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. The Fujisaki-Okamoto transformation is a generic IND-CCA secure transformation widely adopted by post-quantum KEMs including ML-KEM, FrodoKEM, Classic McEliece, and many more. The FO transform achieves CCA security using de-randomization and re-encryption. However, de-randomization might degrade the security if the input PKE is randomized, and re-encryption incurs significant computational cost in decapsulation. In this paper, we propose a generic IND-CCA secure KEM transformation that achieves chose-ciphertext security by applying the encrypt-then-MAC mechanism to an OW-PCA secure PKE and an one-time existentially unforgeable MAC. Compared to the Fujisaki-Okamoto transformation, the encrypt-then-MAC transformation maintains the randomization of the input PKE and replaces the expensive re-encryption with efficient MAC tag computation. We instantiate our proposed KEM with the McEliece cryptosystem, which we call McEliece+. We then implement McEliece+ with a wide selection of MACs including Poly1305, GMAC, CMAC, and KMAC-256. On average, McEliece+ achieves between 9-12% speed up in combined "encapsulation + decapsulation" compared to Classic McEliece.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Introduction will need to be re-written. I will get back to it after writing section 2, 3, 4, 5

2 Preliminaries

2.1 Public-key encryption scheme

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

Correctness. Following the definition in [6], a $\mathtt{PKE}$ is δ-correct if:

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

Where the expectation is taken with respect to the probability distribution of all possible keypairs $(\mathtt{pk},\mathtt{sk}) \xleftarrow{\$} \mathtt{PKE.KeyGen()}$. For many lattice-based cryptosystems, including ML-KEM, decryption failures could leak information about the secret key, although the probability of a decryption failure is low enough that classical adversaries cannot exploit decryption failure more than they can defeat the underlying lattice problem.

Rigidity.

Security. The security of public-key encryption is conventionally discussed within the context of adversarial games played between a challenger and an adversary [8]. There are two main types of games: i) in the one-wayness (OW-ATK) game, the adversary is given a random encryption, then asked to produce the correct decryption; ii) in the indistinguishability (IND-ATK) game, the adversary is given the encryption of one of two adversary-chosen plaintexts, then asked to decide which of the plaintexts corresponds with the given encryption. Depending on the attack model, the adversary may have access to various oracles. Within the context of public-key cryptography, adversaries are always assumed to have the public key with which they can mount chosen-plaintext attack (CPA). If the adversary has access to a plaintext-checking oracle (PCO) [18] then it can mount plaintext-checking attack (PCA). Where the adversary has access to a decryption oracle, it can mount chosen-ciphertext attacks (CCA).

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

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

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

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

2.2 Key encapsulation mechanism (KEM)

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

Security. The security of a KEM is similarly discussed in adversarial games (Figure 2), although the win conditions differ slightly from the win conditions of a PKE indistinguishability game. In a KEM's indistinguishability game, an adversary is given the public key and a challenge ciphertext, then asked to distinguish a pseudorandom shared secret K_0 associated with the challenge ciphertext from a truly random bit string of equal length.

IND-ATK game

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

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

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

Fig. 2: IND-ATK game for KEM and decapsulation oracle $\mathcal{O}_{\mathtt{Decap}}$

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

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

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

2.3 Message authentication code (MAC)

Syntax. A message authentication code $\texttt{MAC}(\texttt{KeyGen}, \texttt{Sign}, \texttt{Verify})$ is a collection of routines defined over some key space $\mathcal{K}$, some message space $\mathcal{M}$, and some tag space $\mathcal{T}$. The signing routine $\texttt{Sign}(k, m)$ authenticates the message m under the secret key k by producing a tag t (also called digest) (we define the process that generates an authentication tag t over message m a *signing routine* in this paper). The verification routine $\texttt{Verify}(k, m, t)$ takes the triplet of secret key k, message m, and tag t, and outputs `1` if the message-tag pair is valid under the secret key, or `0` otherwise. Many MAC constructions are deterministic. For these constructions it is simpler to denote the signing routine by $t \leftarrow \texttt{MAC}(k, m)$ and perform verification using a simple comparison.

