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. Abstract needs to be rewritten because ML-KEM is not a suitable candidate. Instead, section 3 will be about the proof, section 4 will be a survey of existing public-key encryption schemes and a discussion of which ones are suitable, and section 5 will be about implementation and performance analysis with 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 $\texttt{sk}$ 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 [3], a $\texttt{PKE}$ is δ-correct if:

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

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

rigidity. **Security.** The security of public-key encryption is conventionally discussed within the context of adversarial games played between a challenger and an adversary [5]. 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) [9] then it can mount plaintext-checking attack (PCA). Where the adversary has access to a decryption oracle, it can mount chosen-ciphertext attacks (CCA).

$\texttt{OW-ATK}$ Game

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

$\texttt{IND-ATK}$ Game

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

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

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

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

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

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

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

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

2.2 Key encapsulation mechanism (KEM)

Syntax. A key encapsulation mechanism $\texttt{KEM}(\texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$ 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 $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$. $\texttt{Encap}(\texttt{pk})$ is a probabilistic routine that takes a public key $\texttt{pk}$ 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). $\texttt{Decap}(\texttt{sk}, c)$ is a deterministic routine that takes the secret key $\texttt{sk}$ 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 $\texttt{Decap}$ will return a rejection symbol $\perp$; other KEM constructions use implicit rejection, where if c is invalid then $\texttt{Decap}$ 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.

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

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

The decapsulation oracle $\mathcal{O}^{\texttt{Decap}}$ takes a ciphertext c and returns the output of the $\texttt{Decap}$ 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 $\texttt{IND-ATK}$ 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** $[\![\texttt{Verify}(k^*, \hat{m}, \hat{t}) \land (\hat{m}, \hat{t}) \notin \mathcal{O}_{\texttt{MAC}}]\!]$

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 [9] 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.

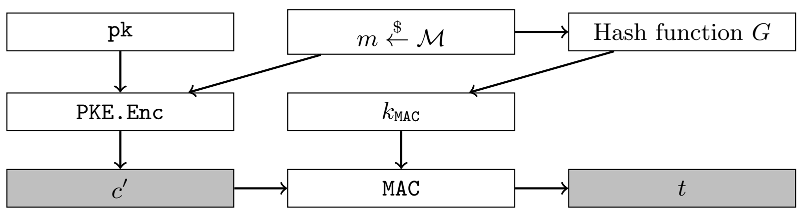

Fig. 4: Combining PKE with MAC using encrypt-then-MAC to ensure ciphertext integrity

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

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

$(\texttt{pk}, \texttt{sk}') \xleftarrow{\$} \texttt{PKE.KeyGen}()$
$z \xleftarrow{\$} \mathcal{M}$
$\texttt{sk} \leftarrow (\texttt{sk}', z)$
return $(\texttt{pk}, \texttt{sk})$

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

$m \xleftarrow{\$} \mathcal{M}$
$k \leftarrow G(m)$
$c' \xleftarrow{\$} \texttt{PKE.Enc}(\texttt{pk}, m)$
$t \leftarrow \texttt{MAC}(k, c')$
$c \leftarrow (c', t)$
$K \leftarrow H(m, c)$
return (c, K)

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

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

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 $\texttt{MAC}$, 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 $\texttt{PKE}$. 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 $c' \xleftarrow{\$} \texttt{Enc}(\texttt{pk}, m)$ can be correctly signed, while ciphertexts that don't decrypt back to m cannot be correctly signed. This means that a decapsulation oracle can be converted into a plaintext-checking oracle, so every chosen-ciphertext attack against the KEM can be converted into a plaintext-checking attack against the underlying PKE.

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

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

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

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 $\texttt{KEM}_{\texttt{EtM}}$:

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.

