoding routine, the secret permutation matrix, and the secret invertible matrix. A plaintext message m is encrypted by first encoding it using G' , then adding an error vector $\mathbf{e}$ to the codeword: $c \leftarrow G' \cdot m + \mathbf{e}$. To decrypt a ciphertext, the permutation is first removed, then the error vector is removed using the decoding routine, and finally, the message is recovered.

The one-wayness of the McEliece cryptosystem reduces to the NP-hard problem of decoding a random linear code [8]. Harald Niederreiter improved the efficiency of the McEliece cryptosystem in 1986 [66] by replacing the generator matrix of a random linear code with the parity-check matrix. Correspondingly, the plaintext space becomes the set of fixed-weight error vectors, which are encrypted by computing the syndrome under the parity-check matrix, $c \leftarrow H\mathbf{e}$. The one-wayness of the Niederreiter variant reduces to the syndrome decoding problem, which is proven NP-hard [8]. Compared to the McEliece formulation, the Niederreiter variant enjoys a smaller public key and smaller ciphertexts.

Remark. Three code-based KEMs—HQC, BIKE, and Classic McEliece ([58, 3, 13]) — were evaluated in the fourth round of NIST’s PQC standardization process. This round concluded in 2025, with HQC selected for standardization. As discussed in Section 3.3, there exist full or partial key-recovery plaintext-checking attacks against HQC and BIKE. Regarding Classic McEliece, Ueno et al. [83] stated in 2022 that *there is no known adaptive attack on the PKE of Classic McEliece*. To our knowledge, no plaintext-checking attack against the underlying Classic McEliece PKE, in the sense required to violate its OW-PCA security, has subsequently been proposed. The CCA security of instantiating the encrypt-then-MAC KEM with Classic McEliece is thus based on the conjectured OW-PCA security of the subroutines described in Section 4.1 below.

4.1 Construction of McElieceEtM

Classic McEliece [13] is an IND-CCA secure post-quantum KEM submitted to the PQC standardization project and is currently one of three viable fourth-round KEM candidates. Classic McEliece is constructed in two layers. The first layer is an OW-CPA secure PKE based on the Niederreiter cryptosystem using a random binary Goppa code, and the second layer is a modified FO transformation. Each instance of Classic McEliece is parameterized by the base field size m (which induces a finite field with order $q = 2^m$), the codeword size n , and the error vector weight t .

Algorithm ?? describes the key generation routine of Classic McEliece. In step 4, **FieldOrdering** takes the input bits and outputs either $\perp$ or a sequence $(\alpha_0, \dots, \alpha_{q-1})$ distinct elements of $\mathbb{F}_q$. In step 5, **Irreducible** takes the input bits and outputs either $\perp$ or a monic irreducible degree- t polynomial $g \in \mathbb{F}_q[x]$. In step 7, **MatGen** outputs either $\perp$ or an $mt \times k$ matrix over $\mathbb{F}_2$. We refer readers to [13] for details of these subroutines.

Algorithms ?? and ?? describe the encoding and decoding subroutines of the Niederreiter layer. Note that our description of the decoding routine differs from [13] in that we did not include the weight and/or re-encryption check. Instead, they are included in the description of the decapsulation subroutines (Algorithm ??). This is because we later plan to substitute these checks with computing a MAC tag when applying the encrypt-then-MAC KEM transformation.

Algorithms ?? and ?? describe Classic McEliece’s KEM construction using the CPA secure Niederreiter cryptosystem described above. In step 1 of Algorithm ??, **FixedWeight** samples a random vector $\mathbf{e} \in \mathbb{F}_2^n$ with Hamming weight t . In this KEM construction, Classic McEliece makes use of re-encryption (step 3 of Algorithm ??) to ensure the integrity of the ciphertext.

Subroutines of McElieceEtM. We instantiate the encrypt-then-MAC KEM transformation using the subroutines **SeededKeyGen**, **FixedWeight**, **Encode**, and **Decode**. The result is called McElieceEtM. Figure 13 presents the encapsulation and decapsulation routines of McElieceEtM; pseudocode for the underlying Classic McEliece subroutines (Algorithms ??–??) is reproduced in Appendix D. Its key generation routine is identical to the key generation of Classic McEliece. Its encapsulation derives a MAC key from the output of **FixedWeight** and signs the syndrome computed from **Encode**. In decapsulation, weight and syndrome check is replaced with re-deriving the MAC key and checking the tag.

By Theorem 1, if the conjectured OW-PCA security of the Niederreiter PKE of Classic McEliece holds, then McElieceEtM as constructed in Figure 13 is IND-CCA secure. Among the instantiations considered in this paper, McElieceEtM is the only one that achieves full IND-CCA security, since Classic McEliece is the only NIST PQC candidate with no known plaintext-checking attack (Table 3).

4.2 Performance benchmark

The subroutines `SeededKeyGen`, `FixedWeight`, `Encode`, and `Decode` were taken from the `vec` implementation submitted to the fourth round of NIST PQC standardization project¹. While there are a total of 10 sets of parameters in Classic McEliece (see Section E), we only modified the 5 standard parameter variants. The only difference between the standard variants and the f-variants is the key generation routine, which is not the focus of this paper. Key pairs are fully interoperable between the standard and the f-variants, and the encapsulation/decapsulation routines are entirely identical.

We implement the McElieceEtM KEM in the C language to benchmark the encapsulation and decapsulation algorithms for above four different MACs. We used the MAC implementations from `OpenSSL 3.5.0`. Our C code is compiled using Apple Clang 15.0.0. The performance measurement is run on Apple Silicon M1 chip. The CPU clock is measured using `mach_absolute_time`. Each routine is run 10000 times, with median times reported in Table 6. Full per-parameter-set results are given in Table 6 in Appendix E.

5 Applying encrypt-then-MAC to ML-KEM

ML-KEM is an IND-CCA secure post-quantum KEM standardized by NIST in FIPS 203 [64]. 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 [72, 56, 73, 69]. Then, ML-KEM applies the FO transformation to convert the IND-CPA secure PKE into an IND-CCA secure KEM.

