IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 2, 34 pages.

https://doi.org/10.62056/ak2i893y6

Binding Security of Implicitly-Rejecting KEMs and Application to BIKE and HQC

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

[1] University of Regensburg, Regensburg, Germany
[2] University of Konstanz, Konstanz, Germany

Abstract. In this work, we continue the analysis of the binding properties of implicitly-rejecting key-encapsulation mechanisms (KEMs) obtained via the Fujisaki-Okamoto (FO) transform. These binding properties, in earlier literature known under the term robustness, thwart attacks that can arise when using KEMs in complex protocols. Recently, Cremers et al. (CCS'24) introduced a framework for binding notions, encompassing previously existing but also new ones. While implicitly-rejecting FO-KEMs have been analyzed with respect to multiple of these notions, there are still several gaps. We complete the picture by providing positive and negative results for the remaining notions. Further, we show how to apply our results to the code-based KEMs BIKE and HQC, which were round-4 candidates in NIST's PQC standardization process. Through this, we close a second gap as our results complete the analysis of the binding notions for the NIST round-4 KEMs. Finally, we give a modified version of the FO transform that achieves all binding notions.

1 Introduction

Encryption is unequivocally the most fundamental concept of cryptography. Nowadays, encryption can be divided into symmetric encryption and asymmetric (public-key) encryption. For efficiency reasons, public-key encryption is typically used to encrypt only a symmetric key (of some fixed length), while the actual payload will be encrypted using that symmetric key. That is, public-key encryption (PKE) is essentially used to encrypt uniformly random messages that are used as symmetric keys. Key-encapsulation mechanisms (KEMs) are a primitive dedicated to this use-case: here, the encapsulation algorithm only takes a public key pk as input and outputs a uniform symmetric key k alongside a ciphertext c that encapsulates the symmetric key.

The standard security notion that KEMs should achieve is IND-CCA security, which is also the requirement of the NIST standardization process [NIS17]. The common method to obtain IND-CCA secure KEMs is to design IND-CPA secure public-key encryption schemes and then apply the Fujisaki-Okamoto (FO) transform [FO99] to it—a method that is used for virtually all KEMs in the NIST standardization process for post-quantum schemes [NIS17]. The FO transform comes in different variants. In this work, we are mainly concerned with the variant $\mathsf{FO}^{\not\perp}$, which computes the shared key as $\mathsf{H}(m, c)$ while ciphertexts are rejected by outputting $\mathsf{H}(\sigma, c)$ instead, where σ is the implicit rejection value contained in the secret key. Another variant (also covered in this work) is $\mathsf{FO}_m^{\not\perp}$,

E-mail: juliane.kraemer@ur.de (Juliane Krämer), patrick.struck@uni.kn (Patrick Struck), maximiliane.weishaeupl@ur.de (Maximiliane Weishäupl)

*We thank Rune Fiedler for helpful discussions regarding the extension of our results to the LEAK^{+r} attack model. This work was funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) – 505500359, by the Hector Foundation II, and by the German Federal Ministry of Research, Technology and Space (BMFTR) under the project Quant-ID (16KISQ111) and QUDIS (16KIS2091).

Received: 2025-04-07 Accepted: 2025-06-02

which works similar except that the shared key is computed as $\mathsf{H}(m)$ as opposed to $\mathsf{H}(m, c)$. The explicitly-rejecting variants $\mathsf{FO}^{\perp}$ and $\mathsf{FO}^{\perp}_{m}$, which are not in the scope of this work, are defined analogously to their implicitly-rejecting counterparts except that invalid ciphertext are rejected by outputting $\perp$ rather than $\mathsf{H}(\sigma, c)$.

Using cryptographic primitives in larger protocols opens up room for misuse which is not prevented by the standard security notions. Several attacks show that this is a problem for digital signatures [JCCS19, Aye15] as well as authenticated encryption [DGRW18, ADG+22, LGR21]. Very recently, this was also shown to be a problem for KEMs: Cremers et al. [CDM24] showed that the authentication protocol presented in [BDK+18] is vulnerable to what is called a "re-encapsulation attack".[1] Here, an adversary can decapsulate a ciphertext c to obtain a shared key k and then "re-encapsulate" that shared key under a different public key. This attack allows an adversary Eve to convince Alice that she shares a key with Eve although the key is actually shared between Alice and Bob—and unknown to Eve. While this does not compromise default security properties like confidentiality, it violates the so-called *implicit key agreement* that the protocol aims to achieve.

To deal with the general threat of misuse, more advanced security properties for core cryptographic primitives have emerged and enjoyed a lot of research in recent years. For authenticated encryption schemes, committing security prevents adversaries from finding ciphertexts that decrypt under more than one key, for which there is already a long line of research, e.g., [BH22, MLGR23, BH24, SW24, KSW24]. For digital signatures, BUFF security [CDF+21] prevents against various attacks that go beyond the unforgeability features of digital signatures and received a lot of attention lately [ADM+24, DFHS24, DFF24, DFH+24]. Both committing security and BUFF security are considered in the new standardization processes by NIST [NIS24, NIS22]. Very recently, Cremers et al. [CDM24]—motivated by the re-encapsulation attack—developed a framework for so-called "binding properties" of KEMs which prevent against various attacks. Note that this framework encompasses also the already existing robustness notions [GMP22], but renames them to match the new notation.

All notions are built following the pattern X-BIND-P-Q, where P and Q describe which elements (P) are binding which other elements (Q). An example (and one notion that we are considering in this work) is that P equals the shared key k of a KEM and Q equals the public key pk. The corresponding binding notion X-BIND-K-PK then formalizes that k binds pk or, simply speaking, an adversary cannot find ciphertexts that decapsulate to the same shared key k under distinct public keys. The other variable, X, describes the attack model and differs in how the keys are selected. Cremers et al. [CDM24] distinguish three cases: the malicious (MAL) setting where the adversary can generate the keys arbitrarily, the leaked (LEAK) setting where the keys are honestly generated and the adversary receives *all* keys, and the honest (HON) setting where the keys are honestly generated but the adversary only receives the public key along with access to decapsulation oracles for the secret keys. Thus, the three attack models exhibit the hierarchy MAL→LEAK→HON ordered from strong to weak. In total, Cremers et al. [CDM24] give 18 different security notions, which arise as the sensible combinations involving the shared key (K), the public key (PK), and the ciphertext (CT). These 18 notions can be expressed by the 6 general notions X-BIND-K-PK, X-BIND-K,CT-PK, X-BIND-K-CT, X-BIND-K,PK-CT, X-BIND-CT-PK, and X-BIND-CT-K for $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$. Note that for X-BIND-K,CT-PK (and analogously for the other notions that include all three components K, PK, and CT), the notation is to be understood as $\mathrm{P} = \{\mathrm{K}, \mathrm{CT}\}$ binding $\mathrm{Q} = \{\mathrm{PK}\}$.

There is a number of results regarding the security of implicitly-rejecting KEMs with respect to the binding notions. In the original work, Cremers et al. [CDM24] show

[1] Note that this attack assumes the instantiation of the protocol with a KEM that does not fulfill the necessary binding notions—the instantiation with Kyber presented in [BDK+18] is not vulnerable.

that the ciphertext cannot bind any other value, i.e., the six notions X-BIND-CT-PK and X-BIND-CT-K with X $\in$ {MAL, LEAK, HON} are unachievable. The reason is that implicitly-rejecting KEMs always output some key $k \neq \perp$. We will generally exclude these six notions in this work which leaves us with 12 notions. Further, it is a direct implication of [CDM24, Theorem D.1] that schemes which use the FO-variant $\mathsf{FO}^{\not\perp}$ do achieve X-BIND-K-CT (and hence also X-BIND-K,PK-CT) security for X $\in$ {MAL, LEAK, HON}. Lastly, Grubbs et al. [GMP22, Theorem 6] prove that $\mathsf{FO}^{\not\perp}$-KEMs do fulfill HON-BIND-K,CT-PK security provided that a certain collision-freeness property holds for the underlying PKE scheme. With respect to $\mathsf{FO}^{\not\perp}_m$, Cremers et al. [CDM24] show that it achieves LEAK-BIND-K,PK-CT security (hence also HON-BIND-K,PK-CT security) while it is neither X-BIND-K-PK nor X-BIND-K-CT secure (for any X $\in$ {HON, LEAK, MAL}).[2] This leaves several notions open for $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. Figure 1 provides an overview of the notions considered in this work, as well as the existing and our new results for $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$.

Next to these general results, the binding notions have been (partially) analyzed for several of the KEMs included in NIST's standardization process: Grubbs et al. [GMP22] analyzed the security of Classic-McEliece [ABC+22], Hqc [AAB+20] (round-3 version), Saber [DKR+20], FrodoKem [NAB+20], and Kyber [BDK+18] with respect to some of the binding notions. The analysis of those four schemes and Ml-Kem [NIS23] was extended by Cremers et al. [CDM24] and more results for Ml-Kem were given by Schmieg [Sch24]. The remaining two round-4 KEMs Bike [ABB+22] and the current version of Hqc [AAB+24][3] (which was very recently selected for standardization by NIST), however, are yet to be analyzed with respect to *any* of the binding properties.[4]

1.1 Contribution

In this work, we close both of the aforementioned gaps. We complete the analysis of the binding notions for implicitly-rejecting KEMs, i.e., those resulting from either $\mathsf{FO}^{\not\perp}$ or $\mathsf{FO}^{\not\perp}_m$. An overview of the results is provided in Table 1 and further illustration is given in Figure 1. Further, we complete the binding analysis of the NIST round-4 KEMs with the results being depicted in Table 2. Finally, we provide different variants of the FO transform (cf. Table 3) that achieve all notions with respect to the different attack models (HON/LEAK/MAL)—two are established variants, while the strongest one is newly developed by us.

Regarding $\mathsf{FO}^{\not\perp}$-KEMs, we show the following. Firstly, for $\mathsf{FO}^{\not\perp}$-KEMs, we prove equivalence between HON-BIND-K-PK and the (weaker) notion HON-BIND-K,CT-PK.[5] This allows us to then leverage [GMP22, Theorem 6], which provides requirements for $\mathsf{FO}^{\not\perp}$-KEMs to achieve HON-BIND-K,CT-PK. The core requirement is that the underlying PKE scheme needs to satisfy a weak form of collision-freeness, called SCFR-CPA. Secondly, we formalize a property of the underlying PKE scheme, called restricted non-rigidity, such that the KEM (resulting from applying the $\mathsf{FO}^{\not\perp}$ transform) is vulnerable with respect to any of the four notions LEAK-BIND-K-PK, LEAK-BIND-K,CT-PK, MAL-BIND-K-PK, and MAL-BIND-K,CT-PK. For the latter two we also give another attack that does not impose any requirement on the underlying encryption. Thirdly, we show that $\mathsf{FO}^{\not\perp}$-KEMs do achieve MAL-BIND-K-CT and MAL-BIND-K,PK-CT security—again not relying on any requirements for the underlying encryption. An overview of these results can be found in Table 1.

We further show that these results can be applied to the code-based key-encapsulation

[2] We exclude the results that are based on a conjecture.

[3] The relevant change for our work is that round-4 Hqc uses implicit rejection, whereas round-3 Hqc uses explicit rejection.

[4] Also a few notions for Classic-McEliece and Ml-Kem remain without analysis.

[5] In fact, we also prove this for the corresponding LEAK notions.

Table 1: Overview of our results for the binding notions of implicitly-rejecting KEMs obtained via either of the two FO transforms. Positive results are marked with ✓ while negative ones are marked with ✗; an asterisk, i.e., ✓*/✗*, indicates that a result (either positive or negative) relies on some mild assumption. Results from prior works, i.e., Theorem 1, Theorem 2, Theorem 3, and Theorem 4, are written in parentheses.

General Notion	Notion	$\mathsf{FO}^{\not\perp}$	$\mathsf{FO}^{\not\perp}_m$
X-BIND-K-PK	MAL-BIND-K-PK	✗ Thm. 5	✗ (Thm. 4)
	LEAK-BIND-K-PK	✗* Thm. 5	✗ (Thm. 4)
	HON-BIND-K-PK	✓* Thm. 6	✗ (Thm. 4)
X-BIND-K,CT-PK	MAL-BIND-K,CT-PK	✗ Thm. 8	✗ Thm. 8
	LEAK-BIND-K,CT-PK	✗* Thm. 5	✓* Thm. 16
	HON-BIND-K,CT-PK	✓* (Thm. 1)	✓* Thm. 16
X-BIND-K-CT	MAL-BIND-K-CT	✓ Thm. 7	✗ (Thm. 4)
	LEAK-BIND-K-CT	✓ (Thm. 2)	✗ (Thm. 4)
	HON-BIND-K-CT	✓ (Thm. 2)	✗ (Thm. 4)
X-BIND-K,PK-CT	MAL-BIND-K,PK-CT	✓ Thm. 7	✓ Thm. 18
	LEAK-BIND-K,PK-CT	✓ (Thm. 2)	✓ (Thm. 4)
	HON-BIND-K,PK-CT	✓ (Thm. 2)	✓ (Thm. 4)

mechanism Bike [ABB+22], as it applies the $\mathsf{FO}^{\not\perp}$ transform. For the code-based scheme Hqc [AAB+22] some of the binding notions require a dedicated analysis due to Hqc using a modified variant of the $\mathsf{FO}^{\not\perp}$ transform. Resulting from this, we observe that Hqc achieves only 2 out of 12 notions, in particular, none of the LEAK or MAL notions. Next to this, we also describe a slight modification to Hqc, resulting in a scheme we call Hqc*, which provides more binding security than Hqc. More precisely, we observe that the general $\mathsf{FO}^{\not\perp}$ results can be applied not only for Bike but also for Hqc*, i.e., it follows from the existing and our new results that both schemes fulfill 8 out of 12 notions: all HON notions plus X-BIND-K-CT and X-BIND-K,PK-CT for X $\in$ {MAL, LEAK}. An overview of these results can be found in Table 2.

Regarding $\mathsf{FO}^{\not\perp}_m$-KEMs, we have the following results. We show that $\mathsf{FO}^{\not\perp}_m$-KEMs do achieve MAL-BIND-K,PK-CT security while they do not achieve MAL-BIND-K,CT-PK security. However, we have a positive result with respect to LEAK-BIND-K,CT-PK (and hence HON-BIND-K,CT-PK) based on an additional requirement of the underlying encryption scheme which is achieved by the encryption schemes underlying Bike and Hqc. Furthermore, we prove that HON-BIND-K-PK and HON-BIND-K,CT-PK (and their corresponding LEAK-variants LEAK-BIND-K-PK and LEAK-BIND-K,CT-PK) are not equivalent which is in contrast to $\mathsf{FO}^{\not\perp}$-KEMs, where we show this to be the case. These results are included in Table 1.

