Binding Security of Combined KEMs
An Analysis of Real-World KEM Combiners

Juliane Krämer[1], Patrick Struck[2], and Maximiliane Weishäupl[1]

[1] Universität Regensburg, Germany
{juliane.kraemer,maximiliane.weishaeupl}@ur.de
[2] Universität Konstanz, Germany
patrick.struck@uni-konstanz.de

Abstract. In this work we analyze the various binding properties of combiners for KEMs. We show that several binding properties follow easily for the most general combiner—assuming a collision-resistant hash function—while more performance-oriented combiners require the respective property from one or both of the underlying KEMs. Other binding properties are not obtained as directly and require either more properties of one underlying KEM or the respective property from both KEMs. Additionally, we provide an overview of the binding properties for several combinations of real-world KEMs.

1 Introduction

Key-encapsulation mechanisms (KEMs) generate a shared key k along with a ciphertext c, using a public key pk. With the corresponding secret key sk, the shared key k can be recovered from c, thereby enabling the receiver of a ciphertext to obtain the same shared key as the sender. The standard security notion for KEMs is IND-CCA which ensures that c does not reveal information about k. A very common method to achieve IND-CCA security is via the Fujisaki-Okamoto transform [HHK17], which was used by most KEM submissions to the NIST standardization process for post-quantum cryptography and is also used in the final standard ML-KEM [NIST24]. Binding properties are additional security notions that prevent misuse of KEMs in more complex protocols. KEMs with binding properties are desirable as they can be deployed without the need of analyzing the protocol against several misuse attacks. Cremers et al. [CDM24] provided a framework, consisting of 6 *general notions* and 3 *attack models*, totalling 18 binding notions. For instance, the notion LEAK-BIND-K,PK-CT formalizes the idea that the shared key (K) and public key (PK) together bind the ciphertext (CT) in the leakage attack model (LEAK), meaning it should be infeasible for any adversary (that receives both public and secret key) to find two different ciphertexts that decapsulate to the same shared key.

* We thank Stefan Winderl for helpful discussions and an anonymous reviewer for valuable comments regarding the proofs. This work was funded by the BMFTR under the projects QUDIS (16KIS2091), 6G-RIC (16KISK033), and Quant-ID (16KISQ111), as well as by the Hector Foundation II.

Combiners allow to combine two or more KEMs into a new KEM. The idea is that the combined KEM should retain security as long as *at least one* of the underlying KEMs is secure. KEM combiners will play a crucial role in the transition towards post-quantum cryptography (PQC), when a classical KEM is combined with a post-quantum KEM [MPR+24]. Such KEMs, which are referred to as hybrid KEMs, maintain security against classical attacks even if quantum computers break the classical KEM or the post-quantum KEM should turn out to be insecure.[3] While KEM combiners are well-studied regarding the standard security notion IND-CCA [GHP18,BBF+19,HV21,CLM25,CP20], they have not been formally analyzed with respect to their binding properties.

1.1 Contribution

In this work, we close the aforementioned gap by analyzing the binding properties of KEM combiners. Our results are twofold as described in the following.

First, we consider a generic combiner—from which several real-world KEM combiners can be derived—and provide bounds regarding the 18 binding security notions. This combiner relies on the underlying KEMs to generate their shared keys and ciphertexts and then computes the final shared key by combining shared key, public key, and ciphertext of both KEMs via some generic function Combine. Specific KEM combiners can be obtained by specifying Combine and which of the inputs it uses, e.g., the OpenPGP-Comb combiner [KRSW25] is obtained by defining Combine to be SHA3-256, applied to the shared keys, together with the ciphertext and public key from only one of the underlying KEMs. We show that the generic combiner can achieve all 12 binding notions of the form where the shared key binds other components, based on Combine being collision-resistant—if both the public keys and ciphertexts are inputs to Combine. If certain inputs to Combine are dropped, security is maintained if the respective KEM achieves the binding notion itself. For the remaining 6 notions, which formalize if the ciphertext binds the shared key or the public key, the results are qualitatively different. For the notions X-Bind-CT-PK and X-Bind-CT-K, security of the combiner is independent of Combine. In case of X-Bind-CT-PK, we obtain a "hybrid" bound, i.e., security of the combined KEM follows if at least one of the underlying KEMs is secure—though only in the Hon and Leak setting. In the Mal setting, both underlying KEMs need to be secure. For X-Bind-CT-K both KEMs need to be secure for the combined KEM to be X-Bind-CT-K, no matter the attack model. However, if one underlying KEM achieves X-Bind-CT-K *and* X-Bind-K,CT-PK, the combined KEM achieves X-Bind-CT-K security, though, again, only in the Hon and Leak setting.

Second, we apply our generic results to concrete KEM combiners and consider instantiations using KEMs for which the binding properties have been analyzed: Ml-Kem, Hqc, DH-Kem, FrodoKem, Bike, and Classic-McEliece. For

[3] Attacks against schemes in the NIST standardization process have shown that even schemes that have been scrutinized might exhibit vulnerabilities.

these KEMs, the relevant binding notions are either achieved entirely or partially but never not at all. This implies that hashing all components (shared key, ciphertext, and public key) of one KEM is typically unnecessary to achieve all binding properties. Instead, it turns out that merely adding the public key is often sufficient, though none of the considered combiners is of that form.

1.2 Related Work

Anonymity, introduced in [BBDP01], ensures that a ciphertext does not reveal under which key it was generated. The notion of robust (public-key) encryption can be traced back to [ABN10, FLPQ13] and plays a crucial role in achieving anonymity. Binding security as formalized in [CDM24] generalizes the idea of robustness. Robustness analyses for the KEMs that were part of the NIST PQC standardization process [NIS17] are given in [GMP22,Xag22]. Binding properties of the implicitly-rejecting Fujisaki-Okamoto transform, together with an analysis of BIKE and HQC, are given in [KSW25].

Several works [GHP18, ADK+22, BBF+19, ABK25, CHH+25] describe and/or analyze various combiners for KEMs. Furthermore, there are numerous IETF drafts, e.g., [Jos25, CBG25, OGP+25, KRSW25, SFG25] which describe concrete KEM combiners.

Concurrent Work. Connolly et al. [CHH+25], in concurrent and independent work, generalize the X-WING combiner [BCD+24]. They give a detailed analysis for the generalized versions (QSF and QSI) both for standard IND-CCA security and binding security. The latter part overlaps with parts of our results with minimal difference in the bounds. In contrast to their work, we are focusing solely on binding security but consider a wider range of KEM combiners.

1.3 Organization

Section 2 provides the necessary background for this work. Our analysis of the generic combiner appears in Section 3, while the application of the results to specific KEM combiners is given in Section 4.

2 Preliminaries

Notation. We denote the empty string by ε. For a randomized algorithm A, we write $y \leftarrow\!\!\$\ \mathrm{A}(x)$ to denote that y is the output of A on input x. We write $y \leftarrow \mathrm{A}(x;r)$, to denote that y is the output of A on input x with randomness r—this means that A first generates some randomness r and then runs $\mathrm{A}(x;r)$. For a deterministic algorithm, we simply write $y \leftarrow \mathrm{A}(x)$. Throughout we leave the security parameter an implicit input for every algorithm. By *adversary*, we refer to a probabilistic algorithm running in polynomial-time (in the security parameter).

Key-Encapsulation Mechanisms. Below we provide the formal definition of key-encapsulation mechanisms.

Definition 1. *A key-encapsulation mechanism is a tuple of three efficient algorithms* $(\textsc{KeyGen}, \textsc{Encaps}, \textsc{Decaps})$*. The (probabilistic) key-generation algorithm outputs a key pair* (pk, sk)*. The (probabilistic) encapsulation algorithm, on input a public key* pk*, outputs a ciphertext* c *and a shared key* k*. The (deterministic) decapsulation algorithm, on input a secret key* sk[4] *and a ciphertext* c*, outputs a shared key* k*. Correctness requires that* $k = \overline{k}$ *for* $(c, k) \leftarrow\!\!\$ \textsc{Encaps}(pk)$ *and* $\overline{k} \leftarrow \textsc{Decaps}(sk, c)$ *with* $(pk, sk) \leftarrow\!\!\$ \textsc{KeyGen}()$*.*

Throughout this work, we will be concerned with two KEMs that are combined via some combiner. To differentiate the two, we will use subscripts 1 and 2. The subscripts are also used to differentiate the inputs/outputs of the different KEMs, e.g., $\textsc{Kem}_i$ denotes a particular KEM while c_i denotes a ciphertext of $\textsc{Kem}_i$. Note that we will consider both classical and post-quantum KEMs as instantiations for $\textsc{Kem}_1$ and $\textsc{Kem}_2$.

Binding Properties. In Fig. 1 we provide the generic binding security game from which the various concrete binding properties can be derived. Every notion is of the form X-Bind-P-Q. Here, $\mathrm{X} \in \{\textsc{Hon}, \textsc{Leak}, \textsc{Mal}\}$ describes the attack model which describes how the keys are generated. Simply speaking, Hon and Leak use honestly generated keys, the difference being that Hon provides only the public key to the adversary while Leak provides also the secret key. In case of $\mathrm{X} = \textsc{Mal}$, the adversary can generate the keys itself and is thus no longer limited to honestly generated ones. The components P and Q describe which values (P) "bind" which other values (Q). For instance, X-Bind-K-CT captures the idea that the shared key (K) binds the ciphertext (CT) which, roughly speaking, formalizes the idea that it is hard to find two different ciphertexts that decapsulate to the same shared key. Similarly, X-Bind-K,CT-PK formalizes that it is hard to find a ciphertext that decapsulates to the same shared key under different key pairs. We refer to [KSW25] for explicit descriptions of the concrete binding properties. Below we define security for the various binding properties.

Definition 2. *Let* Kem *be a KEM and* X-Bind-P-Q *the security game shown in Fig. 1, where* $\mathrm{X} \in \{\textsc{Hon}, \textsc{Leak}, \textsc{Mal}\}$*,* $\mathrm{P} \in \{\{\mathrm{K}\}, \{\mathrm{CT}\}, \{\mathrm{K}, \mathrm{CT}\}, \{\mathrm{K}, \mathrm{PK}\}\}$*, and* $\mathrm{Q} \in \{\{\mathrm{K}\}, \{\mathrm{PK}\}, \{\mathrm{CT}\}\}$*. We say that* Kem *achieves* X-Bind-P-Q *if for any adversary* $\mathcal{A}$*, its probability in winning* X-Bind-P-Q*, i.e., the advantage* $\mathbf{Adv}^{\text{X-Bind-P-Q}}_{\textsc{Kem}}(\mathcal{A})$*, is negligible.*

The following result from [CDM24] establishes a hierarchy between two binding notions that we will use later.

