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

Gorjan Alagic[1,2][0000−0002−0107−6037], Fahran Bajaj[3][0009−0005−1990−5664], and Aybars Kocoglu[3][0009−0007−6687−6442]

[1] QuICS, University of Maryland Institute for Advanced Computer Studies, College Park, MD 20742, USA
galagic@umd.edu
[2] National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
[3] University of Maryland, College Park, MD 20742, USA
{fbajaj, akocoglu}@umd.edu

**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 the "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 quantum random oracle model. This applies to KEMs such as BIKE, Classic McEliece, HQC, and ML-KEM.

**Keywords:** KEM · KEM Combiner · Hybrid KEM · Post-Quantum · IND-CCA

## 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 [25,26].

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 [40] and Signal [17]. Hybrid key-establishment schemes typically take the form of a key-encapsulation mechanism (KEM), 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[4]. The standard approach (and the one we consider here) is then to combine these ingredient KEMs in parallel. Specifically, given two KEMs $\Pi_1$ and $\Pi_2$, one creates a hybrid KEM $\Pi$ whose encapsulation keys, decapsulation keys, and ciphertexts are concatenations of the corresponding elements of $\Pi_1$ and $\Pi_2$. What then remains is to choose how to generate the session key of $\Pi$.

As shown by Giacon, Heuer, and Poettering, generating the session key in a hybrid KEM is not as straightforward as one might expect [30]. 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 $\mathsf{KDF}$ is used. Specifically, if $\Pi_1$ 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 [30] 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 [30,24].

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

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 [27,28,13]. The *Quantum Superiority Fighter* (QSF) framework of [13] 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 either when ML-KEM maintains its own IND-CCA security or when the strong Diffie-Hellman assumption holds in the X25519 nominal group, so long as 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 build upon the above results as follows. We show that combining two KEMs using (2) generically preserves IND-CCA provided the first KEM satisfies C2PRI and $W$ is split-key pseudorandom. Additionally, 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 folklore result that we prove for completeness shows that this is secure in the (quantum-accessible) random oracle model (QROM). 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 $\mathsf{KEM}_1$ and $\mathsf{KEM}_2$ be two $\delta$-correct KEMs and $\Pi$ be a KEM constructed from $\mathsf{KEM}_1$ and $\mathsf{KEM}_2$ using some "combiner" function family $W$. We are concerned with establishing IND-CCA security of $\Pi$ against either classical probabilistic polynomial-time (PPT) adversaries or quantum polynomial-time (QPT) adversaries.

The combined KEM $\Pi$ is constructed as follows. Key generation, encapsulation, and decapsulation run the corresponding algorithms of $\mathsf{KEM}_1$ and $\mathsf{KEM}_2$ in parallel and concatenate the results appropriately to form the encapsulation key, decapsulation key, and ciphertext of $\Pi$. At the end of encapsulation and decapsulation, the session key of $\Pi$ is computed by $k = W(k_1, k_2, c_1, c_2)$. Our

focus is on the security of $\Pi$ 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 $\Pi$ is then CCA-secure provided (i.) $\mathsf{KEM}_1$ is IND-CCA, or (ii.) $\mathsf{KEM}_2$ is IND-CCA and $\mathsf{KEM}_1$ 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 $\Pi$. Then there exist QPT adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{D}} + \delta)\,. \tag{3}$$

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

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1,\mathcal{B}'} + \mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{C}'} + \delta)\,. \tag{4}$$

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 $\Pi$ is then CCA-secure provided $\mathsf{KEM}_1$ is IND-CCA and $\mathsf{KEM}_2$ 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 $\Pi$. Then there exist QPT adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\mathsf{dual\text{-}PRF}}_{F,\mathcal{D}} + \delta)\,. \tag{5}$$

Together, these two theorems tell us that when one out of two ingredient KEMs is C2PRI, it is appropriate to omit the corresponding ciphertext from the key derivation function (Theorem 1), and when both are C2PRI, both ciphertexts can be omitted (Theorem 2). Theorem 1 and Theorem 2 are implied by two technical results stated in Section 3 and proved in Appendix A. 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 $\Pi$, and each adversary runs with at most constant-factor overhead compared to $\mathcal{A}$. 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 Appendix B, 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* $\Pi$, *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*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2\left(\left(\sum_{j=1,j\neq i}^{\ell} \mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_j,\mathcal{B}_j}\right) + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_i,\mathcal{C}} + \mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{D}} + \delta\right). \tag{6}$$

Again, each derived adversary makes at most one additional oracle query 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 [30] 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 $H$ [24]. In the special case where $g$ is the identity function, a straightforward application of existing results shows that a random oracle is a split-key PRF, as follows.

**Theorem 4.** *Let* $H$ *be a random oracle and let* $\mathcal{A}$ *be an 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,*

$$\mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{A}} \leq \max_{i=1,\ldots,n} 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{7}$$

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 C. 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 [14], which include popular methods like HKDF, can be shown to be split-key PRFs in appropriate idealized models.

**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 [32] 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* $\mathsf{KEM}$ *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* $\mathsf{KEM}$*. If* $\mathcal{C}$ *makes at most* $q$ *classical queries to* $H$*, then*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} \tag{8}$$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{9}$$

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 [13]. 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 [10].

Theorem 5 shows directly that some well-known KEMs (like BIKE and HQC) satisfy C2PRI. We also show how the proofs can be adapted to show C2PRI for other prominent KEMs, namely Classic McEliece, the fourth-round NIST PQC standardization submission of HQC, and ML-KEM [7]. The fourth-round submission of HQC is an interesting case, as it is known not to be ciphertext *collision-resistant* [38]. We provide specific advantage bounds for both classical queries and quantum queries.

**Implications.** As discussed by Barbosa et al. in [13], many post-quantum KEMs use large 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 similar in structure to our combiner, we expect similar performance gains when using our combiner with ML-KEM-768 as an ingredient. The standard parameter sets of Classic McEliece specified in [2] 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 notable

post-quantum KEMs 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 [5], BIKE with ciphertext sizes ranging from 1573 to 5154 bytes [3], and HQC with ciphertext sizes ranging from 4433 to 14421 bytes [6]. By establishing C2PRI security for Classic McEliece, BIKE, and HQC, we show that these KEMs can be used in QSF. Moreover, we demonstrate that one may efficiently combine two post-quantum KEMs with different underlying assumptions without including either ciphertext in the hash function. For example, one may be interested in combining ML-KEM and HQC to hedge against lattice- or code-based assumptions being broken. Creating hybrid schemes with C2PRI 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 [30,18,19,42]. Most of these run each KEM in parallel and input each ciphertext into the final KDF, though the XOR-then-MAC approach of [18] and the construction from [42] use distinctive strategies. The authors of [34], [39] and [21] 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. The Quantum Superiority Fighter (QSF) construction of [13] combines a nominal group with a C2PRI KEM and excludes the KEM's ciphertext from the core function. In particular, the authors highlight X-Wing, the instantiation of QSF with X25519 and ML-KEM-768. Alwen et al. consider the problem of combining multi-recipient KEMs (mKEMs) [9]. 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 are further explored as the LEAK-BIND-K,PK-CT property in [23] and [38]. Notably, [38] proves that the round-4 specification of HQC is not LEAK-BIND-K,PK-CT secure, but we prove that it is C2PRI, implying that the distinction between these two notions is practically relevant. Dual PRFs are studied in [15,11,12].

**Recent Related Work.** The present paper was submitted to a conference in Spring 2025, except for Section 4 and Appendix B, which were added shortly after. After that, the closely related independent work [22] appeared on eprint.

Then, the present paper was also posted on eprint, with no modifications except the addition of this paragraph. Several minor changes were made after receiving comments from reviewers.

We now briefly describe the main differences between the present paper and [22] as we see them. The work [22] looks at binding and multi-user properties for their hybrid KEM constructions while our work does not. Our work looks at combining many KEMs (Appendix B) and various KDF choices (Section 4) while [22] 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, the appropriate notion for their purposes). In terms of showing C2PRI for specific KEMs, [22] gives better bounds; they analyze ML-KEM, Classic McEliece, NTRU, FrodoKEM, and HQC, while we analyze ML-KEM, Classic McEliece, HQC (including the fourth-round submission of HQC, which does not include the salt in the key derivation), and BIKE. [22] also considers the salted and key confirmation variants of the FO transform, while we do not.