5.1 ML-KEM under encrypt-then-MAC

We apply the encrypt-then-MAC transformation to the PKE routines of ML-KEM [64]. The resulting KEM is denoted as ML-KEM-EtM (Figure 3). The key generation routines of ML-KEM-EtM and ML-KEM are identical. 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, so applying the encrypt-then-MAC KEM transformation to the PKE subroutines will not achieve full IND-CCA security. However, all known plaintext-checking attacks require on the order of 10^3 adaptively crafted ciphertexts (Section 3.3), and we conjecture the PKE subroutines to be OW- q PCA secure for q substantially below that threshold. Under this conjecture, Theorem 2 implies that ML-KEM-EtM is IND- q CCA secure; in particular it is IND-1CCA secure, which is the notion required for ephemeral key exchange such as post-quantum TLS 1.3 key exchange [76, 45, 89, 48], and more generally it tolerates bounded key reuse. 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 FO transformation.

5.2 Choosing MAC

A key degree of freedom of the encrypt-then-MAC transformation is that the ciphertext may be authenticated by *any* one-time existentially unforgeable MAC, including constructions built from encryption algorithms. The main design families are: (i) universal-hash (Carter-Wegman) MACs such as Poly1305 and GMAC, which evaluate a polynomial over a finite field at roughly one CPU cycle per byte; (ii) block-cipher-based MACs such as CMAC and PMAC [16], which run at one to two cycles per byte on processors with AES instructions; and (iii) sponge-based keyed

¹<https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-4/submissions/mceliece-Round4.tar.gz>

$\text{KeyGen}_{\text{ML-KEM-EtM}}()$	$\text{Encap}_{\text{ML-KEM-EtM}}(\text{pk})$	$\text{Decap}_{\text{ML-KEM-EtM}}(\text{sk}, c)$
1: $z \xleftarrow{\$} \{0, 1\}^{256}$ 2: $(\text{pk}, \text{sk}') \xleftarrow{\$} \text{KeyGen}_{\text{K-PKE}}()$ 3: $h \leftarrow H(\text{pk})$ 4: $\text{sk} \leftarrow (\text{sk}' \parallel \text{pk} \parallel h \parallel z)$ 5: return (pk, sk)	1: $m \xleftarrow{\$} \{0, 1\}^{256}$ 2: $h \leftarrow H(\text{pk})$ 3: $(\bar{K}, k) \leftarrow G(m \parallel h)$ 4: $c' \xleftarrow{\$} \text{Enc}_{\text{K-PKE}}(\text{pk}, m)$ 5: $t \leftarrow \text{MAC}(k, c')$ 6: $K \leftarrow \text{KDF}(\bar{K} \parallel t)$ 7: $c \leftarrow (c', t)$ 8: return (c, K)	1: $(\text{sk}', \text{pk}, h, z) \leftarrow \text{sk}$ 2: $(c', t) \leftarrow c$ 3: $\hat{m} \leftarrow \text{Dec}_{\text{K-PKE}}(\text{sk}', c')$ 4: $(\bar{K}, \hat{k}) \leftarrow G(\hat{m} \parallel h)$ 5: $\hat{t} \leftarrow \text{MAC}(\hat{k}, c')$ 6: if $\hat{t} = t$ then 7: $K \leftarrow \text{KDF}(\bar{K} \parallel t)$ 8: else 9: $K \leftarrow \text{KDF}(z \parallel c)$ 10: end if 11: return K

Figure 3: ML-KEM-EtM applies the encrypt-then-MAC KEM transformation to the PKE subroutines of ML-KEM.

hashes such as KMAC256, whose per-byte cost is that of KECCAK, roughly an order of magnitude higher in software.

This freedom is precisely what hash-tag transformations such as T_{CH} [45] do not have: there, the tag *is* a random-oracle hash of the plaintext and the full ciphertext, so the long input must be processed by a cryptographic hash function. The structural reason deserves emphasis. A Carter-Wegman MAC replaces the per-block invocation of a cryptographic primitive with a single multiplication in a finite field ($\mathbb{F}_{2^{130-5}}$ for Poly1305, $\mathbb{F}_{2^{128}}$ for GMAC), with only a constant number of primitive calls in total, and its unforgeability is information-theoretic *precisely because* the polynomial is evaluated at a secret, single-use key. The encrypt-then-MAC construction supplies exactly this setting: the MAC key $k = G(m)$ is secret, and by Lemma 2 it is used to authenticate only one message. A hash-tag transformation cannot exploit this trade-off even in principle, since its tag is an unkeyed random-oracle output: with no secret key, universal hashing is insecure by design, and the full per-block cost of a cryptographic hash is unavoidable.

Consequently, we expect ML-KEM-EtM instantiated with families (i) and (ii) to be substantially faster than T_{CH} , while the KMAC256 instantiation performs one KECCAK pass over a ciphertext-length input and should be comparable to T_{CH} in speed; the KMAC256 rows of Table 4 thus double as an approximate in-table proxy for the cost of a hash-tag transformation. Even in that case, the construction retains its proof-theoretic distinction: the MAC remains a keyed, standard-model primitive, and the long ciphertext never enters a random oracle.

Moreover, the comparison is robust to the choice of hash function on the hash-tag side: if a hash-tag transformation is instantiated with a different hash H' , e.g. SHA-256 on processors with SHA hardware extensions, then the encrypt-then-MAC construction can be instantiated with HMAC over the same H' , whose dominant cost is the identical single wide pass of H' over the ciphertext (the outer HMAC call processes only a short input). Under a matched hash choice, the encrypt-then-MAC transformation is therefore never slower than the hash-tag transformation, and it is strictly faster whenever a universal-hash or block-cipher MAC is selected — an option the hash-tag design cannot access, and one that HMAC, itself a standard-model PRF, preserves within our security proofs.

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

Poly1305 [9] and GMAC [61] are both Carter-Wegman style MACs [21, 85], which compute the tag using finite field arithmetic. It first parses the message into a sequence of finite field elements, then evaluates 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.

Remark (misuse resistance). The usual misuse hazards of one-time and nonce-based MACs, i.e. authenticating two messages under the same Poly1305 key, or repeating a GMAC nonce under the same key, do not arise in the encrypt-then-MAC KEM *by construction*: the MAC key $k = G(m)$ is derived from the PKE plaintext, which is freshly and uniformly sampled at every encapsulation, so every MAC key is used to authenticate exactly one message except with the negligible collision probability quantified in Lemma 2. In particular, using GMAC with a fixed nonce is safe here, because no key is ever reused. There is no nonce or key-counter state to manage, and hence no state-rollback misuse surface.