Lemma 3. *Let* Kem *be a KEM and* $\mathrm{X} \in \{\textsc{Leak}, \textsc{Hon}\}$*. Then for any adversary* $\mathcal{A}$*, there exists an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\text{X-Bind-CT-K}}_{\textsc{Kem}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{X-Bind-CT-PK}}_{\textsc{Kem}}(\mathcal{B})\,.$$

[4] Note that we assume that the public key can be derived from the corresponding secret key.

Game X-BIND-P-Q

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
if PK $\in$ P :
  $(\overline{pk}, \overline{sk}) \leftarrow (pk, sk)$
if PK $\notin$ P $\cup$ Q :
  $b \in \{0,1\} \leftarrow \mathcal{A}()$
  if $b = 0$: $(\overline{pk}, \overline{sk}) \leftarrow (pk, sk)$
if X = HON :
  $(c, \overline{c}) \leftarrow \mathcal{A}^{\mathsf{Dec},\overline{\mathsf{Dec}}}(pk, \overline{pk})$
if X = LEAK :
  $(c, \overline{c}) \leftarrow \mathcal{A}(pk, sk, \overline{pk}, \overline{sk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
$v_{\mathrm{p}} \leftarrow (\forall \mathrm{p} \in \mathrm{P} : x_{\mathrm{p}} = \overline{x}_{\mathrm{p}})$
$v_{\mathrm{q}} \leftarrow (\exists \mathrm{q} \in \mathrm{Q} : x_{\mathrm{q}} \neq \overline{x}_{\mathrm{q}})$
return $v_{\mathrm{p}} \wedge v_{\mathrm{q}}$

Game MAL-BIND-P-Q

$g \leftarrow \mathcal{A}()$
if $g = 1$: // Scenario (I): DECAPS-DECAPS
  $(sk, \overline{sk}, c, \overline{c}) \leftarrow \mathcal{A}()$
  $k \leftarrow \textsc{Decaps}(sk, c)$
  $\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $g = 2$: // Scenario (II): ENCAPS-DECAPS
  $(pk, \overline{sk}, r, \overline{c}) \leftarrow \mathcal{A}()$
  $(k, c) \leftarrow \textsc{Encaps}(pk; r)$
  $\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $g \notin \{1, 2\}$: // Scenario (III): ENCAPS-ENCAPS
  $(pk, \overline{pk}, r, \overline{r}) \leftarrow \mathcal{A}()$
  $(k, c) \leftarrow \textsc{Encaps}(pk; r)$
  $(\overline{k}, \overline{c}) \leftarrow \textsc{Encaps}(\overline{pk}; \overline{r})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
$v_{\mathrm{p}} \leftarrow (\forall \mathrm{p} \in \mathrm{P} : x_{\mathrm{p}} = \overline{x}_{\mathrm{p}})$
$v_{\mathrm{q}} \leftarrow (\exists \mathrm{q} \in \mathrm{Q} : x_{\mathrm{q}} \neq \overline{x}_{\mathrm{q}})$
return $v_{\mathrm{p}} \wedge v_{\mathrm{q}}$

Fig. 1. Generic security games for the notions X-BIND-P-Q and MAL-BIND-P-Q for P $\in \{\{\mathrm{K}\}, \{\mathrm{CT}\}, \{\mathrm{K}, \mathrm{CT}\}, \{\mathrm{K}, \mathrm{PK}\}\}$, Q $\in \{\{\mathrm{K}\}, \{\mathrm{PK}\}, \{\mathrm{CT}\}\}$, and X $\in \{\textsc{Hon}, \textsc{Leak}\}$. Note that for p $\in$ P (and q $\in$ Q), we denote the corresponding instances by x_{p}, $\overline{x}_{\mathrm{p}}$ and x_{q}, $\overline{x}_{\mathrm{q}}$, respectively. For example, if p = CT, we have $x_{\mathrm{p}} = c$ and $\overline{x}_{\mathrm{p}} = \overline{c}$. For variants (I) and (II) of the MAL notions, it is understood that any check of the form $pk \bowtie \overline{pk}$, for $\bowtie \in \{=, \neq\}$ is performed on the public keys contained in the respective secret keys sk and $\overline{sk}$. The decapsulation oracles Dec and $\overline{\mathsf{Dec}}$ in case of X = HON are omitted and work in the obvious way (without any restrictions) using sk and $\overline{sk}$, respectively.

3 Binding Analysis of Combined KEMs

In this section, we present our results regarding the binding notions of combined KEMs. For this, we introduce a generic combiner in Section 3.1, which covers all real-world KEM combiners we are aware of. In Section 3.2, we analyze the generic combiner with respect to binding notions X-BIND-K-CT, X-BIND-K-PK, X-BIND-K,PK-CT, and X-BIND-K,CT-PK, i.e., the cases in which the shared key is binding. The remaining notions X-BIND-CT-K and X-BIND-CT-PK, i.e., the ones where the ciphertext by itself is binding, are treated in Appendix A. Note that this covers all notions presented in [CDM24].

3.1 A Generic Combiner

We define a generic combiner GC, which combines two KEMs (KEM_1 and KEM_2) into a combined KEM cKEM as displayed in Fig. 2. For key generation, the combiner independently calls the key generation algorithms of the underlying KEMs KEM_1 and KEM_2, and then assembles the public and secret key of the combined scheme as $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$. During encapsulation, the ENCAPS algorithms of KEM_1 and KEM_2 are used to generate the ciphertexts c_1, c_2 and shared keys k_1, k_2 of the underlying KEMs. The ciphertext of the combined scheme is obtained simply as $c = (c_1, c_2)$—in contrast, the computation of the shared key differs depending on the concrete combiner. While the shared keys are always used in the key derivation, this is not necessarily the case for the public keys and ciphertexts of the underlying schemes. To model this, we introduce an abstract function COMBINE that is used to generate the shared key and a filtering function F that describes which of the components (pk_1, pk_2, c_1, c_2) are given as input to COMBINE. Thus, a key encapsulation mechanism cKEM built via the generic combiner GC, computes the shared secret as

$$\text{COMBINE}(k_1, k_2, \text{F}(pk_1, pk_2, c_1, c_2), \Xi)$$

where Ξ describes an abstract label[5] and F filters out the components that are not needed for the combining operation of cKEM. More precisely, F takes as input the tuple (pk_1, pk_2, c_1, c_2) and outputs a sub-tuple. As an example, for the combiner underlying X-WING, we have COMBINE = SHA3-256 and $\text{F}(pk_1, pk_2, c_1, c_2) = (pk_1, c_1)$. In case decapsulation of an underlying KEM fails and returns $\perp$, we assume that COMBINE will simply return $\perp$ as well, which implies that the combined KEM is explicitly-rejecting if at least one of the underlying KEMs is. Table 1 provides an overview of several combiners which can be derived as specific instantiations of our generic combiner.

For ease of exposition, we also consider a second filtering function to track the inputs that are included in the key derivation. For cKEM, the function

$$f\colon \{\text{PK}, \text{CT}\} \times \{1, 2\} \to \{0, 1\}$$

outputs 1 if and only if the component corresponding to the input is part of key derivation. For instance, referring to the X-WING example above, where $\text{F}(pk_1, pk_2, c_1, c_2) = (pk_1, c_1)$, we have $f(\text{PK}, 1) = f(\text{CT}, 1) = 1$ and $f(\text{PK}, 2) = f(\text{CT}, 2) = 0$.

A key-encapsulation mechanism cKEM constructed from the generic combiner GC with filtering function f is denoted by $\text{cKEM}[f]$—for sake of brevity, we drop the two underlying KEMs KEM_1 and KEM_2 from the notation.

Generic Reductions. For the results in the following, we rely on some generic reductions, reducing a certain binding property of a combined KEM to a binding property of one of the underlying KEMs. We identify these reductions by $\mathcal{H}_i$, $\mathcal{L}_i$,

[5] Note that for our analysis Ξ is not relevant.

KeyGen()	Encaps($pk = (pk_1, pk_2)$)
$(pk_1, sk_1) \leftarrow\!\!\$\ \textsc{KeyGen}_1()$	$(c_1, k_1) \leftarrow\!\!\$\ \textsc{Encaps}_1(pk_1)$
$(pk_2, sk_2) \leftarrow\!\!\$\ \textsc{KeyGen}_2()$	$(c_2, k_2) \leftarrow\!\!\$\ \textsc{Encaps}_2(pk_2)$
$pk \leftarrow (pk_1, pk_2)$	$c \leftarrow (c_1, c_2)$
$sk \leftarrow (sk_1, sk_2)$	$k \leftarrow \textsc{Combine}(k_1, k_2, \mathrm{F}(pk_1, pk_2, c_1, c_2), \Xi)$
return (pk, sk)	**return** (c, k)

Decaps($sk = (sk_1, sk_2), c = (c_1, c_2)$)
$k_1 \leftarrow \textsc{Decaps}_1(sk_1, c_1)$
$k_2 \leftarrow \textsc{Decaps}_2(sk_2, c_2)$
$k \leftarrow \textsc{Combine}(k_1, k_2, \mathrm{F}(pk_1, pk_2, c_1, c_2), \Xi)$
return k

Fig. 2. Generic combiner GC which combines two KEMs $\textsc{Kem}_1$ and $\textsc{Kem}_2$ into a KEM $\textsc{cKem}[f]$, where f describes which inputs are filtered out by F. We assume that Combine outputs $\perp$ whenever any of the input is $\perp$.

and $\mathcal{M}_i$—the letter indicating for which attack model (Hon, Leak, Mal) the reduction is. These reductions work in a straightforward way. For instance, $\mathcal{L}_1$ is a reduction playing against some binding notion (in the Leak setting) of $\textsc{Kem}_1$. In case of Leak-Bind-K-CT, it receives two key pairs $(pk_1, sk_1), (\overline{pk}_1, \overline{sk}_1)$ and has to output two ciphertexts $c_1 \neq \overline{c}_1$. This reduction runs an adversary $\mathcal{A}$ playing against Leak-Bind-K-CT security of the *combined* KEM cKem, by providing its own key pairs alongside two key pairs of $\textsc{Kem}_2$ which $\mathcal{L}_1$ locally generates. When $\mathcal{A}$ outputs two distinct ciphertexts $c = (c_1, c_2)$ and $\overline{c} = (\overline{c}_1, \overline{c}_2)$, $\mathcal{L}_1$ outputs the two ciphertexts c_1 and $\overline{c}_1$. For notions that let the adversary decide if it wants to play against two distinct key pairs or a single one (e.g., Leak-Bind-CT-K), the reduction will generate one or two key pairs locally depending on the choice of $\mathcal{A}$ and also ask for the same number of key pairs in its own security game. Simulation of the decapsulation oracles in the Hon setting is straightforward as $\mathcal{H}_i$ can use its own decapsulation oracle for everything related to $\textsc{Kem}_i$ and can locally simulate decapsulation for the other KEM as it has access to the secret key.