Given that neither $\mathsf{FO}^{\not\perp}$ nor $\mathsf{FO}^{\not\perp}_m$ achieve all binding notions, we describe three transforms FO_H, FO_L, and FO_M, that achieve all binding notions wrt different attack models: (1) FO_H is simply the $\mathsf{FO}^{\not\perp}$ transform which, using some mild assumption on the underlying PKE scheme, achieves all HON notions; (2) FO_L is a variant of the $\mathsf{FO}^{\not\perp}$ transform close to the one used by Ml-Kem and can be seen to achieve all LEAK notions; (3) FO_M is a slight modification of FO_L (the only change is that the public key is also hashed in the rejecting case) which we show to achieve all MAL notions—by the hierarchy also all LEAK and HON notions. Table 3 provides an overview of these transforms.

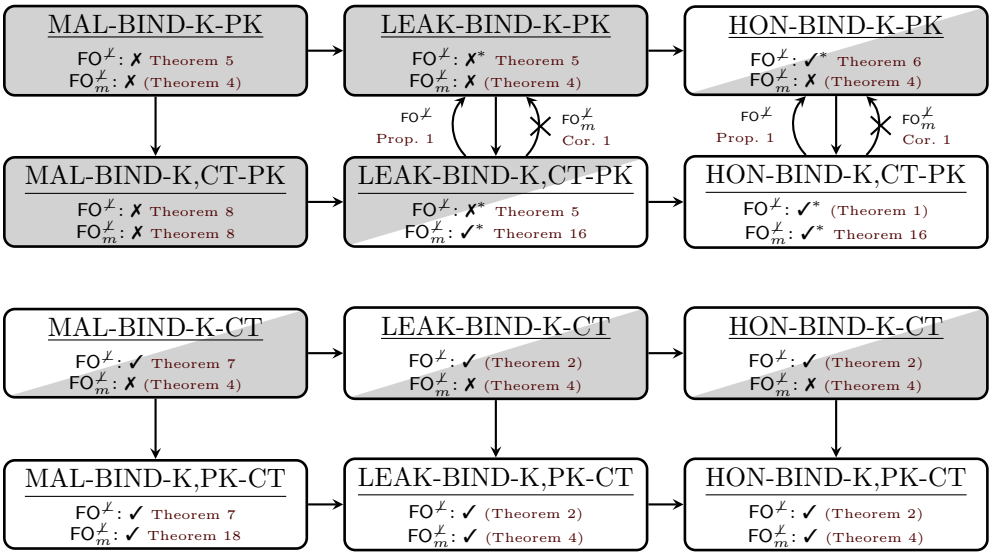

Figure 1: Overview of the 12 binding notions for key-encapsulation mechanisms with their implications. For each notion, we state whether it is achieved (✓) or not achieved (✗) by $\mathsf{FO}^{\not\perp}$ or $\mathsf{FO}^{\not\perp}_m$. If the result relies on an extra condition, it is indicated by an asterisk (✓*/✗*). Results from prior works, i.e., Theorem 1, Theorem 2, Theorem 3, and Theorem 4, are written in parentheses; further, boxes with positive results are left white, while ones with negative results are marked in gray. Boxes with mixed results have both colors; the top left color indicates the result for $\mathsf{FO}^{\not\perp}$ while the bottom right color indicates the result for $\mathsf{FO}^{\not\perp}_m$. Arrows indicate implications while crossed arrow denote separations.

1.2 Related Work

The binding properties generalize the concept of robust encryption, which can be traced back to [ABN10] and ensures that it is hard for an adversary to produce a ciphertext that is valid under more than one key. Robust encryption plays a key role for achieving anonymity of public-key encryption. Anonymity was introduced in [BBDP01] and captures the idea that an adversary, who obtains a ciphertext, cannot distinguish which key was used to generate said ciphertext.

Grubbs et al. [GMP22] and Xagawa [Xag22] studied the anonymity and robustness of post-quantum secure KEMs: the former analyzed robustness for three NIST finalists (Classic-McEliece [ABC+20], Saber [DKR+20], and Kyber [SAB+20]) and one alternate candidate (FrodoKem [NAB+20]); the latter analyzed the anonymity for all round-3 KEMs.

Cremers et al. [CDM24] provide a framework covering various binding notions for KEMs which subsumes existing robustness properties. Similar properties were introduced in [BCD+24] and [AHK+23] which are covered by the general framework from [CDM24]. Besides the introduction of the framework, Cremers et al. [CDM24] also cover a (partial) analysis of Kyber, FrodoKem, Classic-McEliece, and Ml-Kem. For the latter, Ml-Kem, Schmieg [Sch24] analyzed some of the strongest binding notions. Fiedler and Günther [FG25] introduce a fourth attack model for the binding properties analyzing the Signal's PQXDH handshake.

Table 2: Binding security of the round-4 KEMs BIKE, HQC, HQC*, and CLASSIC-MCELIECE, as well as ML-KEM. Note that HQC* is a slightly modified version of HQC, which we introduce in this paper.

Notion	BIKE	HQC / HQC*	CL-MCELIECE	ML-KEM
MAL-BIND-K-PK	✗	✗ / ✗	✗	✗
LEAK-BIND-K-PK	✗	✗ / ✗	✗	✓
HON-BIND-K-PK	✓	✓ / ✓	✗	✓
MAL-BIND-K,CT-PK	✗	✗ / ✗	✗	✗
LEAK-BIND-K,CT-PK	✗	✗ / ✗	✗	✓
HON-BIND-K,CT-PK	✓	✓ / ✓	✗	✓
MAL-BIND-K-CT	✓	✗ / ✓	✓	✗
LEAK-BIND-K-CT	✓	✗ / ✓	✓	✓
HON-BIND-K-CT	✓	✗ / ✓	✓	✓
MAL-BIND-K,PK-CT	✓	✗ / ✓	✓	✓
LEAK-BIND-K,PK-CT	✓	✗ / ✓	✓	✓
HON-BIND-K,PK-CT	✓	✗ / ✓	✓	✓

Table 3: Overview on variants of the FO transform and what notions they achieve. A ✓ means that *all* notion are achieved in the respective attack model, while a ✗ indicates that *at least one* notion in the respective attack model is not achieved. An asterisk denotes that the result relies on a mild assumption. Here, h_{pk} denotes the hash of the public key.

	Computation of the Shared Key		Attack Model		
Transform	accept	reject	HON	LEAK	MAL
FO_H	$\mathsf{H}(m, c)$	$\mathsf{H}(\sigma, c)$	✓*	✗	✗
FO_L	$\mathsf{H}(m, h_{pk})$	$\mathsf{H}(\sigma, c)$	✓	✓	✗
FO_M	$\mathsf{H}(m, h_{pk})$	$\mathsf{H}(\sigma, h_{pk}, c)$	✓	✓	✓

2 Background

This section provides the necessary background for this work. Section 2.1 describes the notation used throughout. The definitions for public-key encryption (PKE) schemes and key-encapsulation mechanisms (KEM) are given in Section 2.2 while the Fujisaki-Okamoto transform to turn a PKE into a KEM is described in Section 2.3. The various binding notions covered in this work appear in Section 2.4.

2.1 Notation

For a set $\mathcal{S}$, $s \leftarrow\!\!\$\, \mathcal{S}$ denotes that a uniform random element from $\mathcal{S}$ is chosen and assigned to s. For a distribution D, we write $x \leftarrow D$ to indicate that x is drawn according to D. For a randomized algorithm ALG, we write $y \leftarrow\!\!\$\, \text{ALG}(x)$ to denote that y is the output of ALG on input x. Often we will be explicit about the random coins of ALG in which case we write $y \leftarrow \text{ALG}(x; r)$ (this means that $y \leftarrow\!\!\$\, \text{ALG}(x)$ picks uniformly random coins r and computes $y \leftarrow \text{ALG}(x; r)$). Throughout this work, we implicitly consider the security parameter λ and will hence not write it as an explicit input for algorithms. Whenever we write adversary $\mathcal{A}$, it means an algorithm that runs in polynomial-time in the security

parameter and security wrt to any notion means that an adversary succeeds at most with negligible probability in the security parameter. For some security notions (those of the form MAL-BIND-P-Q), the security game is a two-stage game for which we assume that the adversaries share an implicit state.

2.2 Public-Key Encryption and Key-Encapsulation Mechanisms

Below we define public-key encryption and key-encapsulation mechanisms.

Definition 1. A public-key encryption scheme PKE consists of three algorithms KEYGEN, ENC, and DEC, where

KEYGEN() is the key generation algorithm that outputs a key pair consisting of a public key pk and a secret key sk.

ENC(pk, m) is the encryption algorithm which takes as input a public key pk and a message m, and outputs a ciphertext c.

DEC(sk, c) is the decryption algorithm which takes as input a secret key sk and a ciphertext c, and outputs a message m (or a special failure symbol $\perp$).

Definition 2. A key-encapsulation mechanism (KEM) KEM is a triple of three algorithms (KEYGEN, ENCAPS, DECAPS) where

KEYGEN() is the key generation algorithm that outputs a key pair consisting of a public key pk and a secret key sk.

ENCAPS(pk) is the encapsulation algorithm which takes as input a public key pk, and outputs a ciphertext c and an encapsulated key k.

DECAPS(sk, c) is the decapsulation algorithm which takes as input a secret key sk and a ciphertext c, and outputs a key k (or a special failure symbol $\perp$).

We write $\mathcal{M}$ for the message space of a public-key encryption scheme, which—for FO-KEMs that we consider in this work—coincides with the randomness space of a key-encapsulation mechanism. As mentioned above, we will typically be explicit about the random coins. This entails that we write $c \leftarrow \text{ENC}(pk, m; r)$ to indicate that c is deterministically computed using r as random coins. Likewise, we write $(c, k) \leftarrow \text{ENCAPS}(pk; r)$ to indicate that (c, k) is deterministically computed using random coins r.

During this work, we will implicitly assume that the PKE schemes achieve IND-CPA security (and hence the corresponding FO-transformed scheme is IND-CCA secure) to guarantee that a random ciphertext will be invalid with overwhelming probability—this property is also exploited in the IND-CCA security proof of the FO transform. We assume that the public-key space is superpolynomial and that the PKE schemes and KEMs are correct with overwhelming probability, i.e., for honestly generated ciphertexts, DEC and DECAPS return the correct message m and key k with overwhelming probability, respectively. We further assume, for both PKE schemes and KEMs, that the public key pk can be derived from the corresponding secret key sk—for honestly generated key-pairs. This is in line with the requirement by NIST. We note that the theoretical analysis then allows the attacker to maliciously choose the actual public key different from the public key contained in the secret key; in practice, however, (part of) the public key is often computed from a shorter seed and the secret key will only contain this seed but not the whole public key.

2.3 The Fujisaki-Okamoto Transform

The Fujisaki-Okamoto transform [FO99] turns a weakly secure public-key encryption scheme into a strongly secure public-key encryption scheme. A variant which transforms a PKE scheme into a KEM was first given by Dent [Den03]. In this work, we consider the modular versions of it given in [HHK17]. Here, the transform is composed of two transforms: $\mathsf{T}_{\not\perp}$ and U. The transform $\mathsf{T}_{\not\perp}$ derandomizes a public-key encryption scheme by deriving the random coins used for encryption from the message. To validate ciphertexts, the decryption additionally checks if re-encrypting a decrypted ciphertext results in the same ciphertext. The transform U transforms a derandomized public-key encryption scheme into a KEM. Here, a random message is chosen and encrypted using the derandomized PKE scheme resulting in a ciphertext. The shared key is obtained by hashing the random message (and this ciphertext). For an FO-KEM, the randomness is a message for the underlying PKE scheme, hence we will typically write $\textsc{Kem.Encaps}(pk; m)$ instead of $\textsc{Kem.Encaps}(pk; r)$. The transform U comes in four different variants $\mathsf{U}^{\not\perp}$,$\mathsf{U}^{\not\perp}_m$, $\mathsf{U}^{\perp}$, and $\mathsf{U}^{\perp}_m$, where the first two are using implicit rejection while the latter two use explicit rejection. Composing the $\mathsf{T}_{\not\perp}$ transform with the four variants of the U transform yields the four FO variants $\mathsf{FO}^{\not\perp}$, $\mathrm{FO}^{\not\perp}_m$, $\mathrm{FO}^{\perp}$, and $\mathsf{FO}^{\perp}_m$.

The $\mathsf{FO}^{\not\perp}$ transform is currently the most commonly used variant, being deployed by all of the round-4 NIST KEMs—in particular by Bike and Hqc[6], which we will consider in Section 4. This variant (decomposed into the two underlying transforms $\mathsf{T}_{\not\perp}$ and $\mathsf{U}^{\not\perp}$) is shown in Figure 2 along with the $\mathsf{FO}^{\not\perp}_m$ transform (decomposed into the two underlying transforms $\mathsf{T}_{\not\perp}$ and $\mathsf{U}^{\not\perp}_m$).

However, during the course of the NIST standardization process, also other FO-variants were used in the submitted KEMs. For instance, LEDACrypt [BBC+19] and Nts-Kem [ACP+19] both relied on $\mathsf{FO}^{\not\perp}_m$, while Hqc in round-3 used $\mathsf{FO}^{\perp}$. In view of the trend towards implicit rejection, we mainly focus on $\mathsf{FO}^{\not\perp}_m$ and $\mathsf{FO}^{\not\perp}$ in this paper.

The security of the FO transform relies on H_{T} and H_{U} to be random oracles. Our results rely on the same assumption and throughout this work, we model H_{T} and H_{U} as random oracles without explicitly stating it each time.

2.4 Binding Properties of Key-Encapsulation Mechanisms

Cremers et al. [CDM24] developed a framework for binding properties of KEMs. Binding notions in this framework are of the form X-BIND-P-Q. Here, P and Q describe which elements (P) are binding which other elements (Q) and $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$ determines the adversary model, more precisely, how the keys are selected.[7] The generic binding security game X-BIND-P-Q is shown in Figure 3.[8] For implicitly-rejecting KEMs, there are a total of twelve binding notions: X-BIND-K-PK and X-BIND-K,CT-PK as well as X-BIND-K-CT and X-BIND-K,PK-CT for $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$ that can be derived from the generic games. For convenience, we also provide the explicit games in Figure 8 and Figure 9 in Appendix A. The relations between the notions are described in Figure 1. Below we define security wrt the various binding notions.

Definition 3. Let Kem be a key-encapsulation mechanism. Further consider $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$, $\mathrm{P} \in \{\{\mathrm{K}\}, \{\mathrm{K,CT}\}\}$, and the game X-BIND-P-PK defined in Figure 3. We say that Kem achieves X-BIND-P-PK if for any adversary $\mathcal{A}$, its probability in winning X-BIND-P-PK is negligible.

