StarHunters— Secure Hybrid Post-Quantum KEMs From IND-CCA2 PKEs

Deirdre Connolly[1], Mike Ounsworth[2], Sophie Schmieg[3], and Douglas Stebila[4]

[1]Oracle, Selkie Cryptography
[2]Entrust
[3]Google
[4]University of Waterloo

March 2, 2026

Abstract

This paper formally specifies and analyzes the CK hybrid key encapsulation mechanism (KEM) construction from the IRTF CFRG's recent draft on hybrid (post-quantum/traditional) KEMs [6]: CK combines two KEMs using a PRF to produce a hybrid KEM. Unlike the QSF framework of Barbosa et al., which combines an IND-CCA KEM with a nominal group (Diffie-Hellman-style), CK combines a C2PRI-secure post-quantum-secure KEM with an IND-CCA traditionlly-secure KEM constructed from an IND-CCA2 public key encryption (PKE) scheme, such as RSA-OAEP. We additionally show how to securely promote an IND-CCA2 PKE into an IND-CCA KEM. We perform two complementary security analyses of CK in the standard model: the first shows CK is IND-CCA assuming the traditional KEM is IND-CCA, the post-quantum KEM is C2PRI, and the KDF is a secure PRF; the second shows CK is IND-CCA assuming the post-quantum KEM is IND-CCA and the KDF is a secure PRF, even if the traditional KEM is completely broken. Neither proof requires the random oracle model.

1 Introduction

With the advent of post-quantum cryptography, it is becoming popular to create *hybrid* schemes from traditional and post-quantum components: if the post-quantum component turns out to be weak, security falls back to the (non-post-quantum) security of traditional constructions. Conversely, when the traditional component becomes weak due to practical quantum attacks, security falls back to the (post-quantum) security of the post-quantum component scheme. [2] [14] [13] [8]. While there are multiple approaches to achieving hybrid security in deployed protocols, producing unified primitives, such as for key agreement, has become a popular approach for their modularity.

There exist multiple motivations for using hybrid algorithms, such as uncertainty about whether the mathematical underpinnings of the new algorithms will stand up to quantum attack in the long term, or for more operationally-focused reasons such as wanting to deploy new algorithm implementations alongside battle-hardened mature algorithm implementations, to hedge against implementation bugs. In the context of key agreement, hybrids typically take the form of combining a key encapsulation mechanism (KEM) with either a public key encryption (PKE) scheme [12] or Diffie-Hellman-style key agreement (DH) scheme over elliptic curves [2].

Contributions. Previous work by Barbosa et al. [2] introduced the QSF framework, which for combines an IND-CCA-secure and ciphertext second pre-image (C2PRI) secure KEM (such as ML-KEM [11]) with a nominal group [2] such as x25519. However, in some circumstances there is a desire to combine an IND-CCA-secure KEM with an IND-CCA2-secure PKE or with a second IND-CCA-secure KEM. This work covers these cases by first providing a simple framework to promote an IND-CCA2-secure PKE into an IND-CCA-secure

KEM, and then a framework for combining the two component KEMs to produce an IND-CCA-secure hybrid KEM.

Outline. Notation and definitions are introduced section 2. We show how to securely construct an IND-CCA key encapsulation mechanism (KEM) from an IND-CCA2 public key encryption scheme (PKE) such as RSA-OAEP in section 3. The CK framework is introduced in section 4; our generic construction is a general framework that can be instantiated with different component algorithms. In section 5 we prove the IND-CCA security when the different component KEMs fail in different ways. Then in section 6 we give concrete instances of CK and show that they are secure.

2 Definitions

2.1 Notation and conventions

$a \leftarrow\!\!\$\ A$ denotes sampling a uniformly at random from a non-empty finite set A. $\leftarrow$ denotes a deterministic assignment of a variable. $\{0,1\}^n$ is the set of all bitstrings of length n. (x, y) denotes a tuple of two elements x and y. $\mathbf{X}[y]$ denotes access into the table $\mathbf{X}$ at position y. Tables are denoted with bold uppercase variable names or $\sum$. An uninitialized position in a table is denoted with the bottom symbol $\perp$. $\mathbf{X}[\cdot] \leftarrow y$ sets all positions of table $\mathbf{X}$ to y. $o \leftarrow\!\!\$\ \mathcal{A}(I)$ denotes running the algorithm $\mathcal{A}$ with input I with uniform random coins and o describing its output.

Definition 1 (Adversary $\mathcal{A}$)**.** An algorithm with explicit inputs and which may have access to an oracle O which will perform computations using a secret value, but not reveal that secret value. Where an adversary is invoked multiple times and is allowed to keep state between invocations, this is denoted with a state parameter *st*.

If $\mathcal{A}$ has additionally access to an oracle O, this is denoted as $o \leftarrow\!\!\$\ \mathcal{A}^{O(\cdot)}(I)$. A security game consists of a main procedure and optionally some oracle procedures. When a game is played, the main procedure is run and adversary $\mathcal{A}$ is given some inputs and access to the oracle procedures. Based on the output of the adversary $\mathcal{A}$ and its oracle calls, the main procedure outputs 1 or 0 depending on whether the adversary $\mathcal{A}$ won the game. If a game aborts at any time, it means that the adversary has no advantage in winning this game. In case of a decision game, this means that the game returns a random bit indicating whether the adversary has won or not. Whenever an adversary algorithm executes "stop with x", it halts returning x to its challenger. We denote an adversary $\mathcal{A}$ playing game G as $G_{\mathcal{A}}$. The probability of the game returning a 1 when played by adversary $\mathcal{A}$ is written as $\Pr[G_{\mathcal{A}} \Rightarrow 1]$. Our security analyses are concrete, which means that we prove that the advantage of an attacker against a construction with fixed parameters is bounded by a real value that is argued to be small. Here, *small* means a summation of statistical terms and advantage terms, the former representing worst-case probabilities below 2^{-128} and the latter representing attacks on lower-level primitives that are assumed to require at least 2^{128} steps to break.

2.2 Key encapsulation mechanisms

Definition 2 (KEM)**.** A *key encapsulation mechanism* scheme consists of three functions:

- KeyGen: $0 \xrightarrow{\$} \mathcal{P} \times \mathcal{S}$
- Encaps: $\mathcal{P} \xrightarrow{\$} \mathcal{C} \times \mathcal{K}$
- Decaps: $\mathcal{S} \times \mathcal{C} \rightarrow \mathcal{K}$

Where $\mathcal{S}$, $\mathcal{P}$, and $\mathcal{C}$ are the sets of private keys, public keys, and ciphertexts respectively and $\mathcal{K}$ is the set of shared secrets encapsulated by the KEM.