## 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 $\mathrm{G}_1$. Similarly, $\triangleright\, \mathrm{G}_1 - \mathrm{G}_3$ means that the relevant line is only executed in games $\mathrm{G}_1$ through $\mathrm{G}_3$.
- $b \leftarrow \mathrm{G}_0$ denotes the security game $\mathrm{G}_0$ running and returning bit $b$.
- $\{\mathcal{X} \rightarrow \mathcal{Y}\}$ refers to the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

### 2.2 Key-Encapsulation Mechanisms (KEMs)

Below, we list standard definitions for KEMs. For further reference, see, e.g., [35]. Many proposed post-quantum KEMs are constructed from variants of the Fujisaki-Okamoto transformation, described in [27,28,32].

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

$$\mathsf{KEM} = (\mathsf{KEM.KeyGen}(1^n), \mathsf{KEM.Encaps}, \mathsf{KEM.Decaps}) \tag{10}$$

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

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

A KEM is said to be $\delta$-correct if the probability that an honestly-performed key establishment process results in $\mathsf{Encaps}$ and $\mathsf{Decaps}$ outputting different keys is at most $\delta$. Specifically,

$$\Pr[k' \neq k \mid (pk, sk) \leftarrow\!\$\ \mathsf{KEM.KeyGen}(1^n); (k, c) \leftarrow\!\$\ \mathsf{KEM.Encaps}(pk); k' \leftarrow \mathsf{KEM.Decaps}(c)] \leq \delta\,. \tag{11}$$

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

$$\text{Adv}^{\mathsf{KEM}}_{\text{IND-CCA},\mathcal{A}} := |\Pr[1 \leftarrow \text{IND-CCA}^0_{\mathsf{KEM},\mathcal{A}}] - \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathsf{KEM},\mathcal{A}}]|\,. \tag{12}$$

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 negligibly small 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 making classical queries to the Decaps oracle.

**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. This notion was formalized as ciphertext second preimage resistance by [13]. 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:

$$\mathrm{Adv}^{\mathsf{KEM}}_{\text{C2PRI},\mathcal{A}} = \Pr[1 \leftarrow \text{C2PRI}_{\mathsf{KEM},\mathcal{A}}] \tag{13}$$

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 and Dual Pseudorandom Functions

**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 were introduced in [30] as an ingredient in hybrid KEM constructions. 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 $i^{\text{th}}$ key as:

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} = \left|\Pr\left[1 \leftarrow \mathrm{PR}^0_{i,F,\mathcal{A}}\right] - \Pr\left[1 \leftarrow \mathrm{PR}^1_{i,F,\mathcal{A}}\right]\right| \tag{14}$$

**Algorithm 3** Split-Key Pseudorandom Function Game

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

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

$$\mathrm{Adv}^{\mathsf{split\text{-}key\text{-}PRF}}_{F,\mathcal{A}} = \max_i \{\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}}\} \tag{15}$$

**Dual Pseudorandom Functions.** The term dual PRF refers to a function of the form $F : \mathcal{K}_1 \times \mathcal{K}_2 \to \mathcal{Y}$ that behaves like a PRF when either input is hidden [15]. Dual PRFs can be thought of as split-key PRFs with two keys and $\mathcal{X} = \emptyset$. For this reason, we use the same notation and conventions for split-key and dual PRFs. In particular, for $i = 1, 2$, we still use the notation of $\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}}$ to refer to adversary $\mathcal{A}$'s ability to distinguish between $F$ and a random function when the $i^{\text{th}}$ key is obscured.

## 3 Security of Hybrid KEMs

### 3.1 Combiner Design

To combine two given KEMs, we follow the "Parallel KEM Combiner" approach of [30]. 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 $\mathsf{KEM}_i$, [30] 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 $c_2$. (A "robust hybrid KEM" retains its IND-CCA security so long as any single ingredient KEM does, see, e.g. [31].) In the case that the first KEM is ciphertext second preimage resistant, we will prove it is secure to output $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$ 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 $W(k_1, k_2, c_1, c_2) = F(k_1, k_2)$. An outline of how to generalize this construction to combine $n$ many KEMs is discussed in Appendix B.

**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** $(sk, pk)$

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

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

### 3.2 Security Theorems

In Theorem 6, we show our construction is secure when $\mathsf{KEM}_1$ is C2PRI and $\mathsf{KEM}_2$ 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 shows that when both KEMs are C2PRI, setting $W = F(k_1, k_2)$ for a dual PRF $F$ produces a robust hybrid KEM. Hence, Theorem 6 is sufficient to show Theorem 2. Notice this result is not enough to prove Theorem 1, 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 $\mathsf{KEM}_2$. Theorem 7 reduces the security of our construction to that of $\mathsf{KEM}_1$ without relying on any properties of $\mathsf{KEM}_2$, assuming that $c_2$ is given as input to the core function. Taken together, Theorems 6 and 7 imply Theorem 1, showing that when only $\mathsf{KEM}_1$ is C2PRI, setting $W = F(k_1, k_2, c_2)$ produces a robust hybrid KEM.

**Theorem 6.** *Let* $\mathsf{KEM}_1$ *be a C2PRI-secure KEM and* $\mathsf{KEM}_2$ *be a* $\delta$*-correct, IND-CCA KEM. Let* $F$ *be a split-key (resp. dual) PRF, and set* $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$ *(resp.* $F(k_1, k_2)$*). Let* $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ *be the combination of these KEMs given in Algorithm 4. Then for all QPT adversaries* $\mathcal{A}$ *against the IND-CCA security of* $\Pi$ *making at most* $q$ *queries to their decapsulation oracle, there exist QPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$*, and* $\mathcal{D}$ *such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\text{PR}_2}_{F,\mathcal{D}} + \delta) \tag{16}$$

*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 compared to* $\mathcal{A}$*.*

**Theorem 7.** *Let* $\mathsf{KEM}_1$ *be a* $\delta$*-correct, IND-CCA KEM and* $\mathsf{KEM}_2$ *be a KEM. Let* $F$ *be a split-key PRF and set* $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$*. Let* $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ *be the combination of these KEMs as described in Algorithm 4. Then for all QPT adversaries* $\mathcal{A}$ *against the IND-CCA security of* $\Pi$ *making at most* $q$ *classical queries to their decapsulation oracle, there exist QPT adversaries* $\mathcal{B}$ *and* $\mathcal{C}$ *such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\mathrm{PR}_1}_{F,\mathcal{C}} + \delta) \tag{17}$$

*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 compared to* $\mathcal{A}$*.*

*Proof.* The proofs of these two theorems are in Appendix A. □

## 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. While we are concerned with post-quantum secure split-key PRFs, the following theorem is stronger in the sense that it applies even when $\mathcal{A}$ is allowed to make quantum queries to its keyed evaluation oracle.

**Theorem 8.** *Let* $H$ *be a random oracle and let* $\mathcal{A}$ *be an adversary against the split-key pseudorandomness of the function* $F(k_1, \ldots, k_n, x) = H(k_1 || \ldots || k_n || x)$ *making at most* $q$ *quantum queries to* $H$*, where* $\mathcal{K}_i$ *denotes the* $i^{th}$ *keyspace. Then,*

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{18}$$

We believe this result is folklore, but we provide a proof in Appendix C for completeness. In what follows, we use this theorem to sketch arguments of split-key pseudorandomness for the key derivation functions recommended by NIST in [14].

**KMAC.** KMAC is a variable-length MAC/PRF scheme based on the customizable extendable output function cSHAKE. Both KMAC and cSHAKE are defined in [36]. In our context, we define KMAC as a split-key PRF to be:

$$\mathsf{KMAC}(k_1, ..., k_n, x) = \mathsf{cSHAKE}(salt||1||k_1||...||k_n||x, L, \text{“KMAC”}, \text{“KDF”}) \tag{19}$$

where *salt* is a fixed, previously agreed-upon, potentially secret string (having been pre-processed as outlined in [36]), $L$ is the fixed desired output length,

and 1, "KMAC", and "KDF" are fixed byte strings acting as domain separators. cSHAKE is a sponge function, so it is known to be indifferentiable from a random oracle against classical [16] and quantum [8] adversaries when Keccak, the underlying permutation, is assumed to be random. Therefore, Theorem 8 can be clearly extended to give a bound on the split-key pseudorandomness of KMAC.

