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

Ganyu Xu[1], Guang Gong[1], and Kalikinkar Mandal[2]

[1] University of Waterloo, Waterloo, Ontario, Canada {g66xu,ggong}@uwaterloo.ca
[2] University of New Brunswick, Canada kmandal@unb.ca

Abstract. TODO: write abstract later

Keywords: First keyword · Second keyword · Another keyword.

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. 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 [47] and RSA signature ISO 9796 [1], were badly broken with sophisticated cryptanalysis [15,19,21]. 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 [26] into the widely adopted RSA-OAEP. The Fujisaki-Okamoto transformation [25,34] 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)[6,39]. 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) [3,4] 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" [43]. DHIES and its variations appeared in international standards such as IEEE P1363a[2] and ANSI X9.63[5].

1.1 Our contributions

Our contributions are as follows:

Generic CCA secure KEM transformation. We propose the "encrypt-then-MAC" KEM transformation. Our transformation constructs a KEM with provable CCA security under the random oracle model using a public-key encryption scheme (PKE) with one-wayness under plaintext-checking attack and a message authentication code with existential unforgeability. Compared to the Fujisaki-Okamoto transformation, which is widely adopted by many KEM candidates in NIST's Post Quantum Cryptography (PQC) standardization project, our transformation replaces *de-randomization* (which might degrade the security of a randomized cryptosystem) and *re-encryption* (which is computationally inefficient and introduces additional risk of side channels) with computing MAC tag. We also provided concrete cryptanalysis on possible real-world attacks.

Instantiation with McEliece cryptosystem. We applied our KEM transformation to the McEliece cryptosystem instantiated with binary Goppa code and surveyed the landscape of plaintext-checking attacks against Classic McEliece. We implemented this instantiation in C and benchmarked its performance. Compared to the reference implementation of Classic McEliece, our scheme achieves comparable performance.

1.2 Related works

OAEP *Optimal Asymmetric Encryption Padding (OAEP)* [7], proposed by Mihir Bellare and Phillip Rogaway in 1994, was one of the earliest provably secure CCA transformations. However, Victor Shoup identified a non-trivial gap in OAEP's security proof that cannot be filled under ROM[51], although Fujisaki et al. later proved that RSA-OAEP is secure under the RSA assumption [26]. 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.

REACT/GEM Okamoto and Pointchval proposed REACT [44] (Figure 1) in 2001, followed by GEM [20] in 2002. Both are generic CCA transformation with security proved under ROM. Okamoto and Pointcheval first defined the security notion of one-wayness under plaintext checking attack (OW-PCA) and reduced

the CCA security of the transformation to the OW-PCA security of the input public-key cryptosystem.

$\mathtt{Enc}_{\mathrm{REACT}}(\mathtt{pk}, m)$

1: $w \leftarrow \mathcal{M}_{\mathtt{PKE}}$
2: $c_1 \leftarrow \mathtt{Enc}(\mathtt{pk}, w)$
3: $k \leftarrow G(w)$
4: $c_2 \leftarrow \mathcal{E}_k(m)$
5: $c_3 \leftarrow H(w, m, c_1, c_2)$
6: **return** (c_1, c_2, c_3)

$\mathtt{Dec}_{\mathrm{REACT}}(\mathtt{sk}, c)$

Require: $(c_1, c_2, c_3) \leftarrow c$
1: $\hat{w} \leftarrow \mathtt{Dec}(\mathtt{sk}, c_1)$
2: $\hat{k} \leftarrow G(\hat{w})$
3: $\hat{m} \leftarrow \mathcal{D}_{\hat{k}}(c_2)$
4: **if** $H(\hat{w}, \hat{m}, c_1, c_2) = c_3$ **then**
5: **return** $\hat{m}$
6: **else**
7: **return** $\perp$
8: **end if**

Fig. 1: Given PKE $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$, SKE $(\mathcal{E}, \mathcal{D})$, and hash functions G, H, REACT constructs a hybrid PKE $(\mathtt{KeyGen}_{\mathrm{REACT}}, \mathtt{Enc}_{\mathrm{REACT}}, \mathtt{Dec}_{\mathrm{REACT}})$

$\mathtt{Enc}_{\mathrm{GEM}}(\mathtt{pk}, m)$

1: $r \leftarrow \mathcal{R}$
2: $s \leftarrow F(m, r)$
3: $w \leftarrow s \| (r \oplus H(s))$
4: $c_1 \leftarrow \mathtt{Enc}(\mathtt{pk}, w)$
5: $k \leftarrow G(w, c_1)$
6: $c_2 \leftarrow \mathcal{E}_k(m)$
7: **return** (c_1, c_2)

$\mathtt{Dec}_{\mathrm{GEM}}(\mathtt{sk}, c)$

Require: $(c_1, c_2) \leftarrow c$
1: $\hat{w} \leftarrow \mathtt{Dec}(\mathtt{sk}, c_1)$
2: $(\hat{s}, \hat{t}) \leftarrow \hat{w}$
3: $\hat{r} \leftarrow \hat{t} \oplus H(\hat{s})$
4: $\hat{k} \leftarrow G(\hat{w}, c_1)$
5: $\hat{m} \leftarrow \mathcal{D}_{\hat{k}}(c_2)$
6: **if** $F(\hat{m}, \hat{r}) = \hat{s}$ **then**
7: **return** $\hat{m}$
8: **else**
9: **return** $\perp$
10: **end if**

Fig. 2: Given PKE $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$, SKE $(\mathcal{E}, \mathcal{D})$, and hash functions F, G, H, GEM constructs a hybrid PKE $(\mathtt{KeyGen}_{\mathrm{GEM}}, \mathtt{Enc}_{\mathrm{GEM}}, \mathtt{Dec}_{\mathrm{GEM}})$

Fujisaki-Okamoto transformation Fujisaki and Okamoto proposed to construct CCA secure hybrid PKE by combining a OW-CPA secure PKE and a semantically secure symmetric-key encryption (SKE) scheme [25]. The main techniques, namely *de-randomization* and *re-encrytion* were both introduced in the original proposal. Under ROM, 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 [52,50]. Further studies [23,34,13,35,56,38] gave tighter security bounds, accounted for decryption failures in the underlying PKE, and analyzed the security under quantum random oracle model (QROM). To this day, the Fujisaki-Okamoto transformation is the only known generic CCA secure transformation that can convert OW-CPA/IND-CPA PKE into a CCA secure KEM. Because of the minimal input requirement and the simple construction, the Fujisaki-Okamoto transformation was widely adopted among post-quantum KEM candidates submitted to the PQC standardization project, including Kyber [17], Saber [22], FrodoKEM [16], and Classic McEliece [11].