Security. The security of a MAC is defined in an adversarial game in which an adversary, with access to a MAC oracle that can answer signing queries $\texttt{MAC}(k, m) \leftarrow \mathcal{O}_{\texttt{MAC}}(m)$, tries to forge a new valid message-tag pair that has never been queried before. The ability to access a MAC oracle is called *chosen-message attack (CMA)*. The ability to produce a valid tag on some arbitrary message is called *existential forgery*. The existential unforgeability under chosen message attack (EUF-CMA) game is shown below:

`EUF-CMA` game
1: $k^* \xleftarrow{\$} \mathcal{K}$
2: $(\hat{m}, \hat{t}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\texttt{MAC}}}()$
3: **return** $[\![\texttt{Verify}(k^*, \hat{m}, \hat{t}) \wedge (\hat{m}, \hat{t}) \notin \mathcal{O}_{\texttt{MAC}}]\!]$

MAC oracle $\mathcal{O}_{\texttt{MAC}}(m)$
1: **return** $\texttt{MAC}(k^*, m)$

Fig. 3: The existential forgery game and the MAC oracle

The advantage $\texttt{Adv}_{\texttt{EUF-CMA}}$ of the existential forgery adversary is the probability that it wins the EUF-CMA game. Some MACs are one-time existentially unforgeable, meaning that each secret key can be used to authenticate only a

single message. The corresponding security game is modified such that the MAC oracle will only answer a single signing query.

3 The encrypt-then-MAC transformation

Our technique. We introduce our encrypt-then-MAC transformation that transforms a OW-PCA secure PKE and an one-time existentially unforgeable MAC into an IND-CCA secure KEM. Our scheme mainly differs from DHIES in its versatility and input requirement. Whereas the IND-CCA security of DHIES reduces specifically to the Gap Diffie-Hellman assumption, the chosen-ciphertext security of the encrypt-then-MAC KEM reduces more generally to the OW-PCA security [18] of the input scheme. In addition, we propose that because each call to encapsulation samples a fresh PKE plaintext, the encrypt-then-MAC KEM can be instantiated with one-time secure MAC such as Poly1305 for further performance improvements (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, see Section **??**). The encapsulation data flow is illustrated in Figure 4.

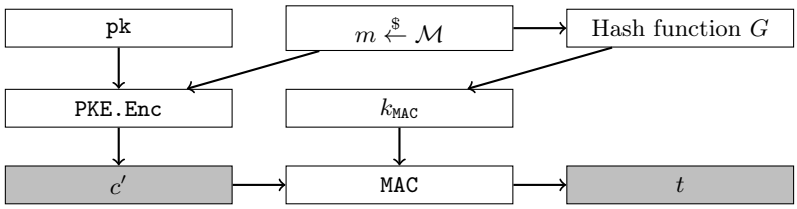

Fig. 4: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

In Section 3.2 we reduce the IND-CCA security of the KEM tightly to the OW-PCA security of the underlying PKE, and non-tightly to the unforgeability of the MAC. In Section **??**, we show that DHIES is a special case of the encrypt-then-MAC transformation by reducing the OW-PCA security of the ElGamal cryptosystem to the Gap Diffie-Hellman assumption.

3.1 The generic KEM construction

Let $\mathcal{B}^*$ denote the set of finite bit strings. Let $\mathtt{PKE}(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be a public-key encryption scheme defined over message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\mathtt{MAC} : \mathcal{K}_{\mathtt{MAC}} \times \mathcal{B}^* \to \mathcal{T}$ be a deterministic message authentication code that takes a key $k \in \mathcal{K}_{\mathtt{MAC}}$, some message $m \in \mathcal{B}^*$, and outputs a tag $t \in \mathcal{T}$. Let $G : \mathcal{M} \to \mathcal{K}_{\mathtt{MAC}}$ be a hash function that maps from PKE's plaintext space to MAC's key space. Let

$H : \mathcal{B}^* \to \mathcal{K}_{\texttt{KEM}}$ be a hash function that maps bit strings into the set of possible shared secrets. The encrypt-then-MAC transformation $\texttt{EtM}[\texttt{PKE}, \texttt{MAC}, G, H]$ constructs a key encapsulation mechanism $\texttt{KEM}_{\texttt{EtM}}(\texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$, whose routines are described in Figure 5.