3.2 Results for Shared Key Binding

In this section, we present our result regarding the X-Bind-K-Q security of the generic combiner. Simply put, a combined KEM achieves X-Bind-K-Q, if Combine is collision-resistant (we denote by CR the collision-resistance game), and either Combine includes the Q-component of the underlying KEM $\textsc{Kem}_i$ or $\textsc{Kem}_i$ is itself X-Bind-K-Q secure.

Theorem 4. *Let* $\textsc{Kem}_1$ *and* $\textsc{Kem}_2$ *be two KEMs. Let further* cKem *denote the KEM resulting from applying* GC *to* $\textsc{Kem}_1$ *and* $\textsc{Kem}_2$ *(with combiner function*

Combine *and filtering function* f*). Furthermore, let* $\mathrm{X} \in \{\mathrm{Hon}, \mathrm{Mal}, \mathrm{Leak}\}$, $\mathrm{P} \in \{\{\mathrm{K}\}, \{\mathrm{K}, \mathrm{CT}\}, \{\mathrm{K}, \mathrm{PK}\}\}$*, and* $\mathrm{Q} \in \{\{\mathrm{CT}\}, \{\mathrm{PK}\}\}$*, such that* $\mathrm{P} \cap \mathrm{Q} = \emptyset$*. Then for any adversary* $\mathcal{A}$ *there exist adversaries* $\mathcal{B}$*,* $\mathcal{C}_1$*, and* $\mathcal{C}_2$*, such that*

$$\mathbf{Adv}^{\text{X-Bind-P-Q}}_{\text{cKem}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{CR}}_{\text{Combine}}(\mathcal{B}) + \sum_{\substack{i \in \{1,2\} \mid \\ f(\mathrm{Q},i)=0}} \mathbf{Adv}^{\text{X-Bind-P-Q}}_{\text{Kem}_i}(\mathcal{C}_i)\,.$$

Proof. Let $\mathcal{A}$ be an adversary that breaks X-Bind-P-Q security of cKem$[f]$. Since $\mathrm{K} \in \mathrm{P}$, the outputs of the adversary $\mathcal{A}$ against X-Bind-P-Q security of cKem$[f]$ yield colliding keys, i.e.,

$$\begin{aligned} k &= \textsc{Combine}(k_1, k_2, \mathrm{F}(pk_1, pk_2, c_1, c_2), \Xi) \\ &= \textsc{Combine}(\overline{k}_1, \overline{k}_2, \mathrm{F}(\overline{pk}_1, \overline{pk}_2, \overline{c}_1, \overline{c}_2), \Xi) = \overline{k}\,. \end{aligned}$$

Further, depending on the value of $\mathrm{Q} \in \{\{\mathrm{CT}\}, \{\mathrm{PK}\}\}$, either $(c_1, c_2) \neq (\overline{c}_1, \overline{c}_2)$ or $(pk_1, pk_2) \neq (\overline{pk}_1, \overline{pk}_2)$ must hold. For sake of simplicity, we focus on the case that $\mathrm{Q} = \{\mathrm{CT}\}$ and note that the argumentation for $\mathrm{Q} = \{\mathrm{PK}\}$ works analogously.

Thus, we know that $(c_1, c_2) \neq (\overline{c}_1, \overline{c}_2)$. For $i \in \{1, 2\}$, we distinguish the cases shown in Fig. 3. Here, CC denotes a collision of Combine—note that if collision-resistance (CR) is given, the probability for CC is negligible—and the event $\mathsf{E}[\textsc{Kem}_i]$ describes that X-Bind-P-CT security of $\textsc{Kem}_i$ has been broken.

Since we know that $k = \overline{k}$, a collision of Combine appears if any of the inputs differ—taking into account which inputs are filtered out by f. In the remaining four cases in Fig. 3, the events $\mathsf{E}[\textsc{Kem}_1]$ and $\mathsf{E}[\textsc{Kem}_2]$ appear. Note that in the case that $\mathrm{P} = \{\mathrm{K}, \mathrm{PK}\}$, the fact that $\mathcal{A}$ breaks the notion X-Bind-P-Q for cKem$[f]$, implies that the public keys of the underlying KEMs also agree, as $pk = (pk_1, pk_2)$. In total, taking into account all cases from the above tree yields the claimed bound as follows.

Firstly, mirroring the tree, we divide the advantage based on whether a collision of Combine appears or not. In the following let Win denote the event that $\mathcal{A}$ wins the game X-Bind-P-CT:

$$\mathbf{Adv}^{\text{X-Bind-P-CT}}_{\text{cKem}[f]}(\mathcal{A}) = \Pr[\mathsf{Win}] = \Pr[\mathsf{Win} \wedge \mathsf{CC}] + \Pr[\mathsf{Win} \wedge \neg\mathsf{CC}]$$

Next, we further split up the second summand, based on the values of $f(\mathrm{CT}, 1)$ and $f(\mathrm{CT}, 2)$—corresponding to the second branching point in the tree:

$$\begin{aligned} \Pr[\mathsf{Win} \wedge \neg\mathsf{CC}] = &\Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT}, 1) = 1 \wedge f(\mathrm{CT}, 2) = 1] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT}, 1) = 1 \wedge f(\mathrm{CT}, 2) = 0] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT}, 1) = 0 \wedge f(\mathrm{CT}, 2) = 1] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT}, 1) = 0 \wedge f(\mathrm{CT}, 2) = 0] \end{aligned}$$

Note that the fact that there is no collision implies that the inputs to Combine must be equal for the two tuples outputted by the adversary. Thus the first summand is zero, as in this case $(c_1, c_2) = (\overline{c}_1, \overline{c}_2)$ must hold, which contradicts

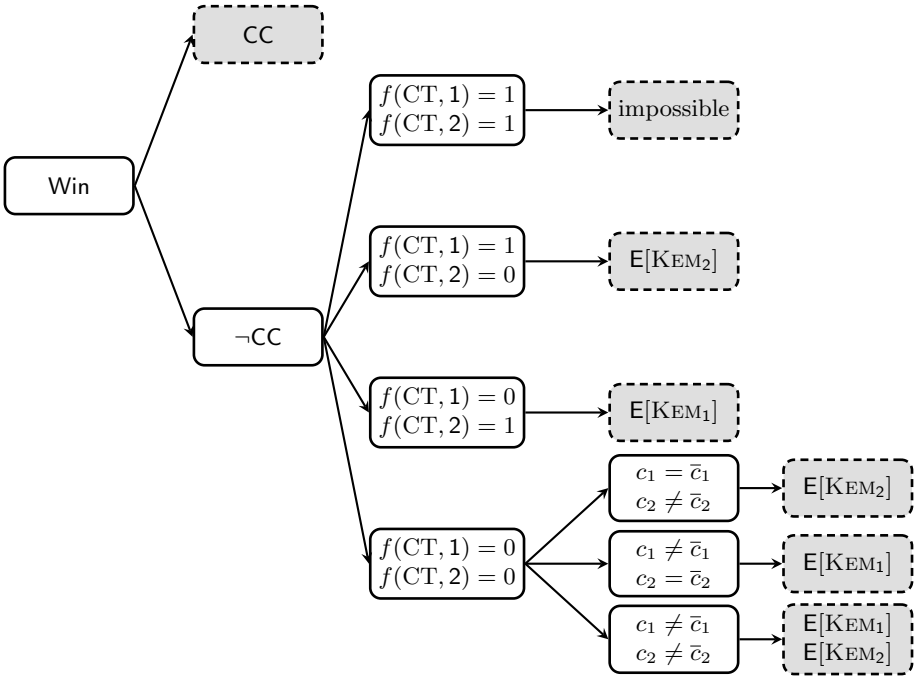


Fig. 3. Case distinction for the proof of Theorem 4. The gray boxes show which of the events (collision of COMBINE (CC) or X-BIND-P-CT security of KEM_i being broken ($\mathsf{E}[\mathrm{KEM}_i]$)) occurs in each case or mark impossible cases.

the fact that $\mathcal{A}$ wins the game X-BIND-P-CT. The summand with $f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0$ is further divided according to the third branching point in the above tree. In this case, neither of the ciphertexts is included in COMBINE, hence the only restriction is $(c_1, c_2) \neq (\overline{c}_1, \overline{c}_2)$, which implies that at least one of the ciphertext components has to differ:

$$
\begin{aligned}
&\Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0] \\
=\,&\Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 = \overline{c}_1 \wedge c_2 \neq \overline{c}_2] \\
+\,&\Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \overline{c}_1 \wedge c_2 = \overline{c}_2] \\
+\,&\Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \overline{c}_1 \wedge c_2 \neq \overline{c}_2]
\end{aligned}
$$

Combining the above yields

$$\begin{aligned}\mathbf{Adv}^{\text{X-Bind-P-CT}}_{\text{cKem}[f]}(\mathcal{A}) &= \Pr[\mathsf{Win} \wedge \mathsf{CC}] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 1 \wedge f(\mathrm{CT},2) = 0] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 1] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 = \bar{c}_1 \wedge c_2 \neq \bar{c}_2] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \bar{c}_1 \wedge c_2 = \bar{c}_2] \\ &+ \Pr[\mathsf{Win} \wedge \neg\mathsf{CC} \wedge f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \bar{c}_1 \wedge c_2 \neq \bar{c}_2]\,.\end{aligned}$$