CMAC [63] is based on the CBC-MAC [15]. 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 subkey 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 constrained computing capacity but hardware support for AES. CMAC is many-time secure.

KMAC256 [51] is a pseudorandom function and keyed hash function based on KECCAK [62], although unlike the extendable output function (XOF) SHAKE, altering the requested output length generates a new unrelated output. Being sponge-based, KMAC256 processes the ciphertext at the per-byte cost of KECCAK and is therefore the slowest of our MACs, with expected performance comparable to a hash-tag transformation; we include it because it is NIST-standardized, it is the only construction with a flexible key and tag length, and it requires no primitives beyond the SHA3 family that ML-KEM implementations already carry.

Remark Beyond the four MACs above, there are parallelizable and lightweight designs such as PMAC [16], EliMAC [27], LightMAC [55], and ZMAC [46], which may offer better performance than the MACs listed above. We restricted our selection to constructions that are standardized (GMAC, CMAC, KMAC256) or widely deployed in mainstream cryptographic libraries (Poly1305), since availability of mature constant-time implementations is a prerequisite for the deployment scenarios we target. In our preliminary measurements, Poly1305 and GMAC outperformed EliMAC, and CMAC was comparable to it. In any case, the performance of the encrypt-then-MAC KEM is dominated by the public-key cryptographic operations, and the choice among fast MACs shifts the totals by only a few percent.

5.3 Experimental results

Our implementation of ML-KEM-EtM extended from Kyber’s reference implementation ². 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. The CPU supports the AES-NI and VAES instruction set extensions, which OpenSSL’s EVP_MAC implementations of GMAC and CMAC use automatically; Poly1305 and KMAC256 do not use AES and do not benefit from these extensions. 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

²<https://github.com/pq-crystals/kyber>

increase of encapsulation cycle count and ciphertext size. Since `K-PKE.Enc` carries a higher computational overhead than `K-PKE.Dec` or any MAC we chose, the performance advantage of the encrypt-then-MAC transformation over the FO transformation is dominated by the runtime saving gained from replacing *re-encryption* with MAC. The performance of ML-KEM-EtM is presented in Appendix B.

A timing comparison between the ML-KEM and the variations of ML-KEM-EtM is provided in Table 4. Because the CPU cycle count reduction in decapsulation is significantly more than the overhead in encapsulation, 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 F).

Classic McEliece parameter sets and benchmarks for McElieceEtM appears in Appendix E. For the different parameters of ML-KEM, for encapsulation, our ML-KEM-EtM is about 2.88 - 4.30% slower than ML-KEM and 1.46 - 2.68% slower than T_{CH} . However, for decapsulation, our ML-KEM-EtM consistently outperforms ML-KEM and T_{CH} , which is about 74.41 - 81.54% faster than ML-KEM and 11.75 - 16.78% faster than T_{CH} . For instance, our ML-KEM-EtM-1024 encapsulation is 2.88% and 1.46% slower than that of ML-KEM-1024 and T_{CH} , respectively. For the decapsulation, our ML-KEM-EtM-1024 is 81.54% and 16.78% faster than that of ML-KEM-1024 and T_{CH} , respectively.

Estimated comparison with hash-tag transformations As discussed in Section 5.2, a hash-tag transformation such as T_{CH} [45] admits essentially one instantiation in the ML-KEM ecosystem: its tag is a random-oracle hash, i.e. a SHA3/SHAKE evaluation over the plaintext and the full ciphertext. (Instantiating the hash tag with SHA-256 instead would not change the comparison: our construction would respond with HMAC-SHA-256 at matching cost; see Section 5.2.) Its decapsulation cost is therefore closely approximated by our KMAC256 rows, which likewise perform one KECCAK pass over a ciphertext-length input. Reading Table 4 with the KMAC256 row as a proxy for T_{CH} , the implied decapsulation advantage of the encrypt-then-MAC transformation with fast MACs is approximately 17% with Poly1305 (consistently across all three parameter sets), 7-12% with GMAC, and a more modest 2-5% with CMAC; the implied encapsulation advantage of Poly1305 is 5-8%. We emphasize that this is an estimate, and a slightly favorable one: T_{CH} performs one wide hash and one short hash per decapsulation, whereas our KMAC256 instantiation performs one wide KMAC call plus two short hashes and a KDF call, so the true T_{CH} baseline is likely marginally faster than the proxy row and the true advantage marginally smaller than the figures above. A directly measured T_{CH} baseline would confirm these estimates and is planned for the final version.

6 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 (one-time, no-message) unforgeability of the underlying MAC, and we introduced the fine-grained notion OW- q PCA under which the transformation achieves IND- q CCA security. We also analyzed generic attacks on the KEM and proposed countermeasures. We then proposed ML-KEM-EtM, an IND- q CCA secure KEM derived from applying the encrypt-then-MAC transformation to the PKE subroutines of ML-KEM, and McElieceEtM, a fully IND-CCA secure (under conjectured OW-PCA) instantiation with Classic McEliece. 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.

Future work Our security proofs are given in the classical random oracle model. A distinctive feature of the encrypt-then-MAC transformation is that the random oracles G and H are evaluated only on short inputs, while the long ciphertext is processed by the MAC, which is a standard-model primitive to which quantum random-oracle techniques need not be applied. We therefore expect a quantum random oracle model (QROM) analysis of the transformation, in the style of the QROM analyses of the T_H/T_{CH} transformations [48, 89], to confine the quantum reprogramming arguments to the short-input hash functions; carrying out this analysis and deriving the appropriate MAC parameters is the main direction of our future work.

References

- [1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHAES: an encryption scheme based on the Diffie-Hellman problem. *IACR Cryptol. ePrint Arch.*, page 7, 1999.
- [2] 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.
- [3] Nicolas Aragon, Paulo L. Barreto, Slim Bettaleb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. NIST post-quantum cryptography standardization round 4 submission: BIKE: Bit flipping key encapsulation. Technical report, National Institute of Standards and Technology (NIST), 2024.
- [4] Ciprian Baetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse attacks on post-quantum cryptosystems. In Yuval Ishai and Vincent Rijmen, editors, *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*, volume 11477 of *Lecture Notes in Computer Science*, pages 747–776. Springer, 2019.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.