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

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

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

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

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

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

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

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

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

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

Fig. 6: Sequence of games in the proof of Theorem 1

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

$z \xleftarrow{\$} \mathcal{M}$
$k \xleftarrow{\$} \mathcal{K}_{\mathtt{MAC}}$
$t \leftarrow \mathtt{MAC}(k, c'^*)$
$c^* \leftarrow (c'^*, t)$
$K \xleftarrow{\$} \mathcal{K}_{\mathtt{KEM}}$
$\hat{b} \leftarrow A^{\mathcal{O}_B^{\mathtt{Decap}}, \mathcal{O}_B^G, \mathcal{O}_B^H}(\mathtt{pk}, c^*, K)$
if $\mathtt{ABORT}(m)$ **then**
  return m
end if

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

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

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

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

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

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

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 $k = G(\texttt{Dec}(\texttt{sk}', c'))$, but k has never been queried. Under the random oracle model, such k is a uniformly random sample of $\mathcal{K}_{\texttt{MAC}}$ that the adversary does not know, so for A to produce a valid tag is to produce a forgery against the `MAC` under an unknown and uniformly random key. Furthermore, the security game does not include a 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:

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

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

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

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

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

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

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

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

We will bound $P[\texttt{QUERY G}]$ and $P[\texttt{QUERY H}]$ by constructing a OW-PCA adversary B against the underlying PKE that uses A as a sub-routine. B's behaviors are summarized in Figure 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 $m^* = \texttt{Dec}(\texttt{sk}', c'^\star)$ has been queried. When m^* is queried, B terminates A and returns m^* to win the OW-PCA game. In other words:

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

Combining all equations above produce the desired security bound.

4 Applications

In this section we will survey a number of existing public-key encryption schemes and discuss the applicability of the encrypt-then-MAC transformation to these schemes.

4.1 RSA

RSA [10] is a public-key cryptosystem proposed by Rivest, Shamir, and Adleman in 1978. A summary of its subroutines can be found in Figure 8.

RSA `KeyGen`	RSA `Enc`	RSA `Dec`

Fig. 8: Textbook RSA

In its original formulation (typically referred to as "textbook RSA"), the RSA cryptosystem is one-way secure against chosen-plaintext attack (OW-CPA) under the conjectured intractability of the RSA problem [10] (Definition 1). Furthermore, it is easy to show that the RSA cryptosystem is a trapdoor permutation on the multiplicative group $\mathbb{Z}_N^*$. In other words, under a given keypair $\texttt{pk} = (N, e), \texttt{sk} = d$, $m^e = c \mod N$ if and only if $c^d \mod N$. Consequently, textbook RSA is one-way secure against plaintext-checking attack, because the plaintext-checking oracle can be perfectly simulated using only the public key, which means that access to a plaintext-checking oracle offers no additional advantage.

Definition 1 (RSA problem). *Let $N = pq$ be the product of two prime numbers. Let e be an integer that is relatively prime to $\phi(N) = (p-1)(q-1)$, and let $d = e^{-1} \mod \phi(N)$. Let m be a uniformly random sample in $\mathbb{Z}_N^*$ and let $c = m^e \mod N$. Given N, e, c, find m.*

Although textbook RSA is OW-PCA secure and thus a suitable candidate for the encrypt-then-MAC transformation, there exists simpler and faster CCA secure KEM transformation. For example, one can apply the $U_m^{\not\perp}$ variant of the modular Fujisaki-Okamoto transformation, which encapsulates the shared secret by directly hashing a randomly sampled plaintext. The resulting KEM

(Figure 9) is largely identical to the RSA-KEM proposed by Victor Shoup in [11]. Compare to the encrypt-then-MAC transformation, RSA-KEM offers identically tight security bound while it removes the need for deriving the MAC key and computing the MAC tag.

RSA-KEM KeyGen	RSA-KEM Enc	RSA-KEM Dec

Fig. 9: RSA-KEM

4.2 ElGamal

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 [4] is parameterized by a cyclic group $G = \langle g \rangle$ of prime order $q > 2$. A summary of the routine is shown in Figure 10:

$\mathtt{KeyGen()}$

$x \xleftarrow{\$} \mathbb{Z}_q$
$\mathtt{sk} \leftarrow x$
$\mathtt{pk} \leftarrow g^x$
return $(\mathtt{pk}, \mathtt{sk})$

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

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

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

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

Fig. 10: ElGamal cryptosystem

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

Definition 2 (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 3 (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 [8]) 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 4 (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.