Definition 3 (Correctness)**.** The correctness of a KEM imposes that, except with small probability drawn over the random coin space of KEM.KeyGen and KEM.Encaps, we have that KEM.Decaps correctly recovers the shared key produced by KEM.Encaps.

Formally, we say that a KEM is δ-correct if:

$$\Pr\left[k \neq k' \;\middle|\; \begin{matrix}(\mathsf{sk},\mathsf{pk}) \leftarrow\!\$ \mathsf{KeyGen}() \\ (k,c) \leftarrow\!\$ \mathsf{Encaps}(\mathsf{pk}) \\ k' \leftarrow \mathsf{Decaps}(c,\mathsf{sk})\end{matrix}\right] \leq \delta.$$

All KEMs in this document are assumed to be δ-correct, even if their other security properties may vary.

Definition 4 (IND-CCA Security)**.** Let KEM be a KEM and $\mathcal{A}$ be an adversary. Define the advantage of $\mathcal{A}$ in breaking the IND-CCA security of KEM by

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}}(\mathcal{A}) = \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}}.\text{Real}() \Rightarrow 1\right] - \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}}.\text{Random}() \Rightarrow 1\right]$$

$\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}}.\text{Real}()$

1 : $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KEM.KeyGen}()$
2 : $(\mathsf{ct}^*,\mathsf{k}) \leftarrow\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk})$
3 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk},\mathsf{ct}^*,\mathsf{k})$

$\mathsf{O}(\mathsf{ct})$

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** $\mathsf{KEM.Decaps}(\mathsf{sk},\mathsf{ct})$

$\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}}.\text{Random}()$

1 : $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KEM.KeyGen}()$
2 : $(\mathsf{ct}^*,\_) \leftarrow\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk})$
3 : $\mathsf{k} \leftarrow\!\$\ \mathsf{KEM}.\mathcal{K}$
4 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk},\mathsf{ct}^*,\mathsf{k})$

$\mathsf{O}(\mathsf{ct})$

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** $\mathsf{KEM.Decaps}(\mathsf{sk},\mathsf{ct})$

Definition 5 (C2PRI security of KEM)**.** Let KEM be a KEM and $\mathcal{A}$ be an adversary. Define the advantage of $\mathcal{A}$ in breaking the C2PRI security of KEM by

$$\mathsf{Adv}^{\text{C2PRI}}_{\mathsf{KEM}}(\mathcal{A}) = \Pr\left[\text{C2PRI}^{\mathcal{A}}_{\mathsf{KEM}}() \Rightarrow 1\right]$$

$\text{C2PRI}^{\mathcal{A}}_{\mathsf{KEM}}()$

1 : $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{KEM.KeyGen}()$
2 : $(\mathsf{ct}^*,\mathsf{k}^*) \leftarrow\!\$\ \mathsf{KEM.Encaps}(\mathsf{pk})$
3 : $\mathsf{ct} \leftarrow\!\$\ \mathcal{A}(\mathsf{pk},\mathsf{sk},\mathsf{ct}^*,\mathsf{k}^*)$
4 : **return** $(\mathsf{KEM.Decaps}(\mathsf{sk},\mathsf{ct}) = \mathsf{k}^*) \wedge (\mathsf{ct} \neq \mathsf{ct}^*)$

2.3 Randomized public key encryptions

Definition 6 (PKE)**.** A *public key encryption* scheme consists of three functions and set :

- $\mathsf{KeyGen}\colon \ \$\!\rightarrow \mathcal{P} \times \mathcal{S}$
- $\mathsf{Enc}\colon \mathcal{P} \times \mathcal{M} \ \$\!\rightarrow \mathcal{C}$
- $\mathsf{Dec}\colon \mathcal{S} \times \mathcal{C} \rightarrow \mathcal{M}$

where $\mathcal{S}$, $\mathcal{P}$, $\mathcal{C}$ as before refer to the sets of private keys, public keys, and ciphertexts respectively, and $\mathcal{M}$ is the set of messages that the PKE can encrypt.

Note that IND-CPAcan only apply to public key encryption schemes with a randomized (non-deterministic) PKE.Enc. Any PKE with a deterministic PKE.Enc cannot win the IND-CPA game because the adversary could run $\mathsf{PKE.Enc}(m_0)$ and $\mathsf{PKE.Enc}(m_1)$ for themselves and compare the output ct to the one given.

Definition 7 (IND-CCA for a (randomized) PKE)**.** Let PKE be a PKE and $\mathcal{A}$ be an adversary. Define the advantage of $\mathcal{A}$ in breaking the IND-CCA security of PKE by

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{PKE}}(\mathcal{A}) = \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Left}() \Rightarrow 1\right] - \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Right}() \Rightarrow 1\right]$$

$\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Left}()$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{PKE.KGen}()$
2 : $(m_0, m_1, st) \leftarrow\!\$\ \mathcal{A}^{\mathsf{O}}(\mathsf{pk})$
3 : $\mathsf{ct}^* \leftarrow\!\$\ \mathsf{PKE.Enc}(m_0)$
4 : **return** $\mathcal{A}^{\mathsf{O}}(st, \mathsf{pk}, \mathsf{ct}^*)$

$\mathsf{O}(\mathsf{ct})$

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$

$\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Right}()$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{PKE.KGen}()$
2 : $(m_0, m_1, st) \leftarrow\!\$\ \mathcal{A}^{\mathsf{O}}(\mathsf{pk})$
3 : $\mathsf{ct}^* \leftarrow\!\$\ \mathsf{PKE.Enc}(m_1)$
4 : **return** $\mathcal{A}^{\mathsf{O}}(st, \mathsf{pk}, \mathsf{ct}^*)$

$\mathsf{O}(\mathsf{ct})$

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** $\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$

2.4 Pseudorandom functions

Definition 8 (PRF)**.** A *PRF* is a function $F : \mathcal{K} \times \{0,1\}^* \to \mathcal{K}_o$.

