The Best of Both KEMs: Securely Combining KEMs in Post-Quantum Hybrid Schemes

Gorjan Alagic[1,2], Fahran Bajaj[3], and Aybars Kocoglu[3]

[1]QuICS, University of Maryland Institute for Advanced Computer Studies
[2]National Institute of Standards and Technology
[3]Department of Mathematics, University of Maryland

August 8, 2025

Abstract

Transitioning secure information systems to post-quantum cryptography (PQC) comes with certain risks, such as the potential for switching to PQC schemes with as yet undiscovered vulnerabilities. Such risks can be mitigated by combining multiple schemes in such a way that the resulting *hybrid* scheme is secure provided *at least one* of the ingredient schemes is secure. In the case of key-encapsulation mechanisms (KEMs), this approach is already in use in practice, where the PQC scheme ML-KEM is combined with "traditional" X25519 key exchange.

Combining multiple KEMs to construct a single hybrid KEM is largely straightforward, except for the crucial choice of how to derive the final shared secret key. A generic method for doing this in a manner that preserves IND-CCA security is to include the keys and ciphertexts of all ingredient KEMs in an appropriate key derivation step. In the specialized X-Wing construction, one instead relies on a special property of ML-KEM to avoid including its ciphertext in key derivation.

In this work, we show that this optimization can be done in a more general setting. Specifically, when combining multiple KEMs one need not include the ciphertext of any KEM that satisfies ciphertext second preimage resistance (C2PRI)—provided the key combination step is performed using a split-key pseudorandom function. We also prove that any KEM constructed from a certain set of Fujisaki-Okamoto (FO) transforms satisfies C2PRI in the random oracle model. This applies to KEMs such as BIKE, Classic McEliece, HQC, and ML-KEM.

1 Introduction

The potential development of large-scale fault-tolerant quantum computers poses a dramatic threat to most public-key cryptographic schemes in use today. This threat has necessitated a transition to post-quantum cryptography (PQC). While most public-key schemes in widespread use today are still insecure against quantum adversaries, they (and their up-to-date implementations) have a long track record of security against classical attacks in the face of years of rigorous theoretical and empirical analysis. The same cannot necessarily be said for their proposed post-quantum replacements. Thus, there is an apparent tradeoff between present and future security guarantees.

In light of this, a popular strategy has been to combine schemes into hybrid constructions that are secure so long as one ingredient scheme remains unbroken. Combining well-established, "traditional" algorithms with newer post-quantum ones can then help with security in the face of both potential vulnerabilities in the new scheme and potential quantum attacks against the traditional one. One can also combine multiple post-quantum schemes to try to eliminate reliance on a single (perhaps not fully established) computational assumption. Using hybrid protocols is explicitly recommended by European agencies [Fed22, Fre24].

Due to the potential for "store-now, decrypt-later" attacks, key establishment has received significant focus in the post-quantum transition, and hence also in hybrid constructions. Hybrid post-quantum key-establishment has been incorporated into mainstream applications like Chrome [O'B23] and Signal [BJKS24]. Hybrid key-establishment schemes typically take the form of a key-encapsulation mechanism (KEMs), owing to the fact that all candidate post-quantum key-establishment protocols are

KEMs. In fact, for most applications it suffices to consider the case where all ingredients are KEMs[1]. The standard approach (and the one we consider here) is then to combine these ingredient KEMs in parallel. Specifically, given two KEMs Π_1 and Π_2, one creates a hybrid KEM Π whose encapsulation keys, decapsulation keys, and ciphertexts are concatenations of the corresponding elements of Π_1 and Π_2. What then remains is to choose how to generate the session key of Π.

As shown by Giacon, Heuer, and Poettering, generating the session key in a hybrid KEM is not as straightforward as one might expect [GHP18]. For example, choosing $k = \mathsf{KDF}(k_1, k_2)$ as the session key corresponding to a ciphertext $c = (c_1, c_2)$ does not generically retain IND-CCA security—no matter what key derivation function KDF is used. Specifically, if Π_1 is broken to such an extent that an adversary can find a ciphertext $c_1' \neq c_1$ that also decapsulates to k_1, then they could perform a decapsulation query on (c_1', c_2) and recover the honest session key. As a result, the generic combiners in [GHP18] include all relevant ciphertexts, computing the session key via

$$k = W(k_1, ..., k_n, c_1, ..., c_n) \tag{1}$$

for a certain *core function* W. They then prove that, provided W is *split-key pseudorandom*, the resulting hybrid KEM will preserve IND-CCA security. The split-key pseudorandom property ensures that W behaves like a PRF so long as at least one key k_j is sampled uniformly at random and hidden from the adversary. Several constructions of split-key PRFs are known [GHP18, DFH22].

For many KEMs, finding a colliding ciphertext as above yields a hash function collision, and is thus arguably much harder than breaking IND-CCA. For example, this appears to be the case for KEMs constructed via the Fujisaki-Okamoto (FO) transform [FO99, FO13, BCD+24]. The *Quantum Superiority Fighter* (QSF) framework of [BCD+24] takes advantage of this to argue that, for combining ML-KEM with X25519, one can use a more efficient combiner than (1), specifically

$$k = W(k_1, k_2, c_2) \tag{2}$$

where c_2 is the X25519 "ciphertext" and W is instantiated with SHA3-256. They prove that this construction preserves IND-CCA under the assumption that ML-KEM satisfies a property they call *ciphertext second preimage resistance* (C2PRI). C2PRI requires an adversary, given an honestly generated public key, secret key, ciphertext, and session key, to find a new ciphertext decapsulating to the same session key. They prove that ML-KEM is C2PRI (against classical adversaries) if the underlying hash functions are modeled as random oracles. They then argue that, due to the significant size of the ML-KEM ciphertext, leaving it out of the core function leads to a notable improvement in performance.

In this work, we generalize the above results as follows. First, we show that combining two KEMs using (2) generically preserves IND-CCA provided the first KEM satisfies C2PRI and W is split-key pseudorandom. Second, we show that choosing the combiner $k = W(k_1, k_2)$ where W is a dual PRF also preserves IND-CCA provided *both* KEMs satisfy C2PRI. In either case, a natural example is to simply choose W to be a hash function; a straightforward application of existing results shows that this is secure in the (quantum-accessible) random oracle model. Finally, we show that a wide range of KEMs, including BIKE, Classic McEliece, HQC, and ML-KEM, satisfy the C2PRI property against quantum adversaries.

1.1 Summary of results

We now briefly summarize our technical results.

Security of KEM combiners. Let KEM_1 and KEM_2 be two δ-correct KEMs and Π a KEM constructed from KEM_1 and KEM_2 using some "combiner" function family W. We are concerned with establishing IND-CCA security of Π against either classical probabilistic polynomial-time (PPT) adversaries or quantum polynomial-time (QPT) adversaries.

The combined KEM Π is constructed as follows. Key generation, encapsulation, and decapsulation run the corresponding algorithms of KEM_1 and KEM_2 in parallel and concatenate the results appropriately to form the encapsulation key, decapsulation key, and ciphertext of Π. At the end of encapsulation and decapsulation, the session key of Π is computed by $k = W(k_1, k_2, c_1, c_2)$. Our focus

[1] Hybrid protocols involving quantum key distribution are a notable exception.

is on the security of Π in the setting where, for efficiency purposes, W ignores one or both of its ciphertext inputs.

We first instantiate W with a split-key PRF $F(k_1, k_2, c_2)$, ignoring the first ciphertext. The split-key property ensures that, provided either k_1 or k_2 is treated as the key, F is pseudorandom on the remaining two inputs. We show that the hybrid KEM Π is then CCA-secure provided (i.) KEM_1 is IND-CCA, or (ii.) KEM_2 is IND-CCA and KEM_1 is ciphertext second preimage resistant (C2PRI).

Theorem 1 (informal)**.** *Let $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$, and suppose $\mathcal{A}$ is a QPT adversary against the IND-CCA security of Π. Then there exist QPT adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\text{split-key-PRF}}_{F,\mathcal{D}} + \delta)\,.$$

Moreover, there also exist QPT adversaries $\mathcal{B}'$ and $\mathcal{C}'$ such that

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1,\mathcal{B}'} + \mathrm{Adv}^{\text{split-key-PRF}}_{F,\mathcal{C}'} + \delta)\,.$$

Next, we instantiate W with a dual PRF $F(k_1, k_2)$, now ignoring both ciphertexts. The dual-PRF property ensures that, provided either k_1 or k_2 is treated as the key, F is pseudorandom on its remaining input. We show that the hybrid KEM Π is then CCA-secure provided KEM_1 is IND-CCA and KEM_2 is C2PRI, or vice-versa.

Theorem 2 (informal)**.** *Let $W(k_1, k_2, c_1, c_2) = F(k_1, k_2)$, and suppose $\mathcal{A}$ is a QPT adversary against the IND-CCA security of Π. Then there exist QPT adversaries $\mathcal{B}$ and $\mathcal{C}$ such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{dual-PRF}}_{F,\mathcal{C}} + \delta)$$

Theorem 1 and Theorem 2 are proved in Section 3. In both theorems, all adversaries make only classical queries to their challenge oracles. This is the appropriate model for post-quantum security, as these oracles implement secretly-keyed functions, and (in practice) the secret key presumably resides only on an honest user's classical machine. In both theorems, the derived QPT adversaries $\mathcal{B}, \mathcal{C}, \mathcal{B}', \mathcal{C}'$ and $\mathcal{D}$ make at most one additional query to their oracles than the number of queries made by $\mathcal{A}$ to the decapsulation oracle of Π, and each adversary runs with at most constant-factor overhead. As the reductions in our proofs are classical, these adversaries are PPT if $\mathcal{A}$ happens to be PPT. This case is important for settings in which one of the two ingredient KEMs is not post-quantum.

Further, we summarize how to extend our construction and security proofs to combine n different KEMs. Briefly put, one may continue running each ingredient KEM in parallel and derive the hybrid session key by evaluating a split-key PRF over all ingredient session keys, as well as the ciphertexts of each KEM that is not believed to be C2PRI. In Section 3.3, we sketch how the proof of Theorem 1 can be extended to this setting:

Theorem 3 (informal)**.** *Let $W(k_1, ..., k_n, c_1, ..., c_n) = F(k_1, ..., k_n, c_{\ell+1}, c_{\ell+2}, ..., c_n)$, and suppose $\mathcal{A}$ is a QPT adversary against the IND-CCA security of Π, and pick i between 1 and n. Then there exist QPT adversaries $\mathcal{B}_1, ..., \mathcal{B}_\ell$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2\left(\left(\sum_{j=1, j\neq i}^{\ell} \mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_j,\mathcal{B}_j}\right) + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_i,\mathcal{C}} + \mathrm{Adv}^{\text{split-key-PRF}}_{F,\mathcal{D}} + \delta\right)\,.$$

Again, each derived adversary makes at most one additional oracle querie than $\mathcal{A}$ and runs with constant-factor overhead.

Post-quantum split-key PRFs. Several constructions of functions that are split-key pseudorandom against QPTs are known. These include the PRF-Then-XOR construction in [GHP18] and the function $F(k_1, \ldots, k_n, x) = H(g(k_1, \ldots, k_n), x)$ for a function g with a certain "well-spread" property and a random oracle H [DFH22]. In the special case where g is the identity function, a straightforward application of existing results[2] shows that a random oracle is a split-key PRF, as follows.

[2]Specifically, this follows from the fact that $F_k(x) := H(k, x)$ is a PRF in the QROM. Although we are quite certain this has been shown before, we have been unable to find a reference. A proof using the O2H lemma is given in Appendix A.

Theorem 4. *Let H be a random oracle and let $\mathcal{A}$ be a QPT adversary against the split-key pseudorandomness of $F(k_1, \ldots, k_n, x) := H(k_1 || \ldots || k_n || x)$ making at most q quantum queries to H, where $\mathcal{K}_i$ denotes the i^{th} keyspace. Then,*

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{3}$$

The adversary here is allowed quantum queries to H, since H is in practice instantiated with a public hash function. This adversary is also allowed to make queries to the (keyed) evaluation oracle that receives $n-1$ keys and one input, but the number of such queries does not enter into the bound. Indeed, a query to H with the incorrect missing key is equally useful to an adversary who has made one evaluation query as it is to an adversary who has the entire evaluation table. Using the above theorem, we outline how NIST's recommended key-derivation functions from [BCD20], which include popular methods like HKDF, can be shown to be split-key PRFs in the random oracle model.

Classical and quantum bounds for C2PRI. Finally, we show that a class of post-quantum KEMs (constructed via a certain type of Fujisaki-Okamoto [FO] transform) satisfy ciphertext second preimage resistance (C2PRI) in the (quantum) random oracle model. Recall that, in the C2PRI experiment, an adversary is given an honestly generated public key, secret key, ciphertext, and session key, and then asked to find a new ciphertext decapsulating to the same session key.

Below we use the notation of [HHK17] to refer to certain types of FO transforms, namely $\mathsf{U}^{\perp}$, $\mathsf{U}^{\not\perp}$, $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$. These transforms involve the use of a single hash function, which we refer to below as H.

Theorem 5. *Let KEM be a $\mathsf{U}^{\perp}$, $\mathsf{U}^{\not\perp}$, $\mathsf{U}^{\perp}_m$ or $\mathsf{U}^{\not\perp}_m$ KEM with keyspace $\mathcal{K}$ and underlying hash function H modeled as a random oracle. Let $\mathcal{C}$ be an adversary against C2PRI for KEM. If $\mathcal{C}$ makes at most q classical queries to H, then*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|}$$

If $\mathcal{C}$ instead makes at most q quantum queries to H, then

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

The proof of Theorem 5 is given in Sections 5 and 6. For the case of classical queries, our proof is a straightforward adaptation of the proof that ML-KEM is classically C2PRI [BCD+24]. This case is particularly relevant in the typical hybrid PQC construction, where security against classical adversaries is based on IND-CCA of a traditional KEM and (classical) C2PRI of a post-quantum KEM. For the case of quantum queries, our proof technique makes key use of the One-Way to Hiding Lemma [AHU21].

