A survey of generic IND-CCA2 transformations

Ganyu Xu

July 8, 2024

1 Preliminaries

1.1 Public-key encryption schemes

A public key encryption scheme $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$ is a collection of three routines. $\texttt{KeyGen}(1^\lambda)$ takes the security parameteras input and returns a keypair $(\texttt{pk}, \texttt{sk})$. $\texttt{E}(\texttt{pk}, m)$ takes some public key and some plaintext message $m \in \mathcal{M}_{\texttt{PKE}}$ and output a ciphertext $c \in \mathcal{C}_{\texttt{PKE}}$. Where the encryption routine is probabilistic, we model the randomness using a coin $r \in \mathcal{R}$ such that $E(\texttt{pk}, m; r)$ is deterministic with an explicit r. Finally, $\texttt{D}(\texttt{sk}, c)$ uses the secret key to decrypt the ciphertext.

1.1.1 Correctness

Conventionally we require a $\texttt{PKE}$ to be perfectly correct. This means that for all possible key pairs $(\texttt{pk}, \texttt{sk})$ and plaintexts m, the decryption routine always correctly inverts the encryption routine: $\texttt{D}(\texttt{sk}, \texttt{E}(\texttt{pk}, m)) = m$.

Where perfect correctness is not achieved, such as with most lattice-based encryption schemes, we need to account for the possiblity that decryption can fail. If under some keypair $(\texttt{pk}, \texttt{sk})$, a plaintext-ciphertext pair (m, c) is such that $c \overset{\$}{\leftarrow} \texttt{E}(\texttt{pk}, m)$ is obtained from encrypting m (probabilistically) but $m \neq \texttt{D}(\texttt{sk}, c)$, we call it a decryption failure. For probabilistic encryption routines where the coin is uniformly sampled from the coin space, we can quantify the probability that m triggers a decryption failure. From here, we can take the distribution of all keypairs and quantify the "correctness" of a (possibly imperfectly correct) encryption scheme. The following definition of δ-correctness is directly taken from [BDK+18].

Definition 1.1 (δ-correctness)**.** *A probabilistic public-key encryption scheme $\mathit{PKE} = (\mathit{KeyGen}, \mathit{E}, \mathit{D})$ is δ-correct if the expected maximal probability of decryption failure taken across the distribution of keypairs is at most δ:*

$$E\left[\max_{m \in \mathcal{M}} P\left[\mathit{D}(\mathit{sk}, \mathit{E}(\mathit{pk}, m))\right]\right] \leq \delta$$

where the expectation is taken over the distribution of keypairs and the probability is taken over the distribution of coins.

[HHK17] also defined an adversarial game in which the adversary's goal is to find some plaintext message to trigger a decryption failure. This adversarial game meaningfully models the real-world scenario in which decryption failure can reveal information about the secret key. Notice that when evaluating the win condition, the coin is uniformly random instead of being chosen by the adversary.

```
Algorithm 1 CORS
(pk, sk) $\overset{\$}{\leftarrow}$ KeyGen($1^\lambda$)
$m \overset{\$}{\leftarrow} A_{\texttt{CORS}}(1^\lambda, \texttt{pk}, \texttt{sk})$
return $[\![\texttt{D}(\texttt{sk}, \texttt{E}(\texttt{pk}, m)) \neq m]\!]$
```

Figure 1: The correctness game CORS

The definition of δ-correctness sets an explicit upper bound on the probability that any plaintext triggers decryption failure. This means that even if the `CORS` adversary actually finds the message m that is the most likely to trigger decryption failure, the probability of winning the `CORS` game is still upper-bounded by δ.