Note that for any given combiner, the values of $f(\mathrm{CT},2)$ and $f(\mathrm{CT},1)$ are predetermined, i.e., $\Pr[f(\mathrm{CT},1) = i \wedge f(\mathrm{CT},2) = j] \in \{0,1\}$ for $i,j \in \{0,1\}$. Using this and the formula for conditional probability, yields for $\mathcal{C}_i \in \{\mathcal{H}_i, \mathcal{L}_i, \mathcal{M}_i\}$ the final bound

$$\begin{aligned}&\mathbf{Adv}^{\text{X-Bind-P-CT}}_{\text{cKem}[f]}(\mathcal{A}) \leq \Pr[\mathsf{CC}] \\ &\quad + \Pr[\mathsf{E}[\text{Kem}_2]] \cdot \Pr[f(\mathrm{CT},1) = 1 \wedge f(\mathrm{CT},2) = 0] \\ &\quad + \Pr[\mathsf{E}[\text{Kem}_1]] \cdot \Pr[f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 1] \\ &\quad + \Pr[\mathsf{E}[\text{Kem}_2]] \cdot \Pr[f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 = \bar{c}_1 \wedge c_2 \neq \bar{c}_2] \\ &\quad + \Pr[\mathsf{E}[\text{Kem}_1]] \cdot \Pr[f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \bar{c}_1 \wedge c_2 = \bar{c}_2] \\ &\quad + (\Pr[\mathsf{E}[\text{Kem}_1]] + \Pr[\mathsf{E}[\text{Kem}_2]]) \cdot \Pr[f(\mathrm{CT},1) = 0 \wedge f(\mathrm{CT},2) = 0 \wedge c_1 \neq \bar{c}_1 \wedge c_2 \neq \bar{c}_2] \\ &\leq \mathbf{Adv}^{\text{CR}}_{\text{Combine}}(\mathcal{B}) + \sum_{\substack{i \in \{1,2\} \mid \\ f(\mathrm{CT},i)=0}} \mathbf{Adv}^{\text{X-Bind-P-CT}}_{\text{Kem}_i}(\mathcal{C}_i)\,.\end{aligned}$$

As this holds analogously for $\mathrm{Q} = \{\mathrm{PK}\}$, we obtain the theorem statement

$$\mathbf{Adv}^{\text{X-Bind-P-Q}}_{\text{cKem}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{CR}}_{\text{Combine}}(\mathcal{B}) + \sum_{\substack{i \in \{1,2\} \mid \\ f(\mathrm{Q},i)=0}} \mathbf{Adv}^{\text{X-Bind-P-Q}}_{\text{Kem}_i}(\mathcal{C}_i)\,.$$

□

To illustrate the above theorem, we give an exemplary application for the notion Hon-Bind-K-CT and the OpenPGP-Comb combiner [KRSW25], for which Combine = SHA3-256 and $\mathrm{F}(pk_1, pk_2, c_1, c_2) = (pk_1, c_1)$ holds. This yields the bound

$$\mathbf{Adv}^{\text{Hon-Bind-K-CT}}_{\text{cKem}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{CR}}_{\text{SHA3-256}}(\mathcal{B}) + \mathbf{Adv}^{\text{Hon-Bind-K-CT}}_{\text{Kem}_2}(\mathcal{C})\,,$$

which implies that the KEM cKem obtained from the OpenPGP-Comb combiner is Hon-Bind-K-CT secure if Kem_2 is, and collision-resistance of SHA3-256 is given.

4 Real-World KEM-Combiners

In this section, we apply our results to real-world combiners. We describe the combiners under consideration in Section 4.1. In Section 4.2, we instantiate the

Table 1. Overview of combiners regarding their optional inputs to COMBINE—the shared keys k_1 and k_2 are always inputs. Note further that subscripts 1 and 2 refer to the classical and post-quantum KEM in the specifications, respectively.

		Optional inputs to COMBINE			
Combiner	f	pk_1	c_1	pk_2	c_2
CHEMPAT [Jos25]	$\left[\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right]$	✓	✓	✓	✓
GHP [CBG25]	$\left[\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right]$	✓	✓	✓	✓
QSF [BCD+24]	$\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]$	✓	✓	✗	✗
LAMPS-COMB [OGP+25]	$\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]$	✓	✓	✗	✗
OPENPGP-COMB [KRSW25]	$\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]$	✓	✓	✗	✗
TLS-COMB [SFG25]	$\left[\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right]$	✗	✓	✗	✓
QSI [CHH+25]	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	✗	✗	✗	✗

combiners and give concrete results regarding the binding security of the resulting KEMs. In Section 4.3, we check what combiner properties are required when combining concrete KEMs.

4.1 Description of the KEM-Combiners

We consider a number of KEM combiners, which are currently described as IETF drafts. This includes the combiners GHP [CBG25], CHEMPAT [Jos25], the combiner developed by LAMPS [OGP+25] and the ones used in the protocols OpenPGP [KRSW25] and TLS [SFG25]. We denote the latter three by LAMPS-COMB, OPENPGP-COMB, and TLS-COMB, respectively. Note that we consider the combiners instantiated with various KEMs, while some are originally only defined for a certain KEMs according to their specification, e.g., GHP [CBG25] uses ML-KEM for KEM_2.

X-WING is another scheme, for which there is an IETF draft [CSW25]. While X-WING comes instantiated with ML-KEM and DH-KEM (using the curve X25519), there is also a more generic version called QSF [BCD+24], which only fixes the elliptic curve component.[6] Moreover, recently, the generic combiner QSI [CHH+25] was introduced, which works for any two KEMs—though additional requirements for the KEMs have to be met. An overview of the combiners under consideration is given in Table 1.

4.2 Binding Security of Real-World KEM-Combiners

In this section, we analyze combiner instantiations with respect to their binding security using our results from Section 3. While in the past, mostly classical and post-quantum schemes were combined, we also consider instantiations using two

[6] Note that we consistently use KEM_1 as the classical KEM and KEM_2 as the post-quantum KEM. In [BCD+24] the order of the KEMs is the other way around.

post-quantum schemes. With their ongoing development and standardization, there is now a number of post-quantum schemes suited for combination: there are schemes that enjoy more trust due to being older (Classic-McEliece) or (soon to be) standardized (Ml-Kem and Hqc). We consider the post-quantum schemes Ml-Kem, FrodoKem, Classic-McEliece, Bike, Hqc, and Hqc*—a modified variant of Hqc proposed in [KSW25] that achieves more binding properties.[7] This selection contains the schemes chosen by NIST for standardization, i.e., Ml-Kem and Hqc, as well as the finalists in NISTs standardization process, i.e., Classic-McEliece and Bike.[8] Regarding classical schemes, we include only DH-Kem as it is, to our knowledge, the only classical scheme for which the binding properties have been analyzed [CDM24]. More generally, the KEMs we consider are the ones for which complete binding analyses exist.

Note that all schemes under consideration use implicit rejection, which implies that the notions X-Bind-CT-PK and X-Bind-CT-K can never be achieved. Moreover, we focus on achieving the Hon and Leak variants of the notions as they seem the most relevant from a practical perspective—both known attacks exploit the absence of certain Leak notions [CDM24,BJKS24]. This leaves us with the notions X-Bind-K-CT and X-Bind-K-PK for $\mathrm{X} \in \{\textsc{Hon}, \textsc{Leak}\}$ as these, by the hierarchy, imply the remaining ones (X-Bind-K,PK-CT and X-Bind-K,CT-PK).

These notions are covered by Theorem 4, which can be applied to obtain the results for concrete combiner instantiations. The theorem states that a combined KEM is X-Bind-P-Q secure, if the combiner function Combine is collision-resistant and each underlying KEM for which the Q-component is not included in Combine (meaning the corresponding entry for f is 0) is required to fulfill the property X-Bind-P-Q itself.

Of the considered KEM-combiners, most instantiate Combine using standard hash functions. More precisely, [Jos25, OGP+25, KRSW25] all mention SHA as an instantiation. In contrast, TLS-Comb relies on a dual-PRF [SFG25, BBF+19]. First of all, the dual-PRF property does not imply collision resistance. As an example, consider the case where a dual-PRF F is modified to a function F^* s.t. $\mathrm{F}^*(k_1, k_2)$ outputs a fixed value whenever $k_1 = k_2$, and otherwise behaves like F; clearly, F^* is not collision-resistant but still works like a dual-PRF as an adversary needs to query the (unknown) key to trigger the special case.

Regarding the binding properties of the KEMs under consideration, we rely on the results from [GMP22,CDM24,KSW25], which cover the KEMs Ml-Kem, FrodoKem, Classic-McEliece, Bike, Hqc, Hqc*, and DH-Kem. We provide an overview of the results relevant for our analysis in Table 2.

Then, Theorem 4 and the existing binding analysis for the selected KEMs enable an easy analysis for concrete combiners. For example, in the case of the TLS combiner, neither of the public keys of the underlying KEMs is in-

[7] Note that the Hqc team adopted the Hqc* variant in the newest version of their specification (from August 22, 2025).

[8] Note that SIKE was another finalist, but it was found to be insecure during the standardization process.

cluded in Combine, i.e., the combiner only achieves Leak-Bind-K-PK if both KEMs do. In particular, this means that the TLS combiner applied to DH-Kem and Ml-Kem would yield an Leak-Bind-K-PK secure scheme (as both KEMs achieve Leak-Bind-K-PK security), while this would not be the case for DH-Kem and Hqc (as Hqc does not fulfill Leak-Bind-K-PK security). Below we describe our results for the combiners under consideration, which are shown in Table 1. We organize combiners with identical or similar outcomes into four groups.

Chempat *and* GHP. Both combiners include shared keys, public keys, and ciphertexts of both underlying KEMs in Combine. Hence, Theorem 4 implies that only collision resistance of Combine is required to achieve Leak-Bind-K-CT and Leak-Bind-K-PK security for the combined KEM. In particular, binding security of Chempat and GHP is guaranteed for *any* two KEMs.

Qsf, Lamps-Comb, *and* OpenPGP-Comb. These three combiners share a common structure, where for the first KEM *all* inputs are fed into Combine, while for the second one only the shared key is used. All three combiners require ciphertext second-preimage resistance (C2PRI) of the second KEM in order to achieve even IND-CCA security. Note that [BCD+24] shows that X-Wing requires ciphertext second-preimage resistance (C2PRI) of the second KEM in order to achieve even IND-CCA security. The C2PRI notion is very similar to Leak-Bind-K,PK-CT with the only difference that the adversary has to find a pre-image instead of a collision—in particular, Leak-Bind-K,PK-CT implies C2PRI. All KEMs except Hqc fulfill Leak-Bind-K-CT and therefore also C2PRI as Leak-Bind-K-CT implies Leak-Bind-K,PK-CT.[9] Therefore, we cannot use Hqc as an instantiation for the second KEM Kem_2. Regarding binding security, Theorem 4 yields that the second KEM has to achieve X-Bind-K-CT and X-Bind-K-PK security in order for the combined KEM to fulfill these notions. Taking this into account, Table 3 shows which KEM combinations are possible when using Qsf, OpenPGP-Comb, and Lamps-Comb.