Theorem 5 shows directly that some well-known KEMs (like BIKE) satisfy C2PRI. We also show how the proofs can be adapted to show C2PRI for other prominent KEMs, namely Classic McEliece, HQC, and ML-KEM. HQC is an interesting case, as it is known to not be ciphertext *collision-resistant* [KSW24]. We provide specific advantage bounds for both classical queries and quantum queries.

Implications. As discussed by Barbosa et al. in [BCD+24] in the case of ML-KEM, many post-quantum KEMs use long ciphertexts, allowing for noticeable improvements in performance when they are not unnecessarily included as an input to the hash function in session-key derivation. For their hybrid scheme X-Wing, they show that omitting ML-KEM-768 ciphertexts in key derivation empirically resulted in 8% and 9% performance gains in encapsulation and decapsulation, respectively. As the Quantum Superiority Fighter (QSF) framework which they used to construct X-Wing is very similar in structure to our combiner, we expect similar performance gains when using our combiner with X25519 and ML-KEM-768 as ingredients. The standard parameter sets of Classic McEliece specified in [McE22] admit ciphertext sizes ranging from 96 to 208 bytes, suggesting more modest performance improvements when it is an ingredient of our combiner. However, several other post-quantum KEMs of interest have ciphertext sizes close to or larger than that of ML-KEM-768, including ML-KEM-512 and ML-KEM-1024 with respective ciphertext sizes of 768 and 1568 bytes [ML-24], BIKE with ciphertext

sizes ranging from 1573 to 5154 bytes [BIK24], and HQC with ciphertext sizes ranging from 4497 to 14485 bytes [HQC24]. Creating hybrid schemes with these KEMs using our combiner is likely to yield substantial improvements over using a combiner that hashes the ciphertext(s) of the post-quantum ingredient KEM(s).

1.2 Related work

Constructions for blackbox KEM combiners that make no assumptions on the ingredient KEMs are given in [GHP18, BBF+19, CP20, XGL+21]. Most of these run each KEM in parallel and input each ciphertext into the final KDF, though the XOR-then-MAC approach of [BBF+19] and the construction from [XGL+21] use distinctive strategies. The authors of [HDV21] and [CLM25] present constructions that transform multiple IND-CPA secure public-key encryption schemes into a single IND-CCA secure robust hybrid KEM. These methods create a hybrid PKE scheme before applying the $\mathsf{U}^{\not\perp}$ transform, so our work implies that hybrid KEMs created in this way are ciphertext second preimage resistant. Alwen et al. consider the problem of combining multi-recipient KEMs (mKEMs) [AHK+23]. They note that the FO transform endows KEMs with a property they call *collision resistance* and construct an mKEM combiner with core function $F(k_1, k_2)$, where F is a dual-PRF. They prove that this construction is IND-CCA secure provided both ingredient mKEMs are collision-resistant and one is IND-CCA. Their requirement of collision resistance is strictly stronger than ours of ciphertext second preimage resistance because it allows an adversary, given a public/secret key pair, to output any two ciphertexts decapsulating to the same session key. This collision resistance property and when it holds is further explored as the LEAK-BIND-K,PK-CT property [CDM24] and [KSW24]. Dual-PRFs are studied in [Bel06, ADK+22, BBGS23].

Recent related work The present paper was submitted to TCC 2025, except for Sections 3.3 and 4, which were added shortly after. After that, the closely related independent work [CHH+25] appeared on eprint. Then, the present paper was also posted on eprint, with no modifications except the addition of this paragraph.

We now briefly describe the main differences between the present paper and [CHH+25], as we see them. The work [CHH+25] looks at binding and multi-user properties for their hybrid KEM constructions while our work does not. Our work looks at combining many KEMs (Section 3.3) and various KDF choices (Section 4) while [CHH+25] does not. For KEM combiners that do not include any ciphertexts, the main results are the same. For combiners involving only one ciphertext, the results are incomparable (our result is proved in the standard model but relies on a KEM being IND-CCA, while theirs allows for the weaker OW-PCVA security notion but requires the classical ROM). In terms of showing C2PRI for specific KEMs, [CHH+25] gives better bounds; they analyze ML-KEM, Classic McEliece, NTRU, FrodoKEM, and a version of HQC that includes the salt in the key derivation, while we analyze ML-KEM, Classic McEliece, HQC, and BIKE. [CHH+25] also considers the salted and key confirmation variants of the FO transform, while we do not.

1.3 Acknowledgments

The authors thank the University of Maryland MathQuantum Research Training Group, supported by NSF grant DMS-2231533, for making this work possible and for their continual support. FB and AK thank Mike Hamburg for his help with formulating the quantum second preimage resistance proofs, and GA's graduate students for their insights during weekly research discussions. GA's work is partly supported by NSF award CNS-21547.

2 Preliminaries

2.1 Basic notation

Below we list the notation that we use throughout this paper:

- $a||b$ denotes the concatenation of a and b
- $a \leftarrow b$ denotes deterministically assigning the value of b to the variable a.

- $a \leftarrow\!\!\$\ A$ denotes sampling a uniformly at random from a set A.
- $a \leftarrow \mathcal{A}(I)$ denotes that the deterministic algorithm $\mathcal{A}$ is run with input I, resulting in output a. Similarly, $b \leftarrow\!\!\$\ \mathcal{B}(I)$ denotes that the randomized algorithm $\mathcal{B}$ is run with input I, resulting in output b.
- $\mathcal{A}^{O(\cdot)}(I)$ denotes that algorithm $\mathcal{A}$ is run with input I and access to oracle O.
- $A[a]$ denotes accessing table A at position a. $A[\cdot]$ denotes all positions of table A.
- $\triangleright\, \mathrm{G}_1$ means that the relevant line of pseudocode is only executed in game G_1. Similarly, $\triangleright\, \mathrm{G}_1 - \mathrm{G}_3$ means that the relevant line is only executed in games G_1 through G_3.
- $b \leftarrow \mathrm{G}_0$ denotes the security game G_0 running and returning bit b.
- $\{\mathcal{X} \to \mathcal{Y}\}$ refers to the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

2.2 Key-Encapsulation Mechanisms (KEMs)

A *key-encapsulation mechanism* KEM with session-key space $\mathcal{K}$, a public-key space $\mathcal{PK}$, a secret-key space $\mathcal{SK}$, and a ciphertext space $\mathcal{C}$ is a triple

$$\mathsf{KEM} = (\mathsf{KEM.KeyGen}(1^n), \mathsf{KEM.Encaps}, \mathsf{KEM.Decaps}) \tag{4}$$

of algorithms called key generation, encapsulation, and decapsulation, respectively. These algorithms must satisfy the following.

- KeyGen is a randomized algorithm that, given a security parameter n in unary, outputs a public key $pk \in \mathcal{PK}$ and a secret key $sk \in \mathcal{SK}$. This is expressed as $(pk, sk) \leftarrow\!\!\$\ \mathrm{KEM.KeyGen}(1^n)$.
- Encaps is a randomized algorithm that, given $pk \in \mathcal{PK}$, outputs a session key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$. This is expressed as $(k, c) \leftarrow\!\!\$\ \mathrm{KEM.Encaps}(pk)$.
- Decaps is a deterministic algorithm that, given a $c \in \mathcal{C}$ and $sk \in \mathcal{SK}$ either returns a session key $k \in \mathcal{K}$ or the reject symbol $\perp$. This is expressed as $y \leftarrow \mathrm{KEM.Decaps}(sk, c)$, where $y \in \mathcal{K} \cup \{\perp\}$.

A KEM is said to be δ-correct if the probability of an honestly-performed key establishment process results in Encaps and Decaps outputting different keys is at most δ. Specifically,

$$\Pr[k' \neq k \,|\, (pk, sk) \leftarrow\!\!\$\ \mathsf{KEM.KeyGen}(1^n); (k, c) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}(pk); k' \leftarrow \mathsf{KEM.Decaps}(c)] \leq \delta\,.$$

Here δ is implicitly a function of n.

Security. The prevailing notion of security for KEMs is that of session-key indistinguishability, i.e., whether an adversary can distinguish between session keys produced by the KEM and bitstrings sampled uniformly at random from the session-key space. In this work we focus on IND-CCA security, which captures session-key indistinguishability against adversaries $\mathcal{A}$ who can make queries to a decapsulation oracle with any ciphertext $c \in \mathcal{C}$ except for the challenge ciphertext c^*. The security game is shown in Algorithm 1.

Algorithm 1 IND-CCA Security Games

Game $\mathrm{IND\text{-}CCA}^b_{\mathsf{KEM},\mathcal{A}}$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{KEM.KeyGen}(1^n)$
$(k_0, c^*) \leftarrow\!\!\$\ \mathsf{KEM.Encaps}(pk)$
$k_1 \leftarrow\!\!\$\ \mathcal{K}$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Decaps}^{\perp c^*}_{sk}(\cdot)}(pk, c^*, k_b)$
return b'

Oracle $\mathsf{Decaps}^{\perp c^*}_{sk}(c)$
if $c = c^*$ **then**
  return $\perp$
$k \leftarrow \mathsf{KEM.Decaps}(sk, c)$
return k

The advantage of an adversary $\mathcal{A}$ against the IND-CCA security of KEM is defined by

$$\mathrm{Adv}^{\mathsf{KEM}}_{\text{IND-CCA},\mathcal{A}} := \left|\Pr[1 \leftarrow \text{IND-CCA}^0_{\mathsf{KEM},\mathcal{A}}] - \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathsf{KEM},\mathcal{A}}]\right|.$$

We say that KEM is IND-CCA if, for all polynomial-time adversaries $\mathcal{A}$, $\mathrm{Adv}^{\mathsf{KEM}}_{\text{IND-CCA},\mathcal{A}}$ is a negligible function of the security parameter n. If we are concerned with post-quantum security, then we require that this holds for all quantum polynomial-time (QPT) adversaries.

Ciphertext Second Preimage Resistance. For certain KEMs, given a ciphertext c^*, it is difficult for an adversary to find a distinct ciphertext decapsulating to the same session key as c^*—under assumptions that are weaker than those required to prove IND-CCA. Consider the game shown in Algorithm 2. Notice that by giving the adversary the secret key as input, the KEM is simulated as being completely broken.

For an adversary $\mathcal{A}$, we define their advantage against the ciphertext second preimage resistance of a KEM KEM as:

$$\mathrm{Adv}^{\mathsf{KEM}}_{\text{C2PRI},\mathcal{A}} = \Pr[1 \leftarrow \text{C2PRI}_{\mathsf{KEM},\mathcal{A}}]$$

In our C2PRI proofs, we distinguish between adversaries with access only to classical computation from those with access to quantum computation.

Algorithm 2 Ciphertext Second Preimage Resistance Game

Game $\text{C2PRI}_{\mathsf{KEM},\mathcal{A}}$
$(pk, sk) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$(k^*, c^*) \leftarrow\$ \mathsf{KEM.Encaps}(pk)$
$c \leftarrow\$ \mathcal{A}(pk, sk, c^*, k^*)$
if $c \neq c^* \wedge \mathsf{KEM.Decaps}(sk, c) = k^*$ **then**
  return 1
return 0

2.3 Split-Key Pseudorandom Functions

A split-key pseudorandom function (PRF) is a function with multiple input keys that behaves like a truly random function as long as one key is obscured from an adversary. Split-key PRFs take n keys and are of the form $F : \mathcal{K}_1 \times \cdots \times \mathcal{K}_n \times \mathcal{X} \to \mathcal{Y}$ for key spaces $\mathcal{K}_1, \ldots, \mathcal{K}_n$, input space $\mathcal{X}$, and output space $\mathcal{Y}$. With respect to the security experiment outlined in Algorithm 3, for $1 \leq i \leq n$, we define the advantage of adversary $\mathcal{A}$ in distinguishing F from a random function F' when keyed on the i^{th} key as:

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} = \left|\Pr\left[1 \leftarrow \mathrm{PR}^0_{i,F,\mathcal{A}}\right] - \Pr\left[1 \leftarrow \mathrm{PR}^1_{i,F,\mathcal{A}}\right]\right|$$

Algorithm 3 Split-Key Pseudorandom Function Game

Game $\mathrm{PR}^b_{i,\mathcal{A}}$
$F' \leftarrow\$ \{\mathcal{K}_1 \times \ldots \times \mathcal{K}_{i-1} \times \mathcal{K}_{i+1} \times \ldots \times \mathcal{K}_n \times \mathcal{X} \to \mathcal{Y}\}$
$k_i \leftarrow\$ \mathcal{K}_i$
$b' \leftarrow\$ \mathcal{A}^{\mathrm{Eval}_b(\cdot)}()$
return b'

Oracle $\mathrm{Eval}_b(k, x)$
$(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n) \leftarrow k$
$y_0 \leftarrow F(k_1, \ldots, k_n, x)$
$y_1 \leftarrow F'(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n, x)$
return y_b

When considering an arbitrary key, we define the advantage of adversary $\mathcal{A}$ in distinguishing F from a random function F' as:

$$\mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{A}} = \max_i\{\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}}\}$$

The term *dual PRF* refers to functions with two inputs that behave like PRFs when either input is hidden. This is similar to the case of a split-key PRF where $\mathcal{X} = \emptyset$. We consider these as functions of the form $F : \mathcal{K}_1 \times \mathcal{K}_2 \to \mathcal{Y}$ and use the same notation and conventions as above.