Lemma 1.0.1. *If `PKE` is δ-correct, then for all `CORS` adversaries A, even computationally unbounded ones, the probability of winning the `CORS` game is at most δ*

The values of δ for Kyber are taken directly from [ABD$^+$19]

security level	δ
Kyber512	2^{-139}
Kyber768	2^{-164}
Kyber1024	2^{-174}

Table 1: Concrete δ for Kyber

1.1.2 Security

An one-way adversary $A = (A_1, A_2)$ consists of two sub-routines A_1 and A_2. $s \xleftarrow{\$} A_1^{\mathcal{O}_1}(1^\lambda, \texttt{pk})$ takes the security parameter, some public-key, access to some oracle(s) $\mathcal{O}_1$, and outputs some intermediate state s. $\hat{m} \leftarrow A_2^{\mathcal{O}_2}(1^\lambda, \texttt{pk}, c^*)$ resumes from the output of A_1 and takes some challenge ciphertext, then tries to guess the corresponding decryption.

The advantage $\texttt{Adv}_{\texttt{OW-ATK}}(A)$ of an `OW-ATK` adversary is the probability that its guess is correct.

Definition 1.2. *A `PKE` is `OW-ATK` secure if for all efficient adversaries A, the advantage in the `OW-ATK` game is neligigble with respect to the security parameter:*

$$\texttt{Adv}_{\texttt{OW-ATK}}(A) \leq \text{negl}(\lambda)$$

An **indistinguishability** adversary $A = (A_1, A_2)$ similarly consists of two sub-routines. The first subroutine adversarially chooses two distinct plaintext messages, and the second sub-routine tries to distinguish which of the two plaintext messages is the decryptino of the challenge encryption. The advantage of an indistinguishability adversary is defined by $\texttt{Adv}_{\texttt{IND-ATK}}(A) = P[\hat{b} = b] - \frac{1}{2}$

Definition 1.3. *A `PKE` is `IND-ATK` secure if for all efficient adversaries A, the advantage in the indistinguishability game is negligible with respect to the security parameter*

$$\texttt{Adv}_{\texttt{IND-ATK}}(A) \leq \text{negl}(\lambda)$$

The security games are described in details in figure 2

Algorithm 2 `OW-ATK` game

```
$(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$
$s \xleftarrow{\$} A_1^{\mathcal{O}_1}(1^\lambda, \texttt{pk})$
$m^* \xleftarrow{\$} \mathcal{M}$
$c^* \xleftarrow{\$} \texttt{E}(\texttt{pk}, m^*)$
$\hat{m} \leftarrow A_2^{\mathcal{O}_2}(1^\lambda, \texttt{pk}, s, c^*)$
return $[\![\hat{m} = m^*]\!]$
```

Algorithm 3 `IND-ATK` game

```
$(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$
$(m_0, m_1) \xleftarrow{\$} A_1^{\mathcal{O}_1}(1^\lambda, \texttt{pk})$
$b \xleftarrow{\$} \{0, 1\}$
$c^* \xleftarrow{\$} \texttt{E}(\texttt{pk}, m_b)$
$\hat{b} \leftarrow A_2^{\mathcal{O}_2}(1^\lambda, \texttt{pk}, s, c^*)$
return $[\![\hat{b} = b]\!]$
```

Figure 2: The indistinguishability security game

The capabilities of the adveraries are modeled using different collections of oracles. In standard security requirements, adversaries with access to no additional oracles can only mount chosen plaintex attacks (CPA), adversaries with access to decryption oracle $\mathcal{O}^{\mathtt{D}}$ only before the receiving challenge ciphertext can mount non-adaptive chosen ciphertext attacks (CCA1), adversaries with access to decryption oracle both before and after receiving the challenge ciphertext can mount adaptive chosen ciphertext attacks (CCA2).

[HHK17] also defined two non-standard oracles and the corresponding attacks. The plaintext checking oracle $\mathtt{PCO}(m, c)$ returns 1 if m is a decryption of c and 0 otherwise. The ciphertext validation oracle $\mathtt{CVO}(c)$ returns 1 if c is a valid ciphertext and 0 otherwise.

```
Algorithm 4 PCO($m, c$)
  return $[\![\mathtt{D}(\mathtt{sk}, c) = m]\!]$
```

```
Algorithm 5 CVO($c$)
  return $[\![\mathtt{D}(\mathtt{sk}, c) \in \mathcal{M}]\!]$
```