Definition 9 (PR security for a PRF)**.** Let F be a PRF and $\mathcal{A}$ be an adversary. Define the advantage of $\mathcal{A}$ in breaking the PR (pseudorandomness) security of F by

$$\mathsf{Adv}^{\text{PR}}_F(\mathcal{A}) = \Pr\left[\text{PR}^{\mathcal{A}}_F.\text{Real}() \Rightarrow 1\right] - \Pr\left[\text{PR}^{\mathcal{A}}_F.\text{Random}() \Rightarrow 1\right]$$

$\text{PR}^{\mathcal{A}}_F.\text{Real}()$

1 : $X \leftarrow \emptyset$
2 : $k \leftarrow\!\$\ \mathcal{K}$
3 : **return** $\mathcal{A}^{\mathsf{O}}$

$\mathsf{O}(x)$

1 : **if** $x \in X$ **then** abort
2 : $X \leftarrow X \cup \{x\}$
3 : $y \leftarrow F(k, x)$
4 : **return** y

$\text{PR}^{\mathcal{A}}_F.\text{Random}()$

1 : $X \leftarrow \emptyset$
2 : **return** $\mathcal{A}^{\mathsf{O}}$

$\mathsf{O}(x)$

1 : **if** $x \in X$ **then** abort
2 : $X \leftarrow X \cup \{x\}$
3 : $y \leftarrow\!\$\ \mathcal{K}_o$
4 : **return** y

Derived PRFs. The CK construction uses a single function $F : \{0,1\}^* \to \mathcal{K}_o$. In our security proofs, we view F as a PRF keyed by different arguments. Define:

- $F' : \mathsf{KEM}_2.\mathcal{K} \times (\mathsf{KEM}_1.\mathcal{K} \times \{0,1\}^*) \to \mathcal{K}_o$ by $F'(\mathsf{k}_2, (\mathsf{k}_1, x)) = F(\mathsf{k}_1 \| \mathsf{k}_2 \| x)$, i.e., F viewed as a PRF keyed by k_2.
- $F'' : \mathsf{KEM}_1.\mathcal{K} \times (\mathsf{KEM}_2.\mathcal{K} \times \{0,1\}^*) \to \mathcal{K}_o$ by $F''(\mathsf{k}_1, (\mathsf{k}_2, x)) = F(\mathsf{k}_1 \| \mathsf{k}_2 \| x)$, i.e., F viewed as a PRF keyed by k_1.

The pre-quantum security analysis (theorem 2) requires F' to be a secure PRF, while the post-quantum security analysis (theorem 3) requires F'' to be a secure PRF. Concrete instantiations of CK (section 6) must demonstrate that the chosen F satisfies both properties.

3 IND-CCA KEM from IND-CCA2 PKE

A fundamental difference between a PKE and a KEM is that a PKE encrypts a message m from the PKE's message space $\mathcal{M}$ chosen by the caller, while a KEM encapsulates a key k chosen uniformly from a key space $\mathcal{K}$. Below is shown a way to promote a PKE into a KEM by uniformly selecting a $k \in \mathcal{K} \subseteq \mathcal{M}$. The shared

secret keys $k \in \mathcal{K}$ are required to be valid inputs to PKE.Enc, but need not cover the PKE's entire message space $\mathcal{M}$. For example, the PKE may be capable of encrypting any message up to 1024 bits, or maybe even strings of any length, yet the key space $\mathcal{K}$ only needs 256 bit strings.

LCK[PKE].KGen()	LCK[PKE].Encaps(pk)	LCK[PKE].Decaps(sk, ct)
1 : **return** PKE.KGen()	1 : $\mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$	1 : **return** PKE.Dec(sk, ct)
	2 : **return** PKE.Enc(pk, k)	

3.1 LCK is an IND-CCA-secure KEM

Claim 1. LCK *is an* IND-CCA-*secure* KEM *so long as* PKE *is an* IND-CCA-*secure* PKE.

3.1.1 Starting and ending games

We start with the LCK construction for promoting a PKE into a KEM. We end with a game in which the adversary is unable to distinguish the key k_0 that was encrypted to form ct^* from a new randomly-chosen key k_1.

Starting game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{LCK[PKE]}}$.Real()

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}()$
2 : $\mathsf{k}_0 \leftarrow\!\!\$\ \mathcal{K}$
3 : $\mathsf{ct}^* \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{k}_0)$
4 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_0)$

O(ct)

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** PKE.Dec(sk, ct)

Ending game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{LCK[PKE]}}$.Random()

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}()$
2 : $\mathsf{k}_0 \leftarrow\!\!\$\ \mathcal{K}$
3 : $\mathsf{ct}^* \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{k}_0)$
4 : $\mathsf{k}_1 \leftarrow\!\!\$\ \mathcal{K}$
5 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_1)$

O(ct)

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** PKE.Dec(sk, ct)

3.1.2 Game 1: give $\mathcal{A}$ random key instead of real

Game 1

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}()$
2 : $\mathsf{k}_0 \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{M}$
3 : $\mathsf{ct}^* \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{k}_0)$
4 : $\mathsf{k}_1 \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{M}$
5 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_1)$

O(ct)

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** PKE.Dec(sk, ct)

Reduction $\mathcal{B}_1^{\mathsf{Opke},\mathcal{A}}(\mathsf{pk})$

1 : $\mathsf{k}_0 \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{M}$
2 : $\mathsf{k}_1 \leftarrow\!\!\$\ \mathsf{PKE}.\mathcal{M}$
3 : output $(\mathsf{k}_0, \mathsf{k}_1, st = \perp)$ to IND-CCA challenger for PKE
4 : receive ct^* from IND-CCA challenger for PKE
5 : **return** $\mathcal{A}^{\mathsf{O}}(\mathsf{pk}, \mathsf{ct}^*, \mathsf{k}_0)$

O(ct)

1 : **if** $\mathsf{ct} = \mathsf{ct}^*$ **then**
2 : **return** $\perp$
3 : **else**
4 : **return** Opke(ct)

We give the reduction $\mathcal{B}_1$ which is an adversary against the IND-CCA security of PKE and which simulates either game 0 or game 1 to the KEM IND-CCA adversary $\mathcal{A}$. $\mathcal{B}_1$ works by using the public key from the PKE challenger, picks two random keys k_0 and k_1, one of which is encrypted by the PKE challenger, and

that ciphertext as the KEM challenge ciphertext. To answer decapsulation queries, $\mathcal{B}_1$ uses the decryption oracle from the PKE challenger; since the challenger ciphertext ct^* is prohibited for the KEM game, there is no concern that $\mathcal{B}_1$ will pass the prohibited ct^* to the PKE decryption oracle.

Claim 2. *When $\mathcal{B}_1$ interacts with* $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Left}()$*, it exactly simulates Game 0 =* $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{LCK[PKE]}}.\text{Real}()$.

Proof by inlining code and checking pseudocode equivalence.

Claim 3. *When $\mathcal{B}_1$ interacts with* $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}.\text{Right}()$*, it exactly simulates Game 1.*

Proof by inlining code and checking pseudocode equivalence.

3.1.3 Conclusion

Observe that Game 1 is already our ending game: Game 1 = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{LCK[PKE]}}.\text{Random}()$.

Thus,

Theorem 1.

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{LCK[PKE]}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{PKE}}(\mathcal{B}_1)$$

Thus the LCK construction can be used to promote an IND-CCA PKE into an IND-CCA KEM.

4 Introducing CK

We consider a construction which combines two KEMs (KEM_1 and KEM_2) together with a PRF F to form a hybrid $\mathsf{KEM} = \mathsf{CK}[\mathsf{KEM}_1, \mathsf{KEM}_2, F]$.