[6] Note, however, that Hqc deploys a modified $\mathsf{FO}^{\not\perp}$ version.

[7] A fourth case was introduced by Fiedler and Günther [FG25]. In this case, dubbed LEAK^{+r}, the adversary is not provided with the key-pairs but the randomness used to generate those; this puts it between LEAK and MAL.

[8] SCFR-CCA and SROB-CCA from [GMP22] correspond to HON-BIND-K,CT-PK and HON-BIND-CT-PK, respectively, in the framework from [CDM24].

$\underline{\text{PKE}^{\not g}.\text{KeyGen}()}$
$(pk, sk) \leftarrow \text{KeyGen}()$
return (pk, sk)

$\underline{\text{PKE}^{\not g}.\text{Enc}(pk, m)}$
$c \leftarrow \text{Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m))$
return c

$\underline{\text{PKE}^{\not g}.\text{Dec}(sk, c)}$
$m \leftarrow \text{Dec}(sk, c)$
$\overline{c} \leftarrow \text{Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m))$
if $\overline{c} \neq c$
  return $\bot$
return m

$\underline{\text{KeyGen}^{\not\perp}()}$
$(pk, sk) \leftarrow \text{KeyGen}()$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$
$sk^{\not\perp} \leftarrow (sk, \sigma)$
return $(pk, sk^{\not\perp})$

$\underline{\text{Encaps}(pk)}$
$m \leftarrow\!\!\$\ \mathcal{M}$
$c \leftarrow \text{PKE}^{\not g}.\text{Enc}(pk, m)$
$k \leftarrow \mathsf{H}_{\mathsf{U}}(m, c)$ // $\mathsf{U}^{\not\perp}$
$k \leftarrow \mathsf{H}_{\mathsf{U}}(m)$ // $\mathsf{U}^{\not\perp}_{m}$
return (c, k)

$\underline{\text{Decaps}(sk^{\not\perp}, c)}$
$(sk, \sigma) \leftarrow sk^{\not\perp}$
$m \leftarrow \text{PKE}^{\not g}.\text{Dec}(sk, c)$
if $m \neq \bot$
  return $\mathsf{H}_{\mathsf{U}}(m, c)$ // $\mathsf{U}^{\not\perp}$
  return $\mathsf{H}_{\mathsf{U}}(m)$ // $\mathsf{U}^{\not\perp}_{m}$
return $\mathsf{H}_{\mathsf{U}}(\sigma, c)$

Figure 2: Top: The derandomized PKE scheme $\text{PKE}^{\not g} = \mathsf{T}_{\not g}[\text{PKE}, \mathsf{H}_{\mathsf{T}}]$ constructed from a PKE scheme PKE and a random oracle H_{T}. **Bottom:** The implicitly-rejecting KEMs $\mathsf{U}^{\not\perp}[\text{PKE}^{\not g}, \mathsf{H}_{\mathsf{U}}]$ and $\mathsf{U}^{\not\perp}_{m}[\text{PKE}^{\not g}, \mathsf{H}_{\mathsf{U}}]$.

Remark 1. In our formalization of the MAL notions, the adversary outputs either the secret key *or* the public key of a key pair, but never both at the same time. Since we assume that the secret key contains the public key, the re-encryption step during decapsulation is still possible. More precisely, for the different variants in the MAL games (see Figure 3), the outputs are (I) sk, $\overline{sk}$; (II) pk, $\overline{sk}$; and (III) pk, $\overline{pk}$. This is equivalent to the formalization by Cremers et al. [CDM24] where the adversary has to output two key pairs (pk, sk) and $(\overline{pk}, \overline{sk})$ and the public key is an explicit input to the decapsulation algorithm.[9]

For some of our results, we need certain requirements for the underlying PKE scheme. More precisely, we require a form of collision-freeness which we establish in three different variants: SCFR-CPA, SCFR-CCA, and SCFR-LEAK as defined below. Note that SCFR-CPA and SCFR-CCA were introduced in [GMP22], while SCFR-LEAK is a new notion introduced in this work.[10]

Definition 4. Consider the games SCFR-CPA, SCFR-CCA, and SCFR-LEAK as defined in Figure 4. We call a public-key encryption scheme PKE SCFR-CPA, SCFR-CCA, and SCFR-LEAK secure, if for any adversary its probability of winning the corresponding game is negligible.

In the following we recall some existing results. The first one by Grubbs et al. [GMP22] establishes HON-BIND-K,CT-PK security for KEMs via the $\mathsf{FO}^{\not\perp}$ transform assuming SCFR-CPA security of the underlying (derandomized) PKE scheme. The other two are general results about implicitly-rejecting FO transforms given by Cremers et al. [CDM24].

Theorem 1 (Adapted from [GMP22, Theorem 6])**.** *Let* PKE *be a public-key encryption scheme that has negligible decryption errors and* H_{T} *and* H_{U} *be random oracles. If* $\text{PKE}^{\not g} = \mathsf{T}_{\not g}[\text{PKE}, \mathsf{H}_{\mathsf{T}}]$ *is* SCFR-CPA *secure, then* $\text{KEM}^{\not\perp} = \mathsf{FO}^{\not\perp}[\text{PKE}, \mathsf{H}_{\mathsf{T}}, \mathsf{H}_{\mathsf{U}}]$ *is secure wrt the notion* HON-BIND-K,CT-PK*.*

[9] Note that an earlier version of [CDM24] did not consider the public key as an explicit input to the decapsulation algorithm which led to MAL attacks as shown by Schmieg [Sch24].

[10] Note that the PKE schemes underlying Bike and Hqc fulfill the new notion SCFR-LEAK (see Proposition 3).

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 $k = \perp \vee \overline{k} = \perp$:
  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 $k = \perp \vee \overline{k} = \perp$:
  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 $v_{\mathrm{p}} \wedge v_{\mathrm{q}}$

Figure 3: Generic security games for the notions X-BIND-P-Q and MAL-BIND-P-Q for $\mathrm{P} = \{\mathrm{K}, \mathrm{CT}, (\mathrm{K}, \mathrm{CT}), (\mathrm{K}, \mathrm{PK})\}$, $\mathrm{Q} = \{\mathrm{PK}, \mathrm{K}, \mathrm{CT}\}$, and $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}\}$. Note that for $\mathrm{p} \in \mathrm{P}$ (and $\mathrm{q} \in \mathrm{Q}$), we denote the corresponding instances by x_{p}, $\overline{x}_{\mathrm{p}}$ (and x_{q}, $\overline{x}_{\mathrm{q}}$, respectively). For example, if $\mathrm{p} = \mathrm{CT}$, 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 $\overline{sk}$. We refer to Remark 1 for more details. The decapsulation oracles Dec and $\overline{\mathsf{Dec}}$ in case of $\mathrm{X} = \mathrm{HON}$ are omitted and work in the obvious way (without any restrictions) using sk and $\overline{sk}$, respectively.

Theorem 2 (Adapted from [CDM24, Theorem D.1])**.** *Let* Pke *be a public-key encryption scheme and* H_T *and* H_U *be random oracles. Then* $\textsc{Kem}^{\not\perp} = \mathsf{FO}^{\not\perp}[\textsc{Pke}, \mathsf{H}_\mathsf{T}, \mathsf{H}_\mathsf{U}]$ *is* LEAK-BIND-K-CT *secure.*

Theorem 3 ([CDM24, Theorem 4.11])**.** *An implicitly-rejecting key-encapsulation mechanism* Kem *cannot be* HON-BIND-CT-PK *or* HON-BIND-CT-K *secure.*

Theorem 4 (Adapted from [CDM24, Section B.4])**.** *Let* Pke *be a public-key encryption scheme and* H_T *and* H_U *be random oracles. Then the key-encapsulation mechanism* $\textsc{Kem}^{\not\perp}_m = \mathsf{FO}^{\not\perp}_m[\textsc{Pke}, \mathsf{H}_\mathsf{T}, \mathsf{H}_\mathsf{U}]$ *is* LEAK-BIND-K,PK-CT *secure, but insecure with respect to* HON-BIND-K-CT *and* HON-BIND-K-PK.

Remark 2. Note that [CDM24, Section B.4] also contains a positive result for the LEAK-BIND-K,CT-PK security of $\mathsf{FO}^{\not\perp}_m$-KEMs. However, a part of the proof is based on a conjecture and left open for future work—this is why we do not include the result in Theorem 4. Further details on this are provided in Section B.1.

Game SCFR-LEAK

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$c \leftarrow \mathcal{A}(pk, \overline{pk}, sk, \overline{sk})$
$m \leftarrow \textsc{Dec}(sk, c)$
$\overline{m} \leftarrow \textsc{Dec}(\overline{sk}, c)$
return $m = \overline{m} \neq \bot$

Game SCFR-CCA

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$c \leftarrow \mathcal{A}^{\mathsf{Dec}, \overline{\mathsf{Dec}}}(pk, \overline{pk})$
$m \leftarrow \textsc{Dec}(sk, c)$
$\overline{m} \leftarrow \textsc{Dec}(\overline{sk}, c)$
return $m = \overline{m} \neq \bot$

Oracle $\mathsf{Dec}(c)$

$k \leftarrow \textsc{Decaps}(sk, c)$
return k

Oracle $\overline{\mathsf{Dec}}(c)$

$k \leftarrow \textsc{Decaps}(\overline{sk}, c)$
return k

Figure 4: Security games SCFR-LEAK and SCFR-CCA for PKEs. By removing the decapsulation oracles in game SCFR-CCA, we obtain the game SCFR-CPA as defined in [GMP22] and required for Theorem 1.

3 General Analysis of Implicitly-Rejecting FO

We give general results about the binding security of implicitly-rejecting FO-KEMs, i.e., $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. These results complete the binding analysis of the two implicitly-rejecting FO variants. Section 3.1 and Section 3.2 cover the notions of the form X-BIND-K,CT-PK and X-BIND-K-PK for $\mathsf{FO}^{\not\perp}$. Section 3.3 completes the picture by showing security with respect to the notions MAL-BIND-K-CT and MAL-BIND-K,PK-CT for $\mathsf{FO}^{\not\perp}$. Section 3.4 presents several MAL-BIND-P-Q attacks for both $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. The results for $\mathsf{FO}^{\perp}_m$ are deferred to Appendix B. More precisely, Appendix B.1 covers the notions of the form X-BIND-K,CT-PK and Appendix B.2 the ones of the form X-BIND-K,PK-CT.

3.1 LEAK-BIND-K,CT-PK Attack for $\mathrm{FO}^{\not\perp}$

Below we define a property of public-key encryption schemes needed for our results. It bears similarities with the rigidity property [BP18], where it is checked whether decrypting a ciphertext and then re-encrypting the resulting message will return the same ciphertext. However, in the definition below, we consider the negated version (non-rigidity) and add a further restriction: we consider only ciphertexts obtained from an honest encryption with a public key that is *different* from the one used for the actual non-rigidity check. This is why we call the property *restricted non-rigidity*.

Rigidity was initially introduced for the $\mathsf{T}_{\not\$}$ transform underlying the FO transform. This transformation derandomizes a PKE scheme by deriving the random coins as the hash of the message. Looking ahead, we will need this property not only for the $\mathsf{T}_{\not\$}$ transform but also for a modified version used by HQC, which we will refer to as $\mathsf{T}_{\$}$.

Definition 5. Let PKE be a public-key encryption scheme. We say that PKE fulfills *restricted non-rigidity* (or PKE is *restricted non-rigid*) if for two honestly generated key pairs $(pk, sk), (\overline{pk}, \overline{sk})$, a randomly chosen message m, and $c \leftarrow\!\!\$\ \textsc{Pke.Enc}(pk, m)$, we have

$$\textsc{Pke.Enc}(\overline{pk}, \textsc{Pke.Dec}(\overline{sk}, c)) \neq c\,. \tag{1}$$

with overwhelming probability. Note that the probability is taken over the random coins used to generate the key pair, to pick the message, and for encryption.

Remark 3. For $\mathsf{T}_{\not\$}$ the re-encrypt transform as defined in Figure 2, the property from Eq. (1) corresponds to the re-encrypt check failing. An honestly generated ciphertext using (pk, sk) will be invalid under a different key pair $(\overline{pk}, \overline{sk})$ with overwhelming probability.

The theorem below shows that applying $\mathrm{FO}^{\not\perp}$ to a PKE scheme that is restricted non-rigid results in a KEM that is *not* LEAK-BIND-K,CT-PK. The attack is conceptually very simple: the adversary takes the implicit-rejection-value σ from one of the key pairs and

honestly encapsulates it under the other key pair, yielding a ciphertext c. By correctness c gets decapsulated to $\mathsf{H}_{\mathsf{U}}(\sigma, c)$ (under the key pair which was used to honestly generate c). By non-rigidity of the underlying PKE scheme, c gets rejected with overwhelming probability under the other key pair, which, however, will result in the same output $\mathsf{H}_{\mathsf{U}}(\sigma, c)$.

Theorem 5. *Let* Pke *be a public-key encryption scheme and* $\textsc{Kem}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}^{\not\perp}$ *to* Pke*. If* Pke *is restricted non-rigid, then* $\textsc{Kem}^{\not\perp}$ *is neither* X-BIND-K,CT-PK *nor* X-BIND-K-PK *secure for* $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}\}$.

Proof. We construct a LEAK-BIND-K,CT-PK adversary $\mathcal{A}$ against $\textsc{Kem}^{\not\perp}$ as follows. As input, $\mathcal{A}$ obtains two honestly generated key pairs $(pk, sk^{\not\perp} = (sk, \sigma)), (\overline{pk}, \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})) \leftarrow\!\!\$\ \textsc{Kem}^{\not\perp}.\textsc{KeyGen}()$ and has to output c such that

$$\textsc{Kem}^{\not\perp}.\textsc{Decaps}(sk^{\not\perp}, c) = k = \overline{k} = \textsc{Kem}^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, c)\,.$$

Note that we have $pk \neq \overline{pk}$ with overwhelming probability. As a first step, $\mathcal{A}$ sets $m = \overline{\sigma}$ where $\overline{\sigma}$ is the implicit rejection value from $\overline{sk}^{\not\perp}$ and computes an honest encapsulation of pk, using not a random message but m instead: $(c, k) \leftarrow \textsc{Kem}^{\not\perp}.\textsc{Encaps}(pk; m)$. Then, $\mathcal{A}$ outputs c.