Figure 3: PCO and CVO

Here is an overview of the various kinds of attacks and their associated oracles

ATK	$\mathcal{O}_1$	$\mathcal{O}_2$
CPA	$-$	
CCA1	$\mathcal{O}^{\mathtt{D}}$	-
CCA2	$\mathcal{O}^{\mathtt{D}}$	
PCVA	PCO, CVO	
PCA	PCO	
VA	CVO	

Table 2: Attacks and associated oracle access

[HHK17] stated a "well-known" result that the IND-CPA security of a scheme with a large message space implies OW-CPA security:

Theorem 1.1. *For every IND-CPA adversary B against some PKE, there exists an OW-CPA adversary A against the same PKE such that:*

$$\mathtt{Adv}_{\mathtt{OW\text{-}CPA}}(A) = \frac{1}{|\mathcal{M}|} + \mathtt{Adv}_{\mathtt{IND\text{-}CPA}}(B)$$

1.1.3 Spread and rigidity

The spread of a public key encryption scheme measures the diffusion the encryption routine's output. The higher the spread, the lower the probability of obtaining any specific ciphertext.

Definition 1.4 (γ-spread)**.** *For a given keypair $(\mathtt{pk}, \mathtt{sk})$ and plaintext message $m \in \mathcal{M}$, the* min-entropy *of the encryption routine is:*

$$\text{min-entropy}(\mathtt{pk}, m) := -\log_2 \left(\max_{c \in \mathcal{C}} P\left[c = \mathtt{E}(\mathtt{pk}, m)\right] \right)$$

A PKE has γ-spread if for all keypairs $(\mathtt{pk}, \mathtt{sk})$ and plaintext $m \in \mathcal{M}$:

$$\text{min-entropy}(\mathtt{pk}, m) \leq \gamma$$

Having γ sperad means that for any keypair $(\mathtt{pk}, \mathtt{sk})$, plaintext m, and ciphertext c:

$$P\left[c = \mathtt{E}(\mathtt{pk}, m)\right] \leq 2^{-\lambda}$$

Finally, *rigidity* conveys the idea that a ciphertext cannot be perturbed without becoming either invalid or decrypting to another plaintext

Definition 1.5 (rigidity)**.** *$\mathtt{PKE}(\mathtt{KeyGen}, \mathtt{E}, \mathtt{D})$ is* rigid *if for all keypairs $(\mathtt{pk}, \mathtt{sk})$ and ciphertext c, either $\mathtt{D}(\mathtt{sk}, c) = \perp$ or $\mathtt{E}(\mathtt{pk}, \mathtt{D}(\mathtt{sk}, c)) = c$*

1.2 Key encapsulation mechanism

A key encapsulation mechanism $\texttt{KEM} = (\texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$ is a collection of three routines. The key generation routine $(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$ takes the security parameter 1^λ and returns a keypair. The encapsulation routine $(c, K) \xleftarrow{\$} \texttt{Encap}(\texttt{pk})$ takes the public key and outputs some ciphertext $c \in \mathcal{C}_{\texttt{KEM}}$ and some shared secret $K \in \mathcal{K}_{\texttt{KEM}}$. Finally, the decapsulation routine $K \leftarrow \texttt{Decap}(\texttt{sk}, c)$ takes the secret key and a ciphertext and outputs the correponding shared secret.

Correctness: similar to a $\texttt{PKE}$, key encapsulation mechanisms are usually required to be perfectly correct, meaning that for all keypairs $(\texttt{pk}, \texttt{sk})$, decapsulation always outputs the same shared secret as the encapsulation

$$\mathrm{P}\left[(c, K_1) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}); K_2 \leftarrow \texttt{Decap}(\texttt{sk}, c); K_1 = K_2\right] = 1$$

However, where decapsulation failure has a non-zero probability, we need to upperbound this quantity.