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

- **Computational inefficiency.** In all variants of Fujisaki-Okamoto transformation, decapsulation routine needs to re-encrypt the decryption to ensure ciphertext non-malleability. For input PKE whose encryption routine carries significant computational cost, such as most lattice-based cryptosystems, re-encryption substantially slows down decapsulation.
- **Side-channel vulnerability.** Re-encryption introduces side-channels that can leak information about the decrypted PKE plaintext. As demonstrated in [54,53,36], these side-channels can be converted into efficient plaintext-checking attacks that can fully recover the secret key
- **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 [8,9].

1.3 Paper organization

In Section 2, we review the preliminary definitions and theorems. In Section 3, we present the encrypt-then-MAC KEM transformation, proves its CCA security, and discusses practical attacks. In Section 4, we show that the encrypt-then-MAC transformation is a generalization of DHIES by applying it to the ElGamal cryptosystem. In Section 5, we present McEliece+, an instantiation of the encrypt-then-MAC transformation, benchmark, and compare the performance of McEliece+ with Classic McEliece.

2 Preliminaries

2.1 Public-key encryption scheme

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

Correctness A PKE is δ-correct if

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

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

Security The security of PKE's is conventionally discussed using adversarial games played between a challenger and an adversary [28]. In the OW-ATK game (Figure 3), 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.

OW-ATK game

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

Fig. 3: The one-wayness game: challenger samples a random keypair and a random encryption, and the adversary wins if it correctly produces the decryption

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

IND-ATK game

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

Fig. 4: IND-ATK game: adversary is asked to distinguish the encryption of one message from another

In the IND-ATK game (Figure 4), 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: $\mathrm{Adv}^{\text{IND-ATK}}_{\mathtt{PKE}}(A) = |P[\hat{b} = b] - \frac{1}{2}|$. A PKE is said to be IND-ATK secure if no efficient adversary can win the IND-ATK game with non-negligible advantage.

In public-key cryptography, all adversaries are assumed to have access to the public key (ATK = CPA). If the adversary has access to a decryption oracle $\mathcal{O}^{\mathtt{Dec}} : c \mapsto \mathtt{Dec}(\mathtt{sk}, c)$, it is said to mount chosen-ciphertext attack (ATK = CCA). If the adversary has access to a plaintext-checking oracle (PCO) $\mathcal{O}^{\mathtt{PCO}} : (m, c) \mapsto [\![m = \mathtt{Dec}(\mathtt{sk}, c)]\!]$, then it is said to mount plaintext-checking attack (ATK = PCA).

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

2.2 Key encapsulation mechanism (KEM)

Syntax A key encapsulation mechanism (KEM) is a collection of three routines $(\mathtt{KeyGen}, \mathtt{Encap}, \mathtt{Decap})$ defined over some ciphertext space $\mathcal{C}$ and some key space $\mathcal{K}$. Key generation $\mathtt{KeyGen} : 1^\lambda \mapsto (\mathtt{pk}, \mathtt{sk})$ is a randomized routine that returns a keypair. Encapsulation $\mathtt{Encap} : \mathtt{pk} \mapsto (c, K)$ is a randomized routine that takes a public encapsulation key and returns a pair of ciphertext c and shared secret K (also commonly referred to as session key). Decapsulation $\mathtt{Decap} : (\mathtt{sk}, c) \mapsto K$ is

a deterministic routine that uses the secret key $\mathtt{sk}$ to recover the shared secret K from the input ciphertext c. Where the KEM chooses to reject invalid ciphertext explicitly, the decapsulation routine can also output the rejection symbol $\perp$. We assume a KEM to be perfectly correct:

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

Security Similar to PKE security, the security of KEM is discussed using adversarial games. In the IND-ATK game (Figure 5), the challenger generates a random keypair and encapsulates a random secret; the adversary is given the public key and the ciphertext, then asked to distinguish the shared secret from a random bit string.

KEM IND-ATK Game

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

Fig. 5: The IND-ATK game for KEM

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

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

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

2.3 Message authentication code (MAC)

Syntax A message authentication code (MAC) is a collection of two routines $(\mathtt{Sign}, \mathtt{Verify})$ defined over some key space $\mathcal{K}$, some message space $\mathcal{M}$, and some

tag space $\mathcal{T}$. The signing routine $\texttt{Sign} : (k, m) \mapsto t$ authenticates the message m under the symmetric key k by producing a tag t. The verification routine $\texttt{Verify}(k, m, t)$ 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 $t \leftarrow \texttt{MAC}(k, m)$, and verification done using a simple comparison. Some MAC constructions require a distinct or randomized nonce $r \leftarrow \mathcal{R}$, and the signing routine will take this additional argument $t \leftarrow \texttt{MAC}(k, m; r)$.

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 $\mathcal{O}^{\texttt{Sign}} : m \mapsto \texttt{Sign}(k, m)$ and tries to produce a valid message-tag pair that has not been queried from the signing oracle (Figure 6).