Below we argue that $\mathcal{A}$ wins the game LEAK-BIND-K,CT-PK with overwhelming probability. Note first that $k = \mathsf{H}_{\mathsf{U}}(m, \cdot)$ which follows by definition of Kem.Encaps. Note further that $\textsc{Kem}^{\not\perp}.\textsc{Decaps}(sk^{\not\perp}, c) = k$ holds with overwhelming probability by correctness of $\textsc{Kem}^{\not\perp}$. For the remaining part of the proof, we assume that the decryption of the base-PKE scheme does not return $\perp$.[11] Next, we show that c decapsulates to k under $\overline{sk}^{\not\perp}$ as well: As the underlying PKE scheme is restricted non-rigid, we have $\mathsf{T}_{\not\$}[\textsc{Pke}, \mathsf{H}_{\mathsf{T}}].\textsc{Dec}(\overline{sk}, c) = \perp$ (as the re-encryption check will fail with overwhelming probability) and hence the ciphertext c will be rejected under the secret key $\overline{sk}$, i.e., $\textsc{Kem}^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, c) = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \cdot)$. This leads to

$$\textsc{Kem}^{\not\perp}.\textsc{Decaps}(sk^{\not\perp}, c) = k = \mathsf{H}_{\mathsf{U}}(m, \cdot) = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \cdot) = \textsc{Kem}^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, c)$$

which shows that $\mathcal{A}$ wins the game LEAK-BIND-K,CT-PK. Insecurity with respect to LEAK-BIND-K-PK, MAL-BIND-K,CT-PK, and MAL-BIND-K-PK follows from the hierarchy of the binding notions. □

3.2 HON-BIND-K-PK Security for $\mathsf{FO}^{\not\perp}$

We establish a positive result for HON-BIND-K-PK security of $\mathsf{FO}^{\not\perp}$. More precisely, we show that—for $\mathsf{FO}^{\not\perp}$—HON-BIND-K-PK is in fact equivalent to the generally weaker notion of HON-BIND-K,CT-PK. This effectively implies that Theorem 1 also yields HON-BIND-K-PK security.

The following proposition establishes equivalence between HON-BIND-K-PK and HON-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}$-KEMs. The difference between the two notions is that the former puts no restriction the the ciphertexts while the latter requires them to be the same. Since $\mathsf{FO}^{\not\perp}$ hashes the ciphertext to derive the shared key, we can conclude that the ciphertexts have to be different (unless there was a hash collision), effectively removing this freedom of the (generally stronger) notion HON-BIND-K-PK.

Proposition 1. *Let* Pke *be a PKE scheme and* $\textsc{Kem}^{\not\perp}$ *the KEM obtained from applying* $\mathrm{FO}^{\not\perp}$ *to* Pke*. If* $\textsc{Kem}^{\not\perp}$ *is* X-BIND-K,CT-PK *secure, then* $\textsc{Kem}^{\not\perp}$ *is also* X-BIND-K-PK *secure for* $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}\}$.

[11] Note that the decryption for Kyber.Pke never returns $\perp$, while for Bike.Pke and Hqc.Pke this is possible due to decoding failures.

Proof. We give the proof in terms of X = HON and argue at the relevant parts what changes for X = LEAK. We show that if $\textsc{Kem}^{\not\perp}$ is not HON-BIND-K-PK secure, then it is also not HON-BIND-K,CT-PK secure. More precisely, given a HON-BIND-K-PK adversary $\mathcal{A}$, we construct a HON-BIND-K,CT-PK adversary $\mathcal{B}$ as follows: $\mathcal{B}$ obtains $(pk, \overline{pk})$, each of which is part of an honestly generated key pair $(pk, sk^{\not\perp} = (sk, \sigma))$, $(\overline{pk}, \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma}))$. $\mathcal{B}$ then calls the HON-BIND-K-PK adversary $\mathcal{A}$ against $\textsc{Kem}^{\not\perp}$ on input $(pk, \overline{pk})$ and simulates the decapsulation oracles for $\mathcal{A}$ using its own ones. When $\mathcal{A}$ outputs $(c, \overline{c})$, $\mathcal{B}$ in turn outputs (c, c). For X = LEAK, $\mathcal{B}$ also gets the secret keys which $\mathcal{B}$ sends to $\mathcal{A}$; in this variant, there are no oracles that need to be simulated.

We assume that $\mathcal{A}$ wins the game HON-BIND-K-PK and show that then the adversary $\mathcal{B}$ as constructed above wins the game HON-BIND-K,CT-PK, except with negligible probability. As the keys are honestly generated, we can assume that $pk \neq \overline{pk}$ and $sk \neq \overline{sk}$, with overwhelming probability. Likewise, we can assume that $\sigma \neq \overline{\sigma}$ with overwhelming probability. $\mathcal{A}$ winning the game HON-BIND-K-PK implies that

$$k = \textsc{Kem}^{\not\perp}.\textsc{Decaps}(sk^{\not\perp}, c) = \textsc{Kem}^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, \overline{c}) = \overline{k}\,,$$

for c and $\overline{c}$ (not necessarily equal) output by $\mathcal{A}$. Let $m = \textsc{Pke}^{\not g}.\textsc{Dec}(sk, c)$ and $\overline{m} = \textsc{Pke}^{\not g}.\textsc{Dec}(\overline{sk}, \overline{c})$. There are three cases to consider, depending on whether the ciphertexts are valid or not. In all cases, the shared keys are computed as $k \leftarrow \mathsf{H}_{\mathsf{U}}(\cdot, c)$ and $\overline{k} \leftarrow \mathsf{H}_{\mathsf{U}}(\cdot, \overline{c})$; the mere difference is whether the first input is the decrypted message (m or $\overline{m}$) or the implicit rejection value (σ or $\overline{\sigma}$). But regardless of the first input—even in the LEAK case, when $\mathcal{A}$ knows both σ and $\overline{\sigma}$—we can deduce $c = \overline{c}$ as otherwise $\mathcal{A}$ has found a collision for H_{U}. This yields that also $\mathcal{B}$ is successful in winning HON-BIND-K,CT-PK, which concludes the proof. □

Note that the equivalence is limited to KEMs obtained via the $\mathsf{FO}^{\not\perp}$ transform. In general, the two notions are not equivalent. Cremers et al. [CDM24] showed a separation between the notions for $\mathsf{FO}^{\perp}_{m}$. Looking ahead, our results for $\mathsf{FO}^{\not\perp}_{m}$ (cf. Corollary 1) also yield the separation between the notions HON-BIND-K-PK and HON-BIND-K,CT-PK as well as between the notions LEAK-BIND-K-PK and LEAK-BIND-K,CT-PK.

As we show that the notions HON-BIND-K-PK and HON-BIND-K,CT-PK are equivalent for $\mathrm{FO}^{\not\perp}$-KEMs in the above theorem, we can leverage Theorem 1 to show security wrt the notion HON-BIND-K-PK: if the underlying PKE scheme is SCFR-CPA secure, we get HON-BIND-K,CT-PK and thus HON-BIND-K-PK security for the $\mathrm{FO}^{\not\perp}$ KEM. This is formalized in the following theorem.

Theorem 6. *Let* Pke *be a public-key encryption scheme that has negligible decryption errors and* $\textsc{Kem}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}^{\not\perp}$ *to* Pke*. If* $\textsc{Pke}^{\not g} = \mathsf{T}_{\not g}[\textsc{Pke}, \mathsf{H}_{\mathsf{T}_{\not g}}]$ *is* SCFR-CPA *secure, then* $\textsc{Kem}^{\not\perp}$ *is* HON-BIND-K,CT-PK *secure.*

3.3 MAL-BIND-K-CT Security for $\mathsf{FO}^{\not\perp}$

We show that key-encapsulation mechanisms obtained via the $\mathsf{FO}^{\not\perp}$ transform are secure wrt the notions MAL-BIND-K-CT and MAL-BIND-K,PK-CT. The argument simply follows from the fact that the notions require the ciphertexts to be distinct while $\mathsf{FO}^{\not\perp}$ derives the shared key by hashing also the ciphertext; any successful adversary therefore has to find a hash collision.

Theorem 7. *Let* Pke *be a public-key encryption scheme and* $\textsc{Kem}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}^{\not\perp}$ *to* Pke*. Then* $\textsc{Kem}^{\not\perp}$ *is* MAL-BIND-K-CT *and* MAL-BIND-K,PK-CT *secure.*

Proof. Let $\mathcal{A}$ be an adversary against MAL-BIND-K-CT. Let c and $\overline{c}$ denote the ciphertexts related to $\mathcal{A}$'s output—which can either be direct outputs of $\mathcal{A}$ (in case of the Decaps

variant) or obtained from an honest encapsulation using the public key and randomness output by $\mathcal{A}$ (in case of the ENCAPS variant). In order to win, these ciphertexts must be distinct, i.e., $c \neq \overline{c}$, while the encapsulated keys must agree, i.e., $k = \overline{k}$. However, by definition of $\mathsf{FO}^{\not\perp}$ this yields $\mathsf{H}_{\mathsf{U}}(\cdot, c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\cdot, \overline{c})$ which means that $\mathcal{A}$ would have found a collision for H_{U}. Security wrt to MAL-BIND-K,PK-CT follows via the hierarchy of the notions. □

While the results from Theorem 7 were claimed in an earlier version of [CDM24][12], the current version shows only LEAK-BIND-K,PK-CT. Our proof agrees with the one in the earlier version; we give it for sake of completeness but do not claim any novelty here.

3.4 MAL-BIND-K,CT-PK Attacks

We establish several negative results regarding MAL-BIND-K,CT-PK security of $\mathsf{FO}^{\not\perp}$ as well as $\mathsf{FO}^{\not\perp}_m$. Given the hierarchy of the notions, this also establishes negative results with respect to MAL-BIND-K-PK and MAL-BIND-CT-PK for both $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. Schmieg [Sch24] shows that ML-KEM is not MAL-BIND-K-PK secure which leaves the possibility that it might achieve MAL-BIND-K,CT-PK security (which is the generally weaker notion of the two). We show, that their attack also applies to MAL-BIND-K,CT-PK and for any KEM constructed via either $\mathsf{FO}^{\not\perp}$ or $\mathsf{FO}^{\not\perp}_m$. In the attack, the adversary generates malicious secret keys that agree in their implicit rejection value. Then any invalid ciphertext (trivially obtained by randomly picking a ciphertext) results in the same shared key for both secret keys and thus breaks MAL-BIND-K,CT-PK security. This is formally stated in the theorem below.

Theorem 8. *Let* PKE *be a public-key encryption scheme and* KEM *be the key-encapsulation mechanism resulting from applying the implicitly-rejecting* FO *transform (regardless of which of the two variants) to* PKE*. Then,* KEM *is neither* MAL-BIND-K,CT-PK *nor* MAL-BIND-K-PK *secure.*

Proof. We give an MAL-BIND-K,CT-PK adversary $\mathcal{A}$ for variant (I) DECAPS-DECAPS. Adversary $\mathcal{A}$ first generates two key pairs for the underlying public-key encryption scheme, i.e., $(pk, sk), (\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$. Furthermore, $\mathcal{A}$ samples $\sigma \leftarrow\!\!\$\ \mathcal{M}$ and sets $sk^{\not\perp} \leftarrow (sk, \sigma)$ as well as $\overline{sk}^{\not\perp} \leftarrow (\overline{sk}, \sigma)$. Finally, $\mathcal{A}$ generates an arbitrary ciphertext c, sets $\overline{c} \leftarrow c$ and outputs $(sk^{\not\perp}, \overline{sk}^{\not\perp}, c, \overline{c})$. The ciphertexts will be invalid (wrt both secret keys) with overwhelming probability[13], hence $\textsc{Kem.Decaps}(sk^{\not\perp}, c) = \mathsf{H}_{\mathsf{U}}(\sigma, c)$ and $\textsc{Kem.Decaps}(\overline{sk}^{\not\perp}, c) = \mathsf{H}_{\mathsf{U}}(\sigma, c)$. This yields that $\mathcal{A}$ wins the game MAL-BIND-K,CT-PK. Insecurity wrt MAL-BIND-K-PK easily follows. □

For ML-KEM and MAL-BIND-K-PK, the theorem above is exactly what is shown in [Sch24].

4 Application to BIKE and HQC

In this section, we analyze the binding properties of the code-based round-4 KEMs BIKE and HQC. We first provide a description of the schemes and the necessary background in Section 4.1. Subsequently, in Section 4.2, we analyze the binding properties of HQC and in Section 4.3, we do the same for BIKE and HQC*, where the latter is a modified version of HQC that we introduce in Section 4.1. While the modification is quite small, it results in improved binding security compared to the original HQC. Lastly, in Section 4.4, we show

[12] The earlier version we are referring to is Version 1.0.6 which is available at https://eprint.iacr.org/archive/2023/1933/20240403:091024.

[13] Recall our assumption that the KEM is IND-CCA secure (cf. Section 2.2).

how our results can be transferred to the third round-4 KEM Classic-McEliece and to Ml-Kem to cover the last gaps left in the analysis of their binding properties.

4.1 Description of Bike and Hqc

Bike [ABB+22] is a round-4 KEM based on Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) codes. More precisely, Bike is obtained by instantiating the Niederreiter scheme with QC-MDPC codes. Hence, the security can be traced back to the quasi-cyclic variants of certain distinguishing problems from coding theory. It results from applying the $\mathsf{FO}^{\not\perp}$ transform to the public-key encryption scheme Bike.Pke. Both Bike.Pke and Bike.Kem are displayed in Figure 5.

Hqc [AAB+22] is a round-4 KEM based on quasi-cyclic codes and the hardness of the syndrome decoding problem. The public-key encryption scheme Hqc.Pke and the key-encapsulation mechanism Hqc.Kem, that results from applying *a variant of* the $\mathsf{FO}^{\not\perp}$ transform to Hqc.Pke, are shown in Figure 6. This variant introduces a random salt *salt* to protect against multi-ciphertext attacks. The salt is appended to the ciphertext, however, it is *not* part of the key derivation. Another change involving the salt is that the randomness for encryption is derived as $\mathsf{H}_{\mathsf{T}_{\$}}(m, pk, salt)$ instead of $\mathsf{H}_{\mathsf{T}_{\$}}(m)$. We describe the transformation used by Hqc in Figure 7. This transformation relies on variants of $\mathsf{T}_{\$}$ and $\mathsf{U}^{\not\perp}$, which we will refer to as $\mathsf{T}_{\$}$ and $\mathsf{U}^{\not\perp}_{\$}$, respectively, where the subscript indicates that they are randomized.