3 Security of Hybrid KEMs

3.1 Combiner design

To combine two given KEMs, we follow the "Parallel KEM Combiner" approach of [GHP18]. We will instantiate each KEM individually. The public key, secret key, and ciphertext of our hybrid mechanism will be the concatenation of each ingredient KEM's public key, secret key, and ciphertext respectively. The session key output will be the result of some *core function* W applied to the ciphertexts and session keys of each ingredient KEM. We denote this method of combining KEMs as $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ and outline pseudocode for it in Algorithm 4. Letting c_i and k_i be the ciphertext and session key returned from KEM_i, [GHP18] proved that setting $W = F(k_1, k_2, c_1||c_2)$ for a split-key pseudorandom function F is sufficient to produce a robust hybrid KEM, where $c_1||c_2$ denotes the concatenation of c_1 and c_2. (Recall that a "robust hybrid KEM" retains its IND-CCA security so long as one ingredient KEM does.) In the case that the first KEM is ciphertext second preimage resistant, we will prove it is secure to output $F(k_1, k_2, c_2)$ for a split-key pseudorandom function F. Similarly, we will show that when both KEMs are C2PRI, one can simply return the result of a dual PRF $F(k_1, k_2)$.

Algorithm 4 Hybrid KEM design

$\underline{\Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W).\mathsf{KeyGen}(1^n)\textbf{:}}$
$(sk_1, pk_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$(sk_2, pk_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$sk \leftarrow (sk_1, sk_2)$
$pk \leftarrow (pk_1, pk_2)$
return (sk, pk)

$\underline{\Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, b).\mathsf{Encaps}(pk)\textbf{:}}$
$(pk_1, pk_2) \leftarrow pk$
$(k_1, c_1) \leftarrow\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$(k_2, c_2) \leftarrow\!\$\ \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$c \leftarrow (c_1, c_2)$
return $W(k_1, k_2, c_1, c_2)$

$\underline{\Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, b).\mathsf{Decaps}(sk, c)\textbf{:}}$
$(sk_1, sk_2) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(c_1, sk_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(c_2, sk_2)$
if $k_1 = \perp \vee k_2 = \perp$ **then**
  return $\perp$
return $W(k_1, k_2, c_1, c_2)$

3.2 Security proofs

In Theorem 6, we show our construction is secure when KEM_1 is C2PRI and KEM_2 retains its IND-CCA security. This proof holds both when W is a split-key PRF taking c_2 as input and when W is a dual PRF receiving neither ciphertext. By symmetry, Theorem 6 show that when both KEMs are C2PRI, setting $W = F(k_1, k_2)$ for a dual PRF F produces a robust hybrid KEM. Notice this theorem is not sufficient to prove security in the other setting, where only the first KEM is C2PRI and we want to set $W = F(k_1, k_2, c_2)$ for a split-key PRF F. We need our hybrid to be as secure as either ingredient KEM, but in that setting, Theorem 6 only reduces the security of the hybrid KEM to that of KEM_2. Theorem 7 reduces the security of our construction to that of KEM_1 without relying on any properties of KEM_2, assuming that c_2 is given as input to the core function. Taken together, Theorems 6 and 7 show that when only KEM_1 is C2PRI, setting $W = F(k_1, k_2, c_2)$ produces a robust hybrid KEM.

Theorem 6. *Let KEM_1 be a KEM and KEM_2 be a δ-correct KEM. Let F be a split-key (resp. dual) PRF with input space $\mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{C}_2$ (resp. $\mathcal{K}_1 \times \mathcal{K}_2$) where $\mathcal{K}_1$ is the key space of KEM_1 and $\mathcal{K}_2$ and $\mathcal{C}_2$ are the key and ciphertext spaces of KEM_2. Set $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$ (resp. $F(k_1, k_2)$), and let $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ be the combination of these KEMs as described in Section 3.1. Then for all quantum polynomial time (QPT) adversaries $\mathcal{A}$ against the IND-CCA security of Π making at most q queries to their decapsulation oracle, there exist QPT adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\text{PR}_2}_{F,\mathcal{D}} + \delta)$$

Moreover, $\mathcal{C}$ makes at most q classical queries to its own decapsulation oracle, $\mathcal{D}$ makes at most $q+1$ classical queries to its evaluation oracle, and all of $\mathcal{B}, \mathcal{C}$ and $\mathcal{D}$ run with constant-factor overhead.

Proof. We will use a hybrid strategy similar to that in the proof of Theorem 1 of [GHP18], as well as Theorems 1 and 2 of [BCD+24]. As an overview of the strategy, we start by considering the behavior of $\mathcal{A}$ when playing against the IND-CCA^0 game, labeling that game as hybrid G_0. We then consider the adversary playing against game G_1, where we slightly change in the decapsulation oracle, and argue that $\mathcal{A}$ can only detect this change with probability δ. In G_2, we have the decapsulation oracle abort when $\mathcal{A}$ inputs some KEM_1 ciphertext c_1 different from the challenge ciphertext c_1^* but decapsulating to the same session key. We show that noticing this change is as difficult as wining the C2PRI experiment for KEM_1. At this point, we can swap the KEM_1 session key used as input to F with a random key, bounding this change with the IND-CCA security of KEM_1. Once this key is generated uniformly at random, we may apply the security of F to replace the challenge key k^* that $\mathcal{A}$ receives with a random key. (Due to a small technical detail, this step requires two hybrid games.) Having acheived this, in G_5, we are nearly done with the proof. In game $G_6 - G_8$, we reverse the changes from the beginning of the proof, arguing in the same way as before that $\mathcal{A}$ cannot detect these modifications. G_8 is equivalent to the IND-CCA^1 game, so having shown it is difficult for $\mathcal{A}$ to notice any changes between G_0 and G_8, we conclude with our bound for $\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}}$. We describe each individual hybrid below, and Algorithm 5 contains their pseudocode.

Algorithm 5 Hybrid games for the proof of Theorem 6

Games $G_0 - G_8$:

$W'[\cdot] \leftarrow \perp$ ▷ G_4
$pk_1, sk_1 \leftarrow\$ \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow\$ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$k_1^*, c_1^* \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$k_2^*, c_2^* \leftarrow\$ \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$k_2^* \leftarrow\$ \mathcal{K}_2$ ▷ $G_3 - G_5$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$ ▷ $G_4 - G_8$
$W'[k_1^*] \leftarrow k^*$ ▷ G_4
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$

Oracle $\mathsf{Decaps}(c)$:

if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 = \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
if $c_2 = c_2^*$ **then** ▷ $G_1 - G_7$
  $k_2 \leftarrow k_2^*$ ▷ $G_1 - G_7$
if $k_1 = \perp \vee k_2 = \perp$ **then**
  return $\perp$
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then** ▷ $G_2 - G_6$
  abort_1 ▷ $G_2 - G_6$
if $c_2 = c_2^*$ **then** ▷ G_4
  if $W'[k_1] = \perp$: ▷ G_4
    $W'[k_1] \leftarrow\$ \mathcal{K}$ ▷ G_4
  return $W'[k_1]$ ▷ G_4
return $W(k_1, k_2, c_1, c_2)$

Game 0: Game G_0 represents $\mathcal{A}$ playing in the IND-CCA experiment for Π with the challenge bit set to 0. We run each KEM in parallel, just as Π does, and provide an honest session key to $\mathcal{A}$. Therefore:

$$\Pr[1 \leftarrow \text{IND-CCA}^0_{\Pi,\mathcal{A}}] = \Pr[1 \leftarrow G_0^{\mathcal{A}}]$$

Game 1: In game G_1, we make a small modification to the decapsulation oracle. When receiving (c_1, c_2) as input such that $c_2 = c_2^*$, it sets k_2 as k_2^* instead of performing $k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$ (The notation used here is the same as in Algorithm 5). Games 0 and 1 should nearly always be identical, and they only differ when KEM_2 would incorrectly decapsulate c_2^*. Decapsulation is deterministic, and we are only changing how the oracle decapsulates c_2^* (not other KEM_2 ciphertexts), so this G_0 and G_1 are different with probability at most δ regardless of how many times $\mathcal{A}$ queries the oracle. Therefore, we have:

$$\left|\Pr[1 \leftarrow G_0^{\mathcal{A}}] - \Pr[1 \leftarrow G_1^{\mathcal{A}}]\right| \leq \delta$$

Game 2: In game G_2, we again modify the decapsulation oracle. Now, it aborts when it receives an input (c_1, c_2) such that $c_1 \neq c_1^*$ but $\mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1) = k_1^*$. Notice this abort condition precisely

matches the win condition for KEM_1's ciphertext second preimage resistance game, so we claim there exists a QPT adversary $\mathcal{B}_1$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_1^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_2^{\mathcal{A}}]\right| \leq \Pr[\mathrm{abort}_1] = \mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}_1}$$

The pseudocode for adversary $\mathcal{B}_1$ is shown in Algorithm 6. When starting its challenge, $\mathcal{B}_1$ receives a public/secret key pair, ciphertext, and session key for KEM_1. It runs key generation and encapsulation for KEM_2, then uses these to instantiate $\mathcal{A}$. Because $\mathcal{B}_1$ has access to both secret keys, it can easily answer $\mathcal{A}$'s decapsulation queries. It perfectly simulates games 1 and 2 for $\mathcal{A}$ right until the abort condition would be triggered in game G_2, and $\mathcal{B}_1$ wins its C2PRI challenge exactly when this abort would occur. (Note if the abort never occurs, $\mathcal{B}_1$ returns c_1^* and loses its challenge.) This justifies the claim.

Algorithm 6 Adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ against the ciphertext second preimage resistance of KEM_1. The line in pink is only executed for $\mathcal{B}_2$.

Adversary $\mathcal{B}_1(pk_1, sk_1, c_1^*, k_1^*)$/$\mathcal{B}_2(pk_1, sk_1, c_1^*, k_1^*)$
$pk_2, sk_2 \leftarrow\$ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$k_2^*, c_2^* \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$
return c_1^*

If $\mathcal{A}$ **Calls** $\mathsf{Decaps}(c)$**:**
if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 = \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
if $c_2 = c_2^*$ **then**
  $k_2 \leftarrow k_2^*$
if $k_1 = \perp \vee k_2 = \perp$ **then**
  return $\perp$
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then**
  stop with c_1
return $W(k_1, k_2, c_1, c_2)$

Game 3: In game 3, we replace k_2^*, the session key for KEM_2, with a key generated uniformly at random. This should be difficult for $\mathcal{A}$ to detect so long at KEM_2 retains IND-CCA security. Indeed, we claim there exists a QPT adversary C_1 such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_2^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}_1}$$

The pseudocode to construct $\mathcal{C}_1$ is shown in Algorithm 7. $\mathcal{C}_1$ receives a public key, ciphertext, and session key for KEM_2. It creates keys and a ciphertext for KEM_1 and uses these to call $\mathcal{A}$. Adversary $\mathcal{C}_1$ can answer most of $\mathcal{A}$'s decapsulation queries by decapsulating c_1 with its own secret key and c_2 with its oracle. However, it cannot call its decapsulation oracle on c_2^*, so in that case, it uses the session key k_2^* it was given. (Notice due to the change in G_1, this is the correct way of simulating games G_2 and G_3 for $\mathcal{A}$.) When $\mathcal{C}$ is instantiated on the IND-CCA^0 game, the session key k_2^* was honestly generated, so $\mathcal{C}_1$ perfectly simulates G_2 for $\mathcal{A}$. When instantiated on the IND-CCA^1 game, it receives a random key and therefore simulates G_3 for $\mathcal{A}$. $\mathcal{C}_1$ returns the same output as $\mathcal{A}$, so the claim follows.

Game 4: Now that k_2^* is generated uniformly at random, we can apply the split-key pseudorandom property of F to replace k^* with a random session key. In this game, on decapsulation queries that include c_2^*, we return a random element of $\mathcal{K}$, using the table W' to maintain consistency across queries. We need to do this to accommodate the PRF adversary we will construct, which does not have access to k_2^* in its challenge. We will reverse this change in game G_5. We claim there exists a QPT adversary $\mathcal{D}_1$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\mathrm{PR}_2}_{F,\mathcal{D}_1}$$

We construct adversary $\mathcal{D}_1$ in Algorithm 8. (That algorithm shows $\mathcal{D}_1$ in the case that F is a split-key PRF taking a ciphertext as input. The dual PRF case is the same, except we don't include a ciphertext

Algorithm 7 IND-CCA adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ against KEM_2. Line in pink is only executed for $\mathcal{C}_2$.