MAC EUF-CMA game

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

Fig. 6: 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

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. Our scheme is inspired by DHIES, but differs from it in two key aspects: whereas DHIES reduces its CCA security specifically to the Gap Diffie-Hellman assumption [43], our construction's CCA security reduces generically to the PCA security of the the input PKE; in addition, we argue that if the PKE's plaintext space is large and the sampling

method has sufficient entropy, then the MAC only needs to be one-time existentially unforgeable (Abdalla, Rogaway, and Bellare originally proposed to use HMAC and CBC-MAC, which are many-time secure MAC but less efficient than one-time MAC). The data flow of the encapsulation is illustrated in Figure 7

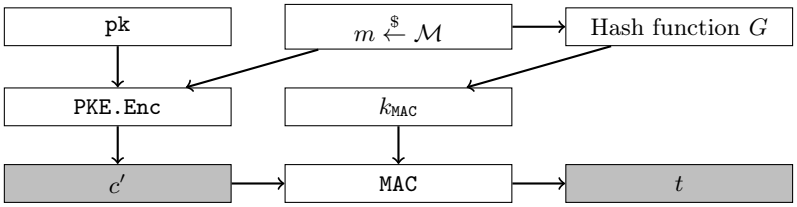

Fig. 7: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

In Section 3.1 we will describe the encrypt-then-MAC KEM routines and state the security reduction. In Section 3.2 we present the proof reducing the IND-CCA security of the KEM tightly to the OW-PCA security of the underlying PKE and non-tightly to the unforgeability of the MAC. In Section 3.3 we discuss some generic attacks on our KEM transformation.

3.1 The construction

Let $\mathcal{B}^*$ denote the set of finite bit strings. Let $\mathcal{K}_{\mathtt{KEM}}$ denote the set of all possible shared secrets. Let $(\mathtt{KeyGen}_{\mathtt{PKE}}, \mathtt{Enc}_{\mathtt{PKE}}, \mathtt{Dec}_{\mathtt{PKE}})$ be a PKE defined over message space $\mathcal{M}_{\mathtt{PKE}}$ and ciphertext space $\mathcal{C}_{\mathtt{PKE}}$. Let $\mathtt{MAC} : \mathcal{K}_{\mathtt{MAC}} \times \mathcal{B}^* \to \mathcal{T}$ be a MAC over key space $\mathcal{K}_{\mathtt{MAC}}$ and tag space $\mathcal{T}$. Let $G : \mathcal{B}^* \to \mathcal{K}_{\mathtt{MAC}}, H : \mathcal{B}^* \to \mathcal{K}_{\mathtt{KEM}}$ be hash functions. The encrypt-then-MAC transformation $\mathtt{EtM}[\mathtt{PKE}, \mathtt{MAC}, G, H]$ constructs a KEM $(\mathtt{KeyGen}_{\mathtt{EtM}}, \mathtt{Encap}_{\mathtt{EtM}}, \mathtt{Decap}_{\mathtt{EtM}})$ (Figure 8).

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

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

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

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

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

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

Fig. 8: The encrypt-then-MAC KEM routines

We chose to construct $\texttt{KEM}_{\texttt{EtM}}$ using implicit rejection $K \leftarrow H(s, c)$: on invalid ciphertexts, the decapsulation routine returns a fake shared secret that depends on the ciphertext and some secret values, though choosing to use explicit rejection should not impact the security of the KEM. In addition, because the underlying PKE can be randomized, the shared secret $K \leftarrow H(m, c)$ must depend on both the plaintext and the ciphertext. According to [23,34], if the input PKE is *rigid* (i.e. $m = \texttt{Dec}(\texttt{sk}, c)$ if and only if $c = \texttt{Enc}(\texttt{pk}, m)$), such as with RSA, then the shared secret may be derived from the plaintext alone $K \leftarrow H(m)$.

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

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

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

3.2 Proof of Theorem 1

We will prove Theorem 1 using a sequence of game. A summary of the the sequence of games can be found in Figure 9 and 10. From a high level we made three incremental modifications to the IND-CCA game for $\texttt{KEM}_{\texttt{EtM}}$:

1. Replace the true decapsulation oracle with a simulated decapsulation oracle
2. Replace the pseudorandom MAC key $k^* \leftarrow G(m^*)$ with a truly random $k^* \xleftarrow{\$} \mathcal{K}_{\texttt{MAC}}$
3. Replace the pseudorandom shared secret $K_0 \leftarrow H(m^*, c)$ with a truly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fig. 9: Sequence of games in the proof of Theorem 1

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

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

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

Game 3 is identical to game 2, except that the challenger samples a uniformly random shared secret $K_0 \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$ instead of deriving it from m^* and t. From A's perspective the two games are indistinguishable, unless A queries H with

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

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

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

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

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

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

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

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

Fig. 10: OW-PCA adversary B simulates game 3 for IND-CCA adversary A in the proof for Theorem 1

$(m^*, \cdot)$. Denote the probability that A queries H with $(m^*, \cdot)$ by $P[\mathtt{QUERY\ H}]$, then:

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

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

We will bound $P[\mathtt{QUERY\ G}]$ and $P[\mathtt{QUERY\ H}]$ by constructing a OW-PCA adversary B against the underlying PKE that uses A as a sub-routine. B's behaviors are summarized in Figure 10.

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

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

Combining all equations above produce the desired security bound.

3.3 Cryptanalysis

OW-PCA security 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 OW-PCA adversary A against the underlying PKE, there exists an IND-CCA adversary B against the encrypt-then-MAC KEM such that:*

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

For a sketch of proof, we observe that the IND-CCA adversary B can perfectly simulate the plaintext-checking oracle for the OW-PCA adversary A (Figure 11), and if A succeeds in breaking the one-wayness of the underlying PKE, then B can compute the shared secret associated with the challenge ciphertext, which allows B to distinguish true shared secret from random bit strings.

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

1: $k \leftarrow G(m)$
2: $t \leftarrow \texttt{MAC}(k, c)$
3: $K \leftarrow H(m, c)$
4: **return** $[\![\mathcal{O}^{\texttt{Decap}}(c, t) = K]\!]$

Fig. 11: Plaintext-checking oracle can be simulated using decapsulation oracle. If m is the decryption of c, then m will hash into the correct MAC key and produce the correct tag, and the decapsulation will accept (c, t) and return the true shared secret. If m is not the decryption of c, then the probability of producing the correct tag is negligible, and the decapsulation will reject (c, t) as invalid.