Note that this deviation has significant impact, as several results for the $\mathsf{FO}^{\not\perp}$ transform no longer hold due to this change. This is why we will give a separate analysis for Hqc in Section 4.2, covering a few positive but more negative results. We also consider a slightly different version of Hqc, called Hqc*, which achieves more binding properties. It differs from Hqc in the fact that the salt is included in the final computation of the shared key (see Figure 6)—note that this is not simply the result of applying $\mathsf{FO}^{\not\perp}$ to the base PKE scheme underlying Hqc due to the salt.[14] Nevertheless, this small modification allows us to apply the general results for the $\mathsf{FO}^{\not\perp}$ transform, i.e., it suffices to check if the necessary prerequisites are fulfilled. The same is the case for Bike, thus we analyze Bike and Hqc* together in Section 4.3.

Lastly, note that both Bike and Hqc sample certain key components using a distribution that is not completely uniform, but close to it. For Bike this affects the secret key components h_0 and h_1 and analogously for Hqc the secret key components $\mathbf{x}$ and $\mathbf{y}$. However, this does not have a relevant impact on security: for two games G and G' that differ only in the way the Bike secret key is sampled—in G using a uniform distribution, in G' using the slightly biased one—the advantage of the adversary improves only by a factor τ. For the proposed parameter sets of Bike [ABB+22, Section C.4] and Hqc [AAB+24, Section 5.3], it holds that $\tau \leq 1.00353$. This is described in more detail in [ABB+22, Section C.4] and can be traced back to techniques by Sendrier [Sen11, Sen21]. For sake of simplicity, we make the assumption that $h_0, h_1, \mathbf{x}$, and $\mathbf{y}$ are uniformly sampled for the following section and omit the factor τ from the bounds.[15]

4.2 Binding Security of Hqc

In this section, we analyze the binding security of Hqc. Since it is an implicitly-rejecting KEM, there are 12 notions to be considered, which are depicted in Figure 1. As described in the previous section, Hqc does not apply the standard $\mathsf{FO}^{\not\perp}$ transform. More precisely, for Hqc a ciphertext contains a salt value, which (together with the message and public

[14] IND-CCA security for the variant of the FO transform used by Hqc* follows from [HHK17, Theorem 3.2 and Theorem 3.4]—in fact, this variant is closer to the original FO transform, as it hashes the entire ciphertext which the variant used in Hqc does not.

[15] We use this assumption in Proposition 3.

BIKE.PKE.KEYGEN()
$(h_0, h_1) \leftarrow\!\!\$\ \mathcal{H}_w$
$h \leftarrow h_0 h_1^{-1}$
$pk \leftarrow h$
$sk \leftarrow (h_0, h_1)$
return (pk, sk)

BIKE.PKE.ENC(pk, m)
$(e_0, e_1) \leftarrow\!\!\$\ \mathcal{E}_t$
$c_0 \leftarrow e_0 + e_1 h$
$c_1 \leftarrow m \oplus \mathsf{H}_L(e_0, e_1)$
$c \leftarrow (c_0, c_1)$
return c

BIKE.PKE.DEC(sk, c)
$\overline{e} \leftarrow$ DECODER$(c_0 h_0, h_0, h_1)$
if $\overline{e} = \perp$
  $m \leftarrow \perp$
else
  $m \leftarrow c_1 \oplus \mathsf{H}_L(\overline{e})$
return m

BIKE.KEYGEN()
$(h_0, h_1) \leftarrow\!\!\$\ \mathcal{H}_w$
$h \leftarrow h_0 h_1^{-1}$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$
$pk \leftarrow h$
$sk \leftarrow (h_0, h_1, \sigma)$
return (pk, sk)

BIKE.ENCAPS(pk)
$m \leftarrow\!\!\$\ \mathcal{M}$
$(e_0, e_1) \leftarrow \mathsf{H}_H(m)$
$c_0 \leftarrow e_0 + e_1 h$
$c_1 \leftarrow m \oplus \mathsf{H}_L(e_0, e_1)$
$c \leftarrow (c_0, c_1)$
$k \leftarrow \mathsf{H}_U(m, c)$
return (c, k)

BIKE.DECAPS(sk, c)
$\overline{e} \leftarrow$ DECODER$(c_0 h_0, h_0, h_1)$
$\overline{m} \leftarrow c_1 \oplus \mathsf{H}_L(\overline{e})$
if $\overline{e} = \mathsf{H}_H(\overline{m})$
  $k \leftarrow \mathsf{H}_U(\overline{m}, c)$
else
  $k \leftarrow \mathsf{H}_U(\sigma, c)$
return k

Figure 5: PKE scheme BIKE.PKE and key-encapsulation mechanism BIKE. Here, $\mathcal{H}_w = \{(h_0, h_1) \in \mathcal{R}^2 \mid |h_0| = |h_1| = w/2\}$ and $\mathcal{E}_t = \{(e_0, e_1) \in \mathcal{R}^2 \mid |e_0| + |e_1| = t\}$ for $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$ for the BIKE parameters w, r and t. Note that H_H, H_L, and H_U are hash functions.

key) is used to derive the randomness for the re-encryption approach. While the general attacks we give for $\mathsf{FO}^{\not\perp}$-KEMs are still applicable[16], the general proofs from Section 2 and Section 3 cannot be transferred this easily. It turns out that HQC only achieves 2 out of the 12 binding notions: HON-BIND-K-PK and HON-BIND-K,CT-PK. Note that the notions X-BIND-K-CT and X-BIND-K,PK-CT (for X $\in$ {LEAK, MAL}) are achieved by *any* $\mathsf{FO}^{\not\perp}$-KEM, thus the modification of the $\mathsf{FO}^{\not\perp}$ transform made by HQC negatively affects the results for the binding properties.

4.2.1 Attacking MAL/LEAK Binding Notions for HQC.

Firstly, all attacks against X-BIND-K-PK and X-BIND-K,CT-PK for X $\in$ {LEAK, MAL} still apply to HQC. This is formulated in the following theorem which is proven in Appendix C.1.

Theorem 9. *The key-encapsulation mechanism* HQC *as shown in Figure 6 is insecure with respect to the following binding notions:* LEAK-BIND-K,CT-PK*,* LEAK-BIND-K-PK*,* MAL-BIND-K,CT-PK*, and* MAL-BIND-K-PK*.*

The following theorem (proven in Appendix C.2) proves HQC to be insecure with respect to the notion LEAK-BIND-K,PK-CT, which is contrary to the results for $\mathsf{FO}^{\not\perp}$. From this, a number of other LEAK/MAL attacks follow. The overall attack idea follows the strategy of our attack against LEAK-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}$-KEMs (cf. Theorem 5): We construct an honest ciphertext—using the implicit rejection value as the random message—and create a second, invalid ciphertext by changing the salt value of the ciphertext. Since the salt is not used to derive the key k, the valid and invalid ciphertexts result in the same key.

[16] These results cover the general notions X-BIND-K-PK and X-BIND-K,CT-PK for X $\in$ {LEAK, MAL}. The attacks described for the $\mathsf{FO}^{\not\perp}$ transform still apply by choosing the salts (contained in the ciphertexts that are output by the adversary) to be equal.

HQC.PKE.KEYGEN()

$\mathbf{h} \leftarrow\!\!\$\ \mathcal{R}$
$\mathbf{G} \leftarrow\!\!\$\ \mathbb{F}_2^{k\times n}$
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_w \times \mathcal{R}_w$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk \leftarrow (\mathbf{h}, \mathbf{s})$
$sk \leftarrow (\mathbf{x}, \mathbf{y})$
return (pk, sk)

HQC.PKE.ENC(pk, m)

$(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \leftarrow\!\!\$\ \mathcal{R}_{w_e} \times \mathcal{R}_{w_r} \times \mathcal{R}_{w_r}$
$\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$
$\mathbf{v} \leftarrow \text{TRUNC}(m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}, \ell)$
$c \leftarrow (\mathbf{u}, \mathbf{v})$
return c

HQC.PKE.DEC(sk, c)

$m \leftarrow \text{DECODER}(\mathbf{v} - \mathbf{uy})$
return m

HQC.KEYGEN()

$\mathbf{h} \leftarrow\!\!\$\ \mathcal{R}$
$\mathbf{G} \leftarrow\!\!\$\ \mathbb{F}_2^{k\times n}$
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_w \times \mathcal{R}_w$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$
$pk \leftarrow (\mathbf{h}, \mathbf{s})$
$sk \leftarrow (\mathbf{x}, \mathbf{y}, \sigma)$
return (pk, sk)

HQC.ENCAPS(pk)

$m \leftarrow\!\!\$\ \mathbb{F}_2^k$
$salt \leftarrow\!\!\$\ \mathbb{F}_2^{128}$
$(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \leftarrow \mathsf{H}_{\mathsf{T}_{\$}}(m, pk, salt)$
$\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$
$\mathbf{v} \leftarrow \text{TRUNC}(m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}, \ell)$
$k \leftarrow \mathsf{H}_{\mathsf{U}}(m, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$
$c \leftarrow (\mathbf{u}, \mathbf{v}, salt)$
return (c, k)

HQC.DECAPS(sk, c)

$\overline{m} \leftarrow \text{DECODER}(\mathbf{v} - \mathbf{uy})$
$(\overline{\mathbf{e}}, \overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2) \leftarrow \mathsf{H}_{\mathsf{T}_{\$}}(\overline{m}, pk, salt)$
$\overline{\mathbf{u}} \leftarrow \overline{\mathbf{r}}_1 + \mathbf{h}\overline{\mathbf{r}}_2$
$\overline{\mathbf{v}} \leftarrow \text{TRUNC}(\overline{m}\mathbf{G} + \mathbf{s}\overline{\mathbf{r}}_2 + \overline{\mathbf{e}}, \ell)$
if $(\overline{\mathbf{u}}, \overline{\mathbf{v}}) = (\mathbf{u}, \mathbf{v})$
  $k \leftarrow \mathsf{H}_{\mathsf{U}}(\overline{m}, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$
else
  $k \leftarrow \mathsf{H}_{\mathsf{U}}(\sigma, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$
return k

Figure 6: PKE scheme HQC.PKE and key-encapsulation mechanism HQC. Here, $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$ and $\mathcal{R}_x = \{v \in \mathcal{R} \text{ with hamming weight } x\}$ for $x \in \{w, w_e, w_r\}$; the latter are HQC parameters. Note that, H_{U} is a hash function. The modified version HQC* includes the highlighted parts, i.e., the *entire* ciphertext is hashed to compute the key k.

Theorem 10. *The key-encapsulation mechanism* HQC *as shown in Figure 6 is neither* X-BIND-K,PK-CT *nor* X-BIND-K-CT *secure for* $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}\}$.

4.2.2 Proving/Attacking HON Binding Notions for HQC.

The theorem below, which is proven in Appendix C.3, shows that HQC achieves both HON-BIND-K-PK and HON-BIND-K,CT-PK.

Theorem 11. *The key-encapsulation mechanism* HQC *as displayed in Figure 6 fulfills the notions* HON-BIND-K-PK *and* HON-BIND-K,CT-PK.

The following theorem shows that HQC does not achieve HON-BIND-K-CT and HON-BIND-K,PK-CT. The proof can be found in Appendix C.4. The attack exploits the fact that for two invalid ciphertexts that differ only in the salt, the same shared key will be computed if the same key pair is used.

Theorem 12. *The KEM* HQC *as displayed in Figure 6 is not* HON-BIND-K-CT *and* HON-BIND-K,PK-CT *secure.*

4.3 Binding Security of BIKE and HQC*

The fact that BIKE uses $\mathsf{FO}^{\not\perp}$ already provides several results for different binding notions. While HQC* does not use FO, its variant is close enough that the same results apply. However, for LEAK-BIND-K,CT-PK, LEAK-BIND-K-PK, HON-BIND-K,CT-PK, and HON-BIND-K,CT-PK our results rely on additional assumptions. To show an attack against the former two notions, we rely on the restricted non-rigidity property while

$\underline{\text{PKE}^{\$}.\text{KeyGen}()}$
$(pk, sk) \leftarrow \text{KeyGen}()$
return (pk, sk)

$\underline{\text{PKE}^{\$}.\text{Enc}(pk, m)}$
$salt \leftarrow\!\!\$\ \mathcal{S}$
$r \leftarrow \mathsf{H}_{\mathsf{T}}(m, pk, salt)$
$c_* \leftarrow \text{Enc}(pk, m; r)$
return $(c_*, salt)$

$\underline{\text{PKE}^{\$}.\text{Dec}(sk, (c_*, salt))}$
$m \leftarrow \text{Dec}(sk, c_*)$
$r \leftarrow \mathsf{H}_{\mathsf{T}}(m, pk, salt)$
$\overline{c}_* \leftarrow \text{Enc}(pk, m; r)$
if $\overline{c}_* \neq c_*$
  return $\perp$
return m

$\underline{\text{KeyGen}^{\not\perp}()}$
$(pk, sk) \leftarrow \text{KeyGen}()$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$
$sk^{\not\perp} \leftarrow (sk, \sigma)$
return $(pk, sk^{\not\perp})$

$\underline{\text{Encaps}(pk)}$
$m \leftarrow\!\!\$\ \mathcal{M}$
$(c_*, salt) \leftarrow \text{PKE}^{\$}.\text{Enc}(pk, m)$
$k \leftarrow \mathsf{H}_{\mathsf{U}}(m, c_*)$
return $((c_*, salt), k)$

$\underline{\text{Decaps}(sk^{\not\perp}, (c_*, salt))}$
$(sk, \sigma) \leftarrow sk^{\not\perp}$
$m \leftarrow \text{PKE}^{\$}.\text{Dec}(sk, (c_*, salt))$
if $m \neq \perp$
  return $\mathsf{H}_{\mathsf{U}}(m, c_*)$
return $\mathsf{H}_{\mathsf{U}}(\sigma, c_*)$

Figure 7: Transformations used by HQC. **Top:** The PKE scheme $\text{PKE}^{\$} = \mathsf{T}_{\$}[\text{PKE}]$. **Bottom:** The KEM $\mathsf{U}^{\not\perp}_{\$}[\mathsf{T}_{\$}[\text{PKE}]]$.

security with respect to the latter two requires the underlying $\mathsf{T}_{\not\$}$-transformed PKE scheme to achieve SCFR-CPA. The two propositions below establish these properties for BIKE and HQC*. Note that the $\mathsf{T}_{\$}$-transformed PKEs underlying HQC and HQC* agree (we will use HQC.PKE as notation for both), i.e., the below results apply for both schemes.

The following proposition is proven in Appendix C.5.