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

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

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

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

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

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

Fig. 5: The encrypt-then-MAC transformation builds a KEM, denoted by $\texttt{KEM}_{\texttt{EtM}}$, using a $\texttt{PKE}(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$, a MAC, and two hash functions G, H

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

Security analysis. The CCA security of the encrypt-then-MAC scheme can be intuitively argued through an adversary's inability to learn additional information from the decapsulation oracle. For an adversary A to produce a valid tag t for some unauthenticated ciphertext c' under the symmetric key $k \leftarrow G(\texttt{Dec}(\texttt{sk}', c'))$ implies that A must either know the symmetric key k or produce a forgery. Under the random oracle model, A also cannot know k without knowing its pre-image $\texttt{Dec}(\texttt{sk}', c')$, so A must either have produced c' honestly, or have broken the one-

way security of PKE. This means that the decapsulation oracle will not give out information on decryption that the adversary does not already know.

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

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

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

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

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 6 and 7. From a high level we made three incremental modifications to the IND-CCA game for $\mathtt{KEM}_{\mathtt{EtM}}$:

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

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

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

$(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KEM}_{\mathtt{EtM}}.\mathtt{KeyGen}()$
$m^* \xleftarrow{\$} \mathcal{M}$
$c' \xleftarrow{\$} \mathtt{PKE.Enc}(\mathtt{pk}, m^*)$
$k^* \leftarrow G(m^*)$ ▷ 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 $\mathcal{O}^{\mathtt{Decap}}(c)$

$(c', t) \leftarrow c$
$\hat{m} = \mathtt{Dec}(\mathtt{sk}', c')$
$\hat{k} \leftarrow G(\hat{m})$
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 $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G : \tilde{m} = \mathtt{Dec}(\mathtt{sk}', c') \wedge \mathtt{MAC}(\tilde{k}, c') = t$ **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. 6: Sequence of games in the proof of Theorem 1

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

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

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

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

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

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

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

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

Fig. 7: OW-PCA adversary B simulates game 3 for IND-CCA adversary A in the proof for Theorem 1

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

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

If the simulated oracle accepts the queried ciphertext as valid, then there is a matching query that also validates the tag, which means that the queried ciphertext is honestly generated. Therefore, the true oracle must also accept 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 $k = G(\mathtt{Dec}(\mathtt{sk}', c'))$.

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 $k = G(\mathtt{Dec}(\mathtt{sk}', c'))$, but k has never been queried. Under the random oracle model, such k is a uniformly random sample of $\mathcal{K}_{\mathtt{MAC}}$ that the adversary does not know, so for A to produce a valid tag is to produce a forgery against the MAC under an unknown and uniformly random key. Furthermore, the security game does not include a MAC oracle, so this is a zero-time forgery. While zero-time forgery is not a standard security definition for a MAC, we can bound it by the advantage of a one-time forgery adversary C:

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

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

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

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 $(m^*, \cdot)$. 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 7.

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

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

Combining all equations above produce the desired security bound.

3.3 Concrete cryptanalysis

PCA security. The security notion of *One-wayness under plaintext-checking attack (OW-PCA)* was introduced by Okamoto and Pointcheval in [18], 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 [4], 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 [2]. 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, and the ElGamal cryptosystem is OW-PCA secure under the Gap Diffie-Hellman assumption [17]. More recently, the modular Fujisaki-Okamoto transformation [14] proposed CCA secure KEM whose security reduces to the OW-PCA security of the input PKE under both ROM and QROM.

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)[9,19,21,20]. KR-PCA against NTRU-HRSS/NTRU-HPS can be adapted from [13,16,5,22]. Among code-based KEMs, HQC has similar structure with LWE-based cryptosystems, so similar KR-PCA applies [15]. 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 [10]. There are many reaction attacks (also called ciphertext-validation attack) [12], 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. In Section 3, we chose to derive the MAC key $k \leftarrow G(m)$ from hashing a randomly sampled plaintext. While this simplifies the security reduction, letting the MAC key depend solely on the plaintext opens the possibility of plaintext recovery attack: 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. Furthermore, if the input PKE is randomized, then the deterministic hashing of MAC key "de-randomizes" the encryption, which might further undermine security [3].