**Hash Functions.** Following NIST's recommendations in [14], to construct a split-key PRF from a hash function $H$, we consider the form:

$$F(k_1, \ldots, k_n, x) = H(1||k_1|| \ldots ||k_n||x)||H(2||k_1|| \ldots ||k_n||x)|| \ldots \tag{20}$$

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 8 can again be used to bound split-key pseudorandomness in this setting.

**HMAC (One-Step).** HMAC is a message authentication code (MAC) defined in [1] over a hash function $H$ as:

$$\mathsf{HMAC}(k, m) = H\left((k \oplus opad)||H((k \oplus ipad)||m)\right) \tag{21}$$

where *opad* and *ipad* are fixed byte strings. Again following from [14], we construct a split-key PRF from HMAC by using the form:

$$F(k_1, \ldots, k_n, x) = \mathsf{HMAC}(salt, 1||k_1|| \ldots ||k_n||x)||\mathsf{HMAC}(salt, 2||k_1|| \ldots ||k_n||x)|| \ldots \tag{22}$$

where *salt* is once again a fixed string. As above, the counter acts as a domain separator that creates distinct input messages for HMAC. HMAC is known to be indistinguishable from random against both classical [29] and quantum [33] adversaries, so again, Theorem 8 applies.

**Two-Step Methods.** In addition to the above, [14] 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:

$$K_{DK} = \mathsf{MAC}(salt, k_1||...||k_n) \tag{23}$$

The allowable MAC algorithms are HMAC and CMAC. Next, $K_{DK}$ is used to derive the final output key material using a PRF-based KDF constructed in [20], which presents options to construct KDFs based on an underlying PRF. (The algorithm used as the PRF in this step must match the MAC from the first step.) In each option, one fixes a length $L$ and a label string and sets $F(k_1, ..., k_n, x) = K(1)||K(2)||...$ for some function $K$ and as many rounds as

necessary to produce a key of the desired length. The first option is *counter mode*, with

$$K(i) = \mathsf{PRF}(K_{DK}, i||Label||0||x||L) \tag{24}$$

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

$$K(i) = \mathsf{PRF}(K_{DK}, K(i-1)||i||Label||0||x||L) \tag{25}$$

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 [37]. Finally, in *double pipeline mode*, two PRF evaluations are used in each step. Specifically, we have:

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

where once again, the inclusion of $i$ is optional. Each of these modes is shown to be indistinguishable from a random function in [41] when instantiated with either HMAC or CMAC, so Theorem 8 shows that they are valid constructions for split-key PRFs.

## 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 [32]. $\mathcal{M}$ consists of bitstrings of the same length, and no properties about the underlying PKE scheme are assumed in our C2PRI proofs. Some well-known KEMs encompassed by these definitions include the $\mathsf{U}^{\not\perp}$ KEMs BIKE and HQC, as respectively specified in [3] and [6]. Other popular KEMs such as Classic McEliece are similar in structure to these KEMs [2]. The pseudocode for these classes of KEMs is shown in Algorithm 5.

### 5.2 Classical and Quantum C2PRI Proofs

**Theorem 9.** *Let* KEM *be a* $\mathsf{U}^{\perp}$ *or* $\mathsf{U}^{\not\perp}$ *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,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} \tag{26}$$

**Algorithm 5** $\mathsf{U}^{\perp}$ 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** $k$

*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 $H$ and messages are fixed-length. 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 10.** *Let* KEM *be a* $\mathsf{U}^{\perp}$ *or* $\mathsf{U}^{\not\perp}$ *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,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2 + q}{|\mathcal{K}|}} \tag{27}$$

*Proof.* First, let us generate a key $k \in \mathcal{K}$ using KEM.KeyGen and KEM.Encaps. Consider random oracles $H^0_0$ and $H^1_0$, where $H^0_0$ is defined to be the random oracle modeling $H$ and $H^1_0$ is defined to be $H^0_0$ punctured at inputs that map to $k$ such that those inputs now map to keys sampled from $\mathcal{K}\backslash\{k\}$. 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 $k \in \mathcal{K}$ 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 [10] any adversary $\mathcal{D}$ has advantage:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 2\sqrt{(d+1) \cdot P_{\mathrm{find}}} = 2\sqrt{(d+1) \cdot \frac{4q}{|\mathcal{K}|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{28}$$

in Algorithm 6, 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 $P_{\mathrm{find}}$ denotes the probability with which $\mathcal{D}$ can find an element of $S$.

**Algorithm 6** $H_0^0$ and $H_0^1$ 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 $H_0^1$ 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 7, where $\mathcal{D}$ calls on an adversary $\mathcal{A}$ against the ciphertext second preimage resistance of $\mathsf{KEM}^b$, which is a modified version of $\mathsf{KEM}$.

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 7 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 $\mathsf{KEM}^b$, which is a KEM with the same construction as $\mathsf{KEM}$ 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

**Algorithm 7** $H_0^0$ and $H_0^1$ Distinguishing Game with Adversary $\mathcal{A}$

**Game** $\text{DIST}_{\mathcal{D}}^{H_0}$
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 $\mathsf{KEM}^b$
$c' \leftarrow\$ \mathcal{A}(pk, sk, c, k)$
**if** $c' \neq c \wedge \mathsf{KEM}'.\mathsf{Decaps}(sk, c') = k$ **then**
  **return** 0
**return** 1

resistance game, yielding the result:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} = \Pr[1 \leftarrow \text{C2PRI}_{\mathsf{KEM}^0,\mathcal{A}}] \leq \Pr[0 \leftarrow \text{DIST}^{H_0^0}_{H_0,\mathcal{D}}] = \text{Adv}^{\text{DIST}}_{H_0,\mathcal{D}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{29}$$

since $\Pr[0 \leftarrow \text{DIST}^{H_0^1}_{H_0,\mathcal{D}}] = 0$ by the construction of $H_0^1$ and $\mathcal{D}$. An adversary $\mathcal{C}$ against the ciphertext second preimage resistance of $\mathsf{KEM}$ must have an advantage less than or equal to the advantage of $\mathcal{A}$ against the ciphertext second preimage resistance of $\mathsf{KEM}^0$, 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:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \text{Adv}^{\text{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{30}$$

as desired.

□

### 5.3 Applications to BIKE, Classic McEliece, and HQC

BIKE [3] and HQC [**?**] are standard $\mathsf{U}^{\not\perp}$ KEMs at the time of writing, so our results can be applied directly. HQC was updated on August 22, 2025 to hash the full ciphertext (instead of a saltless version of the ciphertext) in key derivation. Previous versions of HQC, such as the fourth-round NIST PQC standardization submission of HQC, implemented a variant of the FO Transform that did not include the full ciphertext in key derivation (namely, excluding the salt), enabling attacks that make those versions of HQC not ciphertext collision resistant [38]. However, we will show that the fourth-round submission of HQC is C2PRI,

demonstrating that there are relevant cases in which a KEM is not ciphertext collision resistant but still can be C2PRI.

**Corollary 1.** *Let* $\mathsf{KEM}$ *be an instantiation of BIKE [3] or HQC [6] with keyspace* $\mathcal{K}$ *and underlying hash function* $H$ *modeled as a random oracle. Let* $\mathcal{C}$ *be an adversary against C2PRI for* $\mathsf{KEM}$*. If* $\mathcal{C}$ *makes at most* $q$ *classical queries to* $H$*, then*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} \tag{31}$$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{32}$$

*Proof.* Follows immediately from Theorems 9 and 10. □

**Classic McEliece.** Classic McEliece is another KEM designed with a slight modification of the $\mathsf{U}^{\not\perp}$ transformation, described in [2]. 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), $\mathsf{Encode}$ (similar to encryption), and $\mathsf{Decode}$ (similar to decryption). Those algorithms are omitted here, as well as $\mathsf{KeyGen}$, but the encapsulation and decapsulation algorithms for Classic McEliece are shown in Algorithm 8.