Proof. We will prove by a sequence of games. A summary can be found in Figure 11

$G_0 - G_2$

$x \xleftarrow{\$} \mathbb{Z}_q$
$m^* \xleftarrow{\$} G$
$y \xleftarrow{\$} \mathbb{Z}_q, w \leftarrow g^y$
$v \leftarrow m^* \cdot (g^x)^y$ $\quad \triangleright G_0$ - G_1
$v \xleftarrow{\$} G$ $\quad \triangleright G_2$
$c^* \leftarrow (w, v)$
$\hat{m} \xleftarrow{\$} A^{\mathtt{PCO}}(g^x, c^*)$ $\quad \triangleright G_0$
$\hat{m} \xleftarrow{\$} A^{\mathtt{PCO}_1}(g^x, c^*)$ $\quad \triangleright G_1$ - G_2
return $[\![\hat{m} = m^*]\!]$ $\quad \triangleright G_0$ - G_1
return $[\![\hat{m} = w^{-x} \cdot v]\!]$ $\quad \triangleright G_2$

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

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

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

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

Fig. 11: The sequence of games in proving Lemma 1

Game 0 is the OW-PCA game. Adversary A has access to the plaintext-checking oracle 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 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 PCO and $\mathtt{PCO}_1$ are equivalent, which means that $\mathtt{PCO}_1$ behaves identically to 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 12): B receives as the Gap Diffie-Hellman problem inputs g^x and g^y. g^x simulates an ElGamal public key, where as g^y simulates the first component of the challenge ciphertext. As in game 2, the second component of the challenge ciphertext can be randomly sampled. Finally, the $\mathtt{PCO}_1$ from game 2 can be perfectly simulated using the restricted DDH oracle $\mathcal{O}^{\mathrm{DDH}}$.

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

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

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

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

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

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

Fig. 12: Gap Diffie-Hellman adversary B simulates game 2 for A

If A wins game 2, then its output is $\hat{m} = w^{-x} \cdot v = g^{-xy} \cdot v$, so $m^{-1} \cdot v$ is g^{xy}, the correct answer to the Gap Diffie-Hellman problem. In other words, B solves its Gap Diffie-Hellman problem if and only if A wins the simulated game 2: $\mathtt{Adv}_2(A) = \mathtt{Adv}_{\mathrm{GapDH}}(B)$.

4.3 Lattice-based cryptosystems

Applying the encrypt-then-MAC transformation to the PKE subroutines of Kyber/ML-KEM (which we will call KyberPKE for short) will not produce CCA secure KEM. This is because KyberPKE is not one-way secure against plaintext checking attack. An abundance of literature [13,12,7] described plaintext-checking attacks that can recover the complete secret key using a few thousand oracle queries. Here we review the key-recovery plaintext-checking attack presented in [7].

Let $q = 3329, n = 256$, let $R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. The secret key of KyberPKE is $\mathtt{sk} = \mathbf{s} \in R_q^k$ where $k = 2, 3, 4$ for depending on the security level. The coefficients of the secret key are sampled from centered binomial distribution

CBD_{η_1}. The ciphertext takes the form $c = (\mathbf{u} \in R_q^k, v \in R_q)$. The decryption routine is described in Figure 13

`KyberPKE.Dec`$(\mathtt{sk}, c)$

Require: $c = (\mathbf{u}, v)$
Require: $\mathtt{sk} = \mathbf{s}$
1: $w \leftarrow v - \langle \mathbf{s}, \mathbf{u} \rangle$
2: $m = \mathtt{Round}(w)$ $\triangleright\ m \in Z_2[x]/\langle x^n + 1 \rangle$
3: **return** m

Fig. 13: KyberPKE decryption routine

For $1 \le i \le k, 0 \le j \le n-1$ denote the i-th entry in a polynomial vector $\mathbf{s} \in R_q^k$ by s_i and the coefficient of the j-th power term of a polynomial $v \in R_q$ by $v[j]$. For some fixed i, j, an adversary can learn some information about $s_i[j]$ by crafting malformed ciphertext $\tilde{c} = (\tilde{\mathbf{u}}, \tilde{v})$ where $\tilde{u}_i[1], \tilde{v}[j]$ are chosen by the adversary, and all other coefficients are set to 0. When decrypting this malformed ciphertext, $w[j] = \tilde{v}[j] - s_i[j]\tilde{u}_i[0]$ and all other coefficients of w are 0s. With specific choices of values for $\tilde{u}_i[0], \tilde{v}[j]$, whether $w[j]$ will round to 1 or 0 will depend on the value of $s_i[j]$. This can be used to perform a binary search that recovers the value of $s_i[j]$ in $\lceil \log_2(2 \cdot \eta_1 + 1) \rceil$ plaintext-checking queries. See Table 1 for an example of plaintext-checking query strategy. Such a binary search can then be repeated $n \cdot k$ times to completely recover the secret key.