Adversary $\mathcal{C}_1^{\mathsf{Decaps}^{\mathcal{O}}(\cdot)}(pk_2, c_2^*, k_2^*)$/ $\mathcal{C}_2^{\mathsf{Decaps}^{\mathcal{O}}(\cdot)}(pk_2, c_2^*, k_2^*)$

$pk_1, sk_1 \leftarrow \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$c_1^*, k_1^* \leftarrow \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$c \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$
return b'

If $\mathcal{A}$ Calls $\mathsf{Decaps}(c)$**:**
if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
if $c_2 = c_2^*$ **then**
  $k_2 \leftarrow k_2^*$
else
  $k_2 \leftarrow \mathsf{Decaps}^{\mathcal{O}}(c_2)$
if $k_1 = \perp \vee k_2 = \perp$ **then**
  return $\perp$
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then**
  abort_1
return $W(k_1, k_2, c_1, c_2)$

on queries to Eval.) $\mathcal{D}_1$ receives no input and runs key generation and encapsulation for both KEMs. When instantiated with game PR_2^0, $\mathcal{D}_1$ makes evaluation queries to F keyed with k_1^* and a random element of $\mathcal{K}_2$, so k^* is computed exactly as in G_3. However, $\mathcal{D}_1$ does not have access to the element of $\mathcal{K}_2$ used to compute k^*, so on decapsulation queries including c_2^*, $\mathcal{D}_1$ must make queries to Eval instead of computing the session key with F. Notice in the PR_2^0 setting, $\mathcal{D}_1$ generates k_2^* but never uses it, instead using the randomly-generated PRF key for its challenge. Since G_3 samples k_2^* at random, $\mathcal{D}_1$ is perfectly simulating G_3 for $\mathcal{A}$ in this setting. When $\mathcal{D}_1$ is instantiated on PR_2^1, all of its evaluation queries return random elements of $\mathcal{K}$. In the split-key pseudorandomness experiment, the evaluation oracle always returns consistent results across multiple queries with the same input; this is why we use the table W' in game G_4 instead of returning a randomly sampled element of $\mathcal{K}$ whenever $\mathcal{A}$ queries the decapsulation oracle with c_2^*. In the PR_2^1 case, $\mathcal{D}_1$ is perfectly simulating game G_4 for $\mathcal{A}$, so the claim follows.

Algorithm 8 Adversaries $\mathcal{D}_1$ and $\mathcal{D}_2$ against the split-key pseudorandomness of F, keyed on the second input. The line in pink is only executed for $\mathcal{D}_2$.

Adversary $\mathcal{D}_1^{\mathsf{Eval}(\cdot,\cdot)}/\mathcal{D}_2^{\mathsf{Eval}(\cdot,\cdot)}$

$pk_1, sk_1 \leftarrow \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$c_1^*, k_1^* \leftarrow \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$c_2^*, k_2^* \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$c \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow \mathsf{Eval}(k_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$
return b'

If $\mathcal{A}$ Calls $\mathsf{Decaps}(c)$**:**
if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
if $k_1 = \perp \vee (k_2 = \perp \wedge c_2 \neq c_2^*)$ **then**
  return $\perp$
if $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then**
  abort_1
if $c_2 = c_2^*$ **then**
  return $\mathsf{Eval}(k_1, c_2)$
return $W(k_1, k_2, c_1, c_2)$

Game 5: In game 5, we remove the table W' used in G_4 and answer decapsulation queries as in game 3. We claim there exists a QPT adversary $\mathcal{D}_2$ such that:

$$\left|\Pr[1 \leftarrow \text{G}_4^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_5^{\mathcal{A}}]\right| \leq \text{Adv}_{F,\mathcal{D}_2}^{\text{PR}_2}$$

The pseudocode for $\mathcal{D}_2$ is also shown in Algorithm 8 (as in the previous game, Algorithm 8 shows the case that F is a split-key PRF, but the dual PRF case is very similar). It behaves nearly identically to $\mathcal{D}_1$, but it always samples k^* randomly from $\mathcal{K}$. Similar to $\mathcal{D}_1$, when $\mathcal{D}_2$ is instantiated on PR_2^1, it returns random elements of $\mathcal{K}$ on decapsulation queries including c_2^*, simulating G_4 for $\mathcal{A}$. $\mathcal{D}_2$ does differ slightly from G_4: in G_4, $k^* = F'[k_1^*]$, but $\mathcal{D}_2$ choses k^* randomly with no relation to $\mathsf{Eval}(k_1^*, c_2^*)$. This would be a problem if $\mathcal{A}$ could query its decapsulation oracle in such a way as to recover $\mathsf{Eval}(k_1^*, c_2^*)$ and compare this to k^*. The decapsulation oracle only returns Eval queries when $c_2 = c_2^*$, and due to our abort condition, we cannot have $k_1 = k_1^*$ and $c_2 = c_2^*$ unless $c_1 = c_1^*$ and $c_2 = c_2^*$. But the decapsulation oracle always returns $\perp$ on that specific query. Therefore, when $\mathcal{D}_2$ playing against PR_2^1, it is still simulating G_4 for $\mathcal{A}$. When $\mathcal{D}_2$ is instantiated on PR_2^0, its evaluation queries are once again honest evaluations of F keyed on a random element of $\mathcal{K}_2$. Therefore, $\mathcal{D}_2$ perfectly simulates G_5 for $\mathcal{A}$ in this case, justifying our claim.

In the rest of the proof, we undo the changes made in games 1, 2, and 3. Now that k^* is sampled randomly from $\mathcal{K}$, this will yield the IND-CCA^1 game.

Game 6: In game 6, we undo the change from game G_3 and once again generate k_2^* honestly. Similar to game 3, we claim there exists a QPT adversary $\mathcal{C}_2$ such that:

$$\left|\Pr[1 \leftarrow \text{G}_5^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_6^{\mathcal{A}}]\right| \leq \text{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2, \mathcal{C}_2}$$

$\mathcal{C}_2$ is very similar to $\mathcal{C}_1$ and is shown in Algorithm 7. When instantiated on IND-CCA^1, $\mathcal{C}_2$ uses a random element of $\mathcal{K}_2$ as k_2^* and as such, simulates G_5 for $\mathcal{A}$. When playing against IND-CCA^0, $\mathcal{C}_2$ is uisng the output of $\mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$ as k_2^* and simulating G_6. Therefore, the claim holds.

Game 7: In game 7, we undo the change from G_2 and no longer abort on decapsulation queries with $k_1 = k_1^*$ and $c_1 \neq c_1^*$. We claim there exists a QPT adversary $\mathcal{B}_2$ such that:

$$\left|\Pr[1 \leftarrow \text{G}_6^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_7^{\mathcal{A}}]\right| \leq \Pr[\text{abort}_1] = \text{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1, \mathcal{B}_2}$$

Once again, $\mathcal{B}_2$ is similar to $\mathcal{B}_1$ and is shown in Algorithm 6. $\mathcal{B}_2$ receives a public/secret key pair, ciphertext, and session key for KEM_1 and creates the same for KEM_2. As in games 6 and 7, it samples k^* randomly from $\mathcal{K}$ and runs $\mathcal{A}$, answering decapsulation queries using its two secret keys. Games 6 and 7 are identical except for the abort condition, and $\mathcal{B}_2$ perfectly simulates both of them to $\mathcal{A}$ until the abort is triggered, at which time it wins its own C2PRI challenge. This justifies the claim.

Game 8: In game 8, we undo the change from G_1. On queries including c_2^*, instead of setting $k_2 = k_2^*$, we once again decapsulate c_2^* using sk_2. Similar to game 1, we have:

$$\left|\Pr[1 \leftarrow \text{G}_7^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_8^{\mathcal{A}}]\right| \leq \delta$$

Now, we have that k^* is sampled uniformly at random from $\mathcal{K}$, and we have undone our other changes. Therefore, G_8 is equivalent to the IND-CCA^1 game, and:

$$\Pr[1 \leftarrow \text{G}_8^{\mathcal{A}}] = \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathcal{A}}]$$

The theorem follows by setting $\mathcal{B}$ (respectively $\mathcal{C}$, $\mathcal{D}$) to whichever of $\mathcal{B}_1$ or $\mathcal{B}_2$ (respectively $\mathcal{C}_1$ or $\mathcal{C}_2$, $\mathcal{D}_1$ or $\mathcal{D}_2$) attains a greater advantage and summing the terms in each claim. □

Theorem 7. *Let KEM_1 be a δ-correct KEM and KEM_2 be a KEM. Let $F : \mathcal{K}_1 \times \mathcal{K}_2 \times \mathcal{C}_2 \to K$ be a split-key pseudorandom function, where $\mathcal{K}_1$ is the key space of KEM_1, $\mathcal{K}_2$ and $\mathcal{C}_2$ are the key and ciphertext spaces of KEM_2. Set $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$ and let $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ be the combination of these KEMs as described in Section 3.1. Then for all QPT adversaries $\mathcal{A}$ making at most q queries to their decapsulation oracle, there exist QPT adversaries $\mathcal{B}$ and $\mathcal{C}$ such that:*

$$\text{Adv}^{\text{IND-CCA}}_{\Pi, \mathcal{A}} \leq 2(\text{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1, \mathcal{B}} + \text{Adv}^{\text{PR}_1}_{F, \mathcal{C}} + \delta)$$

Moreover, $\mathcal{B}$ makes at most q queries to its own decapsulation oracles, $\mathcal{C}$ makes at most $q + 1$ queries to its evaluation oracle, and both run with at most constant-factor overhead.

Proof. The proof of this theorem follows in much the same way as Theorem 6 and is once again similar to Theorem 1 of [GHP18] and Theorems 1-2 of [BCD+24]. The largest difference is that because c_2 is part of the input to F, we don't need to rely on ciphertext second preimage resistance. This changes how we analyze adversary $\mathcal{D}$, which plays the same role that $\mathcal{E}$ did in the above proof. We refer to the previous proof for many other details and omit explicit constructions of $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{E}$ here. Once again, we use hybrid games, described in Algorithm 9.

Algorithm 9 Hybrid games for the proof of Theorem 7

Games $G_0 - G_6$**:**

$W'[\cdot] \leftarrow \perp$ ▷ G_3
$pk_1, sk_1 \leftarrow\$ \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow\$ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$k_1^*, c_1^* \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$k_2^*, c_2^* \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$k_1^* \leftarrow\$ \mathcal{K}_1$ ▷ $G_2 - G_4$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$ ▷ $G_3 - G_6$
$W'[k_2^*, c_2^*] \leftarrow k^*$ ▷ G_3
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$

Oracle $\mathsf{Decaps}(c)$**:**

if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 = \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
if $c_1 = c_1^*$ **then** ▷ $G_1 - G_5$
  $k_1 \leftarrow k_1^*$ ▷ $G_1 - G_5$
if $k_1 = \perp \vee k_2 = \perp$ **then**
  return $\perp$
if $c_1 = c_1^*$ **then** ▷ G_3
  if $W'[k_2, c_2] = \perp$: ▷ G_3
    $W'[k_2, c_2] \leftarrow\$ \mathcal{K}$ ▷ G_3
  return $W'[k_2, c_2]$ ▷ G_3
return $W(k_1, k_2, c_1, c_2)$

Games 0 - 3: Games 0, 1, 2, and 3 of this proof play the same roles as games 0, 1, 3, and 4 respectively in the proof of Theorem 6. Game 0 is equivalent to the IND-CCA game with challenge bit set to 0. In G_1, we set $k_1 \leftarrow k_1^*$ for decapsulation queries with $c_1 = c_1^*$, bounding the difference between these two games with δ. In game 2, we sample k_1^* randomly from $\mathcal{K}_1$, justifying this change with the IND-CCA security of KEM_1. In game 3, we choose k^* uniformly at random. Additionally, just like in G_4 of Theorem 6, on decapsulation queries with $c_1 = c_1^*$, we return random elements of $\mathcal{K}$, using the table W' to maintain consistency across queries. None of these changes relied on the abort condition we introduced in game 2 of the previous theorem, so similarly to before, we have that there exist QPT adversaries $\mathcal{B}_1$ and $\mathcal{C}_1$ such that:

$$\left|\Pr[1 \leftarrow \text{IND-CCA}^0_{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}]\right| \leq \delta + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1, \mathcal{B}_1} + \mathrm{Adv}^{\mathrm{PR}_1}_{F, \mathcal{C}_1}$$

Game 4: Game 4 is analogous to game 5 of the previous proof, where we remove the table W'. This time, W' is being used to answer decapsulation queries containing c_1^*. We claim there exists a QPT adversary $\mathcal{C}_2$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\mathrm{PR}_1}_{F, \mathcal{C}_2}$$

The pseudocode to construct $\mathcal{C}_2$ is shown in Algorithm 10. It generates public/secret key pairs, ciphertexts, and session keys for each KEM and selects k^* randomly from $\mathcal{K}$ so as to simulate games 3 and 4. When $\mathcal{C}_2$ is playing against PR_1^0, the Eval queries return honest evaluations of F keyed on a random element of $\mathcal{K}_1$, so $\mathcal{D}$ is properly simulating G_4 for $\mathcal{A}$. In the PR_1^1 setting, the evaluation queries return random elements of $\mathcal{K}$. This nearly simulates G_3, except that in G_3, $k^* = F'[k_2^*, c_2^*]$, but $\mathcal{C}_2$ does not guarantee that $k^* = \mathsf{Eval}(k_2^*, c_2^*)$. This means that $\mathcal{C}_2$ only replicates G_3 so long as $\mathcal{A}$ cannot recover $\mathsf{Eval}(k_2^*, c_2^*)$ from the decapsulation oracle. However, the oracle only returns Eval queries on inputs with $c_1 = c_1^*$. We would also need $c_2 = c_2^*$ for the decapsulation oracle to return an Eval query containing c_2^*, but if $c_1 = c_1^*$ and $c_2 = c_2^*$ then Decaps returns $\perp$. Therefore, when $\mathcal{C}_2$ is instantiated on PR_1^1, it can perfectly simulate G_3 for $\mathcal{A}$, justifying the claim.

Games 5-6: Once again, games 5 and 6 are similar to games 6 and 8 of the previous proof. In game 5, we return to generating k_1^* honestly, bounding this difference with the IND-CCA security of KEM_1.

Algorithm 10 Adversary $\mathcal{C}_2$ against the split-key pseudorandomness of F, keyed on the first input.