$\mathsf{CK}[\mathsf{KEM}_1, \mathsf{KEM}_2, F].\mathsf{KeyGen}()$

1: $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2: $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3: **return** $((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{sk}_1, \mathsf{sk}_2))$

$\mathsf{CK}[\mathsf{KEM}_1, \mathsf{KEM}_2, F].\mathsf{Encaps}(\mathsf{pk})$

1: parse $(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathsf{pk}$
2: $(\mathsf{ct}_1, \mathsf{k}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
3: $(\mathsf{ct}_2, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
4: $\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
5: **return** $((\mathsf{ct}_1, \mathsf{ct}_2), \mathsf{k})$

$\mathsf{CK}[\mathsf{KEM}_1, \mathsf{KEM}_2, F].\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

1: parse $(\mathsf{sk}_1, \mathsf{sk}_2) \leftarrow \mathsf{sk}$
2: parse $(\mathsf{ct}_1, \mathsf{ct}_2) \leftarrow \mathsf{ct}$
3: $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
4: $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
5: $\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
6: **return** k

5 Security of CK

The objective of a hybrid of this form is that the construction remains secure so long as one of the component algorithms remains secure. In other words, one algorithm may be found to contain weaknesses without compromising the hybrid.

Proof-strategy. Our proofs cover multiple scenarios, including the possibility of cryptographically-relevant quantum computers becoming a reality, and thus breaking the IND-CCA security of pre-quantum KEMs, and the possibility of post-quantum cryptography such as a KEM being classically broken, such as by an attack, vulnerability, or implementation error.

We will consider two cases. In the first case, we will analyze the security of QSF relative to the IND-CCA security of the traditional KEM, possibly constructed from an IND-CCA2 PKE scheme, and the C2PRI security of the post-quantum KEM, while in the second case, we analyze the security relative to the IND-CCA security of the post-quantum KEM.

Pre-Quantum Security. This the case where one of the component KEMs is assumed to be classically IND-CCA-secure but where the post-quantum KEM may not offer any IND-CCA security.

We prove in the standard model that CK is IND-CCA-secure when:

1. The KDF is modeled as a secure PRF;
2. The pre-quantum KEM is IND-CCA-secure; and
3. The post-quantum KEM is modeled as a possibly broken KEM that provides no security beyond C2PRI.

Our proof also demonstrates the opposite scenario:

POST-QUANTUM SECURITY: This the case where one of the component KEMs is assumed to be IND-CCA-secure against a quantum adversary but the pre-quantum KEM may not offer any security at all.

We prove in the standard model that CK is IND-CCA-secure when:

1. The KDF is modeled as a PRF;
2. The pre-quantum KEM no longer provides any security; and
3. The post-quantum KEM provides IND-CCA security against a quantum attacker.

These results do not rely on the ROM nor the QROM: it is sufficient to assume the standard model PRF security property for the employed hash function F, when viewed as a keyed function using k_2 as key.

5.1 Reduction to C2PRI security of KEM_1, IND-CCA security of KEM_2, and PRF security in the standard model

Starting assumptions:

- KEM_1 is C2PRI-secure,
- KEM_2 is IND-CCA-secure, and
- F is a secure PRF

5.1.1 Starting and ending games

Starting game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{CK}}$.Real()

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Ending game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{CK}}$.Random()

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k} \leftarrow\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

5.1.2 Game 1: use k_2^* instead of decapsulating ct_2^*

Instead of the oracle decapsulating ct_2^* to obtain k_2, it will return k_2 directly if presented with ct_2^*.

This is equivalent by inlining the result of performing the decapsulation on ct_2^*

Game 1

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \mathsf{k}_2^*) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then** $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
else $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 4. *Starting game and Game 1 are indistinguishable assuming the correctness of* KEM_2.

5.1.3 Game 2: use random key for k_2 in challenge instead of real

We give the reduction $\mathcal{B}_2$ which is an adversary against the IND-CCA security of KEM_2 and which simulates either game 0 or game 1 to the KEM IND-CCA adversary $\mathcal{A}$. $\mathcal{B}_2$ works by using the public key, challenge ciphertext ct_2^*, and challenge shared secret k_2^* from the KEM_2 challenger; it simulates the KEM_1 computations itself. To answer decapsulation queries, $\mathcal{B}_2$ uses the decapsulation oracle from the KEM_2 challenger. If the challenge ciphertext ct_2^* is used in the decapsulation query for the hybrid KEM, the reduction $\mathcal{B}_2$ uses the same challenge shared secret k_2^* as its alleged decapsulation; there is no concern that $\mathcal{B}_2$ will pass the prohibited ct_2^* to the KEM_2 decapsulation oracle.

Game 2

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_2^* \leftarrow\!\$\ \mathsf{KEM}_2.\mathcal{K}$
$\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then** $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
else $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Reduction $\mathcal{B}_2^{\mathsf{Okem2}, \mathcal{A}}(\mathsf{pk}_2, \mathsf{ct}_2^*, \mathsf{k}_2^*)$

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then** $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
else $\mathsf{k}_2 \leftarrow \mathsf{Okem2}(\mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 5. *When $\mathcal{B}_2$ interacts with* IND-CCA$^{\mathcal{A}}_{\mathsf{KEM}_2}$.Real()*, it exactly simulates Game 1.*

Proof by inlining code and checking pseudocode equivalence.

Claim 6. *When $\mathcal{B}_2$ interacts with* IND-CCA$^{\mathcal{A}}_{\mathsf{KEM}_2}$.Random()*, it exactly simulates Game 2.*

Proof by inlining code and checking pseudocode equivalence.

5.1.4 Game 3: rewrite (case decomposition in oracle O)

Game 3

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_2^* \leftarrow\!\$\ \mathsf{KEM}_2.\mathcal{K}$
$\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    return $\perp$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1, \mathsf{k}_2, (\mathsf{ct}_2, \mathsf{pk}_2, \mathit{Label}))$
else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
    return $F(\mathsf{k}_1, \mathsf{k}_2, (\mathsf{ct}_2, \mathsf{pk}_2, \mathit{Label}))$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

This decomposition simply expands the possible outcomes into separate blocks. $\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$ has four possible states depending on whether $\mathsf{ct}_i = \mathsf{ct}_i^*$, each examined in isolation.

Claim 7. *Game 2 and Game 3 are indistinguishable when* $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 = \mathsf{ct}_2^*$.

Proof: both return $\perp$.

Claim 8. *Game 2 and Game 3 are indistinguishable when* $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and k_2 from ct_2.

Claim 9. *Game 2 and Game 3 are indistinguishable when* $\mathsf{ct}_1 \neq \mathsf{ct}_1^*, \mathsf{ct}_2 = \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and the uniformly chosen k_2^*.