The security notion of *One-wayness under plaintext-checking attack (OW-PCA)* was introduced by Okamoto and Pointcheval in [44], 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 [20], 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 strictly easier to achieve: any one-way trapdoor permutation (such as RSA) is automatically OW-PCA secure (Figure 12), while there are cryptosystems that are OW-PCA secure but not one-way trapdoor permutation (Table 1). More recently, the modular Fujisaki-Okamoto transformation [34] proposed CCA secure KEM whose security reduces to the OW-PCA security of the input PKE under both ROM and QROM.

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

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

Fig. 12: If $\mathtt{PKE}(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ is an one-way trapdoor permutation, then for all $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}()$, $\mathtt{Dec}(\mathtt{sk}, c) = m$ if and only if $\mathtt{Enc}(\mathtt{pk}, m) = c$

PKE	**is it OW-PCA?**
RSA	Yes
ElGamal	If Gap Diffie-Hellman assumption holds
Kyber/Saber/FrodoKEM/NTRU	No
HQC	No
BIKE (with MDPC)	No
Classic McEliece (binary Goppa code)	unknown, but there is no known attack

Table 1: The landscape of OW-PCA security

There have been numerous attempts at constructing plaintext-checking attacks on post-quantum KEMs submitted to NIST's PQC projects. Due to the search-decision equivalence of the Ring/Module Learning With Error problem, lattice-based cryptosystems including Kyber, Saber, FrodoKEM, and NTRU Prime are all vulnerable to similar key-recovery plaintext-checking attacks (KR-PCA)[29,46,54,53]. KR-PCA against NTRU-HRSS/NTRU-HPS can be adapted from [33,37,24,57]. Among code-based KEMs, HQC has similar structure with LWE-based cryptosystems, so similar KR-PCA applies [36]. BIKE and Classical McEliece are both based on the Niederreiter cryptosystem, although BIKE is instantiated using Moderate-Density Parity Check (MDPC) code, which was also vulnerable to KR-PCA [30]. There are many reaction attacks (also called

ciphertext-validation attack) [32] against the McEliece/Niedereitter cryptosystem using binary Goppa code, where an adversary flips chosen bits of the ciphertext and learns the locations of the error bits by observing whether the modified ciphertext incurs decryption failures. However, there is no known way of converting such reaction attacks to a plaintext-checking attacks. In other words, there no known plaintext-checking attack against the Niedereitter cryptosystem using binary Goppa code.

Dictionary attack on MAC key. For simplicity of proof, we chose to derive the MAC key $k \leftarrow G(m)$ from hashing a randomly sampled plaintext. However, letting the MAC key depend solely on the plaintext opens the possibility of plaintext recovery attack (in fact, deriving the MAC key from plaintext is the source of non-tightness in Theorem 1). An adversary first pre-computes a large lookup table mapping individual plaintexts to the corresponding MAC key, then checks intercepted KEM ciphertext $c = (c', t)$ against the MAC keys stored in this lookup table. This attack could be more efficient than a brute-force search on the plaintext space, since MAC computation is usually more efficient than PKE encryption. This is especially problematic if the PKE is randomized and the MAC key space is smaller than the combination of plaintext space and coin space. We refer reader to [9,8] for detailed discussion of how de-randomization can reduce the search space of a brute-force attack and thus degrade the security of the KEM.

One straightforward fix is to use a MAC with larger key space and derive the MAC key from hashing both the randomly sampled plaintext and the public key. If the PKE public key is large, then it can be pre-hashed during key generation and appended to the KEM keypair. This increases the cost of the plaintext recovery attack since the attacker will need to compute one lookup table per keypair. However, long-term keypairs (e.g. disk encryption or email encryption) might still have sufficiently long lifespan to be vulnerable to dictionary attack. In this case, we propose salting the hash $k \leftarrow G(\mathtt{pk}\|m\|\mathrm{salt})$ with a random salt at each encapsulation, then appending the salt to the ciphertext. At the cost of increasing ciphertext size by the salt size, the cost of such dictionary attack can be raised arbitrarily high.

Forgery attack on MAC tag. With modern MAC constructions such as Poly1305, an adversary has no better method of forging tags under unknown key than blind guesses. Although the security evaluation criteria in NIST's PQC project limits chosen-ciphertext attack adversaries to no more than 2^{64} classical decapsulation queries [41], having a relatively small tag size (e.g. Poly1305, GMAC, CMAC all use 128-bit tag size) might still present practical weakness. Among existing popular MAC constructions, HMAC and KMAC-256 can straightforwardly increase the tag size with appropriate choices of parameters. Scaling CMAC will require a secure pseudorandom function with lengthier output, and although methods of converting n-bit block cipher into mn-bit block cipher exist [31], they remain unproven in real-world usage. Scaling Carter-

Wegman MACs [18,55] (e.g. Poly1305 and GMAC) is relatively straightforward: first the universal hash function needs to operate on a larger finite field, then the pseudorandom function needs to output longer bit strings.

One-time MAC. Under ROM, deriving the MAC key from freshly sampled plaintext means that at each encaspsulation, the MAC key is statistically indistinguishable from uniformly random unless the pre-image is known. In practice, this means that if the implementation has a high-quality source of randomness, the underlying PKE's plaintext space is sufficiently large, and the hash function is collision resistant, then within some reasonable limit for the number of encapsulations under each keypair, the probability of signing distinct ciphertext with identical MAC key is negligible. Furthermore, within the standard IND-CCA security model for KEMs, an adversary does not have access to a signing oracle or other methods of obtaining MAC tag under an unknown symmetric key. Consequently, using a one-time MAC should not affect the security of the transformed KEM. On the other hand, one-time MACs are usually significantly more efficient than many-time MACs, which can lead to meaningful speedup in both encapsulation and decapsulation.