Definition 1.6 (δ-correctness)**.** *A $\texttt{KEM} = (\texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$ is δ-correct if the probability of decapsulation failure taken across the keypair distribution is at most δ:*

$$P\left[\texttt{Decap}(\texttt{sk}, c) \neq K \mid (\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(); (c, K) \xleftarrow{\$} \texttt{Encap}(\texttt{pk})\right] \leq \delta$$

Security: the security of a $\texttt{KEM}$ is defined in an adversarial game in which the adversary's goal is to distinguish between shared secret derived from encapsulation and uniformly random samples. An adversary $A = (A_1, A_2)$ contains two sub-routines with access to some oracle $\mathcal{O}$ depending on game.

Algorithm 6 $\texttt{IND-CCA2}$ game

```
$(\texttt{pk}, \texttt{sk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$
$s \xleftarrow{\$} A_1^{\mathcal{O}^{\texttt{Decap}}}(1^\lambda, \texttt{pk})$
$(c^*, K_0) \xleftarrow{\$} \texttt{Encap}(\texttt{pk})$
$K_1 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$
$b \xleftarrow{\$} \{0, 1\}$
$\hat{b} \xleftarrow{\$} A_2^{\mathcal{O}^{\texttt{Decap}}}(1^\lambda, \texttt{pk}, s, c^*, K_b)$
return $[\![\hat{b} = b]\!]$
```

Algorithm 7 $\texttt{Decap}$ oracle $\mathcal{O}^{\texttt{Decap}}(c \neq c^*)$

```
return $\texttt{Decap}(\texttt{sk}, c)$
```

Figure 4: The $\texttt{IND-CCA2}$ game for $\texttt{KEM}$

Definition 1.7 ($\texttt{IND-CCA2 security}$)**.** *A $\texttt{KEM} = (\texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$ is $\texttt{IND-CCA2}$ secure if no efficient adversary has non-negligible advantage in the $\texttt{IND-CCA2}$ game.*

2 Modular Fujisaki-Okamoto transformation

The Fujisaki-Okamoto transformation [FO99] and its KEM variations [HHK17] achieve security against adaptive chosen-ciphertext attacks through *de-randomization* and *re-encryption.* In particular, the modular FO transformation contains two steps. The first step (denoted the T transformation) takes a $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$ and outputs a $\texttt{PKE}_1 = (\texttt{KeyGen}, \texttt{E}_1, \texttt{D}_1)$. The key generation remains unchanged, but the encryption routine is *de-randomized* by deriving the coin from the plaintext using a hash function $G : \mathcal{M}_{\texttt{PKE}} \rightarrow \mathcal{R}_{\texttt{PKE}}$, and the decryption routine uses *re-encryption* to reject invalid ciphertexts.

Algorithm 8 $\mathtt{E}_1(\mathtt{pk}, m)$

```
$r \leftarrow G(m)$
$c \leftarrow \mathtt{E}(\mathtt{pk}, m; r)$
return $c$
```

Algorithm 9 $\mathtt{D}_1(\mathtt{sk}, c)$

```
$\hat{m} \leftarrow D(\mathtt{sk}, c)$
if $\hat{m} \in \mathcal{M}_{\mathtt{PKE}} \land \mathtt{E}(\mathtt{pk}, \hat{m}; G(\hat{m})) = c$ then
    return $\hat{m}$
end if
return $\perp$
```

Figure 5: T transformation

[HHK17] states the security property of $\mathtt{PKE}_1$ as follows:

Theorem 2.1. *If PKE is δ-correct and has γ spread, then for every OW-PCVA adversary B against PKE_1 who makes q_G hash queries, q_V ciphertext validation queries, and q_P plaintext checking queries, there exists an OW-CPA adversary A such that:*

$$\mathbf{Adv}(B) \leq q_V \cdot 2^{-\gamma} + (q_G + q_P) \cdot \delta + (1 + q_G + q_P) \cdot \mathbf{Adv}(A)$$

In other words, if PKE is OW-CPA secure, then $\mathtt{PKE}_1$ is OW-PCVA secure, though the security is non-tight. On the other hand, if PKE is IND-CPA secure, the OW-PCVA security is tight:

Theorem 2.2. *for every OW-PCVA adversary B against PKE_1 who makes q_G hash queries, q_V ciphertext validation queries, and q_P plaintext checking queries, there exists an IND-CPA adversary A such that:*

$$\mathbf{Adv}(B) \leq q_V \cdot 2^{-\gamma} + (q_G + q_P) \cdot \delta + \frac{2 \cdot q_G + 1}{|\mathcal{M}_{PKE}|} + 3 \cdot \mathbf{Adv}(A)$$

Proof. We will provide a sketch of proof for theorem 2.1 and 2.2. The main idea is to construct an OW-CPA/IND-CPA adversary A who can simulate the OW-PCVA game and use the OW-PCVA adversary B as a sub-routine. However, there are three main difficulties with constructing a "convincing" simulation:

1. A has no access to PCO
2. A has no access to CVO
3. The challenge ciphertext A receives is obtained using a truly random coin, but the challenge ciphertext B expects is obtained using a pseudorandom coin $r \leftarrow G(m)$

□

2.1 From PKE to KEM

The second part of the modular FO transform (denoted the U transform) takes a PKE and outputs a KEM. Depending on whether the input PKE is rigid and whether the output KEM rejects invalid ciphertext implicitly or explicitly, the security requirements for the input PKE and the construction of the KEM will vary. [HHK17] presents four variations of the U transform, which are listed in table 3

Name	PKE requirements	rejection	shared secret
$U^{\perp}$	OW-PCVA	$\perp$	$H(m,c)$
$U^{\not\perp}$	OW-PCA	$H(s,c)$	$H(m,c)$
$U_m^{\perp}$	rigidity + OW-VA	$\perp$	$H(m)$
$U_m^{\not\perp}$	rigidity + OW-CPA	$H(s,c)$	$H(m)$

Table 3: A summary of variants of U transformations

For the remaining of this section we will present the four transformations and sketch proofs of their securities.

2.1.1 KEM with explicit rejection from randomized PKE

Let $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$ be a public-key encryption scheme, and $H : \mathcal{M}_{\texttt{PKE}} \times \mathcal{C}_{\texttt{PKE}} \rightarrow \mathcal{K}_{\texttt{KEM}}$ be a hash function. The $U^{\perp}$ transformation outputs a $\texttt{KEM}^{\perp} = (\texttt{KeyGen}^{\perp}, \texttt{Encap}^{\perp}, \texttt{Decap}^{\perp})$, where the key generation routine remains unchanged: $\texttt{KeyGen}^{\perp} = \texttt{KeyGen}$. The encapsulation and decapsulation routines are described in figure 6

Algorithm 10 $\texttt{Encap}^{\perp}(\texttt{pk})$

```
$m \overset{\$}{\leftarrow} \mathcal{M}_{\texttt{PKE}}$
$c \overset{\$}{\leftarrow} \texttt{E}(\texttt{pk}, m)$
$K \leftarrow H(m, c)$
return $(c, K)$
```

Algorithm 11 $\texttt{Decap}^{\perp}(\texttt{sk}, c)$

```
$\hat{m} \leftarrow \texttt{D}(\texttt{sk}, c)$
if $\hat{m} \in \mathcal{M}_{\texttt{PKE}}$ then
    $K \leftarrow H(\hat{m}, c)$
    return $K$
end if
return $\perp$
```

Figure 6: $U^{\perp}$ routines

Under the random oracle model, the security of $\texttt{KEM}^{\perp}$ depends on the security of the input $\texttt{PKE}$.

Theorem 2.3. *For every IND-CCA2 adversary B against $\texttt{KEM}^{\perp}$, there exists an OW-PCVA adversary A against the underlying PKE such that*

$$\textit{Adv}(B) \leq \textit{Adv}(A)$$

Proof. For a sketch of proof, we argue that under the random oracle model, $H(m, c)$ is indistinguishable from a uniformly random sample from $\mathcal{K}_{\texttt{KEM}}$ from the adversary B's perspective unless B somehow queries H on (m, c). This means that an OW-PCVA adversary can perfectly simulate the decapsulation oracle using uniformly random samples, as long as the outputs from the simulated decapsulation oracle are consistent with the outputs from the hash functions.