TLS-Comb. The draft [SFG25] describes a combiner that computes the shared key as $k = k_1 \parallel k_2$. In the subsequent TLS-Comb key schedule, further operations involving the ciphertexts are executed. Therefore, the TLS-Comb combiner can be seen to follow the dual-PRF approach from [BBF+19]—this is also explicitly stated in [SFG25]. Due to this, we consider the dual-PRF combiner, which uses the shared keys and ciphertexts of both KEMs as input to Combine. By Theorem 4, TLS-Comb requires X-Bind-K-PK security for both KEMs. We display the results in Table 4.

Qsi. The Qsi combiner includes only the shared keys of both underlying KEMs in Combine. Note that IND-CCA security of the combined KEM is only given

[9] While Hqc is shown to be insecure w.r.t. Leak-Bind-K,PK-CT, it could still fulfill the weaker notion C2PRI. However one easily checks that the Leak-Bind-K,PK-CT attack from [KSW25] also applies against the notion C2PRI (as the shared key and one of the ciphertexts in the attack are honestly generated).

Table 2. Summary of the binding analysis of KEMs given in [GMP22,CDM24,KSW25]. The entries show if a binding notion holds in the Leak and the Hon setting (✓), only in the Hon setting (✓̸), or in neither setting (✗).

KEM	X-Bind-K-CT	X-Bind-K-PK
Ml-Kem	✓	✓
Hqc	✗	✓̸
Hqc*	✓	✓̸
DH-Kem	✓	✓
FrodoKem	✓	✓
Bike	✓	✓̸
McEliece	✓	✗

if the underlying KEMs are IND-CCA secure and fulfill C2PRI—thus, Hqc cannot be used with Qsi. Regarding binding security, Theorem 4 implies that both KEMs have to fulfill X-Bind-K-CT and X-Bind-K-PK. Hence, fewer combinations are possible for the QSI combiner—the details can be found in Table 5.

4.3 How to Combine Specific KEMs

In the previous section, we considered various combiners instantiated with the KEMs for which binding properties have been analyzed. The tables (cf. Table 3, Table 5, and Table 4) show, for each combiner, which combinations of KEMs result in a combined KEM achieving all binding notions.[10] Here, we take a different perspective and consider which components of the two KEMs really need to be inputs to Combine. It turns out, that none of the considered combiners is optimal in the sense that it only feeds the minimum of components to Combine. Table 2 shows that all KEMs, except Hqc, satisfy X-Bind-K-CT, which makes hashing the ciphertext unnecessary—at least from the perspective of achieving binding properties. On the other hand, several KEMs fall short regarding X-Bind-K-PK: Hqc, Hqc*, and Bike achieve it only for X = Hon, while McEliece does not achieve it at all. This means that for several combinations, feeding the public key to Combine is more important than the ciphertext. However, none of the combiners provide solely the public key(s) to Combine. The OpenPGP-Comb combiner gives public key and ciphertext of one underlying KEM to Combine, while TLS-Comb gives the ciphertexts of both underlying KEMs. The results are summarized in Table 6 which shows, for each combination, what f needs to look like to achieve all binding properties.

4.4 Conclusion

Our results from Section 4.2 allow us to draw conclusions about the compatibility of the schemes under consideration when used in different combiners.

[10] Recall that we focus on security up to the Leak setting and, since all KEMs are implicitly-rejecting, only on X-Bind-K-PK and X-Bind-K-CT.

Table 3. The binding results for KEMs obtained using one of the combiners Qsf [BCD+24], OpenPGP-Comb [KRSW25], or Lamps-Comb [OGP+25], where $F(pk_1, pk_2, c_1, c_2) = (pk_1, c_1)$. We consider instantiations using any combination of Ml-Kem, Hqc, Hqc*, DH-Kem, FrodoKem, Bike, and Classic-McEliece (shortened to McEliece in the table)—note that we exclude combinations of a scheme with itself and any combination where $\text{Kem}_2 = \text{Hqc}$, as it fails to achieve the required C2PRI property (■). The entries show if security (w.r.t. both X-Bind-K-CT and X-Bind-K-PK) holds in the Leak and the Hon setting (✓), only in the Hon setting (✓), or in neither setting (✗).

Kem_1 \ Kem_2	Ml-Kem	Hqc[a]	Hqc*	DH-Kem	FrodoKem	Bike	McEliece
Ml-Kem	■	■	✓	✓	✓	✓	✗
Hqc	✓	■	■	✓	✓	✓	✗
Hqc*	✓	■	■	✓	✓	✓	✗
DH-Kem	✓	■	✓	■	✓	✓	✗
FrodoKem	✓	■	✓	✓	■	✓	✗
Bike	✓	■	✓	✓	✓	■	✗
McEliece	✓	■	✓	✓	✓	✓	■

More precisely, we compare the resulting combined KEMs regarding the level of binding security they achieve with respect to the notions X-Bind-K-CT and X-Bind-K-PK.

- Firstly, for Qsf, OpenPGP-Comb, and Lamps-Comb, we observe that using Ml-Kem, DH-Kem, or FrodoKem as Kem_2 yields a combined scheme that achieves both notions in the Leak setting. In use-cases where Hon security suffices, also Bike and Hqc* are suited as Kem_2. In contrast, Hqc and Classic-McEliece are not fit for use as Kem_2; the former as it does not satisfy the necessary assumption C2PRI, and the latter due to the fact X-Bind-K-PK security is not even given in the Hon setting.
- For TLS-Comb, combinations of the schemes Ml-Kem, FrodoKem, and DH-Kem with each other, yield a combined KEM that achieves the notions under consideration in the Leak setting. Further, if Hon security suffices, Bike, Hqc, and Hqc* can additionally be included. Combinations with Classic-McEliece do not fulfill all binding notions due to Classic-McEliece being insecure w.r.t. Hon-Bind-K-PK.
- The results for Qsi agree with the ones for TLS-Comb with the only difference being that Hqc cannot be used with this combiner.

In summary, Ml-Kem, FrodoKem, and DH-Kem are best suited for use-cases in which the Leak attack model is relevant, while also Bike and Hqc* can be used when security in the Hon setting is sufficient. Note that the slightly modified version Hqc* of Hqc performs better than the original scheme. This

Table 4. The binding results for KEMs obtained using the TLS-Comb combiner [SFG25], where $\mathrm{F}(pk_1, pk_2, c_1, c_2) = (c_1, c_2)$. We consider instantiations using any combination of ML-KEM, HQC, HQC*, DH-KEM, FrodoKEM, BIKE, and Classic-McEliece (shortened to McEliece in the table)—note that we exclude combinations of a scheme with itself (■). The entries show if security (w.r.t. both X-Bind-K-CT and X-Bind-K-PK) holds in the Leak and the Hon setting (✓), only in the Hon setting (✓), or in neither setting (✗).

KEM1 \ KEM2	ML-KEM	HQC	HQC*	DH-KEM	FrodoKEM	BIKE	McEliece
ML-KEM	■	✓	✓	✓	✓	✓	✗
HQC	✓	■	■	✓	✓	✓	✗
HQC*	✓	■	■	✓	✓	✓	✗
DH-KEM	✓	✓	✓	■	✓	✓	✗
FrodoKEM	✓	✓	✓	✓	■	✓	✗
BIKE	✓	✓	✓	✓	✓	■	✗
McEliece	✗	✗	✗	✗	✗	✗	■

modification [KSW25] has been adopted by the HQC team in the newest version of the specification, who also recommended this version for the final NIST standard.

Lastly, in Section 4.3, we observed that the real-world combiners are not optimal with respect to the components included in Combine—when focusing on binding security. Taking the results from Table 6, we can describe an optimal combiner for the schemes under consideration, excluding HQC. It turned out that none of the KEMs require the ciphertext to be input to Combine. For any combination of ML-KEM, DH-KEM, and FrodoKEM the public keys are also not required as inputs to Combine, making $\left[\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ (what QSI does) the optimal combiner. In contrast, combinations involving HQC*, BIKE, and Classic-McEliece require the public key of these KEMs to be an input to Combine.[11] This means the optimal combiner for combinations involving those schemes would be $\left[\begin{smallmatrix} x & x \\ 0 & 0 \end{smallmatrix}\right]$, where the entry x corresponding to either of these KEMs must be set to 1.

[11] Note that HQC* and BIKE still achieve security in the Hon setting even if the public key is not an input.

Table 5. The binding results for KEMs obtained using the combiner Qsi [BCD+24], where $\mathrm{F}(pk_1, pk_2, c_1, c_2) = \varepsilon$. We consider instantiations using any combination of Ml-Kem, Hqc, Hqc*, DH-Kem, FrodoKem, Bike, and Classic-McEliece (shortened to McEliece in the table)—note that we exclude combinations of a scheme with itself as well as any combination with Hqc as it fails to achieve the C2PRI requirement of the combiner (■). The entries show if security (w.r.t. both X-Bind-K-CT and X-Bind-K-PK) holds in the Leak and the Hon setting (✓), only in the Hon setting (✓), or in neither setting (✗).

Kem1 \ Kem2	Ml-Kem	Hqc	Hqc*	DH-Kem	FrodoKem	Bike	McEliece
Ml-Kem	■	■	✓	✓	✓	✓	✗
Hqc	■	■	■	■	■	■	■
Hqc*	✓	■	■	✓	✓	✓	✗
DH-Kem	✓	■	✓	■	✓	✓	✗
FrodoKem	✓	■	✓	✓	■	✓	✗
Bike	✓	■	✓	✓	✓	■	✗
McEliece	✗	■	✗	✗	✗	✗	■

Table 6. Overview of which components need to be given as input to Combine to achieve all binding properties for any combination of two KEMs. Entries are in matrix form $\left[\begin{smallmatrix} pk_1 & pk_2 \\ c_1 & c_2 \end{smallmatrix}\right]$. A 0 means the component is not required to be an input to Combine, while a 1 means that the component has to be given as input to Combine. A ★ indicates that the component needs to be input to Combine to achieve Leak security while otherwise Hon security is achieved.