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, otherwise an adversary can find a second ciphertext that decrypts to the same plaintext and obtain the true shared secret from the decapsulation oracle. However, the shared secret does not have to be hashed from the entire ciphertext. We propose that, because the MAC tag is functionally equivalent to a keyed hash of the ciphertext, it is sufficient derive shared secret from the plaintext and the tag $K \leftarrow H(m, t)$. Even though the shared secret does not directly depend on the PKE ciphertext, if an adversary tempered with the challenge ciphertext, it will then need to produce a valid tag under some unknown symmetric key. Assuming that the underlying PKE is OW-PCA secure, the hash function is cryptographically strong, and the MAC is existentially unforgeable, the probability that the adversary can produce valid tag for the tempered challenge ciphertext is negligible. Therefore, deriving shared secret from the tag instead of the entire ciphertext should not impact the CCA security of the KEM. On the other hand, MAC tags are typically much smaller than PKE ciphertexts, so hashing the tag instead of the PKE ciphertext can lead to meaningful speedup, especially in decapsulation. could there be a more rigorous proof?

Side-channel attacks We speculate that the encrypt-then-MAC transformation carries less risk for side-channels than the Fujisaki-Okamoto transformation. This is because Fujisaki-Okamoto transformation re-encrypts the decrypted plaintext in the decapsulation routine, which is more complex than computing a MAC tag. In fact, a number of side-channel attacks [10,14,45] successfully target insecure implementation of the PKE encryption sub-routine. On the other

hand, MAC implementation is usually simpler and thus easier to verify. Furthermore, many MAC constructions such as HMAC and CBC-MAC and benefit from hardware support for SHA-256, AES, or finite field arithmetic, which further decreases the risk of side-channels introduced in software.

4 Application to ElGamal

Applying the encrypt-then-MAC KEM transformation to the ElGamal cryptosystem results in a construction that is highly similar to DHIES [4]. In this section, we will show that the encrypt-then-MAC KEM transformation is a generalization of DHIES by showing that the Gap Diffie-Hellman assumption is a special case of OW-PCA security. Specifically we will sketch a proof of the following Lemma:

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

$$Adv_{GapDH}(B) = Adv^{\texttt{OW-PCA}}_{ElGamal}(A).$$

Each ElGamal cryptosystem [27] is parameterized by a cyclic group $G = \langle g \rangle$ of prime order $q > 2$. A summary of the routine is shown in Figure 13:

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

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

`Dec`$(\mathtt{sk}, c)$
Require: $\mathtt{sk} = x \in \mathbb{Z}_q$
Require: $c = (w, v) \in G \times G$
1: $\hat{m} \leftarrow (w^x)^{-1} \cdot v$
2: **return** $\hat{m}$

Fig. 13: ElGamal cryptosystem over cyclic group $G = \langle g \rangle$ of prime order q

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

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

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

It is also conjectured in [4] (and later extensively studied in [43]) that for certain choice of cyclic group G, the computational Diffie-Hellman problem remains intractable even if the adversary as access to a restricted decisional Diffie-Hellman oracle. This assumption is captured in the Gap Diffie-Hellman problem:

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

We now present the proof for Lemma 2.

Proof. We will prove by a sequence of games. A summary can be found in Figure 14

$G_0 - G_2$

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

$\mathcal{O}^{\mathsf{PCO}}(m, c = (w, v))$

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

$\mathcal{O}_1^{\mathsf{PCO}}(m, c = (w, v))$

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

Fig. 14: The sequence of games in proving Lemma 2

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

Game 1 is identical to game 0, except that the formulation of the $\mathcal{O}^{\mathsf{PCO}}$ is changed. When servicing the plaintext-checking query $(m, c = (w, v))$, PCO_1 checks whether w^x is equal to $m^{-1} \cdot v$. Observe that in the cyclic group G, the algebraic expressions in $\mathcal{O}^{\mathsf{PCO}}$ and $\mathcal{O}_1^{\mathsf{PCO}}$ are equivalent, which means that $\mathcal{O}_1^{\mathsf{PCO}}$ behaves identically to $\mathcal{O}^{\mathsf{PCO}}$.

Game 2 is identical to game 1 except for two modifications: first, when computing the challenge ciphertext, v is no longer computed from m^* but is randomly sampled; second, the win condition changed from $\hat{m} = m^*$ to $\hat{m} = w^{-x} \cdot v$. It is easy to verify that Game 0 through Game 2 are algebraically equivalent:

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

The Gap Diffie-Hellman adversary B can perfectly simulate game 2 for A (see Figure 15): B receives as the Gap Diffie-Hellman problem inputs g^x and g^y. g^x simulates an ElGamal public key, where as g^y simulates the first component of the challenge ciphertext. As in game 2, the second component of the challenge ciphertext can be randomly sampled. Finally, the $\mathcal{O}_1^{\mathtt{PCO}}$ from game 2 can be perfectly simulated using the restricted DDH oracle $\mathcal{O}^{\mathrm{DDH}}$.

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

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

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

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

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

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

Fig. 15: Gap Diffie-Hellman adversary B simulates game 2 for A

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

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

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

Code-based cryptography was first introduced by Robert J. McEliece in 1978 [40]. 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 $\mathtt{pk} = G' \leftarrow PGS$ 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, 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 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, finally the message is recovered. The one-wayness of the McEliece cryptosystem reduces to the NP-hard problem of decoding a random linear code. Harald Niederreiter improved the efficiency of

the McEliece cryptosystem in 1986 [42] 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. Compared to the McEliece formulation, the Niederreiter variant enjoys smaller public key and smaller ciphertexts.

5.1 Classic McEliece

Classic McEliece [12] 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 layer. The first layer is a OW-CPA secure PKE based on the Niederreiter cryptosystem using a random binary Goppa code, and the second layer is a modified Fujisaki-Okamoto 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. Table 2 summarizes the prameter sets.

Parameter set	m	n	t
`mceliece348864/348864f`	12	3488	64
`mceliece460896/460896f`	13	4608	96
`mceliece6688128/6688128f`	13	6688	128
`mceliece6960119/6960119f`	13	6960	119
`mceliece8192128/8192128f`	13	8192	128

Table 2: Classic McEliece parameter sets

Algorithm 1 describes the key generation routine of Classic McEliece. In step 4, `FieldOrdering` takes the input bits and outputs either $\perp$ or a sequence $(\alpha_0, \ldots, \alpha_{q-1})$ 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 [12] for details of these subroutines.