Claim 10. *Game 2 and Game 3 are indistinguishable when* $\mathsf{ct}_1 \neq \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and k_2 from ct_2. Same as the case where $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.

Thus, this decomposition results in an oracle with indistinguishable behaviour.

5.1.5 Game 4: abort if decapsulation of ct_1 matches challenge key k_1^*

Game 4

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \mathsf{k}_1^*) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$
6 : $\mathsf{k} \leftarrow F(\mathsf{k}_1^*, \mathsf{k}_2^*, (\mathsf{ct}_2^*, \mathsf{pk}_2, \mathit{Label}))$
7 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
2 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
3 : **return** $\perp$
4 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
5 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
7 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
8 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
9 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
10 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
11 : abort if $\mathsf{k}_1 = \mathsf{k}_1^*$
12 : $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
13 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
14 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
15 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
16 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
17 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 11. *Game 4 and Game 3 are indistinguishable given that* KEM_1 *is* C2PRI*-secure.*

Proof by contradiction. Assume that $\mathsf{k}_1 = \mathsf{k}_1^*$ in line 11. Since we have $\mathsf{k}_1 = \mathsf{k}_1^*$ when $\mathsf{ct}_1 \neq \mathsf{ct}_1^*$, we have violated the assumption that KEM_1 is C2PRI-secure, therefore it must never be the case that we have $\mathsf{k}_1 = \mathsf{k}_1^*$ in line 11.

5.1.6 Game 5: use random values for keys derived from k_2^*

- Replace two invocations of F with uniformly-chosen keys.

Reduction $\mathcal{B}_5$ is against the (standard) PRF security of PRF $F' : \mathsf{KEM}_2.\mathcal{K} \times (\mathsf{KEM}_1.\mathcal{K} \times \{0,1\}^*)$ defined by

$$F'(\mathsf{k}_2, (\mathsf{k}_1, (\mathsf{ct}_2, \mathsf{pk}_2, \mathit{Label}))) = F(\mathsf{k}_1, \mathsf{k}_2, (\mathsf{ct}_2, \mathsf{pk}_2, \mathit{Label}))$$

Here, $F(a, b)$ represents function inputs, while (a, b) represents association by concatenation.
Oprf is our PRF oracle keyed by k_2.

Game 5

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \mathsf{k}_1^*) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$
6 : $\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$
7 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
2 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
3 : **return** $\perp$
4 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
5 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
7 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
8 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
9 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
10 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
11 : abort if $\mathsf{k}_1 = \mathsf{k}_1^*$
12 : $\mathsf{k} \leftarrow\!\!\$\ F.\mathcal{K}_o$
13 : **return** k
14 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
15 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
16 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
17 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Reduction $\mathcal{B}_5^{\mathsf{Oprf}}$

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \mathsf{k}_1^*) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k}^* \leftarrow \mathsf{Oprf}((\mathsf{k}_1^*, (\mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})))$
6 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k}^*)$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
2 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
3 : **return** $\perp$
4 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
5 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
7 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
8 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
9 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
10 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
11 : abort if $\mathsf{k}_1 = \mathsf{k}_1^*$
12 : $\mathsf{k} \leftarrow \mathsf{Oprf}((\mathsf{k}_1, (\mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})))$
13 : **return** k
14 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
15 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
16 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
17 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

By assuring that $k_1 \neq k_1^*$ in the decapsulation oracle line 11, we are assured that the input queried to Oprf in the decapsulation oracle line 12 is different from the input queried to Oprf in the reduction main code line 5.

Oprf is either invoking $\mathrm{PR}_{F'}$.Real() or $\mathrm{PR}_{F'}$.Random(), where F' is F viewed as a PRF keyed by k_2 as defined in section 2.

By assuring that $\mathsf{k}_1 \neq \mathsf{k}_1^*$ in line 11 of the decapsulation oracle, the input $(\mathsf{k}_1, (\mathsf{ct}_2^*, \mathsf{pk}_2, \mathit{Label}))$ queried to Oprf in line 12 is distinct from the input $(\mathsf{k}_1^*, (\mathsf{ct}_2^*, \mathsf{pk}_2, \mathit{Label}))$ queried in the reduction's main code at line 5. Therefore all inputs to Oprf across all queries are distinct, as required by the PRF security game (Definition 9).

Claim 12. *When $\mathcal{B}_5$ interacts with $\mathrm{PR}_{F'}^{\mathcal{B}_5}$.Real(), it exactly simulates Game 4.*

Proof by inlining code and checking pseudocode equivalence. When Oprf implements Real(), each query $\mathsf{Oprf}(x)$ returns $F'(\mathsf{k}_2^*, x)$. In the main code, $\mathsf{Oprf}((\mathsf{k}_1^*, (\mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})))$ returns $F(\mathsf{k}_1^* \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$, matching the challenge key computation in Game 4. In the decapsulation oracle when $\mathsf{ct}_2 = \mathsf{ct}_2^*$ and $\mathsf{ct}_1 \neq \mathsf{ct}_1^*$, the query returns $F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$, matching Game 4 line 13.

Claim 13. *When $\mathcal{B}_5$ interacts with $\mathrm{PR}_{F'}^{\mathcal{B}_5}$.Random(), it exactly simulates Game 5.*

Proof by inlining code and checking pseudocode equivalence.When Oprf implements Random(), each query returns a uniformly random value from $\mathcal{K}_o$.The challenge key k^* and the oracle responses when $\mathsf{ct}_2 = \mathsf{ct}_2^*$ are therefore uniformly random, matching Game 5.

Thus,

$$|\Pr[\text{Game 4} \Rightarrow 1] - \Pr[\text{Game 5} \Rightarrow 1]| \leq \mathsf{Adv}_{F'}^{\mathrm{PR}}(\mathcal{B}_5)$$

Now that the game matches the desired ending game, we have to "unwind" the oracle back to its original state.

5.1.7 Game 6: use real key instead of random for handling decapsulation queries involving ct_2^*

Game 6

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$
$\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    return $\perp$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    abort if $\mathsf{k}_1 = \mathsf{k}_1^*$
    $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

This game "unwinds" the PRF substitution in the decapsulation oracle while keeping the challenge key random. We construct a reduction $\mathcal{B}_6$ against the PR security of F' (keyed by k_2^*), analogous to $\mathcal{B}_5$ but in reverse: $\mathcal{B}_6$ uses Oprf to answer decapsulation queries when $\mathsf{ct}_2 = \mathsf{ct}_2^*$, and samples the challenge key k^* uniformly at random.