- [9] 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.
- [10] Daniel J. Bernstein. FO derandomization sometimes damages security. Cryptology ePrint Archive, Paper 2021/912, 2021.
- [11] Daniel J. Bernstein. On the looseness of FO derandomization. *IACR Cryptol. ePrint Arch.*, page 912, 2021.
- [12] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Wen Wang, Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. Classic McEliece: Conservative code-based cryptography: Guide for security reviewers, October 23 2022.
- [13] Daniel J. Bernstein, Nadia Heninger, Tanja Lange, Michael van Beirendonck, et al. Classic mceliece: Specification, October 2022. Accessed: 2025-01-29.
- [14] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. *IACR Cryptol. ePrint Arch.*, page 526, 2018.
- [15] John Black and Phillip Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2000.
- [16] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.
- [17] 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.
- [18] 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.
- [19] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrede 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.
- [20] 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.

- [21] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
- [22] 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.
- [23] Jean-Sébastien Coron, David Naccache, and Julien P. Stern. On the security of RSA padding. 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 1–18. Springer, 1999.
- [24] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In Kaoru Kurosawa, editor, *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*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007.
- [25] 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.
- [26] 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.
- [27] Christoph Dobraunig, Bart Mennink, and Samuel Neves. EliMAC: Speeding up LightMAC by around 20%. *IACR Trans. Symmetric Cryptol.*, 2023(2):69–93, 2023.
- [28] Haiyue Dong and Qian Guo. 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.
- [29] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1197–1210. ACM, 2015.
- [30] International Organization for Standardization. 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.
- [31] 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.
- [32] 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.
- [33] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4):469–472, 1985.
 - [34] 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.
 - [35] Guillaume Goy, Julien Maillard, Philippe Gaborit, and Antoine Loiseau. Single trace HQC shared key recovery with SASCA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(2):64–87, 2024.
 - [36] François Grieu. A chosen messages attack on the ISO/IEC 9796-1 signature scheme. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 70–80. Springer, 2000.
 - [37] Qian Guo, Thomas Johansson, and Alexander Nilsson. 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.
 - [38] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *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*, volume 10031 of *Lecture Notes in Computer Science*, pages 789–815, 2016.
 - [39] Qian Guo and Erik Mårtensson. Do not bound to a single position: Near-optimal multi-positional mismatch attacks against Kyber and Saber. In Thomas Johansson and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings*, volume 14154 of *Lecture Notes in Computer Science*, pages 291–320. Springer, 2023.
 - [40] Julius Hermelink, Kai-Chun Ning, Richard Petri, and Emanuele Strieder. The insecurity of masked comparisons: Scas on ml-kem’s fo-transform. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 2430–2444. ACM, 2024.
 - [41] Jeffrey Hoffstein and Joseph H Silverman. Reaction attacks against the NTRU public key cryptosystem. Technical report, Technical Report 15, NTRU Cryptosystems, 1999.
 - [42] 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.
 - [43] 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.
- [44] Loïs Huguenin-Dumittan and Serge Vaudenay. Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I*, volume 12146 of *Lecture Notes in Computer Science*, pages 208–227. Springer, 2020.
 - [45] Loïs Huguenin-Dumittan and Serge Vaudenay. On IND-qCCA security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In Orr Dunkelman and Stefan Dziembowski, editors, *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*, volume 13277 of *Lecture Notes in Computer Science*, pages 613–642. Springer, 2022.
 - [46] Tetsu Iwata, Kazuhiko Minematsu, Thomas Peyrin, and Yannick Seurin. ZMAC: A fast tweakable block cipher mode for highly secure message authentication. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2017.
 - [47] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2000.
 - [48] Haodong Jiang, Zhi Ma, and Zhenfeng Zhang. Post-quantum security of key encapsulation mechanism against CCA attacks with a single decapsulation query. In Jian Guo and Ron Steinfeld, editors, *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*, volume 14441 of *Lecture Notes in Computer Science*, pages 434–468. Springer, 2023.
 - [49] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 96–125. Springer, 2018.
 - [50] Burt Kaliski. PKCS #1: RSA encryption version 1.5. *RFC*, 2313:1–19, 1998.
 - [51] John Kelsey, Shu jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Technical Report NIST Special Publication 800-185, National Institute of Standards and Technology, U.S. Department of Commerce, 2016.
 - [52] 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.
 - [53] Suparna Kundu, Siddhartha Chowdhury, Sayandeep Saha, Angshuman Karmakar, Debdeep Mukhopadhyay, and Ingrid Verbauwhede. 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.

- [54] Jinnuo Li, Chi Cheng, Muyan Shen, Peng Chen, Qian Guo, Dongsheng Liu, Liji Wu, and Jian Weng. Grafted trees bear better fruit: An improved multiple-valued plaintext-checking side-channel attack against kyber. *IACR Cryptol. ePrint Arch.*, page 1174, 2024.
- [55] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC mode for lightweight block ciphers. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2016.
- [56] 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.
- [57] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical report, NASA, 1978.
- [58] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaiieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, and Gilles Zémor. NIST post-quantum cryptography standardization round 4 submission: Hamming quasi-cyclic (HQC). Technical report, National Institute of Standards and Technology (NIST), 2024.
- [59] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. 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*, pages 2069–2073. IEEE, 2013.
- [60] Puja Mondal, Suparna Kundu, Sarani Bhattacharya, Angshuman Karmakar, and Ingrid Verbauwhede. A practical key-recovery attack on lwe-based key-encapsulation mechanism schemes using rowhammer. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III*, volume 14585 of *Lecture Notes in Computer Science*, pages 271–300. Springer, 2024.
- [61] National Institute of Standards and Technology. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. Technical Report NIST Special Publication 800-38D, U.S. Department of Commerce, 2007.
- [62] National Institute of Standards and Technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical report, Federal Information Processing Standards (FIPS) Publication 202, August 2015.
- [63] National Institute of Standards and Technology. Recommendation for block cipher modes of operation: The CMAC mode for authentication. Technical Report NIST Special Publication 800-38B, U.S. Department of Commerce, 2016.
- [64] 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.
- [65] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization: Security evaluation criteria, n.d. Accessed: 2025-01-20.
- [66] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.
- [67] 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.
- [68] 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.
 - [69] 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.
 - [70] Chris Peikert. Lattice cryptography for the internet. Cryptology ePrint Archive, Paper 2014/070, 2014.
 - [71] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on CCA-secure lattice-based PKE and KEM schemes. Cryptology ePrint Archive, Paper 2019/948, 2019.
 - [72] 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.
 - [73] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.
 - [74] E. Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, RFC Editor, August 2018.
 - [75] 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.
 - [76] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1461–1480. ACM, 2020.
 - [77] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.
 - [78] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
 - [79] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer, 2000.
 - [80] Victor Shoup. OAEP reconsidered. 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 239–259. Springer, 2001.
 - [81] Victor Shoup. A proposal for an ISO standard for public key encryption. *IACR Cryptol. ePrint Arch.*, page 112, 2001.