Adversary $\mathcal{C}_2^{\mathsf{Eval}(\cdot,\cdot)}$
$pk_1, sk_1 \leftarrow \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$c_1^*, k_1^* \leftarrow \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$c_2^*, k_2^* \leftarrow \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$c \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow \mathsf{Eval}(k_2^*, c_2^*)$
$k^* \leftarrow\!\!\$\ \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$
return b'

If $\mathcal{A}$ **Calls** $\mathsf{Decaps}(c)$**:**
if $c = c^*$ **then**
  return $\perp$
$c_1, c_2 \leftarrow c$
$k_1 = \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
if $(k_1 = \perp \wedge c_1 \neq c_1^*) \vee k_2 = \perp$ **then**
  return $\perp$
if $c_1 = c_1^*$ **then**
  return $\mathsf{Eval}(k_2, c_2)$
return $W(k_1, k_2, c_1, c_2)$

In game 6, we stop setting $k_1 \leftarrow k_1^*$ on queries $c_1 \leftarrow c_1^*$, justifying this with the correctness of KEM_1. Therefore, there exists a QPT advesary $\mathcal{B}_2$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_6^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1, \mathcal{C}_2}$$

But G_6 is equivalent to the IND-CCA game with challenge bit set to 1, so:

$$\Pr[1 \leftarrow \mathrm{G}_6^{\mathcal{A}}] = \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathcal{A}}]$$

Once again, the theorem follows by setting $\mathcal{B}/\mathcal{C}$ to whichever of $\mathcal{B}_1$ or $\mathcal{B}_2/\mathcal{C}_1$ or $\mathcal{C}_2$ attains a greater advantage and summing each inequality. □

3.3 Extension to Many KEMs

In this section, we briefly outline how our construction and proofs can be generalized to combine n different KEMs. A natural extension of our two-KEM construction to combine n KEMs is to concatenate the n public keys, secret keys, and ciphertexts of the ingredient KEMs whenever those of the two KEMs in the two-KEM construction are concatenated and adapt everything accordingly, as shown in Algorithm 11. Supposing KEMs 1 through ℓ are ciphertext second preimage resistant, one can compute the hybrid session key by applying a split-key PRF to each ingredient session key, as well as the ciphertexts from KEMs $\ell + 1$ through n. We will continue referring to this construction as Π.

Algorithm 11 n-KEM Hybrid design

$\Pi(\mathsf{KEM}_1, \ldots, \mathsf{KEM}_n, W).\mathsf{KeyGen}(1^m)$**:**
$(sk_1, pk_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{KeyGen}(1^m)$
$\ldots$
$(sk_n, pk_n) \leftarrow\!\!\$\ \mathsf{KEM}_n.\mathsf{KeyGen}(1^m)$
$sk \leftarrow (sk_1, \ldots, sk_n)$
$pk \leftarrow (pk_1, \ldots, pk_n)$
return (sk, pk)

$\Pi(\mathsf{KEM}_1, \ldots, \mathsf{KEM}_n, b).\mathsf{Encaps}(pk)$**:**
$(pk_1, \ldots, pk_n) \leftarrow pk$
$(k_1, c_1) \leftarrow\!\!\$\ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$\ldots$
$(k_n, c_n) \leftarrow\!\!\$\ \mathsf{KEM}_n.\mathsf{Encaps}(pk_n)$
$c \leftarrow (c_1, \ldots, c_n)$
return $W(k_1, \ldots, k_n, c_1, \ldots, c_n)$

$\Pi(\mathsf{KEM}_1, \ldots, \mathsf{KEM}_n, b).\mathsf{Decaps}(sk, c)$**:**
$(sk_1, \ldots, sk_n) \leftarrow sk$
$(c_1, \ldots, c_n) \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(c_1, sk_1)$
$\ldots$
$k_n \leftarrow \mathsf{KEM}_n.\mathsf{Decaps}(c_n, sk_n)$
if $k_1 = \perp \vee \ldots \vee k_n = \perp$ **then**
  return $\perp$
return $W(k_1, \ldots, k_n, c_1, \ldots, c_n)$

Theorem 8. *Let* $\mathsf{KEM}_1, ..., \mathsf{KEM}_n$ *be KEMs, where* KEM_i *is* δ*-correct. Let* F *be a split-key PRF with input space* $\mathcal{K}_1 \times ... \times \mathcal{K}_n \times \mathcal{C}_{\ell+1} \times \mathcal{C}_{\ell+2} \times ... \times \mathcal{C}_n$*, where* $\mathcal{K}_j$ *and* $\mathcal{C}_j$ *are the key and ciphertext spaces of* KEM_j*. Set* $W = F(k_1, ..., k_n, c_{\ell+1}, c_{\ell+2}, ..., c_n)$ *and let* $\Pi = \Pi(\mathsf{KEM}_1, ..., \mathsf{KEM}_n, W)$ *be the combination of these KEMs as described above. Then, for all QPT adversaries* $\mathcal{A}$ *against the* IND-CCA *security of* Π *making at most* q *queries to their decapsulation oracle, there exist QPT adversaries* $\mathcal{B}_1, ..., \mathcal{B}_\ell, \mathcal{C}$*, and* $\mathcal{D}$ *such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2\left(\left(\sum_{j=1,j\neq i}^{\ell} \mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_j,\mathcal{B}_j}\right) + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_i,\mathcal{C}} + \mathrm{Adv}^{\text{PR}_i}_{F,\mathcal{D}} + \delta\right)$$

Moreover, $\mathcal{C}$ *makes at most* q *classical queries to its own decapsulation oracle,* $\mathcal{D}$ *makes at most* $q+1$ *classical queries to its evaluation oracle, and each adversary runs with constant-factor overhead.*

Proof sketch. This theorem can be proved using the same structure as Theorem 6, and we only highlight the main differences here. Game 2 is replaced with a series of games $2.1, ..., 2.\ell$. (We exclude $2.i$ if $1 \leq i \leq \ell$.) In game $2.j$, we prohibit $\mathcal{A}$ from making decapsulation queries where the j^{th} coordinate c_j is different from the j^{th} challenge ciphertext c_j^* but decapsulates to k_j^*, the j^{th} challenge session key. We bound the difference between each of these game hops with $\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_j,\mathcal{B}_{1,j}}$ and construct $\mathcal{B}_{1,j}$ similarly to adversary $\mathcal{B}_1$ of Theorem 6. In game 4, we once again select the challenge session key k^* randomly and answer decapsulation queries randomly when $c_i = c_i^*$, using the table W to maintain consistency. W takes as input a session key from each KEM except KEM_i, as well as a ciphertext from each of $\mathsf{KEM}_{\ell+1}, ..., \mathsf{KEM}_n$, again except KEM_i (if applicable). In game 5, we continue selecting k^* randomly but remove the other changes of game 4, bounding this difference with the advantage of a adversary $\mathcal{D}_2$ against the split-key pseudorandomness of F. Similar to game 5 of Theorem 6, $\mathcal{D}_2$ would fail to properly simulate game 5 to $\mathcal{A}$ if $\mathcal{A}$ could query the decapsulation oracle in a way that creates the same preimage of F as was used to initially compute k^*. However, just like in the previous proofs, due to the changes from games $2.1, ..., 2.\ell$, it is impossible to make such a query without submitting the challenge ciphertext, which is always prohibited. So $\mathcal{D}_2$ can properly simulate game 5 for $\mathcal{A}$. To complete the proof, games 6, 7, and 8 once again undo the changes of games 1, 2, and 3.

4 Common Key Derivation Functions are Split-Key PRFs

In this section, we provide an overview on how commonly used KDFs can be shown to be split-key PRFs, assuming idealized primitives. We begin with the assertion that a random oracle applied to concatenated keys and input constitutes a split-key PRF:

Theorem 9. *Let* H *be a random oracle and let* $\mathcal{A}$ *be an adversary against the split-key pseudorandomness of the function* $F(k_1, \ldots, k_n, x) = H(k_1 || \ldots || k_n || x)$ *with keys* $k_1 \in \mathcal{K}_1, \ldots, k_n \in \mathcal{K}_n$ *in keyspaces* $\mathcal{K}_1, \ldots, \mathcal{K}_n$ *making at most* q *quantum queries to* H*. Then,*

$$\mathrm{Adv}^{\text{PR}_i}_{F,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}}$$

We believe this is a straightforward conclusion from previously known results, but we provide a proof in Appendix A for completeness. In what follows, we use this theorem to sketch proofs of split-key pseudorandomness for the key derivation functions discussed in [BCD20].

KMAC: KMAC is a variable-length MAC/PRF scheme based on the customizable extendable output function cSHAKE, both defined in [KCP16]. In our context, we define KMAC as a split-key PRF to be:

$$\mathsf{KMAC}(k_1, ..., k_n, x) = \mathsf{cSHAKE}(salt||1||k_1||...||k_n||x, L, \text{“KMAC”}, \text{“KDF”})$$

where *salt* is a fixed, previously agreed-upon, potentially secret string (having been pre-processed as outlined in [KCP16]), L is the fixed desired output length, and 1, “KMAC”, and “KDF” are fixed byte strings acting as domain separators. When cSHAKE is modeled as a random oracle, Theorem 9 can be clearly extended to give a bound on the split-key pseudorandomness of KMAC.

Hash Functions: Following NIST's recommendations in [BCD20], to construct a split-key PRF from a hash function H, we consider the form:

$$F(k_1, \ldots, k_n, x) = H(1||k_1|| \ldots ||k_n||x)||H(2||k_1|| \ldots ||k_n||x)|| \ldots$$

continuing for as many hash evaluations as necessary to achieve the (fixed) desired output length. The counter at the beginning of each input is represented by a fixed-length byte string and acts as a domain separator, so when H is modeled as a random oracle, each hash output is an independent random value. Therefore, F also behaves like a random oracle, so Theorem 9 can again be used to bound split-key pseudorandomness in this setting.

HMAC (One Step): HMAC is a message authentication code (MAC) defined in [HMA08] over a hash function H as:

$$\mathsf{HMAC}(k, m) = H\left((k \oplus opad)||H((k \oplus ipad)||m)\right)$$

where *opad* and *ipad* are fixed byte strings. Notice that when H is a random oracle, different values of m produce independent, random output values. Again following from [BCD20], we construct a split-key PRF from HMAC by using the form:

$$F(k_1, \ldots, k_n, x) = \mathsf{HMAC}(salt, 1||k_1|| \ldots ||k_n||x)||\mathsf{HMAC}(salt, 2||k_1|| \ldots ||k_n||x)|| \ldots$$

where *salt* is once again a fixed string. As above, the counter acts as a domain separator that creates distinct input messages for HMAC, so once again, if H behaves like a random oracle, then Theorem 9 applies.

Two-Step Methods: In addition to the above, [BCD20] also provides a two-step, extract-then-expand procedure. First, a key-derivation key K_{DK} is computed as a MAC of the initial shared secret, namely:

$$K_{DK} = \mathsf{MAC}(salt, k_1||...||k_n)$$

Though both HMAC and CMAC are allowed, we will focus on HMAC for simplicity. When the underlying hash function behaves like a random oracle, an adversary that does not know one k_i can assume that different choices of $k_1, ..., k_{i-1}, k_{i+1}, ..., k_n$ yield independent random values of K_{DK}. Next, K_{DK} is used to derive the final output key material using a PRF-based KDF constructed in [Che22]. Chen presents three options to construct KDFs based on an underlying PRF. (When HMAC is used as the MAC in the first step, it is also required to be used as the PRF in the second step.) In each option, we fix a length L and a label string and set $F(k_1, ..., k_n, x) = K(1)||K(2)||...$ for some function K and as many rounds as necessary to produce a key of our desired length. The first option is *counter mode*, with

$$K(i) = \mathsf{PRF}(K_{DK}, i||Label||0||x||L)$$

In *feedback mode*, we set $K(0)$ to some initialization vector and

$$K(i) = \mathsf{PRF}(K_{DK}, K(i-1)||i||Label||0||x||L)$$

where the inclusion of i in the PRF is optional. Notably, the popular key derivation function HKDF is constructed with HMAC and a variant of feedback mode [KE10]. Finally, in *double pipeline mode*, we use two PRF evaluations in each step. Specifically, we set:

$$\begin{aligned} A(0) &= Label||0||x||L \\ A(i) &= \mathsf{PRF}(K_{DK}, A(i-1)) \\ K(i) &= \mathsf{PRF}(K_{DK}, A(i)||i||Label||0||x||L) \end{aligned}$$

where once again, the inclusion of i is optional. Whenever i is included in the PRF and the hash function of HMAC is modeled as a random oracle, each $K(i)$ can be regarded as an independent random value, so once again, we may apply Theorem 9 to bound the split-key pseudorandomness of these key derivation functions. When i is not included in feedback and double pipeline modes, the values of each $K(i)$ can no longer be regarded as independent. However, the function output still appears random as long as for any individual query, no two values of $K(i)$ collide; the probability of this event occuring on a given query is roughly $\frac{m^2}{|\mathcal{K}|}$, where m is the number of $K(i)$ blocks computed in one query and $\mathcal{K}$ is the output space of HMAC.

5 Ciphertext Second Preimage Resistance of $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEMs

5.1 $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEM design

We define $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEMs to be the results of converting public-key encryption schemes into key-encapsulation mechanisms via the respective $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ transformations as defined in [HHK17]. In these transforms, encryption is deterministic, with a derandomization of the underlying encryption scheme . A well-known KEM encompassed by these definitions includes the $\mathsf{U}^{\not\perp}$ KEM BIKE, as specified in [BIK24], and other popular KEMs such as Classic McEliece and HQC are similar in structure to these KEMs. The pseudocode for these classes of KEMs is shown in Algorithm 12.