One straightforward fix is to derive the MAC key from hashing both the randomly sampled plaintext and the public key (large public key can be pre-hashed during key generation and added to the 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 used in KEM-DEM combination for bulk encryption (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 $k \leftarrow G(\mathtt{pk}\|m\|\text{salt})$, then appending the salt to the ciphertext. At the cost of increasing ciphertext size by the salt size, the cost of dictionary attack can be raised further.

Forgery attack on MAC tag. With modern MAC constructions such as Carter-Wegman MAC, CMAC, and HMAC, an adversary has no better forgery method than brute-force all possible tags. Although the security evaluation criteria in NIST's PQC project limits chosen-ciphertext attack adversaries to no more than 2^{64} classical decapsulation queries, having a relatively small tag size (e.g. Poly1305, GMAC, CMAC all use 128-bit tag size) might still present cryptographic 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 [11], they remain unproven in real-world usage. This is also the reason why scaling Carter-Wegman MAC $\mathtt{MAC}(k = (k_1, k_2), m) = H_{\text{xpoly}}(k_1, m) \oplus \text{PRF}(k_2, r)$ is also not straightforward. However, notice that honest parties executing the encapsulation routine samples fresh plaintext per encapsulation; assuming sufficiently high-quality source of randomness and cryptographically strong hash functions, there should be no repetition of plaintext, thus no repetition of MAC keys. The CCA security game does not include a signing oracle, either. In other words, we can safely assume that each MAC key will only be used once, meaning that we can use one-time MAC such as keyed polynomial hash functions (i.e. Carter-Wegman MAC without PRF). In addition to being computationally more efficient, polynomial hash function is also easy to scale up in security by switching to a larger finite field.

4 Application to ElGamal cryptosystem

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

Lemma 1. *For every OW-PCA adversary A against the ElGamal cryptosystem, there exists a Gap Diffie-Hellman problem solver B such that:*

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

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

Each ElGamal cryptosystem [7] is parameterized by a cyclic group $G = \langle g \rangle$ of prime order $q > 2$. A summary of the routine is shown in Figure 8:

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

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

Fig. 8: ElGamal cryptosystem

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 [1] (and later extensively studied in [17]) 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 1.

$G_0 - G_2$

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

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

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

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

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

Fig. 9: The sequence of games in proving Lemma 1

Proof. We will prove by a sequence of games. A summary can be found in Figure 9

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

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

Game 2 is identical to game 1 except for two modifications: first, when computing the challenge ciphertext, v is no longer computed from m^* but is randomly sampled; second, the win condition changed from $\hat{m} = m^*$ to $\hat{m} = w^{-x} \cdot v$. 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 10): B receives as the Gap Diffie-Hellman problem inputs g^x and g^y. g^x simulates an ElGamal public key, where as g^y simulates the first component of the challenge ciphertext. As in game 2, the second component of the challenge ciphertext can be randomly sampled. Finally, the $\mathtt{PCO}_1$ from game 2 can be perfectly simulated using the restricted DDH oracle $\mathcal{O}^{\mathrm{DDH}}$.

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

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

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

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

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

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

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

Fig. 10: Gap Diffie-Hellman adversary B simulates game 2 for A

5 Application to Classic McEliece

Classic McEliece is an IND-CCA secure post-quantum KEM submitted to the Post Quantum Cryptoraphy (PQC) standardization project and is currently one of three viable fourth round KEM candidates. Classic McEliece is based on the Niederreiter variant of the McEliece cryptosystem using binary Goppa code, originally proposed by Robert McEliece in 1978 and later improved by Harald Niederreiter in 1986. There are two layers in the construction of the Classic McEliece KEM. The first layer is an OW-CPA secure PKE whose one-wayness reduces to the intractability of the Syndrome Decoding Problem (SDP) and the indistinguishability of random binary Goppa code from random linear code. The second layer is a modified Fujisaki-Okamoto transformation for converting the passively secure PKE into an actively secure KEM.

Each instance of Classic McEliece KEM is parameterized by the base field size m (which induces a binary extension field $\mathbb{F}_{2^m}$), the codeword length n, and the weight t of error vector. Let $\mathcal{G}_{m,t} = \{g(x) \in \mathbb{F}_{2^m}[x] \mid \deg(g) = t, g \text{ is irreducible}\}$ denote the set of degree-t irreducible polynomials in $\mathbb{F}_{2^m}$. Given a Goppa polynomial $g \in \mathcal{G}_{m,t}$ and n distinct elements $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_{2^m}$, the canonical parity check matrix $H \in \mathbb{F}_{2^m}^{t \times n}$ is given by $H_{j,i} = \alpha_i^{j-1}/g(\alpha_i)$ for $1 \leq i \leq n, 1 \leq j \leq t$.