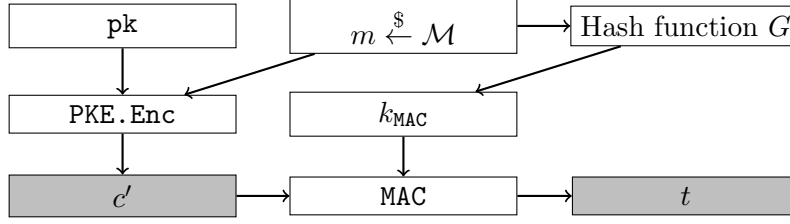


Figure 4: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

- [82] 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.
- [83] 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.
- [84] Ruize Wang, Martin Brisfors, and Elena Dubrova. A side-channel attack on a higher-order masked crystals-kyber implementation. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III*, volume 14585 of *Lecture Notes in Computer Science*, pages 301–324. Springer, 2024.
- [85] 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.
- [86] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. Fault-injection attacks against NIST’s post-quantum cryptography round 3 KEM candidates. In Mehdi Tibouchi and Huaxiong Wang, editors, *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*, volume 13091 of *Lecture Notes in Computer Science*, pages 33–61. Springer, 2021.
- [87] Keita Xagawa and Takashi Yamakawa. (tightly) QCCA-secure key-encapsulation mechanism in the quantum random oracle model. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*, volume 11505 of *Lecture Notes in Computer Science*, pages 249–268. Springer, 2019.
- [88] Xiaohan Zhang, Chi Cheng, and Ruoyu Ding. 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.
- [89] Biming Zhou, Haodong Jiang, and Yunlei Zhao. CPA-secure KEMs are also sufficient for post-quantum TLS 1.3. In Kai-Min Chung and Yu Sasaki, editors, *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*, volume 15486 of *Lecture Notes in Computer Science*, pages 433–464. Springer, 2024.

A Deferred figures and tables

This appendix collects figures and tables referenced from the main body: the standard security games (Figures 6–10), the encapsulation data flow (Figure 4), the plaintext-checking-oracle simulation used in Lemma 1 (Figure 5), and the OW-PCA security landscape (Table 3).

$\mathcal{O}_{\text{Decap}}^{\text{PCO}}(m, c')$
1: $k \leftarrow G(m)$
2: $t \leftarrow \text{MAC}(k, c')$
3: $c \leftarrow (c', t)$
4: $K \leftarrow H(m, c)$
5: return $\llbracket \mathcal{O}^{\text{Decap}}(c) = K \rrbracket$

Figure 5: 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) .

PKE	Category	Is it OW-PCA?
RSA	Number theoretic	Yes [26, 81]
ElGamal	Number theoretic or elliptic curve	If Gap Diffie-Hellman assumption holds [2, 67]
FrodoKEM	Lattice-based	No, full key recovery [37, 4]
Kyber/Saber	Lattice-based	No, full key recovery [44, 86, 39]
NTRU	Lattice-based	No, full key recovery [41, 47, 83]
HQC	Code-based	No, full key recovery [44, 4]
BIKE	Code-based	No, partial key recovery [38]
Classic McEliece	Code-based	Unknown, but there is no known attack [83]

Table 3: The landscape of OW-PCA security

Figures 6–10 give the algorithmic descriptions of the standard security games and oracles referenced in Section 2.

B Full ML-KEM-EtM performance tables

Table 4 summarizes CPU cycles for primitives of ML-KEM and ML-KEM-EtM.

C Sequence-of-games figures and proof of Theorem 1

Figures 11 and 12 show the sequence of games in the proof of Theorem 1 and OW-PCA adversary B simulates game 3 for IND-CCA adversary A , respectively.

Proof of Theorem 1. *Game 0* is the standard KEM IND-CCA game. The decapsulation oracle $\mathcal{O}^{\text{Decap}}$ 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 $\mathcal{O}^{\text{Decap}}$ is replaced with a simulated oracle $\mathcal{O}_1^{\text{Decap}}$. Instead of directly decrypting c' as in the decapsulation routine,

OW-ATK game
1: $(\mathbf{pk}, \mathbf{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$
2: $m^* \xleftarrow{\$} \mathcal{M}$
3: $c^* \xleftarrow{\$} \text{Enc}(\mathbf{pk}, m)$
4: $\hat{m} \xleftarrow{\$} A^{\text{ATK}}(1^\lambda, \mathbf{pk}, c^*)$
5: return $\llbracket \hat{m} = m^* \rrbracket$

Figure 6: 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: $(\mathbf{pk}, \mathbf{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$
2: $(m_0, m_1) \xleftarrow{\$} A^{\text{ATK}}(1^\lambda, \mathbf{pk})$
3: $b \xleftarrow{\$} \{0, 1\}$
4: $c^* \xleftarrow{\$} \text{Enc}(\mathbf{pk}, m_b)$
5: $\hat{b} \xleftarrow{\$} A^{\text{ATK}}(1^\lambda, \mathbf{pk}, c^*)$
6: return $\llbracket \hat{b} = b \rrbracket$

Figure 7: IND-ATK game: adversary is asked to distinguish the encryption of one message from another

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 $k = G(\text{Dec}(\mathbf{sk}', c'))$. Therefore, there could not have been a matching query that also validates the tag, and the simulated oracle must also reject 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 $k = G(\text{Dec}(\mathbf{sk}', c'))$, but k has never been queried. Under the random oracle model, such k is a uniformly random sample of $\mathcal{K}_{\text{MAC}}$ 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, without having seen any tag under that key. We formalize this with a hybrid argument over the decapsulation queries. The forgery adversary C samples an index $i \xleftarrow{\$} \{1, \dots, q_D\}$, generates a KEM_{ETM} keypair, and runs game 0 for A internally (answering all oracle queries itself, which it can do since it holds the secret key). When A submits its i -th decapsulation query $c = (c', t)$, C outputs (c', t) as its forgery.