$u_i[0]$	$v[j]$	if $m[j] = 1$	if $m[j] = 0$
208	416	$s_i[j] \in \{-3\}$	$s_i[j] \in \{-2, -1, 0, 1, 2, 3\}$
208	624	$s_i[j] \in \{-3, -2\}$	$s_i[j] \in \{-1, 0, 1, 2, 3\}$
208	832	$s_i[j] \in \{-3, -2, -1\}$	$s_i[j] \in \{0, 1, 2, 3\}$
208	1040	$s_i[j] \in \{-3, -2, -1, 0\}$	$s_i[j] \in \{1, 2, 3\}$
208	1248	$s_i[j] \in \{-3, -2, -1, 0, 1\}$	$s_i[j] \in \{2, 3\}$
208	1456	$s_i[j] \in \{-3, -2, -1, 0, 1, 2\}$	$s_i[j] \in \{3\}$
208	-1456	$s_i[j] \in \{-2, -1, 0, 1, 2, 3\}$	$s_i[j] \in \{-3\}$
208	-1248	$s_i[j] \in \{-1, 0, 1, 2, 3\}$	$s_i[j] \in \{-3, -2\}$
208	-1040	$s_i[j] \in \{0, 1, 2, 3\}$	$s_i[j] \in \{-3, -2, -1\}$
208	-832	$s_i[j] \in \{1, 2, 3\}$	$s_i[j] \in \{-3, -2, -1, 0\}$
208	-624	$s_i[j] \in \{2, 3\}$	$s_i[j] \in \{-3, -2, -1, 0, 1\}$
208	-416	$s_i[j] \in \{3\}$	$s_i[j] \in \{-3, -2, -1, 0, 1, 2\}$

Table 1: Plaintext-checking query strategy for Kyber512: $\eta_1 = 3, d_u = 10, d_v = 4$

4.4 Code-based cryptosystems

4.5 Other cryptosystems

5 Implementation and performance analysis

In this section, we instantiate the encrypt-then-MAC KEM transformation using the subroutines of classic McEliece and a variety of MAC implementations. We will discuss the modifications and their rationale, as well as the performance implications.

Classic McEliece is an IND-CCA secure post-quantum KEM submitted to NIST's Post Quantum Cryptoraphy (PQC) standardization project and is currently one of three viable round 4 candidate. 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. Specifically, because the encoding subroutine is deterministic, Classic McEliece does not need derandomization, and although Classic McEliece uses re-encryption for ensuring ciphertext integrity, it does not simply include the PKE public key in the KEM secret secret and apply the PKE encryption. We will discuss Classic McEliece's re-encryption in details shortly.

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 14 and 15 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. 14: Classic McEliece uses the Niederreiter variant of the McEliece cryptosystem. When instantiated with binary Goppa code, this PKE achieves OW-CPA security

5.1 Security analysis

PCA security of Niederreiter cryptosystem. The security notion of *One-wayness under plaintext-checking attack (OW-PCA)* was first defined by Okamoto and Pointcheval in [9], 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 [2], another generic CCA secure transformation whose security reduces to the OW-PCA security of the underlying PKE. OW-PCA came up again in [6], where the $U^{\not\perp}$ transformation converts OW-PCA secure PKE into an IND-CCA secure KEM. $U^{\not\perp}$ was used in Kyber's third round submission to NIST PQC project, although ML-KEM switched to using the $U_m^{\not\perp}$ variant because applying de-randomization and re-encryption to Kyber's PKE subroutines have made it a rigid PKE.

$\texttt{CMKEM.keygen}()$

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

$\texttt{CMKEM.enc}(\texttt{pk})$

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

$\texttt{CMKEM.dec}(\texttt{sk}, \mathbf{c})$

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

Fig. 15: Classic McEliece applies a modified Fujisaki-Okamoto transformation to achieve CCA security

[13]

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. 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
3. 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
4. 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
5. 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

6. 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
7. 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
8. 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
9. 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
10. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978). https://doi.org/10.1145/359340.359342, https://doi.org/10.1145/359340.359342
11. 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
12. 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
13. 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