Proposition 2. *Let* PKE *be a public-key encryption scheme with the following property: for public keys $pk, \overline{pk}$ stemming from honest key generations, two (not necessarily different) messages $m, \overline{m}$, and two random coins $r, \overline{r}$, the ciphertexts $c = \text{PKE.Enc}(pk, m; r)$ and $\overline{c} = \text{PKE.Enc}(\overline{pk}, \overline{m}; \overline{r})$ differ with overwhelming probability. Then the following statements hold:*

1. *The PKE scheme $\mathsf{X}[\text{PKE}, \mathsf{H}_{\mathsf{T}}]$ is* SCFR-LEAK *secure for $\mathsf{X} \in \{\mathsf{T}_{\not\$}, \mathsf{T}_{\$}\}$.*
2. *The PKE scheme* PKE *is restricted non-rigid.*

The proof for the proposition below can be found in Appendix C.6.

Proposition 3. *Consider the public-key encryption schemes* BIKE.PKE *and* HQC.PKE *as shown in Figure 5 and Figure 6, respectively. The following statements hold:*

1. *The PKE scheme $\mathsf{T}_{\not\$}[\text{BIKE.PKE}, \mathsf{H}_{\mathsf{T}}]$ is* SCFR-LEAK *secure.*
2. BIKE.PKE *fulfills restricted non-rigidity.*
3. *The PKE scheme $\mathsf{T}_{\$}[\text{HQC.PKE}, \mathsf{H}_{\mathsf{T}}]$ is* SCFR-LEAK *secure.*
4. HQC.PKE *fulfills restricted non-rigidity.*

Having established Proposition 2 and Proposition 3, we get the following theorem regarding the binding properties of BIKE and HQC*—the results are exactly those that are presented for $\mathsf{FO}^{\not\perp}$ in Table 1. Note that HQC* deviates from the standard $\mathsf{FO}^{\not\perp}$ transform only in the way the randomness is derived. However, one can easily check that this change is irrelevant for our $\mathsf{FO}^{\not\perp}$ results. Due to this and the fact that all requirements are fulfilled, we can apply our $\mathsf{FO}^{\not\perp}$ results to HQC*.

Theorem 13. *The key-encapsulation mechanisms* Bike *and* Hqc* *as shown in Figure 5 and Figure 6 are* HON-BIND-K,CT-PK *and* HON-BIND-K-PK *as well as* X-BIND-K-CT *and* X-BIND-K,PK-CT *secure for* $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$. *They are insecure wrt the notions* LEAK-BIND-K,CT-PK, LEAK-BIND-K-PK, MAL-BIND-K,CT-PK, *and* MAL-BIND-K-PK.

4.4 Binding Security of Round-4 KEMs and ML-KEM

In the previous sections, we completed the analysis of the binding properties of the key-encapsulation mechanisms Bike, Hqc, and Hqc*. Furthermore, our results cover the last gaps left in the analysis of Classic-McEliece and Ml-Kem: Theorem 7 proves Classic-McEliece to be MAL-BIND-K-CT secure. Further, Ml-Kem is not MAL-BIND-K,CT-PK secure by Theorem 8 (the attack relies exclusively on invalid ciphertexts for which Ml-Kem behaves like $\mathsf{FO}^{\not\perp}$) while it is MAL-BIND-K,PK-CT secure by Theorem 18 with the following changes to the proof. Case 1 (both ciphertexts are invalid) works exactly the same way. Case 2 (one ciphertext is invalid) is different in two aspects: (1) for the valid ciphertext, the shared key is computed as the hash of *both* message and public key; (2) different hash functions are used to compute k and $\overline{k}$. Nevertheless, one can see that an adversary needs to find a collision to be successful. Case 3 (both ciphertexts are valid) works essentially the same: the difference is that the shared keys are computed as the hash of message and public key (compared to just the message), however, this does not matter for the conclusion that the messages are equal.

The completed results regarding the binding properties of the round-4 KEMs (and Ml-Kem) can be found in Table 2.

5 Achieving Binding Security

In this section, we present ways to achieve binding security for implicitly-rejecting KEMs. In Section 5.1, we describe transforms for the different attack models HON, LEAK, and MAL, while the corresponding results are given in Section 5.2.

5.1 Description of the Transforms FO_H, FO_L, and FO_M

In the following, we describe how different levels of binding security can be achieved in general for implicitly-rejecting KEMs—an overview is given in Table 3. For this, we take into account the general results presented in Section 3 as well as conclusions that can be drawn from the scheme-specific analysis in Section 4. We want to highlight the fact that if we talk about transforms achieving *all* HON/LEAK/MAL binding notions in this section, we mean all notions that are generally possible for implicitly-rejecting KEMs—note that there are further ones, that can never be achieved for such schemes (namely X-BIND-CT-PK and X-BIND-CT-K).

All transforms are variants of the FO transform. More precisely, they are the composition of the T transform and a variant of the U transform; in our description, we focus on the latter. The full transforms are dubbed FO_H, FO_L, and FO_M, the subscript indicating which security level each transform achieves.

FO_H **(accept:** $k = \mathsf{H}_\mathsf{U}(m, c)$**; reject:** $k = \mathsf{H}_\mathsf{U}(\sigma, c)$**).**

We observe that all HON notions are fulfilled by KEMs using the $\mathsf{FO}^{\not\perp}$ transform (Table 1).[17] Henceforth we refer to this as the FO_H transform. In the LEAK setting, this transform fails to achieve the notions LEAK-BIND-K-PK and LEAK-BIND-K,CT-PK; a natural

[17] Note that some results depend on a mild assumption.

way to also attain these notions would be to additionally hash the public key. The results for ML-KEM, however, indicate that one can obtain all LEAK notions by hashing the public key *instead* of the ciphertext in the accepting case.

FO_L **(accept:** $k = \mathsf{H}_\mathsf{U}(m, h_{pk})$**; reject:** $k = \mathsf{H}_\mathsf{U}(\sigma, c)$**).**

We show that the variant of the FO transform used in ML-KEM—to be precise the variant of the U transform which derives the shared key from the message and hash of the public key—achieves all LEAK notions and we will refer to this one as the FO_L transform. This change prevents our LEAK attack (cf. Theorem 5) by using different inputs in the accepting and rejecting case. The FO_L transform, however, fails to achieve some of the MAL notions.

FO_M **(accept:** $k = \mathsf{H}_\mathsf{U}(m, h_{pk})$**; reject:** $k = \mathsf{H}_\mathsf{U}(\sigma, h_{pk}, c)$**).**

We finally describe a transform (FO_M) that achieves all MAL notions and thus all binding notions in general. A simple approach would be to ensure that everything (message, ciphertext, and public key) is hashed to derive the shared key k—in the rejecting case, the implicit rejection value is used rather than the message. Surprisingly, we show that this is not necessary: In the accepting case, it is sufficient to hash *only* the message and the public key. More precisely, the MAL transform can be considered as a modified variant of the $\mathsf{FO}^{\not\perp}_m$ transform, where the public key is added to the final hash computation both in the accepting and rejecting case—equivalently, one can view it as the FO_L transform enhanced by hashing the public key also in the rejection case. This keeps the benefit that the ciphertext does not have to be hashed for valid ciphertexts, which would be much more costly.

5.1.1 Comparison of the Variants.

We want to highlight the fact that the FO_H transform will (in most cases) be more costly than the FO_L transform due to hashing the ciphertext rather than the hash of the public key. Thus, when deciding on which of the binding transforms to use, it makes sense to choose FO_L over FO_H, even if the use-case at hand requires only HON-security. In particular, as many schemes already deploy the $\mathsf{FO}^{\not\perp}$ transform (and thus FO_H), a switch to FO_L is neither very invasive nor produces overhead, but brings the benefits of more binding security. Regarding the cost of FO_M, the transform is as efficient as FO_L in the accepting case and only slightly more expensive in the rejecting case (additional hashing of $h_{pk} = \mathsf{H}(pk)$). In view of this, FO_M seems like a good choice as it achieves all binding notions at a small efficiency loss compared to $\mathsf{FO}^{\not\perp}$.

5.1.2 IND-CCA Security of the Transforms.

Note that the three transforms FO_H, FO_L, and FO_M preserve IND-CCA security. For FO_H, this follows directly from prior results for the $\mathsf{FO}^{\not\perp}$ transform. FO_L is the variant of the FO transform used in ML-KEM, which is proven IND-CCA secure—hence IND-CCA security is inherited from the NIST standard. Lastly, FO_M differs from FO_L only by additionally hashing the public key in the rejection case. Since we just add a public value to the key derivation, this does not affect security, which relies on the implicit rejection value being unknown to the adversary.

5.1.3 A Note on the Hash of the Public Key.

The hash of the public key is often added to the secret key to avoid hashing it for each key decapsulation at the cost of a slightly larger key. This approach, however, opens the possibility for malicious keys to not contain the actual hash value but some maliciously

chosen value. Such attacks are described by Schmieg [Sch24]. Thus, the secret key needs to be stored securely so that an adversary can neither read-out nor tamper with it. This way, the remaining attack vectors arise from malicious implementations, e.g., one where the hash of the public key is incorrectly computed. This threat can be mitigated by explicitly hashing the public key and comparing the result with the hash value that is part of the secret key. Note that this needs to be done only once, right after generating the key-pair. Together with our transform FO_M, one can then achieve full MAL-BIND security.

5.2 LEAK/MAL Security for $\mathsf{FO}_\mathsf{L}/\mathsf{FO}_\mathsf{M}$

The following theorem proves LEAK security of an FO_L-transformed KEM. More precisely, we show that the notions LEAK-BIND-K-PK and LEAK-BIND-K-CT are fulfilled, which then implies that all LEAK and HON notions hold (by the hierarchy visualized in Figure 1). The proof is given in Appendix C.7.

Theorem 14. *Let* PKE *be a PKE scheme and* KEM *the KEM obtained from applying* FO_L *to* PKE*. Then* KEM *is* X-BIND-K-PK*,* X-BIND-K-CT*,* X-BIND-K,CT-PK*, and* X-BIND-K,PK-CT *secure for* $\mathrm{X} \in \{\mathrm{LEAK}, \mathrm{HON}\}$*.*

The next theorem shows that the FO_M transform fulfills MAL-BIND-K-PK and MAL-BIND-K-CT security and hence—by the hierarchy—all binding notions. The proof can be found in Appendix C.8.

Theorem 15. *Let* PKE *be a PKE scheme and* KEM *the KEM obtained from applying* FO_M *to* PKE*. Then* KEM *is* X-BIND-K-PK*,* X-BIND-K-CT*,* X-BIND-K,CT-PK*, and* X-BIND-K,PK-CT *secure for* $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$*.*

References

[AAB+20] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. HQC. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[AAB+24] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, NIST, 2024. available at https://pqc-hqc.org/doc/hqc-specification_2024-02-23.pdf.

[ABB+22] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar-Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards

and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[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. doi:10.1007/978-3-642-11799-2_28.

[ACP+19] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. NTS-KEM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[ADG+22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 3291–3308. USENIX Association, August 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/albertini.

[ADM+24] Thomas Aulbach, Samed Düzlü, Michael Meyer, Patrick Struck, and Maximiliane Weishäupl. Hash your keys before signing - BUFF security of the additional NIST PQC signatures. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 301–335. Springer, Cham, June 2024. doi:10.1007/978-3-031-62746-0_13.

[AHK+23] Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. Post-quantum multi-recipient public key encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1108–1122. ACM Press, November 2023. doi:10.1145/3576915.3623185.

[Aye15] Andrew Ayer. Duplicate signature key selection attack in Let's Encrypt. https://www.agwa.name/blog/post/duplicate_signature_key_selection_attack_in_lets_encrypt, 2015.

[BBC+19] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAcrypt. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[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. doi:10.1007/3-540-45682-1_33.

[BCD+24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karolin Varner, and Bas Westerbaan. X-wing. *IACR Communications in Cryptology*, 1(1), 2024. doi:10.62056/a3qj89n4e.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy*, pages 353–367. IEEE Computer Society Press, April 2018. doi:10.1109/EuroSP.2018.00032.

[BH22] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 845–875. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07085-3_29.

[BH24] Mihir Bellare and Viet Tung Hoang. Succinctly-committing authenticated encryption. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part IV*, volume 14923 of *LNCS*, pages 305–339. Springer, Cham, August 2024. doi:10.1007/978-3-031-68385-5_10.

[BP18] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. URL: https://eprint.iacr.org/2018/526.

[CDF+21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE Computer Society Press, May 2021. doi:10.1109/SP40001.2021.00093.

[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. doi:10.1145/3658644.3670283.

[Den03] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *LNCS*, pages 133–151. Springer, Berlin, Heidelberg, December 2003. doi:10.1007/978-3-540-40974-8_12.

[DFF24] Samed Düzlü, Rune Fiedler, and Marc Fischlin. BUFFing FALCON without increasing the signature size. In Maria Eichlseder and Sébastien Gambs, editors, *SAC 2024, Part I*, volume 15516 of *LNCS*, pages 131–150. Springer, Cham, August 2024. doi:10.1007/978-3-031-82852-2_6.

[DFH+24] Jelle Don, Serge Fehr, Yu-Hsuan Huang, Jyun-Jie Liao, and Patrick Struck. Hide-and-seek and the non-resignability of the BUFF transform. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part III*, volume 15366 of *LNCS*, pages 347–370. Springer, Cham, December 2024. doi:10.1007/978-3-031-78020-2_12.

[DFHS24] Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck. On the (in)security of the BUFF transform. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 246–275. Springer, Cham, August 2024. doi:10.1007/978-3-031-68376-3_8.

[DGRW18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Cham, August 2018. doi:10.1007/978-3-319-96884-1_6.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[FG25] Rune Fiedler and Felix Günther. Security analysis of Signal's PQXDH handshake. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part II*, volume 15675 of *LNCS*, pages 137–169. Springer, Cham, May 2025. doi:10.1007/978-3-031-91823-0_5.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

[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. doi:10.1007/978-3-031-07082-2_15.

[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. doi:10.1007/978-3-319-70500-2_12.

[JCCS19] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2165–2180. ACM Press, November 2019. doi:10.1145/3319535.3339813.

[KSW24] Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Committing AE from sponges security analysis of the NIST LWC finalists. *IACR Trans. Symm. Cryptol.*, 2024(4):191–248, 2024. doi:10.46586/tosc.v2024.i4.191-248.

[LGR21] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 195–212. USENIX Association, August 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/len.

[MLGR23] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 379–407. Springer, Cham, April 2023. doi:10.1007/978-3-031-30634-1_13.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[NIS17] NIST. Post-quantum cryptography standardization process. https://csrc.nist.gov/projects/post-quantum-cryptography, 2017.

[NIS22] NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf, 2022.

[NIS23] NIST. Module-Lattice-based Key-Encapsulation Mechanism Standard. https://doi.org/10.6028/NIST.FIPS.203.ipd, 2023. Draft.

[NIS24] NIST. Accordion mode. https://csrc.nist.gov/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd, 2024.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Sch24] Sophie Schmieg. Unbindable kemmy schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK. Cryptology ePrint Archive, Report 2024/523, 2024. URL: https://eprint.iacr.org/2024/523.