Conditioned on the i -th query being the first query on which the two oracles disagree, the key $k = G(\text{Dec}(\mathbf{sk}', c'))$ has never been queried to G , so it is distributed as a fresh uniformly random MAC key, and (c', t) is a valid no-message forgery under that key. Since i is uniform and independent of A 's view:

$$P \left[\exists \text{ query on which } \mathcal{O}^{\text{Decap}} \text{ and } \mathcal{O}_1^{\text{Decap}} \text{ disagree} \right] \leq q_D \cdot \text{Adv}_{\text{MAC}}(C).$$

By the difference lemma:

$\mathcal{O}^{\text{Dec}}(c)$	$\mathcal{O}^{\text{PCO}}(m, c)$
1: return $\text{Dec}(\text{sk}, c)$	1: return $\llbracket \text{Dec}(\text{sk}, c) = m \rrbracket$

Figure 8: Decryption oracle $\mathcal{O}^{\text{Dec}}$ (left) answers decryption queries by returning the decryption of the queried ciphertext. Plaintext-checking oracle $\mathcal{O}^{\text{PCO}}$ (right) answers whether the queried plaintext is the decryption of the queried ciphertext.

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

Figure 9: The IND-ATK game for KEM

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

Note that C requires no signing oracle: one-time (indeed no-message) existential unforgeability of the MAC suffices for this step.

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

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

Game 3 is identical to game 2, except that the challenger samples a uniformly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ 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 $P[\text{QUERY } H]$, then:

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

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: $\text{Adv}_{G_3}(A) = 0$.

We will bound $P[\text{QUERY } G]$ and $P[\text{QUERY } H]$ by constructing an OW-PCA adversary B against the underlying PKE that uses A as a sub-routine. B 's behaviors are summarized in Figure 12.

B simulates game 3 for A : upon 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 $m^* = \text{Dec}(\text{sk}', c'^*)$ has been queried. When m^* is queried, B terminates A and returns m^* to win the OW-PCA game. In other words:

MAC EUF-CMA game

- 1: $k^* \xleftarrow{\$} \mathcal{K}$
 - 2: $(\hat{m}, \hat{t}) \xleftarrow{\$} A^{\text{CMA}}()$
 - 3: **return** $\llbracket \text{Verify}(k^*, \hat{m}, \hat{t}) \wedge (\hat{m}, \hat{t}) \notin \mathcal{O}^{\text{Sign}} \rrbracket$
-

Figure 10: 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

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 4: CPU cycles of each KEM routine

$$P[\text{QUERY } G] \leq \text{Adv}_{\text{PKE}}^{\text{OW-PCA}}(B),$$

$$P[\text{QUERY } H] \leq \text{Adv}_{\text{PKE}}^{\text{OW-PCA}}(B).$$

Combining all equations produces the desired security bound.

Finally, we count the plaintext-checking queries made by B . Each query to the simulated hash oracles $\mathcal{O}_B^G$ and $\mathcal{O}_B^H$ triggers one plaintext-checking query against the challenge ciphertext c'^* , for a total of at most $q_G + q_H$ such queries. Each simulated decapsulation query (c', t) triggers at most q_G plaintext-checking queries, all against the same ciphertext c' , for a total of at most $q_D \cdot q_G$ queries against at most q_D distinct ciphertexts. Therefore, B makes at most $q_G + q_H + q_D \cdot q_G$ plaintext-checking queries, involving the challenge ciphertext and at most q_D distinct other ciphertexts.

D Classic McEliece subroutines

Below we reproduce the McElieceEtM construction (Figure 13) and the Classic McEliece subroutines it uses (Section 4.1).

IND-CCA game for KEM_{EtM}	Decap oracle $\mathcal{O}^{\text{Decap}}(c)$
1: $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}_{\text{EtM}}()$ 2: $m^* \xleftarrow{\$} \mathcal{M}$ 3: $c' \xleftarrow{\$} \text{Enc}_{\text{PKE}}(\text{pk}, m^*)$ 4: $k^* \leftarrow G(m^*)$ $\triangleright$ Game 0-1 5: $k^* \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$ $\triangleright$ Game 2-3 6: $t \leftarrow \text{MAC}(k^*, c')$ 7: $c^* \leftarrow (c', t)$ 8: $K_0 \leftarrow H(m^*, c^*)$ $\triangleright$ Game 0-2 9: $K_0 \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ $\triangleright$ Game 3 10: $K_1 \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ 11: $b \xleftarrow{\$} \{0, 1\}$ 12: $\hat{b} \leftarrow A^{\mathcal{O}^{\text{Decap}}}(\text{pk}, c^*, K_b)$ $\triangleright$ Game 0 13: $\hat{b} \leftarrow A^{\mathcal{O}_1^{\text{Decap}}}(\text{pk}, c^*, K_b)$ $\triangleright$ Game 1-3 14: return $\llbracket \hat{b} = b \rrbracket$	1: $(c', t) \leftarrow c$ 2: $\hat{m} = \text{Dec}_{\text{PKE}}(\text{sk}', c')$ 3: $\hat{k} \leftarrow G(\hat{m})$ 4: if $\text{MAC}(\hat{k}, c') = t$ then 5: $K \leftarrow H(\hat{m}, c)$ 6: else 7: $K \leftarrow H(z, c)$ 8: end if 9: return K
	$\mathcal{O}_1^{\text{Decap}}(c)$ 1: $(c', t) \leftarrow c$ 2: if $\exists (\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \tilde{m} = \text{Dec}_{\text{PKE}}(\text{sk}', c') \wedge \text{MAC}(\tilde{k}, c') = t$ then 3: $K \leftarrow H(\tilde{m}, c)$ 4: else 5: $K \leftarrow H(z, c)$ 6: end if 7: return K
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: $k \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$ 5: $\mathcal{L}^G \leftarrow \mathcal{L}^G \cup \{(m, k)\}$ 6: 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: $K \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ 5: $\mathcal{L}^H \leftarrow \mathcal{L}^H \cup \{(m, c, K)\}$ 6: return K

Figure 11: Sequence of games in the proof of Theorem 1

E Classic McEliece parameter sets and McElieceEtM benchmark tables

Classic McEliece parameter sets have a total of 10 parameter sets. Five of them are presented in Table 5. The other five (called “f-variants”) are identical to the five presented here, except for the internal implementation of the key generation routines. The f-variant key pairs and ciphertexts are fully interoperable with the normal variants, and the encapsulation/decapsulation routines are entirely identical.