Kem1 \ Kem2	Ml-Kem	Hqc	Hqc*	DH-Kem	FrodoKem	Bike	McEliece
Ml-Kem	■	$\left[\begin{smallmatrix}0&\star\\0&1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right]$
Hqc	$\left[\begin{smallmatrix}\star&0\\1&0\end{smallmatrix}\right]$	■	■	$\left[\begin{smallmatrix}\star&0\\1&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&0\\1&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&\star\\1&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&1\\1&0\end{smallmatrix}\right]$
Hqc*	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	■	■	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&1\\0&0\end{smallmatrix}\right]$
DH-Kem	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	■	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right]$
FrodoKem	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$	■	$\left[\begin{smallmatrix}0&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right]$
Bike	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&\star\\0&1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\star&0\\0&0\end{smallmatrix}\right]$	■	$\left[\begin{smallmatrix}\star&1\\0&0\end{smallmatrix}\right]$
McEliece	$\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1&\star\\0&1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1&\star\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1&\star\\0&0\end{smallmatrix}\right]$	■

References

ABK25. Gorjan Alagic, Fahran Bajaj, and Aybars Kocoglu. The best of both KEMs: Securely combining KEMs in post-quantum hybrid schemes. Cryptology ePrint Archive, Report 2025/1444, 2025. (Cited on page 3.)

ABN10. Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Berlin, Heidelberg, February 2010. (Cited on page 3.)

ADK+22. Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenneth G. Paterson, Eyal Ronen, and Eylon Yogev. Practical (post-quantum) key combiners from one-wayness and applications to TLS. Cryptology ePrint Archive, Report 2022/065, 2022. (Cited on page 3.)

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Berlin, Heidelberg, December 2001. (Cited on page 3.)

BBF+19. Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Cham, 2019. (Cited on pages 2, 3, 12, and 13.)

BCD+24. Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-Wing. *CiC*, 1(1):21, 2024. (Cited on pages 3, 11, 13, 15, and 17.)

BJKS24. Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. Formal verification of the PQXDH post-quantum key agreement protocol for end-to-end secure messaging. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. (Cited on page 12.)

CBG25. Deirdre Connolly, Richard Barnes, and Paul Grubbs. Hybrid PQ/T Key Encapsulation Mechanisms. Internet-Draft draft-irtf-cfrg-hybrid-kems-05, Internet Engineering Task Force, July 2025. Work in Progress. (Cited on pages 3 and 11.)

CDM24. Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping up with the KEMs: Stronger security notions for KEMs and automated analysis of KEM-based protocols. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1046–1060. ACM Press, October 2024. (Cited on pages 1, 3, 4, 5, 12, 14, 20, 25, and 26.)

CHH+25. Deirdre Connolly, Kathrin Hövelmanns, Andreas Hülsing, Stavros Kousidis, and Matthias Meijers. Starfighters—on the general applicability of X-wing. Cryptology ePrint Archive, Report 2025/1397, 2025. (Cited on pages 3 and 11.)

CLM25. Céline Chevalier, Guirec Lebrun, and Ange Martinelli. Spilling-cascade: An optimal PKE combiner for KEM hybridization. In Marc Fischlin and Veelasha Moonsamy, editors, *ACNS 2025, Part I*, volume 15825 of *LNCS*, pages 455–485. Springer, Cham, June 2025. (Cited on page 2.)

CP20. Matthew Campagna and Adam Petcher. Security of hybrid key encapsulation. Cryptology ePrint Archive, Report 2020/1364, 2020. (Cited on page 2.)

CSW25. Deirdre Connolly, Peter Schwabe, and Bas Westerbaan. X-Wing: general-purpose hybrid post-quantum KEM. Internet-Draft draft-connolly-cfrg-xwing-kem-09, Internet Engineering Task Force, September 2025. Work in Progress. (Cited on page 11.)

FLPQ13. Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, Berlin, Heidelberg, February / March 2013. (Cited on page 3.)

GHP18. Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 190–218. Springer, Cham, March 2018. (Cited on pages 2 and 3.)

GMP22. Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 402–432. Springer, Cham, May / June 2022. (Cited on pages 3, 12, and 14.)

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017. (Cited on page 1.)

HV21. Loïs Huguenin-Dumittan and Serge Vaudenay. FO-like combiners and hybrid post-quantum cryptography. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 225–244. Springer, Cham, December 2021. (Cited on page 2.)

Jos25. Simon Josefsson. Chempat: Generic Instantiated PQ/T Hybrid Key Encapsulation Mechanisms. Internet-Draft draft-josefsson-chempat-03, Internet Engineering Task Force, March 2025. Work in Progress. (Cited on pages 3, 11, and 12.)

KMSW26. Juliane Krämer, Yannick Münz, Patrick Struck, and Maximiliane Weishäupl. Binding security of explicitly-rejecting KEMs via plaintext confirmation and robust PKEs. *CiC*, 2(4):34, 2026. (Cited on page 24.)

KRSW25. Stavros Kousidis, Johannes Roth, Falko Strenzke, and Aron Wussler. Post-Quantum Cryptography in OpenPGP. Internet-Draft draft-ietf-openpgp-pqc-13, Internet Engineering Task Force, October 2025. Work in Progress. (Cited on pages 2, 3, 10, 11, 12, and 15.)

KSW25. Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Binding security of implicitly-rejecting KEMs and application to BIKE and HQC. *CiC*, 2(2):19, 2025. (Cited on pages 3, 4, 12, 13, 14, and 16.)

MPR+24. Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. NIST Internal Report NIST IR 8547 ipd, National Institute of Standards and Technology, Gaithersburg, MD, 2024. (Cited on page 2.)

NIS17. NIST. Post-quantum cryptography standardization process. https://csrc.nist.gov/projects/post-quantum-cryptography, 2017. (Cited on page 3.)

NIST24. National Institute of Standards and Technology. Module-Lattice-based Key-Encapsulation Mechanism Standard. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 203 ipd., 2024. (Cited on page 1.)

OGP+25. Mike Ounsworth, John Gray, Massimiliano Pala, Jan Klaußner, and Scott Fluhrer. Composite ML-KEM for use in X.509 Public Key Infrastructure. Internet-Draft draft-ietf-lamps-pq-composite-kem-07, Internet Engineering Task Force, June 2025. Work in Progress. (Cited on pages 3, 11, 12, and 15.)

SFG25. Douglas Stebila, Scott Fluhrer, and Shay Gueron. Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-16, Internet Engineering Task Force, September 2025. Work in Progress. (Cited on pages 3, 11, 12, 13, and 16.)

Xag22. Keita Xagawa. Anonymity of NIST PQC round 3 KEMs. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 551–581. Springer, Cham, May / June 2022. (Cited on page 3.)

A Results for Ciphertext Binding

In this section, we give our results regarding the X-Bind-CT-Q security of the generic combiner. In Theorem 5, we treat X-Bind-CT-PK, while Corollary 6 and Theorem 7 concern X-Bind-CT-K. Note that these notions are unachievable for any KEM using implicit rejection as shown in [CDM24]. Looking ahead, the KEMs we will analyze in Section 4 all use implicit rejection, making the results of this section inapplicable. Nevertheless, they are of relevance for any combination of explicitly-rejecting KEMs.

The theorem below shows under which conditions the ciphertext binds the public key. In the Hon and Leak setting, having the property by at least one of the underlying KEMs is sufficient for the combined KEM to inherit it. In the Mal setting, the property is required from both underlying KEMs as the adversary can freely choose the key pairs: if the property is easy to break for one KEM, the adversary can simply choose distinct key pairs which only disagree on the easy-to-break KEM.

Theorem 5. *Let* KEM_1 *and* KEM_2 *be two KEMs. Let further* cKEM *denote the KEM resulting from applying* GC *to* KEM_1 *and* KEM_2*. Then for any adversary* $\mathcal{A}$ *there exist adversaries* $\mathcal{M}_i$*,* $\mathcal{L}_i$*, and* $\mathcal{H}_i$ *for* $i \in \{1,2\}$*, such that*

$$\mathbf{Adv}^{\text{X-Bind-CT-PK}}_{\text{cKEM}}(\mathcal{A}) \leq \begin{cases} \sum\limits_{i\in\{1,2\}} \mathbf{Adv}^{\text{Mal-Bind-CT-PK}}_{\text{KEM}_i}(\mathcal{M}_i) & \text{if X} = \text{Mal}, \\ \min\limits_{i\in\{1,2\}} \mathbf{Adv}^{\text{Leak-Bind-CT-PK}}_{\text{KEM}_i}(\mathcal{L}_i) & \text{if X} = \text{Leak}, \\ \min\limits_{i\in\{1,2\}} \mathbf{Adv}^{\text{Hon-Bind-CT-PK}}_{\text{KEM}_i}(\mathcal{H}_i) & \text{if X} = \text{Hon}. \end{cases}$$

Proof. Let $\mathcal{A}$ be an adversary that breaks X-Bind-CT-PK security of cKEM. We distinguish three cases based on the attack model:

Case 1: X = Mal

Adversary $\mathcal{A}$ outputs two key pairs $(pk, sk), (\overline{pk}, \overline{sk})$, where $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$, $\overline{pk} = (\overline{pk}_1, \overline{pk}_2)$, and $\overline{sk} = (\overline{sk}_1, \overline{sk}_2)$.[12] For $\mathcal{A}$ to be successful

[12] Strictly speaking, the adversary only needs to output the secret key whereas the public keys are derived from those.

w.r.t. X-BIND-CT-PK, it is required that $pk \neq \overline{pk}$, which implies that *at least one* of the following two inequalities has to hold: $pk_1 \neq \overline{pk}_1$ or $pk_2 \neq \overline{pk}_2$. Thus we get

$$\begin{aligned}\mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{cKEM}}(\mathcal{A}) \leq \Pr[\mathcal{A} \text{ wins X-BIND-CT-PK} \wedge pk_1 \neq \overline{pk}_1] \\ + \Pr[\mathcal{A} \text{ wins X-BIND-CT-PK} \wedge pk_2 \neq \overline{pk}_2]\,.\end{aligned}$$

In the following, we bound the first term; bounding the second term works completely analogue. For a successful $\mathcal{A}$ there are three scenarios: DECAPS–DECAPS, ENCAPS–DECAPS, and ENCAPS–ENCAPS. We give the details for the first scenario (DECAPS–DECAPS) and subsequently discuss how it extends to the others. In the DECAPS–DECAPS scenario, $\mathcal{A}$ outputs a single ciphertext c which decapsulates under both secret keys to some valid shared keys, i.e.,

$$\begin{aligned}\perp \neq k \leftarrow \text{DECAPS}(sk, c)\,, \text{ and} \\ \perp \neq \overline{k} \leftarrow \text{DECAPS}(\overline{sk}, c)\,.\end{aligned}$$