[Sen11] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 51–67. Springer, Berlin, Heidelberg, November / December 2011. doi:10.1007/978-3-642-25405-5_4.

[Sen21] Nicolas Sendrier. Secure sampling of constant-weight words – application to BIKE. Cryptology ePrint Archive, Report 2021/1631, 2021. URL: https://eprint.iacr.org/2021/1631.

[SW24] Patrick Struck and Maximiliane Weishäupl. Constructing committing and leakage-resilient authenticated encryption. *IACR Trans. Symm. Cryptol.*, 2024(1):497–528, 2024. doi:10.46586/tosc.v2024.i1.497-528.

[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. doi:10.1007/978-3-031-07082-2_20.

A Additional Background

Figure 8 and Figure 9 display the various binding notions considered in this work.

Game HON-BIND-K,CT-PK

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(c, \overline{c}) \leftarrow \mathcal{A}^{\mathsf{Dec},\overline{\mathsf{Dec}}}(pk, \overline{pk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
return $k = \overline{k} \wedge pk \neq \overline{pk} \wedge c = \overline{c}$

Game LEAK-BIND-K,CT-PK

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(c, \overline{c}) \leftarrow \mathcal{A}(pk, sk, \overline{pk}, \overline{sk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
return $k = \overline{k} \wedge pk \neq \overline{pk} \wedge c = \overline{c}$

Game MAL-BIND-K,CT-PK

$g \leftarrow \mathcal{A}()$
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
return $k = \overline{k} \wedge pk \neq \overline{pk} \wedge c = \overline{c}$

Figure 8: Security games HON-BIND-K-PK, HON-BIND-K,CT-PK, LEAK-BIND-K-PK, LEAK-BIND-K,CT-PK, MAL-BIND-K-PK, and MAL-BIND-K,CT-PK. The variants involving the ciphertext include the highlighted statements (in which case we slightly abuse the syntax by letting $\mathcal{A}$ only output a single ciphertext c), the others do not.

Game HON-BIND-K,PK-CT

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow (pk, sk)$
$(c, \overline{c}) \leftarrow \mathcal{A}^{\mathsf{Dec}, \overline{\mathsf{Dec}}}(pk, \overline{pk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
return $k = \overline{k} \wedge c \neq \overline{c}$

Game LEAK-BIND-K,PK-CT

$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow (pk, sk)$
$(c, \overline{c}) \leftarrow \mathcal{A}(pk, sk, \overline{pk}, \overline{sk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
if $k = \bot \vee \overline{k} = \bot$:
  return 0
return $k = \overline{k} \wedge c \neq \overline{c}$

Game MAL-BIND-K,PK-CT

$g \leftarrow \mathcal{A}()$
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
return $k = \overline{k} \wedge c \neq \overline{c} \wedge pk = \overline{pk}$

Figure 9: Security games X-BIND-K-CT and X-BIND-K,PK-CT for $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$. The variants involving the public key include the highlighted statements, the other do not. In case of HON-BIND-K,PK-CT and LEAK-BIND-K,PK-CT, we will typically provide the adversary with only one public key or key pair, respectively, for sake of simplicity.

B Analysis of $\mathsf{FO}_m^{\not\perp}$

B.1 LEAK-BIND-K,CT-PK Security for $\mathsf{FO}_m^{\not\perp}$

In the following section, we show that $\mathsf{FO}_m^{\not\perp}$ achieves LEAK-BIND-K,CT-PK security if and only if the underlying derandomized PKE scheme achieves our new notion SCFR-LEAK. In particular, this completes a claim made in [CDM24]: In [CDM24, Appendix B.4] LEAK-BIND-K,CT-PK security of $\mathsf{FO}_m^{\not\perp}$ is reduced to LEAK-BIND-K,CT-PK security of the explicitly-rejecting variant $\mathsf{FO}_m^{\perp}$. The latter is handled in [CDM24, Appendix B.2], which traces LEAK-BIND-K,CT-PK security back to LEAK-BIND-CT-PK security of $\mathsf{FO}_m^{\perp}$. However, so far it is only conjectured that $\mathsf{FO}_m^{\perp}$ fulfills LEAK-BIND-CT-PK if the underlying PKE fulfills some robustness property. Thus, the LEAK-BIND-K,CT-PK proof for $\mathsf{FO}_m^{\not\perp}$ is not complete. We will fill this gap by providing Theorem 16—the proof shows that any output with at least one invalid ciphertext results in a hash collision, while the case of two valid ciphertexts contradicts the assumption on the underlying (derandomized) PKE scheme. Further, we make the robustness assumption on the PKE more precise (SCFR-LEAK) and show that it is not only a sufficient, but a necessary condition for achieving LEAK-BIND-K,CT-PK security (Theorem 17).

Theorem 16. *Let* $\textsc{Pke}$ *be a public-key encryption scheme,* $\textsc{Pke}^{\not g}$ *its derandomized variant, and* $\textsc{Kem}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}_m^{\not\perp}$*. If* $\textsc{Pke}^{\not g}$ *is*

SCFR-LEAK *(resp.* SCFR-CCA*) secure, then* $\textsc{Kem}_m^{\not\perp}$ *is* LEAK-BIND-K,CT-PK *(resp.* HON-BIND-K,CT-PK*) secure.*

Proof. Assume for a contradiction that there exists a successful adversary $\mathcal{A}$ against LEAK-BIND-K,CT-PK: given two honestly generated key pairs

$$(pk, sk^{\not\perp} = (sk, \sigma)), (\overline{pk}, \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})) \leftarrow \textsc{Kem}_m^{\not\perp}.\textsc{KeyGen}()\,,$$

$\mathcal{A}$ outputs a ciphertext c such that

$$k = \textsc{Kem}_m^{\not\perp}.\textsc{Decaps}(sk^{\not\perp}, c) = \textsc{Kem}_m^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, c) = \overline{k}\,.$$

For $m = \textsc{Pke}^{\$}.\textsc{Dec}(sk, c)$ and $\overline{m} = \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, c)$, we distinguish the following cases:

Case 1: $m = \perp \wedge \overline{m} = \perp$ (both ciphertexts are invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(\sigma, c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, c)$. Since the keys are honestly generated, we have $\sigma \neq \overline{\sigma}$ with overwhelming probability which entails that $\mathcal{A}$ has found a collision for H_{U}.

Case 2: $m \neq \perp \wedge \overline{m} = \perp$ (one ciphertext is invalid)[18]

In this case, we have $\mathsf{H}_{\mathsf{U}}(m) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, c)$. Clearly, this yields a collision as $m \neq (\overline{\sigma}, c)$.

Case 3: $m \neq \perp \wedge \overline{m} \neq \perp$ (both ciphertexts are valid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(m) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{m})$. Assuming that $\mathcal{A}$ does not find a collision, we can deduce $m = \overline{m}$, however, this yields that the ciphertext c also allows to win SCFR-LEAK against $\textsc{Pke}^{\$}$ as it validly decrypts to the same message under two different secret keys.

This shows LEAK-BIND-K,CT-PK security.

The very same argument applies for the notion HON-BIND-K,CT-PK, but note that the assumption SCFR-LEAK can be relaxed to SCFR-CCA. The reason is that game HON-BIND-K,CT-PK no longer grants the secret key but a decryption oracle to the adversary. Hence the reduction merely needs access to a decryption oracle to simulate the view of the adversary. This concludes the proof. □

Remark 4. The above theorem extends to the new LEAK^{+r} model, introduced in [FG25], when the underlying PKE scheme satisfies the natural extension of SCFR-LEAK to SCFR-LEAK^{+r}, which provides the adversary with the randomness used to generate the keys.[19] Note that every other binding result from this paper directly extends to LEAK^{+r}, as we either have security in the stronger MAL setting or attacks in the weaker LEAK setting.

The following theorem shows that SCFR-LEAK is not only a sufficient but also a necessary condition for achieving LEAK-BIND-K,CT-PK security.

Theorem 17. *Let* Pke *be a PKE scheme,* $\textsc{Pke}^{\$}$ *its derandomized variant, and* $\textsc{Kem}_m^{\not\perp}$ *the KEM obtained from applying* $\mathrm{FO}_m^{\not\perp}$*. If* $\textsc{Kem}_m^{\not\perp}$ *is* LEAK-BIND-K,CT-PK *secure, then* $\textsc{Pke}^{\$}$ *is* SCFR-LEAK *secure.*

Proof. Assume for a contradiction that $\textsc{Pke}^{\$}$ is not SCFR-LEAK secure, i.e., there is an adversary $\mathcal{A}$ that obtains honestly generated key pairs (pk, sk), $(\overline{pk}, \overline{sk})$ and outputs c s.t. $\textsc{Pke}^{\$}.\textsc{Dec}(sk, c) = m = \overline{m} = \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, c) \neq \perp$. As $m = \overline{m} \neq \perp$, we obtain

[18] Here we assume wlog that $\overline{c}$ is invalid.

[19] Note that the separation example given in [FG25] can be extended to separate SCFR-LEAK and SCFR-LEAK^{+r}.

that $k = \mathsf{H}_{\mathsf{U}}(m) = \mathsf{H}_{\mathsf{U}}(\overline{m}) = \overline{k}$. Then we can construct an adversary $\mathcal{B}$ against $\textsc{Kem}_m^{\not\perp}$ that simply outputs the ciphertext c that $\mathcal{A}$ outputs (after providing $\mathcal{A}$ with the same key-pair it got from the LEAK-BIND-K,CT-PK game minus the implicit rejection values σ and $\overline{\sigma}$). Adversary $\mathcal{B}$ is clearly also successful in winning LEAK-BIND-K,CT-PK which contradicts the assumption that $\textsc{Kem}_m^{\not\perp}$ is LEAK-BIND-K,CT-PK secure and thus finishes the proof. □

The corollary below shows that—unlike for $\mathsf{FO}^{\not\perp}$—the notions X-BIND-K-PK and X-BIND-K,CT-PK are not equivalent (for X = HON and X = LEAK). This follows—for KEMs constructed via the $\mathsf{FO}_m^{\not\perp}$ transform—from our positive result for X-BIND-K,CT-PK (Theorem 16) and the negative result for X-BIND-K,CT-PK (Theorem 4).

Corollary 1. *There is a PKE scheme* Pke*, such that the KEM* $\textsc{Kem}_m^{\not\perp}$ *resulting from applying* $\mathsf{FO}_m^{\not\perp}$ *to* Pke *is* X-BIND-K,CT-PK *secure but not* X-BIND-K-PK *secure, for* $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}\}$.

B.2 MAL-BIND-K,PK-CT Security for $\mathrm{FO}_m^{\not\perp}$

We show that $\mathsf{FO}_m^{\not\perp}$ achieves X-BIND-K,PK-CT for $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$ which, together with Theorem 2, shows that these notions are achieved for any implicitly-rejecting FO-KEM.

The theorem below shows that key-encapsulation mechanisms obtained via the $\mathsf{FO}_m^{\not\perp}$ transform achieve MAL-BIND-K,PK-CT security. By hierarchy, this implies security wrt LEAK-BIND-K,PK-CT and HON-BIND-K,PK-CT. The proof shows that for all possible attacks, the adversary needs to find a collision for the hash function.

Theorem 18. *Let* Pke *be a PKE scheme and* $\textsc{Kem}_m^{\not\perp}$ *the KEM obtained from applying* $\mathrm{FO}_m^{\not\perp}$ *to* Pke*. Then* $\textsc{Kem}_m^{\not\perp}$ *is* X-BIND-K,PK-CT *secure for* $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$.

Proof. Note that for MAL-BIND-K,PK-CT, there are three different variants to consider: (I) Decaps-Decaps, (II) Encaps-Decaps, and (III) Encaps-Encaps. We give the proof for the first variant and subsequently argue why it also covers the other two variants.

Adversary $\mathcal{A}$ needs to output two secret keys $sk^{\not\perp} = (sk, \sigma)$ and $\overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})$ (it must hold that the implicitly contained public keys are not distinct, i.e., $pk = \overline{pk}$) and two distinct ciphertexts $c \neq \overline{c}$ that result in the same shared key k when decrypting under the respective secret keys. We can distinguish between the following cases for the ciphertexts output by $\mathcal{A}$, where $m \leftarrow \textsc{Pke}^{\$}.\textsc{Dec}(sk, c)$ and $\overline{m} \leftarrow \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, \overline{c})$:

Case 1: $m = \bot \wedge \overline{m} = \bot$ (both ciphertexts are invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(\sigma, c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \overline{c})$. Even thought $\mathcal{A}$ has full control over σ and $\overline{\sigma}$, the fact that $c \neq \overline{c}$ holds, implies that $\mathcal{A}$ has to find a collision for H_{U};

Case 2: $m \neq \bot \wedge \overline{m} = \bot$ (one ciphertext is invalid)[20]

In this case, we obtain $\mathsf{H}_{\mathsf{U}}(m) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \overline{c})$. Clearly, this would also entail a collision for H_{U};

Case 3: $m \neq \bot \wedge \overline{m} \neq \bot$ (both ciphertexts are valid)

In this case, it holds that $\mathsf{H}_{\mathsf{U}}(m) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{m})$. We can deduce $m = \overline{m}$ as otherwise, we would again have a collision for $\mathsf{H}_{\mathsf{T}_{\$}}$. However, in this case, validity of both ciphertexts yields

$$c = \textsc{Enc}(pk, m; \mathsf{H}_{\mathsf{T}_{\$}}(m)) = \textsc{Enc}(\overline{pk}, \overline{m}; \mathsf{H}_{\mathsf{T}_{\$}}(\overline{m})) = \overline{c}\,,$$

where the second equality follows since both $pk = \overline{pk}$ and $m = \overline{m}$, hence yielding a contradicting to $c \neq \overline{c}$.

[20] Here we assume wlog that $\overline{c}$ is invalid.

In the following, we argue that the above three cases also cover the other variants, i.e., (II) Encaps-Decaps, and (III) Encaps-Encaps. In a nutshell, if $\mathcal{A}$ provides the randomness for Encaps instead of a ciphertext (and a public key instead of a secret key), correctness of the KEM yields that the resulting ciphertext will be valid, meaning that some of the above cases simply cannot occur.

For the Encaps-Decaps variant, $\mathcal{A}$ will output a message m from which the ciphertext $c \leftarrow \text{KEM}_m^{\not\perp}.\text{Encaps}(pk; m)$ is derived. By correctness of $\text{KEM}_m^{\not\perp}$, this yields that c will be a valid ciphertext, hence excluding the first case above. For the other two scenarios, the above argumentation can be applied.