The only flaw in the simulated decapsulated oracle lies in the case when B queries $H(m^*, c^*)$, but because A has access to PCO, A can detect that B has successfully recovered m^* and uses the recovered m^* to win the OW-PCVA game. In other words, A keeps running B in a simulation until B makes the special hash query, at which time A simply terminates B and outputs the answer.

To make the simulated decapsulation oracle and the hash oracle consistent, A maintains two tapes $\mathcal{L}^{\texttt{Decap}}$ and $\mathcal{L}^{H}$ that record the queries made to the decapsulation oracle and the hash oracles respectively:

$$(c, K) \in \mathcal{L}^{\texttt{Decap}} \Leftrightarrow K = \mathcal{O}^{\texttt{Decap}}(c)$$
$$(m, c, K) \in \mathcal{L}^{H} \Leftrightarrow K = H(m, c)$$

If $\mathcal{O}^{\texttt{Decap}}$ has been queried with some valid ciphertext c, then when H is queried with a corresponding input (m, c) where m is the decryption of c, H needs to output the same value as $\mathcal{O}^{\texttt{Decap}}$. Similarly, if H has been queried with some valid plaintext-ciphertext pair (m, c), then when $\mathcal{O}^{\texttt{Decap}}$ is queried with c, the two oracles need to output the same value. The details of the simulation are described in figure 7

Algorithm 12 $\mathcal{O}_1^{\texttt{Decap}}(c)$

```
if $\exists(\tilde{c}, \tilde{K}) \in \mathcal{L}^{\texttt{Decap}} : \tilde{c} = c$ then
    return $\tilde{K}$
end if
if $\texttt{CVO}(c) = 1$ then
    $K \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$
    Append $(c, K)$ to $\mathcal{L}^{\texttt{Decap}}$
    return $K$
end if
return $\bot$
```

Algorithm 13 $H_1(m, c)$

```
if $\exists(\tilde{m}, \tilde{c}, \tilde{K}) \in \mathcal{L}^H : (\tilde{m}, \tilde{c}) = (m, c)$ then
    return $\tilde{K}$
end if
$K \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$
if $\texttt{PCO}(m, c) = 1$ then
    if $\exists(\tilde{c}, \tilde{K}) \in \mathcal{L}^{\texttt{Decap}} : \tilde{(c)} = c$ then
        return $\tilde{K}$
    else
        Append $(c, K)$ to $\mathcal{L}^{\texttt{Decap}}$
    end if
end if
Append $(m, c, K)$ to $\mathcal{L}^H$
return $K$
```

Figure 7: Patched oracles in $U^{\perp}$

□

2.1.2 KEM with implicit rejection from randomized PKE

Given $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$, the $U^{\not\perp}$ transformation outputs $\texttt{KEM} = (\texttt{KeyGen}^{\not\perp}, \texttt{Encap}^{\not\perp}, \texttt{Decap}^{\not\perp})$. With implicit rejection, decapsulation routine returns some output from H even when the input ciphertext c is malformed. Specifically, the implicit rejection value depends on the ciphertext and some secret value s that is a part of the secret key.

Algorithm 14 $\texttt{KeyGen}^{\not\perp}$

```
$(\texttt{pk}, \texttt{sk}') \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$
$s \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$
$\texttt{sk} = (\texttt{sk}', s)$
return $(\texttt{pk}, \texttt{sk})$
```

Algorithm 15 $\texttt{Encap}^{\not\perp}(\texttt{pk})$

```
$m \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$
$c \xleftarrow{\$} \texttt{E}(\texttt{pk}, m)$
$K \leftarrow H(m, c)$
return $(c, K)$
```