Algorithm 12 $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEMs

$\underline{\mathsf{KEM.KeyGen}(1^n)}$
$(pk', sk') \leftarrow\$ \mathsf{PKE.KeyGen}(1^n)$
$s \leftarrow\$ \mathcal{M}$
$sk \leftarrow (sk', s)$
return (pk', sk)

$\underline{\mathsf{KEM.Encaps}(pk)}$
$m \leftarrow\$ \mathcal{M}$
$c \leftarrow\$ \mathsf{PKE.Encrypt}(pk, m)$
$k \leftarrow H(m, c)$
return (k, c)

$\underline{\mathsf{KEM.Decaps}(sk, c)\ (\mathsf{U}^{\perp})}$
$(sk', s) \leftarrow sk$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk', c)$
if $m' = \perp$ **then**
  return $\perp$
$k \leftarrow H(m', c)$
return k

$\underline{\mathsf{KEM.Decaps}(sk, c)\ (\mathsf{U}^{\not\perp})}$
$(sk', s) \leftarrow sk$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk', c)$
if $m' = \perp$ **then**
  $k \leftarrow H(s, c)$
  return k
$k \leftarrow H(m', c)$
return k

5.2 Classical and quantum C2PRI proofs

Theorem 10. *Let* KEM *be a* $\mathsf{U}^{\perp}$ *or* $\mathsf{U}^{\not\perp}$ *KEM with keyspace* $\mathcal{K}$ *where the hash function used in encapsulation and decapsulation* H *is modeled as a random oracle, and let* $\mathcal{C}$ *be an adversary against the ciphertext second preimage resistance of* KEM *making at most* q *classical queries to* H*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|}$$

Proof. There are two cases where there could be a collision: on inputs to H or on outputs of H. It is impossible for inputs to H to collide for distinct ciphertexts, as the ciphertext is an argument of H. Hence, any collisions are limited to the output of H.

Without loss of generality (at a cost of one extra query) we can assume that the adversary only outputs a ciphertext that it has previously queried to H (along with some arbitrary m). We simulate the entire experiment by lazy sampling H. The first query occurs during the encapsulation step performed by the challenger, resulting in some session key k. The remaining queries are made by $\mathcal{C}$. For any fresh query x, the probability of $H(x) = k$ is $1/|\mathcal{K}|$. By a union bound over the $q+1$ queries made by $\mathcal{C}$, the probability of at least one query mapping to k is at most $(q+1)/|\mathcal{K}|$. □

Theorem 11. *Let* KEM *be a* $\mathsf{U}^{\perp}$ *or* $\mathsf{U}^{\not\perp}$ *KEM with keyspace* $\mathcal{K}$ *where the hash function used in encapsulation and decapsulation* H *is modeled as a random oracle, and let* $\mathcal{C}$ *be an adversary against the ciphertext second preimage resistance of* KEM *making at most* q *quantum queries to* H*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

Proof. First, let us generate a key $k \in \mathcal{K}$ using $\mathsf{KEM.KeyGen}$ and $\mathsf{KEM.Encaps}$. Consider random oracles H_0^0 and H_0^1, where H_0^0 is defined to be the random oracle modeling H and H_0^1 is defined to be H_0^0 punctured at inputs that map to k such that those inputs now map to keys sampled from $K \backslash \{k\}$. In other words, if S is the set of $x \in X$ (where X is the input space of the random oracles) that map to $k \in K$ under H_0^0, then $H_0^0(x) = H_0^1(x)$ for $x \in X \backslash S$ and $H_0^0(x) = k \neq H_0^1(x)$ for $x \in S$. In the random oracle model, S is random, H_0^0 and H_0^1 are random functions, and S and x are independent (i.e. any given input to H_0^0 has the same probability of being in S), so by Theorem 1 and Corollary 1 of [AHU21] any adversary $\mathcal{D}$ has advantage:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 2\sqrt{(d+1) \cdot P_{\mathrm{find}}} = 2\sqrt{(d+1) \cdot \frac{4q}{|\mathcal{K}|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

in Algorithm 13, shown below. The quantity d denotes the query depth of $\mathcal{D}$, for which $d \leq q$ holds by the definition of query depth, and P_{find} denotes the probability with which $\mathcal{D}$ can find an element of S.

Algorithm 13 H_0^0 and H_0^1 Distinguishing Game

Game $\mathrm{DIST}_{\mathcal{D}}^{H_0}$
Prepare random oracle H_0^0
$(pk, sk) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$(k, c) \leftarrow\$ \mathsf{KEM.Encaps}(pk)$
Prepare H_0^1 as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{D}^{H_0^b(\cdot)}(k)$
return b'

The case where $b = 0$ corresponds to H_0^0 and the case where $b = 1$ corresponds to H_0^1. $\mathcal{D}$ attempts to output the value of b corresponding to the oracle to which it is given access, winning if and only if $b' = b$.

We now examine the game in Algorithm 14, where $\mathcal{D}$ calls on an adversary $\mathcal{A}$ against the ciphertext second preimage resistance of KEM^b, which is a modified version of KEM.

Algorithm 14 H_0^0 and H_0^1 Distinguishing Game with Adversary $\mathcal{A}$

Game $\mathrm{DIST}_{\mathcal{D}}^{H_0}$
Prepare random oracle H_0^0
$(pk^*, sk^*) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$(k, c^*) \leftarrow\$ \mathsf{KEM.Encaps}(pk)$
Prepare H_0^1 as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{D}^{H_0^b(\cdot)}(k)$
return b'

Adversary $\mathcal{D}^{H_0^b(\cdot)}(k)$
$(pk, sk) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$m \leftarrow\$ \mathcal{M}$
$c \leftarrow\$ \mathsf{PKE.Encrypt}(pk, m)$
$x \leftarrow m||c$
if $H_0^b(x) = k$ **then**
  return 0
Prepare H_1^b
Prepare KEM^b
$c' \leftarrow\$ \mathcal{A}(pk, sk, c, k)$
if $c' \neq c \wedge \mathsf{KEM'.Decaps}(sk, c') = k$ **then**
  return 0
return 1

By construction, there is at least one preimage of k under H_0^0 and there are no preimages of k under H_0^1. The strategy that $\mathcal{D}$ employs in Algorithm 14 is to modify H_0^b to have an arbitrary honestly generated input to the hash function in KEM encapsulation map to k to create a random oracle H_1^b, then to construct KEM^b, which is a KEM with the same construction as KEM but uses H_1^b in place of H everywhere H is used, and then finally to query $\mathcal{A}$ and post-process its output. $\mathcal{D}$ is allowed to prepare H_1^b as it does since H_1^b is modeled as a reversible quantum circuit. As H_1^b is constructed from H_0^b and has one input mapped to k, in the $b = 0$ setting there could be second preimages of k

under H_1^b and in the $b = 1$ setting there can be none. Thus, if $\mathcal{A}$ finds a second preimage, then $\mathcal{D}$ can conclude that it has access to H_0^0. While it is clear that such a strategy is not optimal, as there could exist second preimages among hash inputs with different public and secret keys (or a secret string in the case of implicit rejection for a $\mathsf{U}^{\not\perp}$ KEM), $\mathcal{D}$ perfectly simulates the view of $\mathcal{A}$ in the ciphertext second preimage resistance game, yielding the result:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} = \Pr[1 \leftarrow \mathrm{C2PRI}_{\mathsf{KEM}^0,\mathcal{A}}] \leq \Pr[0 \leftarrow \mathrm{DIST}^{H_0^0}_{H_0,\mathcal{D}}] = \mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

since $\Pr[0 \leftarrow \mathrm{DIST}^{H_0^1}_{H_0,\mathcal{D}}] = 0$ by the construction of H_0^1 and $\mathcal{D}$. An adversary $\mathcal{C}$ against the ciphertext second preimage resistance of KEM must have an advantage less than or equal to the advantage of $\mathcal{A}$ against the ciphertext second preimage resistance of KEM^0, as $H(x) = H_0^1(x)$ for all but one $x \in X$, and for the one $x \in X$ for which they could differ, it is known that $H_0^1(x) = k$ but nothing can be assumed about $H(x)$. Therefore, we obtain:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

as desired.

□

5.3 Applications to BIKE, Classic McEliece, and HQC

BIKE is a standard $\mathsf{U}^{\not\perp}$ KEM, so our results can be applied directly.

Classic McEliece. Classic McEliece is another KEM designed with a slight modification of the $\mathsf{U}^{\not\perp}$ transformation, described in [McE22]. The notable difference is that Classic McEliece includes distinct domain separators in its hash function when outputting a session key via regular decapsulation versus implicit rejection. Additionally, it calls the subroutines $\mathsf{FixedWeight}$ (roughly corresponding to choosing a random message at the beginning of encapsulation), Encode (similar to encryption), and Decode (similar to decryption). Those algorithms are omitted here, as well as KeyGen, but the encapsulation and decapsulation algorithms for Classic McEliece are shown in Algorithm 15.

Algorithm 15 Classic McEliece KEM

$\underline{\mathsf{KEM.Encaps}(pk)}$
$m \leftarrow\!\!\$\ \mathsf{FixedWeight}()$
$c = \mathsf{Encode}(m, pk)$
$k = H(1, m, c)$
return (k, c)

$\underline{\mathsf{KEM.Decaps}(sk, c)}$
$(sk', s) \leftarrow sk$
$b \leftarrow 1$
$m \leftarrow \mathsf{Decode}(sk', c)$
if $m = \perp$ **then**
  $e \leftarrow s$
  $b \leftarrow 0$
$k \leftarrow H(b, m, c)$
return k

The domain separation used in session-key generation does not affect either proof beyond prepending the inputs of the random oracles with a 0 or a 1 accordingly anywhere they are prepared or queried, so these results also hold for Classic McEliece.

HQC. The Hamming Quasi-Cyclic (HQC) KEM is a KEM that is the result of converting the HQC public-key encryption scheme into a KEM via a variant of the FO Transform described in [HQC24]. It has the same structure as a $\mathsf{U}^{\not\perp}$ KEM, but it derandomizes the underlying HQC PKE scheme in the KEM by using a function H_T and does not include part of the ciphertext in session-key derivation, posing important considerations in analyses of its ciphertext second preimage resistance (for simplicity, we consider the salt to be part of the ciphertext rather than separate from it, at no consequence to

the applicability of our results). The pseudocode for this KEM in notation similar to that of [KSW24] is shown in Algorithm 16. The parameters e, r_1, r_2, h, u, v, G, and ℓ are parameters of the HQC PKE scheme, and H_T derandomizes the scheme by using a random oracle G to generate a randomness parameter θ given arguments m, pk, and $salt$ and then using θ to generate e, r_1, and r_2 with prespecified Hamming weights.

Algorithm 16 HQC KEM

$\underline{\mathsf{KEM.KeyGen}(1^n)}$
$(pk', sk') \leftarrow\$ \mathsf{PKE.KeyGen}(1^n)$
$s \leftarrow\$ \mathcal{M}$
$sk \leftarrow (sk', s)$
return (pk', sk)

$\underline{\mathsf{KEM.Encaps}(pk)}$
$m \leftarrow\$ \mathcal{M}$
$salt \leftarrow\$ \mathcal{SL}$
$(e, r_1, r_2) \leftarrow H_T(m, \text{firstBytes}(pk, 32), salt)$
$u \leftarrow r_1 + hr_2$
$v \leftarrow \text{truncate}(mG + sr_2 + e, \ell)$
$c \leftarrow (u, v, salt)$
$k \leftarrow H(m, u, v)$
return (k, c)

$\underline{\mathsf{KEM.Decaps}(sk, c)}$
$(sk', s) \leftarrow sk$
$(u, v, salt) \leftarrow c$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk', u, v)$
$(e', r'_1, r'_2) \leftarrow H_T(m', pk, salt)$
$u' \leftarrow r'_1 + hr'_2$
$v' \leftarrow \text{truncate}(m'G + sr'_2 + e', \ell)$
if $(u', v') \neq (u, v)$ **then**
  $k \leftarrow H(s, u, v)$
  return k
$k \leftarrow H(m', u, v)$
return k

HQC has a structure that introduces additional terms into the advantage bound. Both the classical and quantum C2PRI proofs rely on the fact that the entire ciphertext is used in key derivation to demonstrate that finding a second preimage entails finding a preimage of k under the hash function H. However, only part of the ciphertext is included in key derivation in HQC, which can be exploited by attacks like the one described in [KSW24] against the ciphertext collision resistance of HQC. Fortunately, the implications are not nearly as drastic for the ciphertext second preimage resistance of HQC.

To adapt the C2PRI proofs to HQC, we must account for all cases where an adversary could find ciphertext second preimages that result in the same preimage of the hash function as the challenge hash preimage at the end of decapsulation. An adversary can only exploit implicit rejection in such a way that the adversary and challenge hash preimages are equal when $m = s$, in which case $\mathcal{A}$ could find a ciphertext with a salt such that implicit rejection is triggered but the parts of the ciphertext included in key derivation are equal to those of the challenge ciphertext. The probability that $m = s$ for any given instance of the C2PRI game is $\frac{1}{|\mathcal{M}|}$.