**Algorithm 8** 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** $k$

**Corollary 2.** *Let* $\mathsf{KEM}$ *be an instantiation of Classic McEliece [2] with keyspace* $\mathcal{K}$ *and underlying hash function* $H$ *modeled as a random oracle. Let* $\mathcal{C}$ *be an adversary against C2PRI for* $\mathsf{KEM}$*. If* $\mathcal{C}$ *makes at most* $q$ *classical queries to* $H$*, then*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} \tag{33}$$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{34}$$

*Proof.* The domain separation used in session-key generation does not affect the proof of either Theorem 9 or 10 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 (Fourth-Round Submission).** The fourth-round submission of the Hamming Quasi-Cyclic (HQC) KEM in the NIST Post-Quantum Cryptography Standardization Process 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 [4]. 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 [38] is shown in Algorithm 9. The parameters $e, r_1, r_2, h, u, v, G$, and $\ell$ are parameters of the HQC PKE scheme, and $H_T$ derandomizes the scheme by using a random oracle $G$ to generate a randomness parameter $\theta$ given arguments $m, pk$, and *salt* and then using $\theta$ to generate $e, r_1$, and $r_2$ with prespecified Hamming weights. Henceforth, in this section, we will use "HQC" to exclusively refer to this particular version of HQC, and not the most recent version of HQC at the time of writing.

**Corollary 3.** *Let* $\mathsf{HQC}$ *be an instantiation of the fourth-round submission of HQC [4] with keyspace $\mathcal{K}$, message space $\mathcal{M}$, and underlying hash functions $H$ and $H_T$ modeled as random oracles. Let $\mathcal{C}$ be an adversary against C2PRI for* $\mathsf{HQC}$*. If $\mathcal{C}$ makes at most $q$ classical queries to $H$ and $q_{H_T}$ classical queries to $H_T$, then*

$$\mathrm{Adv}^{C2PRI}_{\mathsf{HQC},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} + \frac{q_{H_T}+1}{\binom{n}{w_r}} + \frac{1}{|\mathcal{M}|} \tag{35}$$

*where $n$ is the length of $h$, $r_1$, and $r_2$, and $w_r$ is the Hamming weight of $r_1$ and $r_2$. If $\mathcal{C}$ instead makes at most $q$ quantum queries to $H$ and $q_{H_T}$ quantum queries to $H_T$, then*

$$\mathrm{Adv}^{C2PRI}_{\mathsf{HQC},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} + 4\sqrt{\frac{q_{H_T}^2+q_{H_T}}{\binom{n}{w_r}}} + \frac{1}{|\mathcal{M}|} \tag{36}$$

*Proof.* 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

**Algorithm 9** HQC KEM (Fourth-Round Submission)

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

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 [38] 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 10.

**Algorithm 10** 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

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

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} + \text{Adv}^{\text{S2PRI}}_{\mathsf{HQC},\mathcal{B}} + \frac{1}{|\mathcal{M}|} \tag{37}$$

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:

$$\text{Adv}^{\text{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} + \text{Adv}^{\text{S2PRI}}_{\mathsf{HQC},\mathcal{B}} + \frac{1}{|\mathcal{M}|} \tag{38}$$

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 game. 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 [10] 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:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} + \frac{q_{H_T}+1}{\binom{n}{w_r}} + \frac{1}{|\mathcal{M}|} \tag{39}$$

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:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{HQC},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} + 4\sqrt{\frac{q_{H_T}^2 + q_{H_T}}{\binom{n}{w_r}}} + \frac{1}{|\mathcal{M}|} \tag{40}$$

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 [32]. $\mathcal{M}$ consists of bitstrings of the same length, and no properties of the underlying PKE scheme are assumed except for (1) deterministic encryption and (2) rejection in decryption if re-encrypting the decrypted message does not yield the same ciphertext as the input ciphertext (i.e. if normal decryption before the re-encryption check returns $m$ on inputs $sk$

and $c$ and if $m = \perp$ or $\mathsf{PKE.Encrypt}(pk, m) \neq c$, then $\mathsf{PKE.Decrypt}(sk, c) = \perp$). Together, these properties ensure that $\mathsf{PKE.Decrypt}$ does not output the same message for distinct ciphertexts and the same secret key, a fact which is used in the C2PRI proofs.

These properties are guaranteed for PKE schemes that are the result of the $\mathsf{T}$ transform in [32], where an initial PKE scheme $\mathsf{PKE}_0$ is derandomized with a random oracle $G$ and a re-encryption check. In [32], Hofheinz et al. use the $\mathsf{T}$ transform to transform OW-CPA secure PKE schemes into OW-PCA secure PKE schemes, which can then be transformed into IND-CCA KEMs, but $\mathsf{PKE}_0$ can be an arbitrary PKE scheme with no assumed properties for our results to hold. The pseudocode for the $\mathsf{T}$ transform is shown in Algorithm 11.

**Algorithm 11** The $\mathsf{T}$ Transform

$\underline{\mathsf{T}[\mathsf{PKE}_0, G].\mathsf{Encrypt}(pk, m)}$
$c \leftarrow \mathsf{PKE}_0.\mathsf{Encrypt}(pk, m; G(m))$
**return** $c$

$\underline{\mathsf{T}[\mathsf{PKE}_0, G].\mathsf{Decrypt}(sk, c)}$
$m \leftarrow \mathsf{PKE}_0.\mathsf{Decrypt}(sk, c)$
**if** $m = \perp \vee \mathsf{PKE}_0.\mathsf{Encrypt}(pk, m; G(m)) \neq c$
  **return** $\perp$
**return** $m$

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 12.

**Algorithm 12** $\mathsf{U}^{\perp}_m$ 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 11.** *Let* $\mathsf{KEM}$ *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* $\mathsf{KEM}$ *making at most* $q$ *classical queries to* $H$*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+1}{|\mathcal{K}|} \tag{41}$$

*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.$\mathsf{Decrypt}$ 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 9, 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 12.** *Let* $\mathsf{KEM}$ *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* $\mathsf{KEM}$ *making at most* $q$ *quantum queries to* $H$*. Then,*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{42}$$

*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:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 2\sqrt{(d+1)\cdot P_{\mathrm{find}}} = 2\sqrt{(d+1)\cdot\frac{4q}{|\mathcal{K}|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{43}$$

in the game depicted in Algorithm 13, where $b = 0$ corresponds to $H_0^0$ and $b = 1$ corresponds to $H_0^1$, 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 13** $H_0^0$ and $H_0^1$ 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 $H_0^1$ as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{D}^{H_0^b(\cdot)}(k)$
**return** $b'$

As in the proof in Section 5.2, we now examine the game in Algorithm 14, where $\mathcal{D}$ calls on an adversary $\mathcal{A}$ against the ciphertext second preimage resistance of $\mathsf{KEM}^b$, which is a modified version of $\mathsf{KEM}$.

**Algorithm 14** $H_0^0$ and $H_0^1$ 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 $H_0^1$ as above
$b \leftarrow\$ \{0,1\}$
$b' \leftarrow\$ \mathcal{D}^{H_0^b(\cdot)}(k)$
**return** $b'$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} = \Pr[1 \leftarrow \mathrm{C2PRI}_{\mathsf{KEM}^0,\mathcal{A}}] \leq \Pr[0 \leftarrow \mathrm{DIST}^{H_0^0}_{H_0,\mathcal{D}}] = \mathrm{Adv}^{\mathrm{DIST}}_{H_0,\mathcal{D}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{44}$$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}^0,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{45}$$

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 [5]. Notably, it uses 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.