Algorithm 16 $\texttt{Decap}^{\not\perp}(\texttt{sk}, c)$

```
$\hat{m} \leftarrow \texttt{D}(\texttt{sk}, c)$
if $\hat{m} \in \mathcal{M}_{\texttt{PKE}}$ then
    $K \leftarrow H(\hat{m}, c)$
    return $K$
end if
return $H(s, c)$
```

Figure 8: $U^{\not\perp}$ routines

The security of $\texttt{KEM}^{\not\perp}$, similar to the security of $\texttt{KEM}^{\perp}$, derives from the indistinguishability between pseudorandom $K \leftarrow H(m, c)$ and truly random $K \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$ under the random oracle model, unless the KEM adversary somehow queries H on (m^*, c^*), in which case an adversary against the underlying PKE with access to a PCO can detect this special query use it to win the OW-PCA game. In addition, because $\texttt{Decap}^{\not\perp}$ will always return random-looking values regardless of the validity of the input ciphertext, the decapsulation oracle can be simulated without needing a ciphertext validation oracle. The KEM adversary will be able to distinguish between a true decapsulation oracle and a simulated decapsulation oracle, but only by querying H on (s, c) for some c. Since s is uniformly random, each query has a $|\mathcal{M}_{\texttt{PKE}}|^{-1}$ chance of hitting s, so across q_H hash queries to H, the probability of hitting s at least once is at most $\frac{q_H}{|\mathcal{M}_{\texttt{PKE}}|}$.

Theorem 2.4. *For every IND-CCA adversary B against $\texttt{KEM}^{\not\perp}$ that makes q_H hash queries to H, there exists an OW-PCA adversary A against the underyling PKE such that:*

$$\texttt{Adv}(B) \leq \frac{q_H}{|\mathcal{M}_{\texttt{PKE}}|} + \texttt{Adv}(A)$$

2.1.3 KEM with explicit rejection from rigid PKE

If the input PKE is rigid (definition 1.5), then the PCO can be simulated with indistinguishability from the true PCO except for when decryption failure occurs. Given $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$ and hash function $H : \mathcal{M}_{\texttt{PKE}} \to \mathcal{K}_{\texttt{KEM}}$, the $U_m^{\perp}$ transformation outputs a $\texttt{KEM}_m^{\perp} = (\texttt{KeyGen}, \texttt{Encap}_m^{\perp}, \texttt{Decap}_m^{\perp})$ where the key generation routine remains unchanged. The encapsulation and decapsulation routines are described in figure 9

Algorithm 17 $\texttt{Encap}_m^{\perp}(\texttt{pk})$

```
$m \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$
$c \leftarrow \texttt{E}(\texttt{pk}, m)$
$K \leftarrow H(m)$
return $(c, K)$
```

Algorithm 18 $\texttt{Decap}_m^{\perp}(\texttt{sk}, c)$

```
$\hat{m} \leftarrow \texttt{D}(\texttt{sk}, c)$
if $\hat{m} \in \mathcal{M}_{\texttt{PKE}}$ then
    return $H(m)$
end if
return $\perp$
```

Figure 9: $\texttt{KEM}_m^{\perp}$ routines

Under the random oracle model, no adversary B can distinguish between the pseudorandom $K_0 \leftarrow H(m)$ and the truly random $K_1 \xleftarrow{\$} \mathcal{K}_{\texttt{KEM}}$ unless B queries H on m^*. Because rigidity means $\texttt{E}(\texttt{pk}, m^*) = c^*$ is equivalent to $\texttt{D}(\texttt{sk}, c^*) = m^*$, if H is queried on m^*, a second adversary A against the PKE can detect the special query and use it to win the OW-VA game.

Theorem 2.5. *For every IND-CCA2 adversary B against $\texttt{KEM}_m^{\perp}$, there exists an OW-VA adversary A against the underlying PKE such that*

$$\texttt{Adv}(B) \leq \texttt{Adv}(A) + \delta \cdot (q_H + q_D)$$

2.1.4 KEM with implicit rejection from rigid PKE

References

[ABD$^+$19] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2(4):1–43, 2019.

[BDK$^+$18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual international cryptology conference*, pages 537–554. Springer, 1999.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.