Analyzing the case where an adversary exploits normal key derivation in this way is slightly more involved. In normal key derivation during decapsulation of the challenge parameters, m, u, and v are hashed, and for the adversary hash preimage to be equal to the challenge hash preimage, for adversary preimage paramaters m', u', and v', we must have $m' = m, u' = u$, and $v' = v$. As a result of PKE decryption, $m' = m$ if $u' = u$ and $v' = v$ assuming the correctness of decryption, so it suffices for the adversary to find $c' = u||v||salt'$ where $c = u||v||salt$ and $salt' \neq salt$ such that rejection is not triggered in decapsulation for c' to fall in this case. For it to be true that $(u', v') = (u, v)$ in the rejection check, it must be true that $r'_1 + hr'_2 = u$ and that $\text{truncate}(m'G + sr'_2 + e', \ell) = v$ in the decapsulation of c'. Moreover, such e', r'_1, and r'_2 must be generated by H_T given m', the first 32 bytes of pk, and $salt'$, where the adversary can only control the $salt'$ parameter. Thus, to quantify the probability of this case, we set up an adversary $\mathcal{B}$ against what we will denote as the salt second preimage resistance (S2PRI) of the process by which u' and v' are computed. The adversary $\mathcal{B}$ attempts to find a salt $salt'$ distinct from the challenge salt such that e', r'_1, and r'_2 by H_T with the property that $(r'_1, r'_2) \in U^{-1}(u)$ and $(e', r'_2) \in V^{-1}(v)$, where $U(x_1, x_2) = x_1 + hx_2$ and $V(x_1, x_2) = \text{truncate}(m'G + sx_2 + x_1, \ell)$ for inputs x_1 and x_2. $\mathcal{B}$ has advantage $\Pr[1 \leftarrow \text{S2PRI}_{\mathsf{HQC},\mathcal{B}}]$ in the game shown in Algorithm 17.

As $\mathcal{C}$ and thus $\mathcal{B}$ are computationally unbounded, $\mathcal{B}$ can compute the preimages of u and v with

Algorithm 17 Salt Second Preimage Resistance Game

Game $\text{S2PRI}_{\mathsf{HQC},\mathcal{B}}$
$(pk, sk') \leftarrow\$ \mathsf{PKE.KeyGen}(1^n)$
$s \leftarrow\$ \mathcal{M}$
$sk \leftarrow (sk', s)$
$m \leftarrow\$ \mathcal{M}$
$salt \leftarrow\$ \mathcal{SL}$
$(e, r_1, r_2) \leftarrow H_T(m, \text{firstBytes}(pk, 32), salt)$
$u \leftarrow U(r_1, r_2)$
$v \leftarrow V(e, r_2)$
$c \leftarrow (u, v, salt)$
$k \leftarrow H(m, u, v)$
$salt' \leftarrow\$ \mathcal{B}(pk, sk, c, k)$
$(e', r_1', r_2') \leftarrow H_T(m, pk, salt')$
if $salt' \neq salt \wedge (r_1', r_2') \in U^{-1}(u) \wedge (e', r_2') \in V^{-1}(v)$ **then**
  return 1
return 0

respect to e', r_1', and r_2' without any effects on advantage bounds. To account for all of the mentioned cases, we can model H_T as a random oracle and update the advantage bound for a classical adversary against the ciphertext second preimage resistance of HQC to be:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} + \text{Adv}^{\text{S2PRI}}_{\mathsf{HQC},\mathcal{B}} + \frac{1}{|\mathcal{M}|}$$

where $\mathcal{B}$ makes at most q_{H_T} classical queries to H_T, and the advantage bound for a quantum adversary against the ciphertext second preimage resistance of HQC to be:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} + \text{Adv}^{\text{S2PRI}}_{\mathsf{HQC},\mathcal{B}} + \frac{1}{|\mathcal{M}|}$$

where $\mathcal{B}$ makes at most q_{H_T} quantum queries to H_T.

Now, it remains to bound the advantage of $\mathcal{B}$ in the S2PRI games. One way to do so is to bound the cardinalities of $U^{-1}(u)$ and/or $V^{-1}(v)$, after which a classical lazy sampling bound or a quantum search bound like that in the One-Way to Hiding Lemma in [AHU21] can be applied. For simplicity, we consider only the cardinality of $U^{-1}(u)$. We know that $u = U(r_1, r_2) = r_1 + hr_2$, where $h, r_1, r_2 \in \mathbb{F}_2^n$ are bit vectors of length n, with r_1 and r_2 having Hamming weight w_r, and hr_2 is the vector representation of the product of the polynomials corresponding to h and r_2. If $h = 0$, then $u = r_1$ and so the cardinality of $U^{-1}(u)$ is $\binom{n}{w_r}$. As for $h \neq 0$, given an arbitrary r_2', we wish to quantify how many r_1' exist such that $r_1' + hr_2' = u$. Rearranging this equation, we obtain $r_1' = u - hr_2'$. The product hr_2' is unique for each r_2', as is the difference of u and hr_2'. Thus, for each r_2', if r_1' exists, then it is unique (alternatively, such an r_1' does not exist if $u - hr_2'$ does not have weight w_r). Since every r_2' has at most one r_1' such that $(r_1', r_2') \in U^{-1}(u)$, an upper bound on the cardinality of $U^{-1}(u)$ is the number of possible vectors r_2'. This number is $\binom{n}{w_r}$, so $|U^{-1}(u)| \leq \binom{n}{w_r}$. The total number of pairs is $\binom{n}{w_r}^2$, so the maximum probability of arbitrary parameters (r_1', r_2') being in $U^{-1}(u)$ is $\frac{\binom{n}{w_r}}{\binom{n}{w_r}^2} = \frac{1}{\binom{n}{w_r}}$. This probability allows us to apply bounds on finding a salt second preimage with respect to q_{H_T} according to classical lazy sampling bounds and the One-Way to Hiding Lemma. More specifically expressed, with n defined to be the length of h, r_1, and r_2 and w_r defined to be the Hamming weight of r_1 and r_2, the advantage bound for a classical adversary against the ciphertext second preimage resistance of HQC is:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} + \frac{q_{H_T}+1}{\binom{n}{w_r}} + \frac{1}{|\mathcal{M}|}$$

where $\mathcal{B}$ makes at most q_{H_T} classical queries to H_T, and the advantage bound for a quantum adversary against the ciphertext second preimage resistance of HQC is:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} + 4\sqrt{\frac{q_{H_T}^2+q_{H_T}}{\binom{n}{w_r}}} + \frac{1}{|\mathcal{M}|}$$

where $\mathcal{B}$ makes at most q_{H_T} quantum queries to H_T.

6 Ciphertext Second Preimage Resistance for $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$ KEMs

6.1 $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$ KEM design

We define $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$ KEMs to be the results of converting public-key encryption schemes into key-encapsulation mechanisms via the respective $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$ transformations as defined in [HHK17]. Encryption is deterministic in these transforms as well, with a derandomization of the underlying encryption scheme. A well-known KEM which has a similar structure to a $\mathsf{U}^{\not\perp}_m$ KEM is ML-KEM. The pseudocode for these classes of KEMs is shown in Algorithm 18.

Algorithm 18 $\mathsf{U}^{\perp}_m$ and $\mathsf{U}^{\not\perp}_m$ KEMs

$\underline{\mathsf{KEM.KeyGen}(1^n)}$
$(pk', sk') \leftarrow\$ \mathsf{PKE.KeyGen}(1^n)$
$s \leftarrow\$ \mathcal{M}$
$sk \leftarrow (sk', s)$
return (pk', sk)

$\underline{\mathsf{KEM.Encaps}(pk)}$
$m \leftarrow\$ \mathcal{M}$
$c \leftarrow \mathsf{PKE.Encrypt}(pk, m)$
$k \leftarrow H(m)$
return (k, c)

$\underline{\mathsf{KEM.Decaps}(sk, c)\ (\mathsf{U}^{\perp}_m)}$
$(sk', s) \leftarrow sk$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk', c)$
if $m' = \perp$ **then**
  return $\perp$
$k \leftarrow H(m')$
return k

$\underline{\mathsf{KEM.Decaps}(sk, c)\ (\mathsf{U}^{\not\perp}_m)}$
$(sk', s) \leftarrow sk$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk', c)$
if $m' = \perp$ **then**
  $k \leftarrow H(s, c)$
  return k
$k \leftarrow H(m')$
return k

6.2 Classical and quantum C2PRI proofs

Theorem 12. *Let* KEM *be a* $\mathsf{U}^{\perp}_m$ *or* $\mathsf{U}^{\not\perp}_m$ *KEM with keyspace* $\mathcal{K}$ *where the hash function* H *is modeled as a random oracle, and let* $\mathcal{C}$ *be an adversary against the ciphertext second preimage resistance of* KEM *making at most* q *classical queries to* H*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|}$$

Proof. As in the proof that $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEMs are classically C2PRI, there can either be a collision on inputs to H or on outputs of H. PKE encryption is well-defined and deterministic, so for a given public key pk and message m it can only output one ciphertext c. It follows that during decapsulation, PKE.Decrypt cannot output the same message m' for distinct ciphertexts c_1 and c_2 and the same secret key sk, so checking the result of PKE decryption with encryption will result in the rejection of at least one of these ciphertexts. Thus, the case where there is a collision on inputs to H during normal key derivation in decapsulation is impossible and can be reduced to the case where the output of H collides with the challenge key during implicit rejection, so the contribution of this specific case to the advantage bound is 0. Meanwhile, the case that there is a collision on inputs to H exploiting implicit rejection could result if $m' = s||c'$ for message m', secret s, and ciphertext c' in decapsulation. However, m' and s have the same length and c' must have a positive length by construction, so implicit rejection hash inputs are strictly longer than normal key derivation hash inputs and such a collision

is impossible. Hence, the contribution of this case to the advantage bound is 0. As for collisions on outputs of H, by the same reasoning in the proof of Theorem 10, the probability of an adversary finding a second preimage of k is less than or equal to $\frac{q}{|\mathcal{K}|}$. Therefore, as in that proof, the advantage of a C2PRI adversary is bounded by $\frac{q+1}{|\mathcal{K}|}$.

□

Theorem 13. *Let* KEM *be a* $\mathsf{U}^{\perp}_{m}$ *or* $\mathsf{U}^{\not\perp}_{m}$ *KEM with keyspace* $\mathcal{K}$ *where the hash function* H *is modeled as a random oracle, and let* $\mathcal{C}$ *be an adversary against the ciphertext second preimage resistance of* KEM *making at most* q *quantum queries to* H*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

Proof. This proof proceeds much in the same manner as the proof of the quantum ciphertext second preimage resistance of $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ KEMs in Section 5.2. Let H^0_0 and H^1_0 be defined as they are in Section 5.2 so that by the same reasoning in that proof any adversary $\mathcal{D}$ has advantage:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 2\sqrt{(d+1)\cdot P_{\mathrm{find}}} = 2\sqrt{(d+1)\cdot\frac{4q}{|\mathcal{K}|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

in the game depicted in Algorithm 19, where $b=0$ corresponds to H^0_0 and $b=1$ corresponds to H^1_0, and $\mathcal{D}$ attempts to output the value of b corresponding to the oracle to which it is given access, winning if and only if $b'=b$.

Algorithm 19 H^0_0 and H^1_0 Distinguishing Game

Game $\mathrm{DIST}^{H_0}_{\mathcal{D}}$
Prepare random oracle H^0_0
$(pk, sk) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$(k, c) \leftarrow\$ \mathsf{KEM.Encaps}(pk)$
Prepare H^1_0 as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{D}^{H^b_0(\cdot)}(k)$
return b'

As in the proof in Section 5.2, we now examine the game in Algorithm 20, where $\mathcal{D}$ calls on an adversary $\mathcal{A}$ against the ciphertext second preimage resistance of KEM^b, which is a modified version of KEM.

Algorithm 20 H^0_0 and H^1_0 Distinguishing Game with Adversary $\mathcal{A}$

Game $\mathrm{DIST}^{H_0}_{\mathcal{D}}$
Prepare random oracle H^0_0
$(pk^*, sk^*) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$(k, c^*) \leftarrow\$ \mathsf{KEM.Encaps}(pk)$
Prepare H^1_0 as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow \mathcal{D}^{H^b_0(\cdot)}(k)$
return b'

Adversary $\mathcal{D}^{H^b_0(\cdot)}(k)$
$(pk, sk) \leftarrow\$ \mathsf{KEM.KeyGen}(1^n)$
$m \leftarrow\$ \mathcal{M}$
$c \leftarrow\$ \mathsf{PKE.Encrypt}(pk, m)$
$x \leftarrow m$
if $H^b_0(x) = k$ **then**
  return 0
Prepare H^b_1
Prepare KEM^b
$c' \leftarrow\$ \mathcal{A}(pk, sk, c, k)$
if $c' \neq c \wedge \mathsf{KEM'.Decaps}(sk, c') = k$ **then**
  return 0
return 1

There is a notable difference between this game and the corresponding game in Section 5.2, namely that H_1^b maps m to k instead of $m||c$ to k. We claim that this difference does not affect the advantage bound of $\mathcal{A}$. Consider the ciphertext c' returned by $\mathcal{A}$. If c' truly constitutes a second preimage, then there are three possibilities for how such a second preimage exists: the inputs to $\mathsf{PKE.Decrypt}$ collide, the outputs of $\mathsf{PKE.Decrypt}$ collide, or the outputs of H_1^b collide. It is impossible for the inputs to $\mathsf{PKE.Decrypt}$ to collide, since $c' \neq c$. Meanwhile, $\mathsf{PKE.Decrypt}$ has an internal check to ensure that $\mathsf{PKE.Encrypt}$ returns c' given pk and the message m' it computes from sk and c', but if $m' = m$, as $\mathsf{PKE.Encrypt}(pk, m) = (k, c)$ and $c' \neq c$ this check will fail and $\mathsf{PKE.Decrypt}$ will return $\perp$, so it must be the case that $m' \neq m$. Thus, the only way there can be a second preimage is if there is a collision in the outputs of $H_1^b(\cdot)$ and it is still the case that if $\mathcal{A}$ finds a second preimage then $\mathcal{D}$ knows that it has access to H_0^0. It also still holds that $\Pr[0 \leftarrow \mathrm{DIST}^{H_0^1}_{H_0,\mathcal{D}}] = 0$ by the construction of H_0^1 and $\mathcal{D}$, so:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} = \Pr[1 \leftarrow \mathrm{C2PRI}_{\mathsf{KEM}^0,\mathcal{A}}] \leq \Pr[0 \leftarrow \mathrm{DIST}^{H_0^0}_{H_0,\mathcal{D}}] = \mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

Again, as in the proof in Section 5.2, an adversary $\mathcal{C}$ against the ciphertext second preimage resistance of KEM must have an advantage less than or equal to the advantage of $\mathcal{A}$ against the ciphertext second preimage resistance of KEM^0. Therefore, we obtain:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}}$$

as desired.