**Corollary 4.** *Let* $\mathsf{KEM}$ *be an instantiation of ML-KEM [5] with keyspace* $\mathcal{K}$ *and underlying hash functions* $G$ *and* $J$ *modeled as random oracles. Let* $\mathcal{C}$ *be an adversary against C2PRI for* $\mathsf{KEM}$*. If* $\mathcal{C}$ *makes at most* $q$ *total classical queries to* $G$ *and* $J$*, then*

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq \frac{q+2}{|\mathcal{K}|} \tag{46}$$

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

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM},\mathcal{C}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}|}} \tag{47}$$

**Algorithm 15** 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** $k$

*Proof.* 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 [13]. 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 $H^b_1$ oracle inputs with 0 and 1 everywhere $G$ and $J$ would be respectively prepared and queried in $\mathsf{KEM}'$ and giving the oracle access to $G(\cdot) = H^b_1(0||\cdot)$ and $J(\cdot) = H^b_1(1||\cdot)$. 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. □

**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.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## A Proofs of Theorems 6 and 7

**Theorem 13 (Theorem 6 Restated).** *Let* $\mathsf{KEM}_1$ *be a C2PRI-secure KEM and* $\mathsf{KEM}_2$ *be a* $\delta$*-correct, IND-CCA KEM. Let* $F$ *be a split-key (resp. dual) PRF,*

*and set* $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$ *(resp.* $F(k_1, k_2)$*). Let* $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ *be the combination of these KEMs given in Algorithm 4. Then for all QPT adversaries* $\mathcal{A}$ *against the IND-CCA security of* $\Pi$ *making at most* $q$ *queries to their decapsulation oracle, there exist QPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$*, and* $\mathcal{D}$ *such that:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1,\mathcal{B}} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2,\mathcal{C}} + \mathrm{Adv}^{\mathrm{PR}_2}_{F,\mathcal{D}} + \delta) \tag{48}$$

*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 compared to* $\mathcal{A}$*.*

*Proof.* We will use a hybrid strategy similar to that in the proof of Theorem 1 of [30], as well as Theorems 1 and 2 of [13]. 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 $\mathrm{G}_0$. We then consider the adversary playing against game $\mathrm{G}_1$, where we slightly change in the decapsulation oracle, and argue that $\mathcal{A}$ can only detect this change with probability $\delta$. In $\mathrm{G}_2$, we have the decapsulation oracle abort when $\mathcal{A}$ inputs some $\mathsf{KEM}_1$ 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 $\mathsf{KEM}_1$. At this point, we can swap the $\mathsf{KEM}_1$ session key used as input to $F$ with a random key, bounding this change with the IND-CCA security of $\mathsf{KEM}_1$. 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 $\mathrm{G}_5$, we are nearly done with the proof. In game $\mathrm{G}_6 - \mathrm{G}_8$, we reverse the changes from the beginning of the proof, arguing in the same way as before that $\mathcal{A}$ cannot detect these modifications. $\mathrm{G}_8$ is equivalent to the IND-CCA$^1$ game, so having shown it is difficult for $\mathcal{A}$ to notice any changes between $\mathrm{G}_0$ and $\mathrm{G}_8$, we conclude with our bound for $\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}}$. We describe each individual hybrid below, and Algorithm 16 contains their pseudocode.

**Game 0.** Game $\mathrm{G}_0$ represents $\mathcal{A}$ playing in the IND-CCA experiment for $\Pi$ with the challenge bit set to 0. We run each KEM in parallel, just as $\Pi$ does, and provide an honest session key to $\mathcal{A}$. Therefore:

$$\Pr[1 \leftarrow \text{IND-CCA}^0_{\Pi,\mathcal{A}}] = \Pr[1 \leftarrow \mathrm{G}_0^{\mathcal{A}}] \tag{49}$$

**Game 1.** In game $\mathrm{G}_1$, 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 16). Games 0 and 1 should nearly always be identical, and they only differ when $\mathsf{KEM}_2$ would incorrectly decapsulate $c_2^*$. Decapsulation is deterministic, and we are only changing how the oracle decapsulates $c_2^*$ (not other $\mathsf{KEM}_2$ ciphertexts), so this $\mathrm{G}_0$ and $\mathrm{G}_1$ are different with probability at most $\delta$ regardless of how many times $\mathcal{A}$ queries the oracle. Therefore, we have:

$$\left|\Pr[1 \leftarrow \mathrm{G}_0^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_1^{\mathcal{A}}]\right| \leq \delta \tag{50}$$

**Algorithm 16** Hybrid games for the proof of Theorem 13

**Games** $\mathrm{G}_0 - \mathrm{G}_8$**:**
$W'[\cdot] \leftarrow \perp$ ▷ $\mathrm{G}_4$
$pk_1, sk_1 \leftarrow\$ \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow\$ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$k_1^*, c_1^* \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$k_2^*, c_2^* \leftarrow\$ \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$k_2^* \leftarrow\$ \mathcal{K}_2$ ▷ $\mathrm{G}_3 - \mathrm{G}_5$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$ ▷ $\mathrm{G}_4 - \mathrm{G}_8$
$W'[k_1^*] \leftarrow k^*$ ▷ $\mathrm{G}_4$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$

**Oracle** $\mathsf{Decaps}(c)$**:**
**if** $c = c^*$ **then**
  **return** $\perp$
$c_1, c_2 \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
**if** $c_2 = c_2^*$ **then** ▷ $\mathrm{G}_1 - \mathrm{G}_7$
  $k_2 \leftarrow k_2^*$ ▷ $\mathrm{G}_1 - \mathrm{G}_7$
**if** $k_1 = \perp \vee k_2 = \perp$ **then**
  **return** $\perp$
**if** $k_1 = k_1^* \wedge c_1 \neq c_1^*$ **then** ▷ $\mathrm{G}_2 - \mathrm{G}_6$
  $\mathrm{abort}_1$ ▷ $\mathrm{G}_2 - \mathrm{G}_6$
**if** $c_2 = c_2^*$ **then** ▷ $\mathrm{G}_4$
  **if** $W'[k_1] = \perp$: ▷ $\mathrm{G}_4$
    $W'[k_1] \leftarrow\$ \mathcal{K}$ ▷ $\mathrm{G}_4$
  **return** $W'[k_1]$ ▷ $\mathrm{G}_4$
**return** $W(k_1, k_2, c_1, c_2)$

**Game 2.** In game $\mathrm{G}_2$, 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 $\mathsf{KEM}_1$'s ciphertext second preimage resistance game, so we claim there exists a QPT adversary $\mathcal{B}_1$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_1^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_2^{\mathcal{A}}]\right| \leq \Pr[\mathrm{abort}_1] = \mathrm{Adv}^{\mathrm{C2PRI}}_{\mathsf{KEM}_1, \mathcal{B}_1} \tag{51}$$

The pseudocode for adversary $\mathcal{B}_1$ is shown in Algorithm 17. When starting its challenge, $\mathcal{B}_1$ receives a public/secret key pair, ciphertext, and session key for $\mathsf{KEM}_1$. It runs key generation and encapsulation for $\mathsf{KEM}_2$, 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 $\mathrm{G}_2$, 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.

**Game 3.** In game 3, we replace $k_2^*$, the session key for $\mathsf{KEM}_2$, with a key generated uniformly at random. This should be difficult for $\mathcal{A}$ to detect so long at $\mathsf{KEM}_2$ retains IND-CCA security. Indeed, we claim there exists a QPT adversary $C_1$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_2^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2, \mathcal{C}_1} \tag{52}$$

The pseudocode to construct $\mathcal{C}_1$ is shown in Algorithm 18. $\mathcal{C}_1$ receives a public key, ciphertext, and session key for $\mathsf{KEM}_2$. It creates keys and a ciphertext for

**Algorithm 17** Adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ against the ciphertext second preimage resistance of $\mathsf{KEM}_1$. The commented line is only executed for $\mathcal{B}_2$.

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

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

$\mathsf{KEM}_1$ 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 $\mathrm{G}_1$, this is the correct way of simulating games $\mathrm{G}_2$ and $\mathrm{G}_3$ 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 $\mathrm{G}_2$ for $\mathcal{A}$. When instantiated on the IND-CCA$^1$ game, it receives a random key and therefore simulates $\mathrm{G}_3$ 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 $\mathrm{G}_5$. We claim there exists a QPT adversary $\mathcal{D}_1$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\mathrm{PR}_2}_{F,\mathcal{D}_1} \tag{53}$$

We construct adversary $\mathcal{D}_1$ in Algorithm 19. (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 on queries to $\mathsf{Eval}$.) $\mathcal{D}_1$ receives no input and runs key generation and encapsulation for both KEMs. When instantiated with game $\mathrm{PR}_2^0$, $\mathcal{D}_1$ makes evaluation queries to $F$ keyed with $k_1^*$ and a random element of $\mathcal{K}_2$, so $k^*$ is computed exactly as in $\mathrm{G}_3$. 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 $\mathsf{Eval}$ instead of computing the

**Algorithm 18** IND-CCA adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ against $\mathsf{KEM}_2$. The commented line is only executed for $\mathcal{C}_2$.