By construction, this means that the ciphertexts of the underlying KEMs validly decapsulate under the underlying secret keys, i.e., for $i \in \{1, 2\}$, we have

$$\perp \neq k_i \leftarrow \text{DECAPS}(sk_i, c_i)\,.$$

Using $\mathcal{M}_1$ from our generic reductions then yields

$$\Pr[\mathcal{A} \text{ wins X-BIND-CT-PK} \wedge pk_1 \neq \overline{pk}_1] \leq \mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{KEM}_1}(\mathcal{M}_1)\,.$$

In the other scenarios, $\mathcal{A}$ either outputs a ciphertext and a secret key for the first tuple and a public key and random coins for the second tuple (ENCAPS–DECAPS) or two tuples consisting of public key and random coins each (ENCAPS–ENCAPS). Regardless of which, the ciphertexts are required to be equal and must decrypt to valid shared keys. Our adversary $\mathcal{M}_1$ can simply take the parts belonging to KEM_1 from $\mathcal{A}$'s output to also break MAL-BIND-CT-PK of KEM_1 (in the same scenario as $\mathcal{A}$ breaks MAL-BIND-CT-PK of cKEM). For the second term, a similar argument yields

$$\Pr[\mathcal{A} \text{ wins X-BIND-CT-PK} \wedge pk_2 \neq \overline{pk}_2] \leq \mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{KEM}_2}(\mathcal{M}_2)\,.$$

Collecting the above then yields

$$\begin{aligned}\mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{cKEM}}(\mathcal{A}) &\leq \sum_{i\in\{1,2\}} \Pr[\mathcal{A} \text{ wins X-BIND-CT-PK} \wedge pk_i \neq \overline{pk}_i] \\ &\leq \sum_{i\in\{1,2\}} \mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{KEM}_i}(\mathcal{M}_i)\,.\end{aligned}$$

Case 2: X = LEAK

Adversary $\mathcal{A}$ receives two key pairs $(pk, sk), (\overline{pk}, \overline{sk})$, where $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$, $\overline{pk} = (\overline{pk}_1, \overline{pk}_2)$, and $\overline{sk} = (\overline{sk}_1, \overline{sk}_2)$. With overwhelming probability, we have $pk_i \neq \overline{pk}_i$ and $sk_i \neq \overline{sk}_i$ for $i \in \{1, 2\}$. Adversary $\mathcal{A}$ outputs a

single ciphertext $c = (c_1, c_2)$ such that

$$\bot \neq k \leftarrow \text{cKEM}(sk, c) \text{ and}$$
$$\bot \neq \overline{k} \leftarrow \text{cKEM}(\overline{sk}, c)\,.$$

By construction, this implies, for $i \in \{1, 2\}$, that

$$\bot \neq k_i \leftarrow \text{KEM}_i(sk_i, c_i) \text{ and}$$
$$\bot \neq \overline{k}_i \leftarrow \text{KEM}_i(\overline{sk}_i, c_i)\,,$$

as otherwise one of the inputs to COMBINE would be $\bot$. For $\mathcal{L}_i$ from our generic reductions, we then obtain

$$\mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{cKEM}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{KEM}_1}(\mathcal{L}_1) \text{ and}$$
$$\mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{cKEM}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{KEM}_2}(\mathcal{L}_2)\,,$$

which yields

$$\mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{cKEM}}(\mathcal{A}) \leq \min_{i \in \{1,2\}} \mathbf{Adv}^{\text{LEAK-BIND-CT-PK}}_{\text{KEM}_i}(\mathcal{L}_i)\,.$$

Case 3: X = HON

The HON setting works essentially like the LEAK setting. The mere difference is that $\mathcal{A}$ has access to a decapsulation oracle which the reduction can simulate by locally evalutating every computation related to the KEM that it generated itself and its own decapsulation oracle for every computation related to the KEM that it plays against. □

The hierarchy between X-BIND-CT-PK and X-BIND-CT-K (cf. Lemma 3) in the HON and LEAK setting allows us to derive a bound for HON-BIND-CT-K and LEAK-BIND-CT-K from Theorem 5.

Corollary 6. *Let* KEM_1 *and* KEM_2 *be two KEMs. Let further* cKEM *denote the KEM resulting from applying* GC *to* KEM_1 *and* KEM_2*. Then for any adversary* $\mathcal{A}$ *and* $\text{X} \in \{\text{HON}, \text{LEAK}\}$*, there exist adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, such that*

$$\mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{cKEM}}(\mathcal{A}) \leq \min_{i \in \{1,2\}} \mathbf{Adv}^{\text{X-BIND-CT-PK}}_{\text{KEM}_i}(\mathcal{B}_i)\,.$$

Proof. This follows directly from Lemma 3 and Theorem 5. □

Although the above corollary yields an initial bound for HON-BIND-CT-K and LEAK-BIND-CT-K, a better bound—using weaker assumptions on the underlying KEMs—can be derived through a dedicated analysis. Further, the results so far do not cover MAL-BIND-CT-K.

The theorem below provides two bounds on the X-BIND-CT-K security of a KEM obtained via the generic combiner GC, subject to the public keys *not* being an input to the combine function COMBINE. The first bound holds for $\text{X} \in \{\text{HON}, \text{LEAK}\}$ and states that if *any* of the two underlying KEMs is

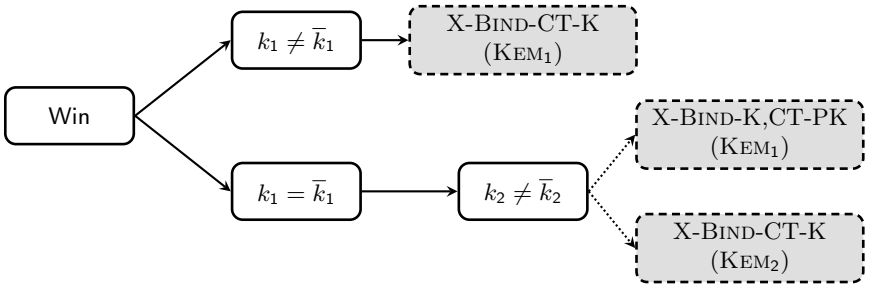


Fig. 4. Case distinction for the proof of Theorem 7. The gray boxes show which of the events $\mathsf{E}[\text{KEM}_i]$ (X-IND-P-CT security of KEM_i being broken) occurs in each case. The dotted lines indicate that both cases hold in the HON and LEAK setting, while in the MAL setting only the implication corresponding to the lower dotted line is true.

both X-IND-CT-K *and* X-IND-K,CT-PK secure, then the combined KEM is X-IND-CT-K secure. The second bound covers $\text{X} \in \{\text{HON}, \text{LEAK}, \text{MAL}\}$ and states that the combined KEM is X-IND-CT-K secure if *both* of the underlying KEMs are X-IND-CT-K secure.

Theorem 7. *Let* KEM_1 *and* KEM_2 *be two KEMs. Let further* cKEM *denote the KEM resulting from applying* GC *to* KEM_1 *and* KEM_2 *(with combiner function* COMBINE *and input-selection function* f *for which* $f(\text{PK}, \cdot) = 0$*). Then for any adversary* $\mathcal{A}$*, there exist adversaries* $\mathcal{B}_i$ *and* $\mathcal{C}_i$ *for* $i \in \{1, 2\}$*, such that the following bounds hold*
(1) for $\text{X} \in \{\text{HON}, \text{LEAK}\}$*:*

$$\begin{aligned}&\mathbf{Adv}^{\text{X-IND-CT-K}}_{\text{cKEM}[f]}(\mathcal{A})\\ &\leq \min_{i\in\{1,2\}} \left(\mathbf{Adv}^{\text{X-IND-CT-K}}_{\text{KEM}_i}(\mathcal{B}_i) + \mathbf{Adv}^{\text{X-IND-K,CT-PK}}_{\text{KEM}_i}(\mathcal{C}_i)\right)\end{aligned}$$

(2) for $\text{X} \in \{\text{HON}, \text{LEAK}, \text{MAL}\}$*:*

$$\mathbf{Adv}^{\text{X-IND-CT-K}}_{\text{cKEM}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{X-IND-CT-K}}_{\text{KEM}_1}(\mathcal{B}_1) + \mathbf{Adv}^{\text{X-IND-CT-K}}_{\text{KEM}_2}(\mathcal{B}_2)\,.$$

Proof. We first handle the HON and LEAK case and subsequently discuss how the results extend to the MAL setting. Let $\mathcal{A}$ be an adversary against the X-IND-CT-K security of cKEM for $\text{X} \in \{\text{HON}, \text{LEAK}\}$. This adversary receives two honestly generated public keys pk, $\overline{pk}$ (in the case that $\text{X} = \text{HON}$) and two key pairs $(pk, sk), (\overline{pk}, \overline{sk})$ (in the case that $\text{X} = \text{LEAK}$), where $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$, $\overline{pk} = (\overline{pk}_1, \overline{pk}_2)$, and $\overline{sk} = (\overline{sk}_1, \overline{sk}_2)$.[13] It outputs a ciphertext

[13] The X-IND-CT-K notion technically allows $\mathcal{A}$ to also play against just a single key pair by setting the key pairs to be the same. In this case, however, $\mathcal{A}$ cannot win because it requires that the (deterministic) decapsulation algorithm outputs different shared keys on identical inputs.