Claim 14. *When $\mathcal{B}_6$ interacts with $\mathrm{PR}_{F'}^{\mathcal{B}_6}$.Real(), it exactly simulates Game 6.*

Claim 15. *When $\mathcal{B}_6$ interacts with $\mathrm{PR}_{F'}^{\mathcal{B}_6}$.Random(), it exactly simulates Game 5.*

Proof of both claims by inlining code and checking pseudocode equivalence.When Oprf implements Real(), queries in the decapsulation oracle return $F(\mathsf{k}_1 \| \mathsf{k}_2^* \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$, matching Game 6.When Oprf implements Random(), queries return uniformly random values, matching Game 5.

Thus,

$$|\Pr[\text{Game } 5 \Rightarrow 1] - \Pr[\text{Game } 6 \Rightarrow 1]| \leq \mathsf{Adv}_{F'}^{\mathrm{PR}}(\mathcal{B}_6)$$

5.1.8 Game 7: don't abort if decapsulation of ct_1 matches challenge key k_1^*

Game 7

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$
$\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    return $\perp$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
  if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    (no abort)
    $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
    return $F(\mathsf{k}_1, \mathsf{k}_2, (\mathsf{ct}_2, \mathsf{pk}_2, \mathit{Label}))$
  else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Follows from C2PRI security of KEM_1 that $\mathsf{k}_1 = \mathsf{k}_1^*$ can never occur, so removing this abort causes no behaviour difference in the oracle.

5.1.9 Game 8: rewrite (consolidate case decomposition in oracle O)

Game 8

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$
$\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then** $\mathsf{k}_2 \leftarrow \mathsf{k}_2^*$
else $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

This consolidation just collapses the possible outcomes into a more compact form. $\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$ has four possible states depending on whether $\mathsf{ct}_i = \mathsf{ct}_i^*$, each examined in isolation.

Claim 16. *Game 7 and Game 8 are indistinguishable when* $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 = \mathsf{ct}_2^*$.

Proof: both return $\perp$.

Claim 17. *Game 7 and Game 8 are indistinguishable when* $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and k_2 from ct_2.

Claim 18. *Game 7 and Game 8 are indistinguishable when* $\mathsf{ct}_1 \neq \mathsf{ct}_1^*, \mathsf{ct}_2 = \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and the uniformly chosen k_2^*.

Claim 19. *Game 7 and Game 8 are indistinguishable when* $\mathsf{ct}_1 \neq \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.

Proof: both return the CK combination of decapsulating k_1 from ct_1 and k_2 from ct_2. Same as the case where $\mathsf{ct}_1 = \mathsf{ct}_1^*, \mathsf{ct}_2 \neq \mathsf{ct}_2^*$.
Thus, this consolidation results in an oracle with indistinguishable behaviour.

5.1.10 Game 9: decapsulate ct_2^* instead of using previous k_2^*

- Remove the unused $\mathsf{k}_2^* \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathcal{K}$.
- Collapse the if-else relating to whether $\mathsf{ct}_2 = \mathsf{ct}_2^*$.

Game 9

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$

$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$

$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$

$(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$

$\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$

return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**

  return $\perp$

$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$

$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$

return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 20. *Game 8 and Game 9 are indistinguishable assuming the* IND-CCA*-security of* KEM_2.

5.1.11 Conclusion

Game 9 is the ending game.

By accumulating the advantages from every hop, we can conclude:

Theorem 2.

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{CK}[\mathsf{KEM}_1,\mathsf{KEM}_2,F]}(\mathcal{A}) \leq 2\delta_{\mathsf{KEM}_2} + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2}(\mathcal{B}_2) + \mathsf{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1}(\mathcal{B}_4) + \mathsf{Adv}^{\text{PR}}_{F'}(\mathcal{B}_5) + \mathsf{Adv}^{\text{PR}}_{F'}(\mathcal{B}_6) + \mathsf{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1}(\mathcal{B}_7) + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2}(\mathcal{B}_9)$$

5.2 Reduction to IND-CCA security of KEM_1 and PRF security in the standard model

Intuitively, this is the scenario where we assume KEM_2 to be completely compromised and the security of the CK hybrid rests on KEM_1. Unlike the scenario above, here we do not make any assumptions at all on KEM_2.

Starting assumptions:

- KEM_1 is IND-CCA-secure, and
- F is a hash function that acts as a secure PRF when viewed as a keyed function with k_1 as the key.

5.2.1 Starting and ending games

Starting game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{CK}}$.Real()

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \mathsf{k}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k} \leftarrow F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
6 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
2 : **return** $\perp$
3 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
4 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
5 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Ending game = $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{CK}}$.Random()

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k} \leftarrow\!\!\$\ F.\mathcal{K}_o$
6 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
2 : **return** $\perp$
3 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
4 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
5 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

5.2.2 Game 1: use k_1^* instead of decapsulating ct_1^*

Game 1

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \mathsf{k}_1^*) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k} \leftarrow F(\mathsf{k}_1^* \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
6 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
2 : **return** $\perp$
3 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
4 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then** $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
5 : **else** $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 21. *Starting game and Game 1 are indistinguishable assuming the correctness of* KEM_1.

5.2.3 Game 2: use random key for k_1 in challenge instead of real

We give the reduction $\mathcal{B}_1$ which is an adversary against the IND-CCA security of KEM_1 and which simulates either game 0 or game 1 to the KEM IND-CCA adversary $\mathcal{A}$. $\mathcal{B}_1$ works by using the public key, challenge ciphertext ct_1^*, and challenge shared secret k_1^* from the KEM_1 challenger; it simulates the KEM_2 computations itself. To answer decapsulation queries, $\mathcal{B}_1$ uses the decapsulation oracle from the KEM_1 challenger. If the challenge ciphertext ct_1^* is used in the decapsulation query for the hybrid KEM, the reduction $\mathcal{B}_1$ uses the same challenge shared secret k_1^* as its alleged decapsulation; there is no concern that $\mathcal{B}_1$ will pass the prohibited ct_1^* to the KEM_1 decapsulation oracle.

Game 2

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_1^* \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathcal{K}$
$\mathsf{k} \leftarrow F(\mathsf{k}_1^* \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then** $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
else $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Reduction $\mathcal{B}_1^{\mathsf{Okem1}, \mathcal{A}}(\mathsf{pk}_1, \mathsf{ct}_1^*, \mathsf{k}_1^*)$

$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k} \leftarrow F(\mathsf{k}_1^* \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then** $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
else $\mathsf{k}_1 \leftarrow \mathsf{Okem1}(\mathsf{ct}_1)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 22. *When $\mathcal{B}_1$ interacts with* $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}_1}.\text{Real}()$*, it exactly simulates Game 1.*

Proof by inlining code and checking pseudocode equivalence.

Claim 23. *When $\mathcal{B}_1$ interacts with* $\text{IND-CCA}^{\mathcal{A}}_{\mathsf{KEM}_1}.\text{Random}()$*, it exactly simulates Game 2.*

Proof by inlining code and checking pseudocode equivalence.