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

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

session key with $F$. Notice in the $\mathrm{PR}_2^0$ setting, $\mathcal{D}_1$ generates $k_2^*$ but never uses it, instead using the randomly-generated PRF key for its challenge. Since $\mathrm{G}_3$ samples $k_2^*$ at random, $\mathcal{D}_1$ is perfectly simulating $\mathrm{G}_3$ for $\mathcal{A}$ in this setting. When $\mathcal{D}_1$ is instantiated on $\mathrm{PR}_2^1$, 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 $\mathrm{G}_4$ instead of returning a randomly sampled element of $\mathcal{K}$ whenever $\mathcal{A}$ queries the decapsulation oracle with $c_2^*$. In the $\mathrm{PR}_2^1$ case, $\mathcal{D}_1$ is perfectly simulating game $\mathrm{G}_4$ for $\mathcal{A}$, so the claim follows.

**Game 5.** In game 5, we remove the table $W'$ used in $\mathrm{G}_4$ and answer decapsulation queries as in game 3. We claim there exists a QPT adversary $\mathcal{D}_2$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_5^{\mathcal{A}}]\right| \leq \mathrm{Adv}_{F,\mathcal{D}_2}^{\mathrm{PR}_2} \tag{54}$$

The pseudocode for $\mathcal{D}_2$ is also shown in Algorithm 19 (as in the previous game, Algorithm 19 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 $\mathrm{PR}_2^1$, it returns random elements of $\mathcal{K}$ on decapsulation queries including $c_2^*$, simulating $\mathrm{G}_4$ for $\mathcal{A}$. $\mathcal{D}_2$ does differ slightly from $\mathrm{G}_4$: in $\mathrm{G}_4$, $k^* = F'[k_1^*]$, 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 $\mathsf{Eval}$ 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 $\mathrm{PR}_2^1$, it is still simulating $\mathrm{G}_4$ for $\mathcal{A}$. When

**Algorithm 19** Adversaries $\mathcal{D}_1$ and $\mathcal{D}_2$ against the split-key pseudorandomness of $F$, keyed on the second input. The commented line is only executed for $\mathcal{D}_2$.

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

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

**If** $\mathcal{A}$ **Calls** $\mathsf{Decaps}(c)$**:**

**if** $c = c^*$ **then**
  **return** $\perp$
$c_1, c_2 \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
**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**
  $\text{abort}_1$
**if** $c_2 = c_2^*$ **then**
  **return** $\mathsf{Eval}(k_1, c_2)$
**return** $W(k_1, k_2, c_1, c_2)$

$\mathcal{D}_2$ is instantiated on $\text{PR}_2^0$, 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 $\text{G}_5$ 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 $\text{G}_3$ and once again generate $k_2^*$ honestly. Similar to game 3, we claim there exists a QPT adversary $\mathcal{C}_2$ such that:

$$\left|\Pr[1 \leftarrow \text{G}_5^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_6^{\mathcal{A}}]\right| \leq \text{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_2, \mathcal{C}_2} \tag{55}$$

$\mathcal{C}_2$ is very similar to $\mathcal{C}_1$ and is shown in Algorithm 18. When instantiated on IND-CCA$^1$, $\mathcal{C}_2$ uses a random element of $\mathcal{K}_2$ as $k_2^*$ and as such, simulates $\text{G}_5$ 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 $\text{G}_6$. Therefore, the claim holds.

**Game 7.** In game 7, we undo the change from $\text{G}_2$ 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:

$$\left|\Pr[1 \leftarrow \text{G}_6^{\mathcal{A}}] - \Pr[1 \leftarrow \text{G}_7^{\mathcal{A}}]\right| \leq \Pr[\text{abort}_1] = \text{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_1, \mathcal{B}_2} \tag{56}$$

Once again, $\mathcal{B}_2$ is similar to $\mathcal{B}_1$ and is shown in Algorithm 17. $\mathcal{B}_2$ receives a public/secret key pair, ciphertext, and session key for $\mathsf{KEM}_1$ and creates the same for $\mathsf{KEM}_2$. 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:

$$\left|\Pr[1 \leftarrow G_7^{\mathcal{A}}] - \Pr[1 \leftarrow G_8^{\mathcal{A}}]\right| \leq \delta \tag{57}$$

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:

$$\Pr[1 \leftarrow G_8^{\mathcal{A}}] = \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathcal{A}}] \tag{58}$$

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 14 (Theorem 7 Restated).** *Let* $\mathsf{KEM}_1$ *be a* $\delta$*-correct, IND-CCA KEM and* $\mathsf{KEM}_2$ *be a KEM. Let* $F$ *be a split-key PRF and set* $W(k_1, k_2, c_1, c_2) = F(k_1, k_2, c_2)$*. Let* $\Pi = \Pi(\mathsf{KEM}_1, \mathsf{KEM}_2, W)$ *be the combination of these KEMs as described in Algorithm 4. Then for all QPT adversaries* $\mathcal{A}$ *against the IND-CCA security of* $\Pi$ *making at most* $q$ *classical queries to their decapsulation oracle, there exist QPT adversaries* $\mathcal{B}$ *and* $\mathcal{C}$ *such that:*

$$\text{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2(\text{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1,\mathcal{B}} + \text{Adv}^{\text{PR}_1}_{F,\mathcal{C}} + \delta) \tag{59}$$

*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 compared to* $\mathcal{A}$*.*

*Proof.* The proof of this theorem follows in much the same way as Theorem 13 and is once again similar to Theorem 1 of [30] and Theorems 1-2 of [13]. 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 20.

**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 13. 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 $\delta$. In game 2, we sample $k_1^*$ randomly from $\mathcal{K}_1$, justifying this change with the IND-CCA security of $\mathsf{KEM}_1$. In game 3, we choose $k^*$ uniformly at random. Additionally, just like in $G_4$ of Theorem 13, on decapsulation queries with $c_1 = c_1^*$, we return random elements of $\mathcal{K}$, using the table $W'$ to maintain

**Algorithm 20** Hybrid games for the proof of Theorem 14

**Games** $\mathrm{G}_0 - \mathrm{G}_6$**:**

$W'[\cdot] \leftarrow \perp$ ▷ $\mathrm{G}_3$
$pk_1, sk_1 \leftarrow\$ \mathsf{KEM}_1.\mathsf{KeyGen}(1^n)$
$pk_2, sk_2 \leftarrow\$ \mathsf{KEM}_2.\mathsf{KeyGen}(1^n)$
$pk \leftarrow (pk_1, pk_2)$
$k_1^*, c_1^* \leftarrow\$ \mathsf{KEM}_1.\mathsf{Encaps}(pk_1)$
$k_2^*, c_2^* \leftarrow\$ \mathsf{KEM}_2.\mathsf{Encaps}(pk_2)$
$k_1^* \leftarrow\$ \mathcal{K}_1$ ▷ $\mathrm{G}_2 - \mathrm{G}_4$
$c^* \leftarrow (c_1^*, c_2^*)$
$k^* \leftarrow W(k_1^*, k_2^*, c_1^*, c_2^*)$
$k^* \leftarrow\$ \mathcal{K}$ ▷ $\mathrm{G}_3 - \mathrm{G}_6$
$W'[k_2^*, c_2^*] \leftarrow k^*$ ▷ $\mathrm{G}_3$
$b' \leftarrow\$ \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, k^*)$