Algorithms 2 and 3 describe the encoding and decoding subroutines of the Niederreiter layer. Note that our description of the decoding routine differs from [12] in that we did not include the weight and/or re-encryption check. Instead, they are included in the description of the KEM subroutines. This is because we later plan to substitute these checks with computing a MAC tag when applying the encrypt-then-MAC KEM transformation.

Algorithms 4 and 5 describe Classic McEliece's KEM construction using the CPA secure Niedereitter cryptosystem descrbied above. In step 1 of Algorithm 4, `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 5) to ensure the integrity of the ciphertext.

Algorithm 1 `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 $\alpha_0, \ldots, \alpha_{q-1}$ 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: $\Gamma \leftarrow (g, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
7: Compute $(T, c_{mt-\mu}, \ldots, c_{mt-1}, \Gamma') \leftarrow$ `MatGen`(Γ). If this fails, set $\delta \leftarrow \delta'$ and restart
8: Write Γ' as $(g, \alpha'_0, \alpha'_1, \ldots, \alpha'_{n-1})$
9: Set $\alpha = (\alpha'_0, \ldots, \alpha'_{n-1}, \alpha_n, \ldots, \alpha_{q-1})$ and $c = (c_{mt-\mu}, \ldots, c_{mt-1})$
10: Output T as the public key and $(\delta, c, g, \alpha, s)$ as private key

Algorithm 2 `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 $\mathbf{c} \leftarrow H\mathbf{e} \in \mathbb{F}_2^{mt}$

Algorithm 3 `Decode`$(\Gamma, \mathbf{c})$

Require: Secret key $\Gamma = (g, \alpha_0, \ldots, \alpha_{n-1})$
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: Set $\mathbf{e} \leftarrow \mathbf{v} + \mathbf{w}$
4: **return** $\mathbf{e}$

Algorithm 4 `Encap`$(\mathtt{pk})$

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

Algorithm 5 `Decap`$(\mathtt{sk}, \mathbf{c})$

Require: Secret key `sk` contains $s \in \mathbb{F}_2^n$ and $\Gamma = (g, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
Require: Ciphertext $\mathbf{c} \in \mathbb{F}_2^{mt}$
1: Set $b \leftarrow 1$
2: Compute $\mathbf{e} \leftarrow$ `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** $K \leftarrow H(b, \mathbf{e}, \mathbf{c})$

We instantiated the encrypt-then-MAC KEM transformation using `SeededKeyGen`, `FixedWeight`, `Encode`, and `Decode` as sub-routines. The result is called McEliece+. 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.

Algorithm 6 $\texttt{Encap}_{\text{McEliece+}}(\texttt{pk})$

Require: Public key $\texttt{pk} = T \in \mathbb{F}_2^{mt \times n}$
1: $\mathbf{e} \xleftarrow{\$} \texttt{FixedWeight}()$
2: $k \leftarrow G(\mathbf{e})$
3: $\mathbf{c} \leftarrow \texttt{Encode}(T, \mathbf{e})$
4: $t \leftarrow \texttt{MAC}(k, c)$
5: $K \leftarrow H(\mathbf{e}, \mathbf{c})$
6: **return** $(\mathbf{c}, K)$

Algorithm 7 $\texttt{Decap}_{\text{McEliece+}}(\texttt{sk}, c)$

Require: Secret key $\texttt{sk} = (\Gamma, s)$
Require: Ciphertext $c = (\mathbf{c}, t)$
1: $\hat{\mathbf{e}} \leftarrow \texttt{Decode}(\Gamma, \mathbf{c})$
2: $\hat{k} \leftarrow G(\hat{\mathbf{e}})$
3: **if** $\texttt{MAC}(\hat{k}, \mathbf{c}) \neq t$ **then**
4: $\quad K \leftarrow H(s, \mathbf{c})$
5: **else**
6: $\quad K \leftarrow H(\mathbf{e}, \mathbf{c})$
7: **end if**
8: **return** K

Fig. 16: McEliece+: applying encrypt-then-MAC to the Niedereitter cryptosystem

5.2 Choosing MAC

For concrete instantiation of McEliece+, we chose four MACs covering a variety of architectures. All MACs are parameterized with a 256-bit key and 128-bit tag, except for KMAC-256, which can have variable key and tag length.

Poly1305 and GMAC are both Carter-Wegman style MACs [18,55], which compute the tag using finite field arithmetic. It first parses the message into a sequence of finite field elements, then evalutes a polynomial whose coefficients are the message blocks and whose indeterminate is the secret key. Specifically, Poly1305 oerates in the prime field $\mathbb{F}_q$ where $q = 2^{130} - 5$. GMAC operates in the binary extension field $\mathbb{F}_{2^{128}}$.

In 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, which includes a nonce and is therefore many-time secure.

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

KMAC is based on the SHA-3 family of sponge functions. We chose KMAC-256, which uses Shake256 as the underlying extendasble output function. Although KMAC generally has worse performance than other MACs, it is the only construction with flexible key and tag length. We fixed the key length at 256 bits, and varied the tag length to 128, 192, and 256 bits depending on the desired security levels.

Other MACs TODO: Kali's part

5.3 Performance benchmark

We implemented McEliece+ by modifying the reference implementation. MAC implementations are taken from OpenSSL. C code is compiled using Apple Clang 15.0.0. Performance measurement is run on Apple Silicon M1 chip. CPU clock is measured using kernel clock `mach_absolute_time`. Each rouine is run 10000 times, with median time reported in Table 3.

6 Conclusion and future works

In this paper we presented "encrypt-then-MAC", a KEM constructed from a PKE and a MAC. We reduced the IND-CCA security of the KEM tightly to the OW-PCA security of the underlying PKE and non-tightly to the security of the underlying MAC. We also analyzed generic attacks on the KEM and proposed countermeasures. We then applie the encrypt-then-MAC transformation to the ElGamal cryptosystem and the McEliece cryptosystem, and implemented the transformed McEliece+ in C using a variety of MACs. On average, McEliece+ achieved 9-12% increase in throughput compared to Classic McEliece, which uses the Fujisaki-Okamoto transformation to convert OW-CPA encryption scheme into a CCA secure KEM.