Key generation in Classic McEliece involves sampling a random binary Goppa code and computing the canonical parity check matrix. The decoding algorithm, parameterized by the Goppa polynomial $g(x)$ and the support $L = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ with secret permutation, is the secret key. The canonical parity check matrix H is row-reduced, and if the output $H' \leftarrow SH$ (S is some invertible matrix encoding the row reduction) has systematic form $H' = [I_t \mid T]$ for some identity matrix $I_t \in \mathbb{F}_{2^m}^{t \times t}$ and $T \in \mathbb{F}_{2^m}^{t \times (n-t)}$, then T is returned as the public key. However, if the canonical parity check matrix cannot be reduced to systematic form, then the entire key generation is restarted. According to (TODO: citation needed), such randomly sampled H has an estimated 30% probability of being reduced to sysmatic form, meaning that key generation usually needs to run multiple times before succeeding. Some modifications introduced in the

f-variants of Classic McEliece improves the estimated probability of success and consequently speeds up key generation, but they are not the focus of this work and will not be discussed in details.

Let $\mathcal{S}_t = \{\mathbf{e} \in \mathbb{F}_2^n \mid wt(\mathbf{e}) = t\}$ denote the Hamming sphere of weight t. The message space of CMPKE is exactly $\mathcal{S}_t$. A plaintext $\mathbf{e} \in \mathcal{S}_t$ is encrypted by interpreting it as a noisy codeword and computing its syndrome under the row-reduced parity check matrix. At decryption, the ciphertext is first zero-padded to transform it into a noisy codeword (this works because the parity-check matrix is in systematic form), then feeding the noisy codeword into the secret Goppa decoder, which identifies the locations of the errors. Because binary Goppa codesare in $\mathbb{F}_2^n$, knowing the error locations is sufficient for recovering the error vector $\mathbf{e} \in \mathcal{S}_t$. Decryption is perfectly correct. Figure 11 and 12 breaks down the two layers of Classic McEliece KEM.

$\texttt{CMKeyGen}()$
1: $g \leftarrow \mathcal{G}$
2: $L \leftarrow \mathbb{F}_{2^m}^n$
3: $H \leftarrow \texttt{Parity}(L, g)$
4: $H' \leftarrow \texttt{RowReduce}(H)$
5: **if** $H' = [I_t \mid T]$ **then**
6: $\quad \texttt{pk} \leftarrow T$
7: $\quad \texttt{sk} \leftarrow (g, L)$
8: $\quad$ **return** $(\texttt{pk}, \texttt{sk})$
9: **end if**
10: Go to line 1

$\texttt{CMEnc}(\texttt{pk}, \mathbf{e})$
Require: $\mathbf{e} \in \mathcal{S}_t$
Require: $\texttt{pk} \in \mathbb{F}_{2^m}^{t \times (n-t)}$
1: $T \leftarrow \texttt{pk}$
2: $c \leftarrow [I_m \mid T] \cdot \mathbf{e}$
3: **return** c

$\texttt{CMDec}(\texttt{sk}, \mathbf{c})$
Require: $\mathbf{c} \in \mathbb{F}_{2^m}^t$
Require: $\texttt{sk} \in \mathbf{F}_{2^m}^n \times \mathcal{G}_{m,t}$
1: $\mathbf{c} \leftarrow [\mathbf{c} \mid \mathbf{0}^{n-t}]$
2: $\mathbf{e} \leftarrow \texttt{GoppaDecode}(\texttt{sk}, \mathbf{c})$
3: **return** $\mathbf{e}$

Fig. 11: Classic McEliece uses the Niederreiter variant of the McEliece cryptosystem. When instantiated with binary Goppa code, this PKE achieves OW-CPA security

We apply the encrypt-then-MAC KEM transformation to the the OW-CPA subroutines ($\texttt{CMKeyGen}, \texttt{CMEnc}, \texttt{CMDec}$) and call it `mceliece+` (Figure 13).

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 1.