□

6.3 Application to ML-KEM

ML-KEM is designed with a slight modification of the $\mathsf{U}^{\not\perp}_m$ transform and is defined in [ML-24]. Notably, it the key derivation function $G(m, H(pk))$ for hash functions G and H. In the case of implicit rejection, it outputs $J(s, c)$ for another hash function J. Additionally, it re-encrypts the message and checks if this equals the original ciphertext; if not, it implicitly rejects.

Algorithm 21 ML-KEM

$\underline{\mathsf{KEM.KeyGen}(1^n)}$
$(pk, sk') \leftarrow\!\$\ \mathsf{PKE.KeyGen}(1^n)$
$s \leftarrow\!\$\ \mathcal{M}$
$sk \leftarrow (sk', pk, H(pk), s)$
return (pk, sk)

$\underline{\mathsf{KEM.Encaps}(pk))}$
$m \leftarrow\!\$\ \mathcal{M}$
$(k, r) \leftarrow G(m, H(pk))$
$c \leftarrow \mathsf{PKE.Encrypt}(pk, m, r)$
return (k, c)

$\underline{\mathsf{KEM.Decaps}(sk, c)}$
$(sk', pk, h, s) \leftarrow sk$
$m' \leftarrow \mathsf{PKE.Decrypt}(sk, c)$
$(k, r') \leftarrow G(m', h)$
$c' \leftarrow \mathsf{PKE.Encrypt}(pk, m', r')$
if $c \neq c'$ **then**
  return $J(s, c)$
return k

The advantage of an adversary against the ciphertext second preimage resistance of ML-KEM making at most q_G classical queries to G and at most q_J classical queries to J was proved to be bounded above by $\frac{q_G+q_J+2}{|\mathcal{K}|}$ in [BCD+24]. As for its quantum second preimage resistance, the structure of ML-KEM is similar to that of $\mathsf{U}^{\not\perp}_m$ KEMs, but differs in that randomness generated as a byproduct of key derivation is included as a parameter in PKE encryption, a hash of the public key is included in normal key derivation, and distinct hash functions G and J are used for normal key derivation and implicit rejection, respectively. The first difference does not affect the quantum C2PRI proof beyond how $\mathcal{D}$ sets up the adversary $\mathcal{A}$, as the randomness does not affect session-key generation. The second difference also does not affect the bound, as $\mathcal{A}$ cannot change the challenge public key hashed and used in key derivation. Finally, to accommodate for G and J, we can use domain separation and adjust the oracles in the proof by prepending H_1^b oracle inputs with 0 and 1 everywhere G and J would be respectively prepared and queried in KEM' and giving the oracle access to $G(\cdot) = H_1^b(0||\cdot)$

and $J(\cdot) = H_1^b(1||\cdot)$. These changes do not affect anything else in the proof, so the quantum C2PRI bound holds for adversaries against the ciphertext second preimage resistance of ML-KEM.

References

[ADK+22] Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenneth G. Paterson, Eyal Ronen, and Eylon Yogev. Practical (post-quantum) key combiners from one-wayness and applications to TLS. Cryptology ePrint Archive, Paper 2022/065, 2022.

[AHK+23] Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. Post-quantum multi-recipient public key encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1108–1122, New York, NY, USA, 2023. Association for Computing Machinery.

[AHU21] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In *Advances in Cryptology – CRYPTO 2019*, 2021.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 206–226, Cham, 2019. Springer International Publishing.

[BBGS23] Matilda Backendal, Mihir Bellare, Felix Günther, and Matteo Scarlata. When messages are keys: Is hmac a dual-prf? In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 661–693, Cham, 2023. Springer Nature Switzerland.

[BCD20] Elaine Barker, Lily Chen, and Richard Davis. Recommendation for key-derivation methods in key-establishment schemes. NIST SP 800-56C Rev. 2, August 2020.

[BCD+24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karolin Varner, and Bas Westerbaan. X-wing. *IACR Communications in Cryptology*, 1(1), 2024.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 602–619, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BIK24] BIKE: Bit flipping key encapsulation, 2024. https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf.

[BJKS24] Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 469–486, Philadelphia, PA, August 2024. USENIX Association.

[CDM24] Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping up with the kems: Stronger security notions for kems and automated analysis of kem-based protocols. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1046–1060, New York, NY, USA, 2024. Association for Computing Machinery.

[Che22] Lily Chen. Recommendation for key derivation using pseudorandom functions. NIST SP 800-108r1-upd1, August 2022.

[CHH+25] Deirdre Connolly, Kathrin Hövelmanns, Andreas Hülsing, Stavros Kousidis, and Matthias Meijers. Starfighters — on the general applicability of x-wing. Cryptology ePrint Archive, Paper 2025/1397, 2025.

[CLM25] Céline Chevalier, Guirec Lebrun, and Ange Martinelli. Spilling-Cascade: an Optimal PKE Combiner for KEM Hybridization. In *23rd International Conference on Applied Cryptography and Network Security (ACNS'25)*, Munich, Germany, June 2025.

[CP20] Matthew Campagna and Adam Petcher. Security of hybrid key encapsulation. Cryptology ePrint Archive, Paper 2020/1364, 2020.

[DFH22] Jelle Don, Serge Fehr, and Yu-Hsuan Huang. Adaptive versus static multi-oracle algorithms, and quantum security of a split-key prf. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I*, page 33–51, Berlin, Heidelberg, 2022. Springer-Verlag.

[Fed22] Federal Office for Information Security (BSI). *Quantum-safe cryptography - fundamentals, current developments and recommendations*, May 2022.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 537–554, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, 2013.

[Fre24] French Cybersecurity Agency (ANSSI) and Federal Office for Information Security (BSI) and Netherlands National Communications Security Agency (NLNCSA) and Swedish National Communications Security Authority, Swedish Armed Forces. *Position Paper on Quantum Key Distribution*, January 2024.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. Kem combiners. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 190–218, Cham, 2018. Springer International Publishing.

[HDV21] Loïs Huguenin-Dumittan and Serge Vaudenay. Fo-like combiners and hybrid post-quantum cryptography. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 225–244, Cham, 2021. Springer International Publishing.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 341–371, Cham, 2017. Springer International Publishing.

[HMA08] The keyed-hash message authentication code (HMAC). FIPS PUB 198-1, July 2008. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.198-1.pdf.

[HQC24] Hamming quasi-cyclic (HQC), 2024. https://pqc-hqc.org/doc/hqc-specification_2024-10-30.pdf.

[KCP16] John Kelsey, Shu-jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash . NIST SP 800-185, 2016.

[KE10] Hugo Krawczyk and Pasi Eronen. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869, May 2010.

[KSW24] Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Binding security of implicitly-rejecting KEMs and application to BIKE and HQC. Cryptology ePrint Archive, Paper 2024/1233, 2024.

[McE22] Classic McEliece: conservative code-based cryptography: cryptosystem specification, 2022. https://classic.mceliece.org/mceliece-spec-20221023.pdf.

[ML-24] Module-lattice-based key-encapsulation mechanism standard. FIPS 203, August 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.

[O'B23] Devon O'Brien. Protecting chrome traffic with hybrid kyber KEM. Chromium Blog, August 2023.

[XGL+21] Jia Xu, Yiwen Gao, Hoon Wei Lim, Hongbing Wang, and Ee-Chien Chang. Stateful KEM: Towards optimal robust combiner for key encapsulation mechanism. Cryptology ePrint Archive, Paper 2021/989, 2021.

A A proof of Theorem 4

Theorem 14. *Let H be a random oracle and let $\mathcal{A}$ be an adversary against the split-key pseudorandomness of the function $F(k_1, \ldots, k_n, x) = H(k_1 || \ldots || k_n || x)$ with keys $k_1 \in \mathcal{K}_1, \ldots, k_n \in \mathcal{K}_n$ in keyspaces $\mathcal{K}_1, \ldots, \mathcal{K}_n$ making at most q quantum queries to H. Then,*

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}}$$

Proof. Denote H by H_0. We start by considering the game in Algorithm 22, where an adversary $\mathcal{D}$ is tasked with distinguishing whether it has access to H_0 or a reprogrammed version denoted by H_1. Specifically, H_1 is defined such that $H_1(x) = H_0(x)$ for $x \in X \setminus S$, where X is the input space of H and S is the set of inputs that include the k_i uniformly randomly sampled from $\mathcal{K}_i$ by the challenger, and such that $H_1(x)$ is sampled from $\mathcal{Y}$ for $x \in S$ and output space $\mathcal{Y}$. The adversary has access to an Eval oracle with which it can query H_0 with k_i fixed in the i^{th} key position in the argument of H_0.

Algorithm 22 Random Oracle Distinguishing Game

Game $\mathrm{DIST}^b_{\mathcal{D}}$
$k_i \leftarrow\$ \mathcal{K}_i$
Prepare Eval as above
$b' \leftarrow \mathcal{D}^{H_b(\cdot),\mathrm{Eval}(\cdot)}$
return b'

Oracle $\mathrm{Eval}(k', x)$
$k_1 ... k_{i-1} k_{i+1} ... k \leftarrow k'$
$y \leftarrow H_0(k_1, ..., k_i, ..., k_n, x)$
return y

The case where $b = 0$ corresponds to H_0, and the case where $b = 1$ corresponds to H_1. The adversary $\mathcal{D}$ attempts to output the value of b corresponding to the version of the H_b oracle to which it is given access. By Theorem 1 and Corollary 1 of [AHU21], any adversary $\mathcal{D}$ has a bounded advantage:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_b,\mathcal{D}} \leq 2\sqrt{(d+1)\cdot P_{\mathrm{find}}} = 2\sqrt{(d+1)\cdot\frac{4q}{|\mathcal{K}_i|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}_i|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}}$$

in this game. We now consider a modified version of this game, as shown in Algorithm 23, where instead of distinguishing whether it has access to H_0 or H_1, the adversary is tasked with distinguishing whether it has access to Eval_0 or Eval_1, with Eval_0 being the Eval oracle in Algorithm 22 and Eval_1 having the same construction as Eval_0 but using H_1 rather than H_0. The adversary is also given access to H_0.

Algorithm 23 Eval Oracle Distinguishing Game

Game $\mathrm{DIST}^b_{\mathcal{E}}$
$k_i \leftarrow\$ \mathcal{K}_i$
Prepare Eval_b as above
$b' \leftarrow \mathcal{E}^{H_0(\cdot),\mathrm{Eval}_b(\cdot)}$
return b'

Oracle $\mathrm{Eval}_b(k', x)$
$k_1 ... k_{i-1} k_{i+1} ... k \leftarrow k'$
$y \leftarrow H_b(k_1, ..., k_i, ..., k_n, x)$
return y

As in the previous game, the adversary $\mathcal{E}$ attempts to output the value of b corresponding to the version of the H_b oracle to which it is given access, winning if and only if $b' = b$. We claim that these games are distributionally identical. It is clear that the adversaries in both games have equivalent tasks: determining whether the Eval oracle is instantiated with the same version of H to which it has access, with the $b = 0$ case being that they are the same and the $b = 1$ case being that they are different. When $b = 0$, the adversaries in both games both have access to H_0 and Eval_0, so when $b = 0$ the two games are equivalent. Meanwhile, when $b = 1$, the adversary in the H_b distinguishing game

has access to H_1 and Eval_0 while the adversary in the Eval_b distinguishing game has access to H_0 and Eval_1. By the fact that H_0 and H_1 are random oracles that can differ only when k_i is in the i^{th} key position in the argument, the setting where the adversary has access to H_1 and Eval_0 and the setting where the adversary has access to H_0 and Eval_1 are identical from the perspective of the adversary up to labeling the oracles, since in both settings the adversary is given access to a random oracle and an Eval oracle that is constructed to return random outputs that are independent of the random oracle. Hence, when $b = 1$ the two games are also equivalent, so the games are distributionally identical and the advantage of an adversary in the Eval_b distinguishing game must be bounded by the advantage of an adversary in the H_b distinguishing game:

$$\text{Adv}^{\text{DIST}}_{\text{Eval}_b,\mathcal{E}} \leq \text{Adv}^{\text{DIST}}_{H_b,\mathcal{D}}$$

We now claim that the Eval_b distinguishing game is identical to the split-key pseudorandom function game instantiated with the split-key construction with H_0, as when $b = 0$, the two games are the same, and when $b = 1$, returning $H_1(k_1, \ldots, k_i, \ldots, k_n, x)$ is equivalent to returning a random value from the output space $\mathcal{Y}$ by the construction of H_1. Therefore, we obtain the following bound for an adversary $\mathcal{A}$ in the split-key pseudorandom function game:

$$\text{Adv}^{\text{PR}_i}_{F,\mathcal{A}} = \text{Adv}^{\text{DIST}}_{\text{Eval}_b,\mathcal{E}} \leq \text{Adv}^{\text{DIST}}_{H_b,\mathcal{D}} \leq 4\sqrt{\frac{q^2 + q}{|\mathcal{K}_i|}}$$

as desired. □