We speculate that because Shor's algorithm [48,49] is inapplicable to most MAC constructions, the encrypt-then-MAC KEM transformation can be applied to a quantum-resistant PKE scheme to derive a post-quantum CCA-secure KEM scheme. In future works, we plan to analyze the security reduction under the quantum random oracle model, which can inform us of the appropriate parameters for the MAC.

References

1. 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
2. Ieee standard specifications for public-key cryptography - amendment 1: Additional techniques. IEEE Std 1363a-2004 (Amendment to IEEE Std 1363-2000) pp. 1–167 (2004). https://doi.org/10.1109/IEEESTD.2004.94612
3. Abdalla, M., Bellare, M., Rogaway, P.: DHAES: an encryption scheme based on the diffie-hellman problem. IACR Cryptol. ePrint Arch. p. 7 (1999), http://eprint.iacr.org/1999/007
4. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2020, pp. 143–158. Springer (2001). https://doi.org/10.1007/3-540-45353-9_12, https://doi.org/10.1007/3-540-45353-9_12
5. ANSI, X.: 63: Public key cryptography for the financial services industry, key agreement and key transport using elliptic curve cryptography. American National Standards Institute (1998)
6. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000). https://doi.org/10.1007/3-540-44448-3_41, https://doi.org/10.1007/3-540-44448-3_41
7. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Lecture Notes in Computer Science, vol. 950, pp. 92–111. Springer (1994). https://doi.org/10.1007/BFB0053428, https://doi.org/10.1007/BFb0053428
8. Bernstein, D.J.: FO derandomization sometimes damages security. Cryptology ePrint Archive, Paper 2021/912 (2021), https://eprint.iacr.org/2021/912
9. Bernstein, D.J.: On the looseness of FO derandomization. IACR Cryptol. ePrint Arch. p. 912 (2021), https://eprint.iacr.org/2021/912
10. Bernstein, D.J., Bhargavan, K., Bhasin, S., Chattopadhyay, A., Chia, T.K., Kannwischer, M.J., Kiefer, F., Paiva, T., Ravi, P., Tamvada, G.: KyberSlash: Exploiting secret-dependent division timings in kyber implementations. Cryptology ePrint Archive, Paper 2024/1049 (2024), https://eprint.iacr.org/2024/1049
11. Bernstein, D.J., Chou, T., Schwabe, P.: Mcbits: Fast constant-time code-based cryptography. In: Bertoni, G., Coron, J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8086, pp. 250–272. Springer (2013). https://doi.org/10.1007/978-3-642-40349-1_15, https://doi.org/10.1007/978-3-642-40349-1_15
12. Bernstein, D.J., Heninger, N., Lange, T., van Beirendonck, M., et al.: Classic mceliece: Specification (October 2022), https://classic.mceliece.org/mceliece-spec-20221023.pdf, accessed: 2025-01-29

13. Bernstein, D.J., Persichetti, E.: Towards KEM unification. IACR Cryptol. ePrint Arch. p. 526 (2018), https://eprint.iacr.org/2018/526
14. Berzati, A., Viera, A.C., Chartouny, M., Vigilant, D.: Simple power analysis assisted chosen cipher-text attack on ML-KEM. Cryptology ePrint Archive, Paper 2024/2051 (2024), https://eprint.iacr.org/2024/2051
15. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1462, pp. 1–12. Springer (1998). https://doi.org/10.1007/BFB0055716, https://doi.org/10.1007/BFb0055716
16. Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 1006–1018. ACM (2016). https://doi.org/10.1145/2976749.2978425, https://doi.org/10.1145/2976749.2978425
17. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018. pp. 353–367. IEEE (2018). https://doi.org/10.1109/EUROSP.2018.00032, https://doi.org/10.1109/EuroSP.2018.00032
18. Carter, L., Wegman, M.N.: Universal classes of hash functions. J. Comput. Syst. Sci. **18**(2), 143–154 (1979). https://doi.org/10.1016/0022-0000(79)90044-8, https://doi.org/10.1016/0022-0000(79)90044-8
19. Coppersmith, D., Halevi, S., Jutla, C.: Iso 9796-1 and the new forgery strategy. rump session of Crypto **99** (1999)
20. Coron, J., Handschuh, H., Joye, M., Paillier, P., Pointcheval, D., Tymen, C.: GEM: A generic chosen-ciphertext secure encryption method. In: Preneel, B. (ed.) Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2271, pp. 263–276. Springer (2002). https://doi.org/10.1007/3-540-45760-7_18, https://doi.org/10.1007/3-540-45760-7_18
21. Coron, J., Naccache, D., Stern, J.P.: On the security of RSA padding. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 1–18. Springer (1999). https://doi.org/10.1007/3-540-48405-1_1, https://doi.org/10.1007/3-540-48405-1_1
22. D'Anvers, J., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 282–305. Springer (2018). https://doi.org/10.1007/978-3-319-89339-6_16, https://doi.org/10.1007/978-3-319-89339-6_16
23. Dent, A.W.: A designer's guide to kems. In: Paterson, K.G. (ed.) Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-

18, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2898, pp. 133–151. Springer (2003). https://doi.org/10.1007/978-3-540-40974-8_12, https://doi.org/10.1007/978-3-540-40974-8_12
24. Ding, J., Deaton, J., Schmidt, K., Vishakha, Zhang, Z.: A simple and efficient key reuse attack on NTRU cryptosystem. Cryptology ePrint Archive, Paper 2019/1022 (2019), https://eprint.iacr.org/2019/1022
25. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer (1999). https://doi.org/10.1007/3-540-48405-1_34, https://doi.org/10.1007/3-540-48405-1_34
26. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 260–274. Springer (2001). https://doi.org/10.1007/3-540-44647-8_16, https://doi.org/10.1007/3-540-44647-8_16
27. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074, https://doi.org/10.1109/TIT.1985.1057074
28. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA. pp. 365–377. ACM (1982). https://doi.org/10.1145/800070.802212, https://doi.org/10.1145/800070.802212
29. Guo, Q., Johansson, T., Nilsson, A.: A key-recovery timing attack on post-quantum primitives using the fujisaki-okamoto transformation and its application on FrodoKEM. Cryptology ePrint Archive, Paper 2020/743 (2020), https://eprint.iacr.org/2020/743
30. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on MDPC with CCA security using decoding errors. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 789–815 (2016). https://doi.org/10.1007/978-3-662-53887-6_29, https://doi.org/10.1007/978-3-662-53887-6_29
31. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Okamoto, T. (ed.) Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2964, pp. 292–304. Springer (2004). https://doi.org/10.1007/978-3-540-24660-2_23, https://doi.org/10.1007/978-3-540-24660-2_23
32. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystems. In: Varadharajan, V., Mu, Y. (eds.) Information and Communication Security, Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1726, pp. 2–12. Springer (1999). https://doi.org/10.1007/978-3-540-47942-0_2, https://doi.org/10.1007/978-3-540-47942-0_2

33. Hoffstein, J., Silverman, J.H.: Reaction attacks against the ntru public key cryptosystem. Tech. rep., NTRU Technical Report (1999), https://ntru.org/resources.shtml, available at https://ntru.org/resources.shtml
34. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer (2017). https://doi.org/10.1007/978-3-319-70500-2_12, https://doi.org/10.1007/978-3-319-70500-2_12
35. Hövelmanns, K., Hülsing, A., Majenz, C.: Failing gracefully: Decryption failures and the fujisaki-okamoto transform. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13794, pp. 414–443. Springer (2022). https://doi.org/10.1007/978-3-031-22972-5_15, https://doi.org/10.1007/978-3-031-22972-5_15
36. Huguenin-Dumittan, L., Vaudenay, S.: Classical misuse attacks on NIST round 2 PQC - the power of rank-based schemes. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12146, pp. 208–227. Springer (2020). https://doi.org/10.1007/978-3-030-57808-4_11, https://doi.org/10.1007/978-3-030-57808-4_11
37. Jaulmes, É., Joux, A.: A chosen-ciphertext attack against NTRU. In: Bellare, M. (ed.) Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 20–35. Springer (2000). https://doi.org/10.1007/3-540-44598-6_2, https://doi.org/10.1007/3-540-44598-6_2
38. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 96–125. Springer (2018). https://doi.org/10.1007/978-3-319-96878-0_4, https://doi.org/10.1007/978-3-319-96878-0_4
39. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is ssl?). In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 310–331. Springer (2001). https://doi.org/10.1007/3-540-44647-8_19, https://doi.org/10.1007/3-540-44647-8_19
40. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Technical report, NASA (1978), https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF
41. National Institute of Standards and Technology (NIST): Post-quantum cryptography standardization: Security evaluation criteria (nd), https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria), accessed: 2025-01-20

42. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory **15**(2), 159–166 (1986)
43. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings. Lecture Notes in Computer Science, vol. 1992, pp. 104–118. Springer (2001). https://doi.org/10.1007/3-540-44586-2_8, https://doi.org/10.1007/3-540-44586-2_8
44. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2020, pp. 159–175. Springer (2001). https://doi.org/10.1007/3-540-45353-9_13, https://doi.org/10.1007/3-540-45353-9_13
45. Purnal, A.: clangover. https://github.com/antoonpurnal/clangover (2025), accessed: 2025-01-20
46. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on CCA-secure lattice-based PKE and KEM schemes. Cryptology ePrint Archive, Paper 2019/948 (2019), https://eprint.iacr.org/2019/948
47. RSA Data Security, I.: PKCS 1: RSA Encryption Standard Version 1.5. Request for Comments: 2313 (Mar 1998), https://www.rfc-editor.org/rfc/rfc2313
48. Shor, P.W.: 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. pp. 124–134. IEEE Computer Society (1994). https://doi.org/10.1109/SFCS.1994.365700, https://doi.org/10.1109/SFCS.1994.365700
49. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172, https://doi.org/10.1137/S0097539795293172
50. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding. Lecture Notes in Computer Science, vol. 1807, pp. 275–288. Springer (2000). https://doi.org/10.1007/3-540-45539-6_19, https://doi.org/10.1007/3-540-45539-6_19
51. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 239–259. Springer (2001). https://doi.org/10.1007/3-540-44647-8_15, https://doi.org/10.1007/3-540-44647-8_15
52. Shoup, V.: A proposal for an ISO standard for public key encryption. IACR Cryptol. ePrint Arch. p. 112 (2001), http://eprint.iacr.org/2001/112
53. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum kems. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2023**(3), 473–503 (2023). https://doi.org/10.46586/TCHES.V2023.I3.473-503, https://doi.org/10.46586/tches.v2023.i3.473-503
54. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. IACR Trans.

Cryptogr. Hardw. Embed. Syst. **2022**(1), 296–322 (2022). https://doi.org/10.46586/TCHES.V2022.I1.296-322, https://doi.org/10.46586/tches.v2022.i1.296-322

55. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. **22**(3), 265–279 (1981). https://doi.org/10.1016/0022-0000(81)90033-7, https://doi.org/10.1016/0022-0000(81)90033-7
56. Xagawa, K., Yamakawa, T.: (tightly) qcca-secure key-encapsulation mechanism in the quantum random oracle model. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers. Lecture Notes in Computer Science, vol. 11505, pp. 249–268. Springer (2019). https://doi.org/10.1007/978-3-030-25510-7_14, https://doi.org/10.1007/978-3-030-25510-7_14
57. Zhang, X., Cheng, C., Ding, R.: Small leaks sink a great ship: An evaluation of key reuse resilience of PQC third round finalist NTRU-HRSS. Cryptology ePrint Archive, Paper 2021/168 (2021), https://eprint.iacr.org/2021/168

KEM	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%)

KEM	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%)

KEM	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%)

KEM	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%)

KEM	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 3: McEliece+ achieves 9-12% speedup in combined "encapsulate + decapsulate" compared to Classic McEliece