5.2.4 Game 3: rewrite (case decomposition in oracle O)

Game 3

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_1^* \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathcal{K}$
$\mathsf{k} \leftarrow F(\mathsf{k}_1^* \| \mathsf{k}_2 \| \mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
  if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
    return $\perp$
  else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
  if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
    $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
  else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

5.2.5 Game 4: use random values for keys derived from k_1^*

Reduction $\mathcal{B}_4$ is against the (standard) PRF security of $F'' : \mathsf{KEM}_1.\mathcal{K} \times (\mathsf{KEM}_2.\mathcal{K} \times \{0,1\}^*)$ as defined in section 2, i.e., F viewed as a PRF keyed by k_1:

$$F''(\mathsf{k}_1, (\mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})) = F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$$

Game 4

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \_) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \mathsf{k}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k}_1^* \leftarrow\!\$\ \mathsf{KEM}_1.\mathcal{K}$
6 : $\mathsf{k}^* \leftarrow\!\$\ F.\mathcal{K}_o$
7 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
2 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
3 : **return** $\perp$
4 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
5 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
7 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
8 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
9 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
10 : $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
11 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
12 : $\mathsf{k} \leftarrow\!\$\ F.\mathcal{K}_o$
13 : **return** k
14 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
15 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
16 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
17 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Reduction $\mathcal{B}_4^{\mathsf{Oprf}}$

1 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
2 : $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
3 : $(\mathsf{ct}_1^*, \_) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
4 : $(\mathsf{ct}_2^*, \mathsf{k}_2^*) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
5 : $\mathsf{k}^* \leftarrow \mathsf{Oprf}((\mathsf{k}_2^*, (\mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label})))$
6 : **return** $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k}^*)$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

1 : **if** $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
2 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
3 : **return** $\perp$
4 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
5 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
6 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
7 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
8 : **else** $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
9 : **if** $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
10 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
11 : $\mathsf{k} \leftarrow \mathsf{Oprf}((\mathsf{k}_2, (\mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})))$
12 : **return** k
13 : **else** $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
14 : $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
15 : $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
16 : **return** $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Oprf is a PRF oracle for F'' keyed by k_1^*. The input $(\mathsf{k}_2^*, (\mathsf{ct}_2^* \| \mathsf{pk}_2 \| \mathit{Label}))$ queried to Oprf in the reduction's main code at line 5 is distinct from the input $(\mathsf{k}_2, (\mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label}))$ queried in the decapsulation oracle at line 11, since when $\mathsf{ct}_1 = \mathsf{ct}_1^*$ and $\mathsf{ct}_2 \neq \mathsf{ct}_2^*$, either $\mathsf{ct}_2 \neq \mathsf{ct}_2^*$ (so the second component differs) or $\mathsf{k}_2 \neq \mathsf{k}_2^*$ (since the inputs to $\mathsf{KEM}_2.\mathsf{Decaps}$ differ). Therefore all inputs to Oprf across all queries are distinct, as required by the PRF security game (Definition 9).

Claim 24. *When $\mathcal{B}_4$ interacts with $\mathrm{PR}_{F''}^{\mathcal{B}_4}.\mathrm{Real}()$, it exactly simulates Game 3.*

Claim 25. *When $\mathcal{B}_4$ interacts with $\mathrm{PR}_{F''}^{\mathcal{B}_4}.\mathrm{Random}()$, it exactly simulates Game 4.*

Proof of both claims by inlining code and checking pseudocode equivalence. When Oprf implements Real(), each query $\mathsf{Oprf}(x)$ returns $F''(\mathsf{k}_1^*, x) = F(\mathsf{k}_1^* \| x)$, matching Game 3. When Oprf implements Random(), each query returns a uniformly random value from $\mathcal{K}_o$, matching Game 4.

Thus,

$$|\Pr[\text{Game } 3 \Rightarrow 1] - \Pr[\text{Game } 4 \Rightarrow 1]| \leq \mathsf{Adv}_{F''}^{\mathrm{PR}}(\mathcal{B}_4)$$

Now we have to "unwind" a bunch of changes to get to the ending game.

5.2.6 Game 5: use real key instead of random for handling decapsulation queries involving ct_2^*

Game 5

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_1^* \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathcal{K}$
$\mathsf{k}^* \leftarrow\!\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct}_2 = \mathsf{ct}_2^*$ **then**
  if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
    return $\perp$
  else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
else $(\mathsf{ct}_2 \neq \mathsf{ct}_2^*)$
  if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then**
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$
  else $(\mathsf{ct}_1 \neq \mathsf{ct}_1^*)$
    $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
    $\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
    return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

This game "unwinds" the PRF substitution in the decapsulation oracle while keeping the challenge key random. We construct a reduction $\mathcal{B}_5$ against the PR security of F'' (keyed by k_1^*), analogous to $\mathcal{B}_4$ but in the reverse direction: $\mathcal{B}_5$ uses Oprf to answer decapsulation queries when $\mathsf{ct}_1 = \mathsf{ct}_1^*$, and samples the challenge key k^* uniformly at random.

Claim 26. *When $\mathcal{B}_5$ interacts with $\mathrm{PR}_{F''}^{\mathcal{B}_5}$.Real(), it exactly simulates Game 5.*

Claim 27. *When $\mathcal{B}_5$ interacts with $\mathrm{PR}_{F''}^{\mathcal{B}_5}$.Random(), it exactly simulates Game 4.*

Proof of both claims by inlining code and checking pseudocode equivalence. When Oprf implements Real(), queries in the decapsulation oracle return $F(\mathsf{k}_1^* \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$, matching Game 5. When Oprf implements Random(), queries return uniformly random values, matching Game 4.

Thus,

$$|\Pr[\text{Game } 4 \Rightarrow 1] - \Pr[\text{Game } 5 \Rightarrow 1]| \leq \mathsf{Adv}_{F''}^{\mathrm{PR}}(\mathcal{B}_5)$$

5.2.7 Game 6: rewrite (consolidate case decomposition in oracle O)

Game 6

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}_1^* \leftarrow\!\$\ \mathsf{KEM}_1.\mathcal{K}$
$\mathsf{k}^* \leftarrow\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
if $\mathsf{ct}_1 = \mathsf{ct}_1^*$ **then** $\mathsf{k}_1 \leftarrow \mathsf{k}_1^*$
else $\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

5.2.8 Game 7: decapsulate ct_1^* instead of using previous k_1^*

Game 7

$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}()$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}()$
$(\mathsf{ct}_1^*, \_) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(\mathsf{pk}_1)$
$(\mathsf{ct}_2^*, \_) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(\mathsf{pk}_2)$
$\mathsf{k}^* \leftarrow\!\$\ F.\mathcal{K}_o$
return $\mathcal{A}^{\mathsf{O}}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), \mathsf{k})$

$\mathsf{O}((\mathsf{ct}_1, \mathsf{ct}_2))$

if $(\mathsf{ct}_1, \mathsf{ct}_2) = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ **then**
  return $\perp$
$\mathsf{k}_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(\mathsf{sk}_1, \mathsf{ct}_1)$
$\mathsf{k}_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(\mathsf{sk}_2, \mathsf{ct}_2)$
return $F(\mathsf{k}_1 \| \mathsf{k}_2 \| \mathsf{ct}_2 \| \mathsf{pk}_2 \| \mathit{Label})$

Claim 28. *Game 6 and Game 7 are indistinguishable assuming the* IND-CCA*-security of* KEM_1.