CMKEM.keygen()

$(\mathtt{pk}, \mathtt{sk}_{\mathtt{PKE}}) \leftarrow \mathtt{CMKeyGen}()$
$\mathbf{s} \leftarrow \mathbb{F}_2^n$
$\mathtt{sk} \leftarrow (\mathtt{sk}_{\mathtt{PKE}}, \mathbf{s})$

CMKEM.enc(pk)

$\mathbf{e} \leftarrow \mathcal{S}_t$
$\mathbf{c} \leftarrow \mathtt{CMEnc}(\mathtt{pk}, \mathbf{e})$
$K \leftarrow H(1, \mathbf{e}, \mathbf{c})$
return $\mathbf{c}, K$

CMKEM.dec(sk, c)

$(\mathtt{sk}_{\mathtt{PKE}}, \mathbf{s}) \leftarrow \mathtt{sk}$
$\hat{\mathbf{e}} \leftarrow \mathtt{CMDec}(\mathtt{sk}, \mathbf{c})$
if $\mathtt{Synd}(\mathtt{sk}, \hat{\mathbf{e}}) = \mathbf{c}$ **then**
    return $H(1, \hat{\mathbf{e}}, \mathbf{c})$
else
    return $H(0, \mathbf{s}, \mathbf{c})$
end if

Fig. 12: Classic McEliece applies a modified Fujisaki-Okamoto transformation to achieve CCA security

mceliece+.keygen()

$(\mathtt{pk}, \mathtt{sk}_{\mathtt{PKE}}) \leftarrow \mathtt{CMKeyGen}()$
$h \leftarrow H(\mathtt{pk})$
$s \leftarrow \mathbb{F}_2^n$
$\mathtt{sk} \leftarrow (\mathtt{sk}_{\mathtt{PKE}}, s, h)$
return $(\mathtt{pk}, \mathtt{sk})$

mceliece+.enc(pk)

$\mathbf{e} \leftarrow \mathcal{S}_t$
$k \leftarrow G(H(\mathtt{pk}), \mathbf{e})$
$c' \leftarrow \mathtt{CMEnc}(\mathtt{pk}, \mathbf{e})$
$t \leftarrow \mathtt{MAC}(k, c')$
$K \leftarrow \mathtt{KDF}(\mathbf{e}, t)$
$c \leftarrow (c', t)$
return (c, t)

mceliece+.dec(sk, c)

$(c', t) \leftarrow c$
$(\mathtt{sk}_{\mathtt{PKE}}, s, h) \leftarrow \mathtt{sk}$
$\hat{\mathbf{e}} \leftarrow \mathtt{CMDec}(\mathtt{sk}_{\mathtt{PKE}}, c')$
$\hat{k} \leftarrow G(h, \hat{\mathbf{e}})$
if $\mathtt{MAC}(\hat{k}, c') = t$ **then**
    $K \leftarrow \mathtt{KDF}(\hat{\mathbf{e}}, t)$
else
    $K \leftarrow \mathtt{KDF}(\mathbf{s}, \hat{c})$
end if
return K

Fig. 13: `mceliece+`: applying encrypt-then-MAC to the Niedereitter/McEliece cryptosystem

6 Conclusion

References

1. 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
2. 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

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

3. Bernstein, D.J.: FO derandomization sometimes damages security. Cryptology ePrint Archive, Paper 2021/912 (2021), https://eprint.iacr.org/2021/912
4. 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
5. 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
6. Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology -

EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3027, pp. 342–360. Springer (2004). https://doi.org/10.1007/978-3-540-24676-3_21, https://doi.org/10.1007/978-3-540-24676-3_21

7. 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
8. 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
9. 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. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12171, pp. 359–386. Springer (2020). https://doi.org/10.1007/978-3-030-56880-1_13, https://doi.org/10.1007/978-3-030-56880-1_13
10. 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
11. 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
12. 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
13. 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
14. 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

15. 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
16. 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
17. 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
18. 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
19. Ravi, P., Roy, S.S., Chattopadhyay, A., Bhasin, S.: Generic side-channel attacks on cca-secure lattice-based PKE and kems. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 307–335 (2020). https://doi.org/10.13154/TCHES.V2020.I3.307-335, https://doi.org/10.13154/tches.v2020.i3.307-335
20. 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
21. 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
22. 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