NIST evaluates concrete security levels against the difficulty of AES brute-force attack and/or SHA-2 collision attacks [65]. There are five categories of concrete securities (in increasing security levels):

1. Key search on AES-128, 2^{143} classical gates.
2. Finding collision on SHA256/SHA3-256, 2^{146} classical gates.
3. Key search on AES-192, 2^{207} classical gates.
4. Finding collision on SHA384/SHA3-384, 2^{210} classical gates.
5. Key search on AES-256, 2^{272} classical gates.

$B(\text{pk}, c'^*)$	$\mathcal{O}_B^{\text{Decap}}(c)$
1: $z \xleftarrow{\$} \mathcal{M}$ 2: $k \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$ 3: $t \leftarrow \text{MAC}(k, c'^*)$ 4: $c^* \leftarrow (c'^*, t)$ 5: $K \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ 6: $\hat{b} \leftarrow A^{\mathcal{O}_B^{\text{Decap}}, \mathcal{O}_B^G, \mathcal{O}_B^H}(\text{pk}, c^*, K)$ 7: if $\text{ABORT}(m)$ then 8: return m 9: end if	1: $(c', t) \leftarrow c$ 2: if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \mathcal{O}^{\text{PCO}}(\tilde{m}, c') = 1 \wedge \text{MAC}(\tilde{k}, c') = t$ then 3: $K \leftarrow H(\tilde{m}, c)$ 4: else 5: $K \leftarrow H(z, c)$ 6: end if 7: return K
$\mathcal{O}_B^H(m, c)$	$\mathcal{O}_B^G(m)$
if $\mathcal{O}^{\text{PCO}}(m, c'^*) = 1$ then $\text{ABORT}(m)$ 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 $K \xleftarrow{\$} \mathcal{K}_{\text{KEM}}$ $\mathcal{L}^H \leftarrow \mathcal{L}^H \cup \{(m, c, K)\}$ return K	1: if $\mathcal{O}^{\text{PCO}}(m, c'^*) = 1$ then 2: $\text{ABORT}(m)$ 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: $k \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$ 8: $\mathcal{L}^G \leftarrow \mathcal{L}^G \cup \{(m, k)\}$ 9: return k

Figure 12: OW-PCA adversary B simulates game 3 for IND-CCA adversary A in the proof for Theorem 1

Parameter set	public key	secret key	ciphertext	security level [12]
mceliece348864	261120	6492	96	category 1
mceliece460896	524160	13608	156	category 3
mceliece6688128	1044992	13932	208	category 5
mceliece6960119	1047319	13948	194	category 5
mceliece8192128	1357824	14120	208	category 5

Table 5: Public key, secret key, and ciphertext sizes are measured in bytes.

F Key exchange protocols

The CRYSTALS-Kyber team [19] 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 [20], 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

Parameter set	Enc	Dec	Enc + Dec
mceliece348864	215	2471	2686
mceliece348864 + Poly1305	316 (+46.98%)	2074 (-16.07%)	2390 (-11.02%)
mceliece348864 + GMAC	335 (+55.81%)	2087 (-15.54%)	2422 (-9.83%)
mceliece348864 + CMAC	340 (+58.14%)	2092 (-15.34%)	2432 (-9.46%)
mceliece348864 + KMAC256	304 (+41.40%)	2093 (-15.30%)	2397 (-10.76%)

Parameter set	Enc	Dec	Enc + Dec
mceliece460896	487	6694	7181
mceliece460896 + Poly1305	514 (+5.54%)	5784 (-13.59%)	6298 (-12.30%)
mceliece460896 + GMAC	565 (+16.02%)	5809 (-13.22%)	6374 (-11.24%)
mceliece460896 + CMAC	544 (+11.70%)	5905 (-11.79%)	6449 (-10.19%)
mceliece460896 + KMAC256	570 (+17.04%)	5760 (-13.95%)	6330 (-11.85%)

Parameter set	Enc	Dec	Enc + Dec
mceliece6688128	816	7500	8316
mceliece6688128 + Poly1305	889 (+8.95%)	6509 (-13.21%)	7398 (-11.04%)
mceliece6688128 + GMAC	890 (+9.07%)	6521 (-13.05%)	7411 (-10.88%)
mceliece6688128 + CMAC	900 (+10.29%)	6540 (-12.80%)	7440 (-10.53%)
mceliece6688128 + KMAC256	901 (+10.42%)	6546 (-12.72%)	7447 (-10.45%)

Parameter set	Enc	Dec	Enc + Dec
mceliece6960119	699	7262	7961
mceliece6960119 + Poly1305	735 (+5.15%)	6389 (-12.02%)	7124 (-10.51%)
mceliece6960119 + GMAC	753 (+7.73%)	6450 (-11.18%)	7203 (-9.52%)
mceliece6960119 + CMAC	763 (+9.16%)	6428 (-11.48%)	7191 (-9.67%)
mceliece6960119 + KMAC256	765 (+9.44%)	6303 (-13.21%)	7068 (-11.22%)

Parameter set	Enc	Dec	Enc + Dec
mceliece8192128	858	7464	8322
mceliece8192128 + Poly1305	955 (+11.31%)	6547 (-12.29%)	7502 (-9.85%)
mceliece8192128 + GMAC	957 (+11.54%)	6550 (-12.25%)	7507 (-9.79%)
mceliece8192128 + CMAC	945 (+10.14%)	6546 (-12.30%)	7491 (-9.99%)
mceliece8192128 + KMAC256	957 (+11.54%)	6574 (-11.92%)	7531 (-9.50%)