5.2.9 Conclusion

Game 7 is the ending game.

By accumulating the advantages from every hop, we can conclude:

Theorem 3.

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{CK}[\mathsf{KEM}_1,\mathsf{KEM}_2,F]}(\mathcal{A}) \leq 2\delta_{\mathsf{KEM}_1} + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1}(\mathcal{B}_1) + \mathsf{Adv}^{\text{PR}}_{F''}(\mathcal{B}_4) + \mathsf{Adv}^{\text{PR}}_{F''}(\mathcal{B}_5) + \mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1}(\mathcal{B}_7)$$

where δ_{KEM_1} is the correctness bound for KEM_1.

6 Concrete Instantiations

In this section, we provide concrete bounds for instantiations of the CK framework with some specific choices of KEM_1, KEM_2, and F.

6.1 Components

6.1.1 ML-KEM

ML-KEM has three parameter sets; ML-KEM-512, ML-KEM-768, and ML-KEM-1024 which are assumed to have classical and post-quantum IND-CCA security equivalent to performing a key search on a 128-bit, 192-bit, and 256-bit block cipher respectively [11]. The C2PRI security of all parameter sets of ML-KEM is given in [7] as:

$$\mathsf{Adv}^{\text{C2PRI}}_{\mathsf{ML-KEM}}(\mathcal{B}) \leq \frac{q+1}{|\mathcal{K}|} \tag{1}$$

where q is the number of queries afforded the adversary, and $|\mathcal{K}|$ is the size of the KEM output key space, which for all parameter sets of ML-KEM is 2^{256}. For the purposes of obtaining concrete values in this section, we assume a single-query attack setting, which yields a C2PRI advantage of 2^{-255} for all ML-KEM parameter sets.

The following table gives concrete values for the ML-KEM-related advantage terms from [11] as used in the security advantage inequalities in Theorem 2 and 3 above.

Param set	$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{ML-KEM}}(\mathcal{B}_1)$	$\delta_{\mathsf{ML-KEM}}$	$\mathsf{Adv}^{\text{C2PRI}}_{\mathsf{ML-KEM}}(\mathcal{B})$
ML-KEM-512	2^{-128}	$2^{-138.8}$	2^{-255}
ML-KEM-768	2^{-192}	$2^{-164.8}$	2^{-255}
ML-KEM-1024	2^{-256}	$2^{-174.8}$	2^{-255}

6.1.2 RSA-OAEP

RSA-OAEP has been shown to be IND-CCA2 secure under multiple assumptions ([5], [9]).RSA-OAEP is perfectly-correct in terms of δ-correctness [10].

6.1.3 SHA3-256

In the security analyses of CK, SHA3-256 is required to have PR (pseudorandomness) security as defined in Definition 9 under two keying conventions (section 2): as F' (keyed by the KEM_2 shared secret k_2, as used in theorem 2) and as F'' (keyed by the KEM_1 shared secret k_1, as used in theorem 3).

SHA-3 and other sponge constructions have been shown to be indifferentiable from random oracles ([4], [1]) and usable as secure PRFs [3].

When instantiating CK using ML-KEM as KEM_1, the entire hybrid construction is at least as IND-CCA-secure as ML-KEM.

References

[1] G. Alagic, J. Carolan, C. Majenz, and S. Tokat. The sponge is quantum indifferentiable, 2025. URL: `https://eprint.iacr.org/2025/731.pdf`.

[2] M. Barbosa, D. Connolly, J. D. Duarte, A. Kaiser, P. Schwabe, K. Varner, and B. Westerbaan. X-wing: The hybrid kem you've been looking for. *Cryptology ePrint Archive*, 2024.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Cryptographic sponge functions, Jan. 2011. URL: `https://keccak.team/files/CSF-0.1.pdf`.

[4] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Berlin, Heidelberg, Apr. 2008. doi:10.1007/978-3-540-78967-3_11.

[5] N. Cao, A. O'Neill, and M. Zaheri. *Toward RSA-OAEP without random oracles*, pages 279–308. Lecture notes in computer science. Springer International Publishing.

[6] D. Connolly, R. Barnes, and P. Grubbs. Hybrid post-quantum key encapsulation mechanisms. draft-irtf-cfrg-hybrid-kems, 2026. URL: https://datatracker.ietf.org/doc/draft-irtf-cfrg-hybrid-kems/.

[7] D. Connolly, K. Hövelmanns, A. Hülsing, S. Kousidis, and M. Meijers. Starfighters—on the general applicability of x-wing. Cryptology ePrint Archive, Paper 2025/1397, 2025. URL: https://eprint.iacr.org/2025/1397.

[8] D. Connolly, P. Schwabe, and B. Westerbaan. X-Wing: general-purpose hybrid post-quantum KEM. draft-connolly-cfrg-xwing-kem, 2024. URL: https://datatracker.ietf.org/doc/draft-connolly-cfrg-xwing-kem/.

[9] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. 17:81–104.

[10] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, Mar. 2004. doi:10.1007/s00145-002-0204-y.

[11] NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. FIPS 203 (Initial Public Draft), Aug. 2023. doi:10.6028/NIST.FIPS.203.ipd.

[12] M. Ounsworth, J. Gray, M. Pano, J. Klaußner, and S. Fluhrer. Composite ML-KEM for use in X.509 Public Key Infrastructure. draft-ietf-lamps-pq-composite-kem, 2026. URL: https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-kem/12.

[13] B. Westerbaan and D. Stebila. X25519Kyber768Draft00 hybrid post-quantum key agreement. draft-westerbaan-cfrg-hpke-xyber768d00, 2023. URL: https://datatracker.ietf.org/doc/draft-tls-westerbaan-xyber768d00/03/.

[14] B. Westerbaan and C. A. Wood. X25519Kyber768Draft00 hybrid post-quantum KEM for HPKE. draft-westerbaan-cfrg-hpke-xyber768d00, 2023. URL: https://datatracker.ietf.org/doc/draft-westerbaan-cfrg-hpke-xyber768d00/.