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

**if** $c = c^*$ **then**
  **return** $\perp$
$c_1, c_2 \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
**if** $c_1 = c_1^*$ **then** ▷ $\mathrm{G}_1 - \mathrm{G}_5$
  $k_1 \leftarrow k_1^*$ ▷ $\mathrm{G}_1 - \mathrm{G}_5$
**if** $k_1 = \perp \vee k_2 = \perp$ **then**
  **return** $\perp$
**if** $c_1 = c_1^*$ **then** ▷ $\mathrm{G}_3$
  **if** $W'[k_2, c_2] = \perp$: ▷ $\mathrm{G}_3$
    $W'[k_2, c_2] \leftarrow\$ \mathcal{K}$ ▷ $\mathrm{G}_3$
  **return** $W'[k_2, c_2]$ ▷ $\mathrm{G}_3$
**return** $W(k_1, k_2, c_1, c_2)$

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:

$$\left|\Pr[1 \leftarrow \text{IND-CCA}^0_{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}]\right| \leq \delta + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1, \mathcal{B}_1} + \mathrm{Adv}^{\mathrm{PR}_1}_{F, \mathcal{C}_1} \tag{60}$$

**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:

$$\left|\Pr[1 \leftarrow \mathrm{G}_3^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\mathrm{PR}_1}_{F, \mathcal{C}_2} \tag{61}$$

The pseudocode to construct $\mathcal{C}_2$ is shown in Algorithm 21. 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 $\mathrm{PR}_1^0$, the $\mathsf{Eval}$ queries return honest evaluations of $F$ keyed on a random element of $\mathcal{K}_1$, so $\mathcal{D}$ is properly simulating $\mathrm{G}_4$ for $\mathcal{A}$. In the $\mathrm{PR}_1^1$ setting, the evaluation queries return random elements of $\mathcal{K}$. This nearly simulates $\mathrm{G}_3$, except that in $\mathrm{G}_3$, $k^* = F'[k_2^*, c_2^*]$, but $\mathcal{C}_2$ does not guarantee that $k^* = \mathsf{Eval}(k_2^*, c_2^*)$. This means that $\mathcal{C}_2$ only replicates $\mathrm{G}_3$ so long as $\mathcal{A}$ cannot recover $\mathsf{Eval}(k_2^*, c_2^*)$ from the decapsulation oracle. However, the oracle only returns $\mathsf{Eval}$ queries on inputs with $c_1 = c_1^*$. We would also need $c_2 = c_2^*$ for the decapsulation oracle to return an $\mathsf{Eval}$ query containing $c_2^*$, but if $c_1 = c_1^*$ and $c_2 = c_2^*$ then $\mathsf{Decaps}$ returns $\perp$. Therefore, when $\mathcal{C}_2$ is instantiated on $\mathrm{PR}_1^1$, it can perfectly simulate $\mathrm{G}_3$ 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

**Algorithm 21** 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** $\perp$
$c_1, c_2 \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}_1.\mathsf{Decaps}(sk_1, c_1)$
$k_2 \leftarrow \mathsf{KEM}_2.\mathsf{Decaps}(sk_2, c_2)$
**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** $W(k_1, k_2, c_1, c_2)$

difference with the IND-CCA security of $\mathsf{KEM}_1$. In game 6, we stop setting $k_1 \leftarrow k_1^*$ on queries $c_1 \leftarrow c_1^*$, justifying this with the correctness of $\mathsf{KEM}_1$. Therefore, there exists a QPT advesary $\mathcal{B}_2$ such that:

$$\left|\Pr[1 \leftarrow \mathrm{G}_4^{\mathcal{A}}] - \Pr[1 \leftarrow \mathrm{G}_6^{\mathcal{A}}]\right| \leq \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_1, \mathcal{C}_2} \tag{62}$$

But $\mathrm{G}_6$ is equivalent to the IND-CCA game with challenge bit set to 1, so:

$$\Pr[1 \leftarrow \mathrm{G}_6^{\mathcal{A}}] = \Pr[1 \leftarrow \text{IND-CCA}^1_{\mathcal{A}}] \tag{63}$$

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. □

## B 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 22. Supposing KEMs 1 through $\ell$ 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 $\Pi$.

**Theorem 15.** *Let* $\mathsf{KEM}_1, ..., \mathsf{KEM}_n$ *be KEMs, where* $\mathsf{KEM}_i$ *is* $\delta$*-correct. Let* $F$ *be a split-key PRF, and set* $W(k_1, ..., k_n, c_1, ..., c_n) = F(k_1, ..., k_n, c_{\ell+1}, c_{\ell+2}, ..., c_n)$. *Let* $\Pi = \Pi(\mathsf{KEM}_1, ..., \mathsf{KEM}_n, W)$ *be the combination of these KEMs given in Algorithm 22. Then for all QPT adversaries* $\mathcal{A}$ *against the* IND-CCA *security*

**Algorithm 22** $n$-KEM Hybrid design

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

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

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

*of $\Pi$ 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:*

$$\mathrm{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq 2\left(\left(\sum_{j=1, j\neq i}^{\ell} \mathrm{Adv}^{\text{C2PRI}}_{\mathsf{KEM}_j,\mathcal{B}_j}\right) + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_i,\mathcal{C}} + \mathrm{Adv}^{\text{PR}_i}_{F,\mathcal{D}} + \delta\right) \quad (64)$$

*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 compared to $\mathcal{A}$.*

*Proof sketch.* This theorem can be proved using the same structure as Theorem 13, 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 $j^{\text{th}}$ coordinate $c_j$ is different from the $j^{\text{th}}$ challenge ciphertext $c^*_j$ but decapsulates to $k^*_j$, the $j^{\text{th}}$ 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 13. 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 $\mathsf{KEM}_i$, as well as a ciphertext from each of $\mathsf{KEM}_{\ell+1}, ..., \mathsf{KEM}_n$, again except $\mathsf{KEM}_i$ (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 13, $\mathcal{D}_2$ 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.

## C A proof of Theorem 4

**Theorem 16 (Theorem 4/Theorem 8 Restated).** *Let $H$ be a random oracle and let $\mathcal{A}$ be an adversary against the split-key pseudorandomness of the function $F(k_1, \ldots, k_n, x) = H(k_1 || \ldots || k_n || x)$ making at most $q$ quantum queries to $H$, where $\mathcal{K}_i$ denotes the $i^{th}$ keyspace. Then,*

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{65}$$

*Proof.* Denote $H$ by $H_0$. We start by considering the game in Algorithm 23, 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 $i^{\text{th}}$ key position in the argument of $H_0$.

**Algorithm 23** Random Oracle Distinguishing Game

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

**Oracle** $\mathrm{Eval}^{k_i}(k', x)$
$k_1 ... k_{i-1} k_{i+1} ... k \leftarrow k'$
$y \leftarrow H_0(k_1, ..., k_i, ..., k_n, x)$
**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 [10], any adversary $\mathcal{D}$ has a bounded advantage:

$$\mathrm{Adv}^{\mathrm{DIST}}_{H_b,\mathcal{D}} \leq 2\sqrt{(d+1)\cdot P_{\mathrm{find}}} = 2\sqrt{(d+1)\cdot\frac{4q}{|\mathcal{K}_i|}} = 4\sqrt{\frac{q(d+1)}{|\mathcal{K}_i|}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{66}$$

in this game. We now consider a modified version of this game, as shown in Algorithm 24, 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 $\text{Eval}_0^{k_i}$ or $\text{Eval}_1^{k_i}$, with $\text{Eval}_0^{k_i}$ being the Eval oracle in Algorithm 23 and $\text{Eval}_1^{k_i}$ having the same construction as $\text{Eval}_0$ but using $H_1$ rather than $H_0$. The adversary is also given access to $H_0$.