Table 6: McElieceEtM achieves 9-12% speedup in combined “encapsulate + decapsulate” compared to Classic McEliece

Encap _{McElieceEtM} (pk)	Decap _{McElieceEtM} (sk, c)
Require: Public key $\text{pk} = T \in \mathbb{F}_2^{mt \times n}$	Require: Secret key $\text{sk} = (\Gamma, s)$
1: $\mathbf{e} \xleftarrow{\$} \text{FixedWeight}()$	Require: Ciphertext $c = (\mathbf{c}, t)$
2: $k \leftarrow G(\mathbf{e})$	1: $\hat{\mathbf{e}} \leftarrow \text{Decode}(\Gamma, \mathbf{c})$
3: $\mathbf{c} \leftarrow \text{Encode}(T, \mathbf{e})$	2: $\hat{k} \leftarrow G(\hat{\mathbf{e}})$
4: $t \leftarrow \text{MAC}(k, \mathbf{c})$	3: if $\text{MAC}(\hat{k}, \mathbf{c}) \neq t$ then
5: $c \leftarrow (\mathbf{c}, t)$	4: $K \leftarrow H(s, \mathbf{c})$
6: $K \leftarrow H(\mathbf{e}, \mathbf{c})$	5: else
7: return (c, K)	6: $K \leftarrow H(\hat{\mathbf{e}}, \mathbf{c})$
	7: end if
	8: return K

Figure 13: McElieceEtM: applying encrypt-then-MAC to the Niederreiter cryptosystem

Table 7: 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

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 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 7).

Unilaterally authenticated key exchange In unilaterally authenticated key exchange (UAKE), the authenticating party proves its identity to the other party by demonstrating

Table 8: 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

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 $\mathbf{pk}_S$ of the server, and the server authenticates itself by correctly decrypting the challenge encryption sent by the client using its long-term secret key $\mathbf{sk}_S$. 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 8).

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 $\mathbf{pk}_S$ and its own long-term secret key $\mathbf{sk}_C$, while the server possesses client’s long-term public key $\mathbf{pk}_C$ and its own long-term secret key $\mathbf{sk}_S$. 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 9).

Table 9: 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

SeededKeyGen(δ)

Require: l -bit seed δ

- 1: Expand δ to $n + \sigma_2 q + \sigma_1 t + l$ bits, where $\delta_1 = 16, \delta_2 = 32$
 - 2: Denote the last l bits by δ'
 - 3: Denote the first n bits by s
 - 4: Compute $\alpha_0, \dots, \alpha_{q-1}$ from the next $\sigma_2 q$ bits using the **FieldOrdering** algorithm. If this fails, set $\delta \leftarrow \delta'$ and restart
 - 5: Compute g from the next $\delta_1 t$ bits using the **Irreducible** algorithm. If this fails, set $\delta \leftarrow \delta'$ and restart
 - 6: $\Gamma \leftarrow (g, \alpha_0, \alpha_1, \dots, \alpha_{n-1})$
 - 7: Compute $(T, c_{mt-\mu}, \dots, c_{mt-1}, \Gamma') \leftarrow \mathbf{MatGen}(\Gamma)$. If this fails, set $\delta \leftarrow \delta'$ and restart
 - 8: Write Γ' as $(g, \alpha'_0, \alpha'_1, \dots, \alpha'_{n-1})$
 - 9: Set $\alpha = (\alpha'_0, \dots, \alpha'_{n-1}, \alpha_n, \dots, \alpha_{q-1})$ and $c = (c_{mt-\mu}, \dots, c_{mt-1})$
 - 10: Output T as the public key and $(\delta, c, g, \alpha, s)$ as private key
-

Encode($T, \mathbf{e}$)

Require: Public key $T \in \mathbb{F}_2^{mt \times (n-mt)}$

Require: Weight- t vector $\mathbf{e} \in \mathbb{F}_2^n$

- 1: Define $H = (I_{mt} \mid T)$
 - 2: Compute and return $\mathbf{c} \leftarrow H\mathbf{e} \in \mathbb{F}_2^{mt}$
-

Decode($\Gamma, \mathbf{c}$)

Require: Secret key $\Gamma = (g, \alpha_0, \dots, \alpha_{n-1})$

Require: Syndrome $\mathbf{c} \in \mathbb{F}_2^{mt}$

- 1: Extend $\mathbf{c}$ to $\mathbf{v} \in \mathbb{F}_2^n$ by appending 0's
 - 2: Find the unique Goppa codeword $\mathbf{w} \in \mathbb{F}_2^n$ such that $H\mathbf{w} = 0$ and $\mathbf{w}$ has Hamming distance no more than t from $\mathbf{v}$. If there is no such $\mathbf{w}$, return $\perp$
 - 3: **return** $\mathbf{e} \leftarrow \mathbf{v} + \mathbf{w}$
-

Encap(pk)

Require: Public key $\mathbf{pk} = T \in \mathbb{F}_2^{mt \times (n-mt)}$

- 1: Use **FixedWeight** to generate a vector $\mathbf{e} \in \mathbb{F}_2^n$ with Hamming weight t
 - 2: Compute $\mathbf{c} \leftarrow \mathbf{Encode}(T, \mathbf{e})$
 - 3: Compute $K \leftarrow H(1, \mathbf{e}, \mathbf{c})$
 - 4: **return** $(\mathbf{c}, K)$
-

Decap(sk, $\mathbf{c}$)

Require: Secret key $\mathbf{sk}$ contains $s \in \mathbb{F}_2^n$ and $\Gamma = (g, \alpha_0, \alpha_1, \dots, \alpha_{n-1})$

Require: Ciphertext $\mathbf{c} \in \mathbb{F}_2^{mt}$

- 1: Set $b \leftarrow 1$
 - 2: Compute $\mathbf{e} \leftarrow \mathbf{Decode}(\Gamma, \mathbf{c})$
 - 3: **if** $wt(\mathbf{e}) \neq t \vee H\mathbf{e} \neq \mathbf{c}$ **then**
 - 4: Set $\mathbf{e} \leftarrow s, b \leftarrow 0$
 - 5: **end if**
 - 6: **return** $K \leftarrow H(b, \mathbf{e}, \mathbf{c})$
-