$c = (c_1, c_2)$ such that $k \neq \overline{k}$ (with $k, \overline{k} \notin \{\bot\}$), where

$$k \leftarrow \text{cKEM.DECAPS}(sk, c) \quad \text{and}$$
$$\overline{k} \leftarrow \text{cKEM.DECAPS}(\overline{sk}, c)\,.$$

Note that the shared keys are computed as

$$k \leftarrow \text{COMBINE}(k_1, k_2, \text{F}(pk_1, pk_2, c_1, c_2), \Xi) \quad \text{and}$$
$$\overline{k} \leftarrow \text{COMBINE}(\overline{k}_1, \overline{k}_2, \text{F}(\overline{pk}_1, \overline{pk}_2, c_1, c_2), \Xi)\,,$$

where $f(\text{PK}, \cdot) = 0$ holds by assumption. To ensure different values for k and $\overline{k}$, we need at least one of the following two inequalities to hold: $k_1 \neq \overline{k}_1$ or $k_2 \neq \overline{k}_2$. We consider different cases as illustrated in Fig. 4 (note that the argument works exactly the same with flipped indices). We have

$$\Pr[\mathsf{Win}] = \Pr[\mathsf{Win} \wedge k_1 \neq \overline{k}_1] + \Pr[\mathsf{Win} \wedge k_1 = \overline{k}_1]\,.$$

Using the generic reductions, we obtain the following bound for the first term.

$$\Pr[\mathsf{Win} \wedge k_1 \neq \overline{k}_1] \leq \mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{KEM}_1}(\mathcal{B}_1)\,.$$

For the second term, there are two options. First, we obtain

$$\Pr[\mathsf{Win} \wedge k_1 = \overline{k}_1] \leq \mathbf{Adv}^{\text{X-BIND-K,CT-PK}}_{\text{KEM}_1}(\mathcal{C}_1)\,.$$

Alternatively, as $k_1 = \overline{k}_1$, we know that the other keys have to differ, i.e., $k_2 \neq \overline{k}_2$. In that case, we can conclude that

$$\Pr[\mathsf{Win} \wedge k_1 = \overline{k}_1 \wedge k_2 \neq \overline{k}_2] \leq \mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{KEM}_2}(\mathcal{B}_2)\,.$$

Collecting the above and using the observation that the argument works the same if the indices are flipped, yields the final two bounds:

$$\begin{aligned}&\mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{cKEM}[f]}(\mathcal{A})\\ &\leq \min_{i \in \{1,2\}} \left(\mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{KEM}_i}(\mathcal{B}_i) + \mathbf{Adv}^{\text{X-BIND-K,CT-PK}}_{\text{KEM}_i}(\mathcal{C}_i)\right)\end{aligned}$$

and

$$\mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{cKEM}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{KEM}_1}(\mathcal{B}_1) + \mathbf{Adv}^{\text{X-BIND-CT-K}}_{\text{KEM}_2}(\mathcal{B}_2)\,.$$

Lastly, we consider the MAL setting, in which the adversary can choose the two key pairs $(pk, sk), (\overline{pk}, \overline{sk})$ with $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$, $\overline{pk} = (\overline{pk}_1, \overline{pk}_2)$, and $\overline{sk} = (\overline{sk}_1, \overline{sk}_2)$ itself. In the HON and LEAK setting, we could argue that the underlying key pairs are different as $\mathcal{A}$ cannot win otherwise. However, this argument does not apply in the MAL setting.[14] The adversary could generate *different* key pairs for one of the KEMs while using *the same* key pair for the

[14] We refer to [KMSW26, Theorem 16] for a counterexample.

other KEM. Thus, using the generic reduction, we do not necessarily get an adversary that breaks Mal-Bind-K,CT-PK security of one of the underlying KEMs. The other argument still applies. This yields

$$\mathbf{Adv}^{\text{Mal-Bind-CT-K}}_{\text{cKEM}[f]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{Mal-Bind-CT-K}}_{\text{KEM}_1}(\mathcal{B}_1) + \mathbf{Adv}^{\text{Mal-Bind-CT-K}}_{\text{KEM}_2}(\mathcal{B}_2)\,.$$

This concludes the proof. □

Remark 8. Note that the first bound given in Theorem 7 for $\text{X} \in \{\text{Hon}, \text{Leak}\}$, i.e.,

$$\begin{aligned}&\mathbf{Adv}^{\text{X-Bind-CT-K}}_{\text{cKEM}[f]}(\mathcal{A})\\ &\leq \min_{i\in\{1,2\}} \left(\mathbf{Adv}^{\text{X-Bind-CT-K}}_{\text{KEM}_i}(\mathcal{B}_i) + \mathbf{Adv}^{\text{X-Bind-K,CT-PK}}_{\text{KEM}_i}(\mathcal{C}_i)\right),\end{aligned}$$

also follows from Corollary 6, i.e.,

$$\mathbf{Adv}^{\text{X-Bind-CT-K}}_{\text{cKEM}[f]}(\mathcal{A}) \leq \min_{i\in\{1,2\}} \left(\mathbf{Adv}^{\text{X-Bind-CT-PK}}_{\text{KEM}_i}(\mathcal{B}_i)\right).$$

This can be seen using the fact that

$$\text{X-Bind-CT-PK} \Leftrightarrow (\text{X-Bind-CT-K,PK} \wedge \text{X-Bind-K,CT-PK})\,,$$

which follows from [CDM24, Lemma 4.4 and Theorem 4.5] and the implication X-Bind-CT-K $\Rightarrow$ X-Bind-CT-K,PK, which holds in the Hon and Leak setting by Lemma 3.[15]

The following theorem shows that having Leak-Bind-CT-K security for only one of the underlying KEMs is insufficient for the combined KEM to inherit this security notion.

Theorem 9. *Let* KEM_1 *and* KEM_2 *be two KEMs. Let further* cKEM *denote the KEM resulting from applying* GC *to* KEM_1 *and* KEM_2*. Assume that* KEM_1 *is* Leak-Bind-CT-K *secure but not* Leak-Bind-K,CT-PK *secure, i.e., there exists an adversary* $\mathcal{A}_1$ *such that* $\mathbf{Adv}^{\text{Leak-Bind-K,CT-PK}}_{\text{KEM}_1}(\mathcal{A}_1)$ *is non-negligible. Assume further that* KEM_2 *is not* Leak-Bind-CT-K *secure, i.e., there exists an adversary* $\mathcal{A}_2$ *such that* $\mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{KEM}_2}(\mathcal{A}_2)$ *is non-negligible. Then there exists an adversary* $\mathcal{A}$ *such that*

$$\begin{aligned}&\mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{cKEM}}(\mathcal{B})\\ &\quad\geq \mathbf{Adv}^{\text{Leak-Bind-K,CT-PK}}_{\text{KEM}_1}(\mathcal{A}_1) \cdot \mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{KEM}_2}(\mathcal{A}_2) - \text{negl}\,.\end{aligned}$$

Proof. The idea of the proof is as follows. Adversary $\mathcal{A}$—receiving two combiner key pairs, each consisting of one key pair per underlying KEM—simply runs

[15] Note that the notion X-Bind-CT-K,PK is not explicitly mentioned in the framework from [CDM24]. However, it easily arises as the special case of the notion X-Bind-CT-PK, additionally requiring that the shared keys have to differ.

the adversaries against the underlying KEMs. By assumption, both output a *single* ciphertext that decrypts validly under both key pairs; this condition is necessary to ensure that cKem does not output $\perp$. Since one of the adversaries (in our case $\mathcal{A}_2$ which breaks Leak-Bind-CT-K of Kem_2) outputs a ciphertext that decrypts to *different* shared keys, this is sufficient to ensure that Combine receives different inputs, thus resulting in different shared keys for the combined KEM.

Let $\mathcal{A}$ be an adversary against the Leak-Bind-CT-K security of cKem. It requests two honestly generated key pairs $(pk, sk), (\overline{pk}, \overline{sk})$, where $pk = (pk_1, pk_2)$, $sk = (sk_1, sk_2)$, $\overline{pk} = (\overline{pk}_1, \overline{pk}_2)$, and $\overline{sk} = (\overline{sk}_1, \overline{sk}_2)$, as input. With overwhelming probability, the individual key pairs are distinct, i.e., $(pk_1, sk_1) \neq (\overline{pk}_1, \overline{sk}_1)$ and $(pk_2, sk_2) \neq (\overline{pk}_2, \overline{sk}_2)$. Then, $\mathcal{A}$ runs $\mathcal{A}_1((pk_1, sk_1), (\overline{pk}_1, \overline{sk}_1))$ to obtain c_1 and $\mathcal{A}_2((pk_2, sk_2), (\overline{pk}_2, \overline{sk}_2))$[16] to obtain c_2. Finally, $\mathcal{A}$ outputs the ciphertext $c \leftarrow (c_1, c_2)$.

By assumption on the adversaries $\mathcal{A}_1$ (against Leak-Bind-K,CT-PK) and $\mathcal{A}_2$ (against Leak-Bind-CT-K), the following two equations hold with probability $\mathbf{Adv}^{\text{Leak-Bind-K,CT-PK}}_{\text{Kem}_1}(\mathcal{A}_1)$ and $\mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{Kem}_2}(\mathcal{A}_2)$, respectively:

$$\perp \neq \text{Kem}_1.\text{Decaps}(sk_1, c_1) = \text{Kem}_1.\text{Decaps}(\overline{sk}_1, c_1) \neq \perp$$
$$\perp \neq \text{Kem}_2.\text{Decaps}(sk_2, c_2) \neq \text{Kem}_2.\text{Decaps}(\overline{sk}_2, c_2) \neq \perp$$

For $i \in \{1, 2\}$, let $k_i = \text{Kem}_i.\text{Decaps}(sk_i, c_i)$ and $\overline{k}_i = \text{Kem}_i.\text{Decaps}(\overline{sk}_i, c_i)$. By construction of cKem, it holds that

$$k = \text{cKem.Decaps}(sk, c) = \text{Combine}(k_1, k_2, \text{F}(pk_1, pk_2, c_1, c_2), \Xi)$$

and

$$\overline{k} = \text{cKem.Decaps}(\overline{sk}, c) = \text{Combine}(\overline{k}_1, \overline{k}_2, \text{F}(\overline{pk}_1, \overline{pk}_2, c_1, c_2), \Xi)\,.$$

Since $k_2 \neq \overline{k}_2$, we have $k \neq \overline{k}$ except when the inputs form a collision for Combine—which happens only with negligible probability. Collecting everything then yields

$$\begin{aligned}&\mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{cKem}}(\mathcal{B})\\ &\quad\geq \mathbf{Adv}^{\text{Leak-Bind-K,CT-PK}}_{\text{Kem}_1}(\mathcal{A}_1) \cdot \mathbf{Adv}^{\text{Leak-Bind-CT-K}}_{\text{Kem}_2}(\mathcal{A}_2) - \text{negl}\,,\end{aligned}$$

which concludes the proof. □

Note that the above theorem holds regardless of the choice of f. Thus, it is also applicable in case of $f(\text{PK}, \cdot) = 0$ as was the case for Theorem 7, thereby showing that having Leak-Bind-CT-K security on only one underlying KEM is insufficient to achieve the same security on the combined KEM. Note further that Theorem 9 requires a KEM that is Leak-Bind-K,CT-PK secure but insecure w.r.t. Leak-Bind-CT-K; Cremers et al. [CDM24, Appendix B.6] show that KEMs with this property exist.

[16] In theory, $\mathcal{A}$ can request to play against a *single* key pair. However, in this case, it effectively plays Leak-Bind-CT,PK-K which is trivially achieved by *any* KEM. Hence, w.l.o.g. we can assume that $\mathcal{A}_2$ requests two different key pairs.