**Algorithm 24** Eval Oracle Distinguishing Game

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

**Oracle** $\text{Eval}_b^{k_i}(k', x)$
$k_1 ... k_{i-1} k_{i+1} ... k \leftarrow k'$
$y \leftarrow H_b(k_1, ..., k_i, ..., k_n, x)$
**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 $\text{Eval}_0^i$, 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 $\text{Eval}_0^{k_i}$ while the adversary in the $\text{Eval}_b$ distinguishing game has access to $H_0$ and $\text{Eval}_1$. By the fact that $H_0$ and $H_1$ are random oracles that can differ only when $k_i$ is in the $i^{\text{th}}$ key position in the argument, the setting where the adversary has access to $H_1$ and $\text{Eval}_0^{k_i}$ and the setting where the adversary has access to $H_0$ and $\text{Eval}_1^{k_i}$ 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 $\text{Eval}_b^i$ distinguishing game must be bounded by the advantage of an adversary in the $\text{H}_b$ distinguishing game:

$$\text{Adv}^{\text{DIST}}_{\text{Eval}_b,\mathcal{E}} \leq \text{Adv}^{\text{DIST}}_{H_b,\mathcal{D}} \tag{67}$$

We now claim that the $\text{Eval}_b^{k_i}$ 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 out-

put 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:

$$\mathrm{Adv}^{\mathrm{PR}_i}_{F,\mathcal{A}} = \mathrm{Adv}^{\mathrm{DIST}}_{\mathrm{Eval}_b,\mathcal{E}} \leq \mathrm{Adv}^{\mathrm{DIST}}_{H_b,\mathcal{D}} \leq 4\sqrt{\frac{q^2+q}{|\mathcal{K}_i|}} \tag{68}$$

as desired. □

## References


1. The keyed-hash message authentication code (HMAC). FIPS PUB 198-1 (Jul 2008), https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.198-1.pdf
2. Classic McEliece: conservative code-based cryptography: cryptosystem specification (2022), https://classic.mceliece.org/mceliece-spec-20221023.pdf
3. BIKE: Bit flipping key encapsulation (2024), https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf
4. Hamming quasi-cyclic (HQC) (Oct 2024), can be downloaded from https://pqc-hqc.org/doc/archive_submissions.zip
5. Module-lattice-based key-encapsulation mechanism standard. FIPS 203 (Aug 2024), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf
6. Hamming quasi-cyclic (HQC) (2025), https://pqc-hqc.org/doc/hqc_specifications_2025_08_22.pdf
7. Alagic, G., Bros, M., Ciadoux, P., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Silberg, H., Smith-Tone, D., Waller, N.: Status report on the fourth round of the NIST post-quantum cryptography standardization process (2025-03-11 04:03:00 2025). https://doi.org/https://doi.org/10.6028/NIST.IR.8545, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=959556
8. Alagic, G., Carolan, J., Majenz, C., Tokat, S.: The sponge is quantum indifferentiable (2025), https://arxiv.org/abs/2504.16887
9. Alwen, J., Hartmann, D., Kiltz, E., Mularczyk, M., Schwabe, P.: Post-quantum multi-recipient public key encryption. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 1108–1122. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3623185, https://doi.org/10.1145/3576915.3623185
10. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Advances in Cryptology – CRYPTO 2019 (2021)
11. Aviram, N., Dowling, B., Komargodski, I., Paterson, K.G., Ronen, E., Yogev, E.: Practical (post-quantum) key combiners from one-wayness and applications to TLS. Cryptology ePrint Archive, Paper 2022/065 (2022), https://eprint.iacr.org/2022/065
12. Backendal, M., Bellare, M., Günther, F., Scarlata, M.: When messages are keys: Is hmac a dual-prf? In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 661–693. Springer Nature Switzerland, Cham (2023)
13. Barbosa, M., Connolly, D., Duarte, J.D., Kaiser, A., Schwabe, P., Varner, K., Westerbaan, B.: X-wing. IACR Communications in Cryptology **1**(1) (2024). https://doi.org/10.62056/a3qj89n4e
14. Barker, E., Chen, L., Davis, R.: Recommendation for key-derivation methods in key-establishment schemes. NIST SP 800-56C Rev. 2 (Aug 2020), https://doi.org/10.6028/NIST.SP.800-56Cr2

15. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006. pp. 602–619. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
16. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 181–197. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
17. Bhargavan, K., Jacomme, C., Kiefer, F., Schmidt, R.: Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 469–486. USENIX Association, Philadelphia, PA (Aug 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/bhargavan
18. Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D.: Hybrid key encapsulation mechanisms and authenticated key exchange. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 206–226. Springer International Publishing, Cham (2019)
19. Campagna, M., Petcher, A.: Security of hybrid key encapsulation. Cryptology ePrint Archive, Paper 2020/1364 (2020), https://eprint.iacr.org/2020/1364
20. Chen, L.: Recommendation for key derivation using pseudorandom functions. NIST SP 800-108r1-upd1 (Aug 2022), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1-upd1.pdf
21. Chevalier, C., Lebrun, G., Martinelli, A.: Spilling-Cascade: an Optimal PKE Combiner for KEM Hybridization. In: 23rd International Conference on Applied Cryptography and Network Security (ACNS'25). Munich, Germany (Jun 2025), https://hal.science/hal-05027882
22. Connolly, D., Hövelmanns, K., Hülsing, A., Kousidis, S., Meijers, M.: Starfighters — on the general applicability of x-wing. Cryptology ePrint Archive, Paper 2025/1397 (2025), https://eprint.iacr.org/2025/1397
23. Cremers, C., Dax, A., Medinger, N.: 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. p. 1046–1060. CCS '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3658644.3670283, https://doi.org/10.1145/3658644.3670283
24. Don, J., Fehr, S., Huang, Y.H.: 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. p. 33–51. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-22318-1_2, https://doi.org/10.1007/978-3-031-22318-1_2
25. Federal Office for Information Security (BSI): Quantum-safe cryptography - fundamentals, current developments and recommendations (May 2022), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf?__blob=publicationFile&v=6
26. 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 (Jan 2024), https://cyber.gouv.fr/sites/default/files/document/Quantum_Key_Distribution_Position_Paper.pdf
27. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 537–554. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

28. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (2013). https://doi.org/10.1007/s00145-011-9114-1, https://doi.org/10.1007/s00145-011-9114-1
29. Gaži, P., Pietrzak, K., Rybár, M.: The exact PRF-security of NMAC and HMAC. In: Garay, J., Gennaro, R. (eds.) Advances in Cryptology — CRYPTO 2014. pp. 113–130. Springer (2014)
30. Giacon, F., Heuer, F., Poettering, B.: Kem combiners. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography – PKC 2018. pp. 190–218. Springer International Publishing, Cham (2018)
31. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. pp. 96–113. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
32. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 341–371. Springer International Publishing, Cham (2017)
33. Hosoyamada, A., Iwata, T.: On tight quantum security of HMAC and NMAC in the quantum random oracle model. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 585–615. Springer International Publishing, Cham (2021)
34. Huguenin-Dumittan, L., Vaudenay, S.: Fo-like combiners and hybrid post-quantum cryptography. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security. pp. 225–244. Springer International Publishing, Cham (2021)
35. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press, third edn. (2021)
36. Kelsey, J., Chang, S.j., Perlner, R.: SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash . NIST SP 800-185 (2016), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf
37. Krawczyk, H., Eronen, P.: HMAC-based extract-and-expand key derivation function (HKDF). RFC 5869 (May 2010). https://doi.org/10.17487/RFC5869, https://www.rfc-editor.org/info/rfc5869
38. Krämer, J., Struck, P., Weishäupl, M.: Binding security of implicitly-rejecting KEMs and application to BIKE and HQC. Cryptology ePrint Archive, Paper 2024/1233 (2024), https://eprint.iacr.org/2024/1233
39. Liu, Y., Zhou, B., Jiang, H.: CuKEM: A concise and unified hybrid key encapsulation mechanism. Cryptology ePrint Archive, Paper 2025/1862 (2025). https://doi.org/10.1145/3719027.3744863, https://eprint.iacr.org/2025/1862
40. O'Brien, D.: Protecting chrome traffic with hybrid kyber KEM. Chromium Blog (Aug 2023), https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html
41. Shen, Y., Wang, L., Gu, D.: Security analysis of NIST key derivation using pseudorandom functions. Cryptology ePrint Archive, Paper 2025/815 (2025), https://eprint.iacr.org/2025/815
42. Xu, J., Gao, Y., Lim, H.W., Wang, H., Chang, E.C.: Stateful KEM: Towards optimal robust combiner for key encapsulation mechanism. Cryptology ePrint Archive, Paper 2021/989 (2021), https://eprint.iacr.org/2021/989