For the Encaps-Encaps variant, $\mathcal{A}$ will output two messages m and $\overline{m}$ from which the ciphertext will be derived as $c \leftarrow \text{KEM}_m^{\not\perp}.\text{Encaps}(pk; m)$ and $\overline{c} \leftarrow \text{KEM}_m^{\not\perp}.\text{Encaps}(\overline{pk}; \overline{m})$, respectively. We thus get that both ciphertexts will be valid by correctness, which means we merely need the final case from the three above.

By the hierarchy of the notions, the above also shows LEAK-BIND-K,PK-CT security and HON-BIND-K,PK-CT security. This concludes the proof. □

C Postponed Proofs

C.1 Proof of Theorem 9

Proof. Insecurity with respect to LEAK-BIND-K,CT-PK follows easily from Theorem 5 using the fact that Hqc.Pke fulfills restricted non-rigidity. The latter is proven in Proposition 3, which covers Bike, Hqc, and the modified variant Hqc*. While Theorem 5 is formulated for $\mathsf{FO}^{\not\perp}$, it directly translates to the variant used by Hqc. Then Hqc is also insecure wrt the notions LEAK-BIND-K-PK, MAL-BIND-K,CT-PK, and MAL-BIND-K-PK by the established hierarchy. □

C.2 Proof of Theorem 10

Proof. We construct the following LEAK-BIND-K,PK-CT adversary $\mathcal{A}$ against Hqc. Its input is a key-pair $(pk, sk^{\not\perp} = (sk, \sigma))$, where $pk = (\mathbf{h}, \mathbf{s})$ and $sk = (\mathbf{x}, \mathbf{y})$, and it is supposed to output distinct ciphertexts c and $\overline{c}$ such that

$$\text{Hqc.Decaps}(sk^{\not\perp}, c) = k = \overline{k} = \text{Hqc.Decaps}(sk^{\not\perp}, \overline{c})\,.$$

Adversary $\mathcal{A}$ first picks $salt \leftarrow\!\!\$\ \mathcal{S}$, sets $m \leftarrow \sigma$, and computes the ciphertext $c = (\mathbf{u}, \mathbf{v}, salt) \leftarrow \text{Hqc.Encaps}(pk; m, salt)$. By construction, we have $\mathbf{u} = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$ for $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) = \mathsf{H}_{\mathsf{T}_{\mathsf{g}}}(m, pk, salt)$. Next, $\mathcal{A}$ picks $\overline{salt} \leftarrow\!\!\$\ \mathcal{S}\backslash\{salt\}$, and outputs the two ciphertexts $c \leftarrow (\mathbf{u}, \mathbf{v}, salt)$ and $\overline{c} \leftarrow (\mathbf{u}, \mathbf{v}, \overline{salt})$.

The two ciphertexts are distinct as we have $salt \neq \overline{salt}$. It remains to argue that they result in the same key. By correctness, with overwhelming probability, we have

$$k = \text{Hqc.Decaps}(sk^{\not\perp}, (\mathbf{u}, \mathbf{v}, salt)) = \mathsf{H}_{\mathsf{U}}(m, (\mathbf{u}, \mathbf{v}))\,.$$

Next, we consider the ouput of the decapsulation algorithm for the second ciphertext $(\mathbf{u}, \mathbf{v}, \overline{salt})$, i.e., $\text{Hqc.Decaps}(sk^{\not\perp}, (\mathbf{u}, \mathbf{v}, \overline{salt}))$. We argue that the recomputation of the ciphertext-part $(\mathbf{u}, \mathbf{v})$ will fail with overwhelming probability. By correctness, we get $m = \text{Decoder}(\mathbf{v} - \mathbf{u}\mathbf{y})$ as the ciphertext was honestly generated and is decrypted (using the underlying base PKE Hqc.Pke) using the same secret key. For the recomputation, the randomness $(\overline{\mathbf{e}}, \overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2) = \mathsf{H}_{\mathsf{T}_{\mathsf{g}}}(m, pk, \overline{salt})$ is used which will be different from $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) = \mathsf{H}_{\mathsf{T}_{\mathsf{g}}}(m, pk, salt)$ as otherwise, $\mathcal{A}$ would have found a collision for $\mathsf{H}_{\mathsf{T}_{\mathsf{g}}}$. In the following, we assume that $(\overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2) \neq (\mathbf{r}_1, \mathbf{r}_2)$, which can be easily achieved by letting $\mathcal{A}$ sample $\overline{salt}$

until that is the case. Let $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$ denote the recomputed ciphertext (using randomness $(\overline{\mathbf{e}}, \overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2)$). This leads to

$$(\mathbf{u}, \mathbf{v}) = ((\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2), \mathbf{v}) \neq ((\overline{\mathbf{r}}_1 + \mathbf{h}\overline{\mathbf{r}}_2), \overline{\mathbf{v}}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})\,.$$

Thus the ciphertext gets rejected by outputting $\overline{k} \leftarrow \mathsf{H}_{\mathsf{U}}(\sigma, (\mathbf{u}, \mathbf{v}))$. Using the choice of m, we get $\overline{k} = \mathsf{H}_{\mathsf{U}}(\sigma, (\mathbf{u}, \mathbf{v})) = \mathsf{H}_{\mathsf{U}}(m, (\mathbf{u}, \mathbf{v})) = k$ which shows that adversary $\mathcal{A}$ wins the game LEAK-BIND-K,PK-CT. □

C.3 Proof of Theorem 11

Proof. We give the proof for HON-BIND-K-PK which extends to HON-BIND-K,CT-PK by the hierarchy of the notions.

Assume, for sake of contradiction, that there is an adversary $\mathcal{A}$ that wins the game HON-BIND-K-PK, i.e., given two honestly generated key pairs $(pk, sk^{\not\perp})$ and $(\overline{pk}, \overline{sk}^{\not\perp})$, with $sk^{\not\perp} = (sk, \sigma)$ and $\overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})$, it outputs c, $\overline{c}$ such that $\textsc{Hqc.Decaps}(sk^{\not\perp}, c) = k = \overline{k} = \textsc{Hqc.Decaps}(\overline{sk}^{\not\perp}, \overline{c})$. We distinguish between the cases $c \neq \overline{c}$ and $c = \overline{c}$ and start with the former.

Firstly note that $c \neq \overline{c}$ can be further divided in the following cases:

$$(\mathbf{u}, \mathbf{v}) \neq (\overline{\mathbf{u}}, \overline{\mathbf{v}}) \wedge \mathit{salt} \neq \overline{\mathit{salt}}$$
$$(\mathbf{u}, \mathbf{v}) \neq (\overline{\mathbf{u}}, \overline{\mathbf{v}}) \wedge \mathit{salt} = \overline{\mathit{salt}}$$
$$(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}}) \wedge \mathit{salt} \neq \overline{\mathit{salt}}$$

Independent of the fact whether the ciphertexts are rejected or not, the computation of the keys includes $(\mathbf{u}, \mathbf{v})$ and $(\overline{\mathbf{u}}, \overline{\mathbf{v}})$, more precisely $k = \mathsf{H}_{\mathsf{U}}(\cdot, (\mathbf{u}, \mathbf{v}))$ and $\overline{k} = \mathsf{H}_{\mathsf{U}}(\cdot, (\overline{\mathbf{u}}, \overline{\mathbf{v}}))$. As $k = \overline{k}$, this implies that the first two cases mentioned above cannot occur, as otherwise the adversary would have found a collision for H_{U}.[21] This leaves us with the case that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $\mathit{salt} \neq \overline{\mathit{salt}}$. Then, we can distinguish between the following cases for the ciphertexts output by $\mathcal{A}$ and $m \leftarrow \textsc{Pke}^{\$}.\textsc{Dec}(sk, c)$, $\overline{m} \leftarrow \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, \overline{c})$:

Case 1: $m = \perp \wedge \overline{m} = \perp$ (both ciphertexts are invalid)

Then $k = \mathsf{H}_{\mathsf{U}}(\sigma, (\mathbf{u}, \mathbf{v})) = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, (\overline{\mathbf{u}}, \overline{\mathbf{v}})) = \overline{k}$, which implies $\sigma = \overline{\sigma}$ as otherwise $\mathcal{A}$ would have found a collision for H_{U}. However, the keys are honestly generated, thus the randomly chosen rejection values will differ with overwhelming probability.

Case 2: $m \neq \perp \wedge \overline{m} = \perp$ (one ciphertext is invalid)[22]

Then $k = \mathsf{H}_{\mathsf{U}}(m, (\mathbf{u}, \mathbf{v})) = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, (\overline{\mathbf{u}}, \overline{\mathbf{v}})) = \overline{k}$, which implies $m = \overline{\sigma}$ as otherwise $\mathcal{A}$ would have found a collision for H_{U}. However, $m = \overline{\sigma}$ can be excluded with overwhelming probability, as $\mathcal{A}$ does not get $\overline{\sigma}$ and its only access is via querying invalid ciphertexts to the decryption oracle in which case the response will be the output of a random oracle on $\overline{\sigma}$ and the queried ciphertext which also does not reveal $\overline{\sigma}$.

Case 3: $m \neq \perp \wedge \overline{m} \neq \perp$ (both ciphertexts are valid)

Then $k = \mathsf{H}_{\mathsf{U}}(m, (\mathbf{u}, \mathbf{v})) = \mathsf{H}_{\mathsf{U}}(\overline{m}, (\overline{\mathbf{u}}, \overline{\mathbf{v}})) = \overline{k}$, which implies $m = \overline{m}$ as otherwise $\mathcal{A}$ would have found a collision for H_{U}. In total $\mathcal{A}$ found $c, \overline{c}$ such that $\textsc{Pke}^{\$}.\textsc{Dec}(sk, (\mathbf{u}, \mathbf{v})) = \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, (\overline{\mathbf{u}}, \overline{\mathbf{v}})) \neq \perp$, however, this contradicts the fact that Hqc is SCFR-CCA secure, which is proven in Proposition 3.

Lastly, we consider the case that $c = \overline{c}$, i.e., $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $\mathit{salt} = \overline{\mathit{salt}}$. For this we can apply the same argument as was used above for the case that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $\mathit{salt} \neq \overline{\mathit{salt}}$ holds. This part of the proof relies only on the fact that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$—at no point it is used that $\mathit{salt} \neq \overline{\mathit{salt}}$. This finishes the proof for HON-BIND-K-PK. □

[21] Note that this argument is essentially the proof of Theorem 2.

[22] Here we assume wlog that $\overline{c}$ is invalid.

C.4 Proof of Theorem 12

Proof. We give an attack against HON-BIND-K,PK-CT which, by the hierarchy of the notions, extends to HON-BIND-K-CT. We construct the following adversary $\mathcal{A}$. It receives a single public key pk as input. Adversary $\mathcal{A}$ picks an arbitrary ciphertext $c = (\mathbf{u}, \mathbf{v}, salt)$ and sets $\overline{c} = (\mathbf{u}, \mathbf{v}, \overline{salt})$ with $\overline{salt} \neq salt$. With overwhelming probability these two ciphertexts will be invalid. In this case, we obtain the same shared key

$$k = \text{HQC.DECAPS}(sk, c) = \mathsf{H}_{\mathsf{U}}(\sigma, (\mathbf{u}, \mathbf{v})) = \text{HQC.DECAPS}(sk, \overline{c}) = \overline{k}\,.$$

Thus, $\mathcal{A}$ wins the game HON-BIND-K,PK-CT. □

C.5 Proof of Proposition 2

Proof. We will give the proof for $\mathsf{T}_{\not\perp}$ and $\mathsf{T}_{\$}$ simultaneously and mark the changes from $\mathsf{T}_{\not\perp}$ to $\mathsf{T}_{\$}$ using gray boxes.

1. Let $\mathcal{A}$ be an SCFR-LEAK adversary against $\mathsf{X}[\text{PKE}, \mathsf{H}_{\mathsf{T}}]$. That means given (pk, sk) and $(\overline{pk}, \overline{sk})$, both stemming from honest key generations, $\mathcal{A}$ outputs a ciphertext c.[23] To show SCFR-LEAK security, we need to check that

$$m = \mathsf{X}[\text{PKE}, \mathsf{H}_{\mathsf{T}}].\text{DEC}(sk, c) = \mathsf{X}[\text{PKE}, \mathsf{H}_{\mathsf{T}}].\text{DEC}(\overline{sk}, c) \neq \perp\,.$$

   happens only with negligible probability. For X the $\mathsf{T}_{\not\perp}$ transform (and the $\mathsf{T}_{\$}$ transform, respectively), due to the re-encryption check, this translates to

$$\text{PKE.ENC}(pk, m; \mathsf{H}_{\mathsf{T}}(m,pk,salt)) = c = \text{PKE.ENC}(\overline{pk}, m; \mathsf{H}_{\mathsf{T}}(m,\overline{pk},salt))\,.$$

   The probability for this to happen is negligible by the assumption that two encryptions using honestly generated public keys pk, $\overline{pk}$ and random values $\mathsf{H}_{\mathsf{T}}(m,pk,salt)$, $\mathsf{H}_{\mathsf{T}}(m,\overline{pk},salt)$ will result in different ciphertexts with overwhelming probability.

2. We consider two honestly generated key pairs (pk, sk), $(\overline{pk}, \overline{sk})$, a randomly chosen message m, and $c \leftarrow_{\$} \text{PKE.ENC}(pk, m)$. To check restricted non-rigidity for PKE, we need to verify that

$$\text{PKE.ENC}(\overline{pk}, \text{PKE.DEC}(\overline{sk}, c)) \neq c\,,$$

   holds with overwhelming probability. This follows directly from our assumption that encryption with the two honestly generated public keys pk, $\overline{pk}$ using random coins will result in different ciphertexts with overwhelming probability.

□

C.6 Proof of Proposition 3

Proof. Using Proposition 2 it suffices to show that ciphertexts of the PKE schemes $\mathsf{T}_{\not\perp}[\text{BIKE.PKE}, \mathsf{H}_{\mathsf{T}}]$ and $\mathsf{T}_{\$}[\text{HQC.PKE}, \mathsf{H}_{\mathsf{T}}]$ have the following property: For two honestly generated public keys $pk, \overline{pk}$ and two (not necessarily different) messages $m, \overline{m}$, the ciphertexts $c = \text{ENC}(pk, m)$ and $\overline{c} = \text{ENC}(\overline{pk}, \overline{m})$ differ.

In the following, we focus on the first ciphertext component of the PKE schemes $\mathsf{T}_{\not\perp}[\text{BIKE.PKE}, \mathsf{H}_{\mathsf{T}}]$ and $\mathsf{T}_{\$}[\text{HQC.PKE}, \mathsf{H}_{\mathsf{T}}]$ to show that we obtain different ciphertexts in the above situation[24]: (1) For BIKE, consider $r_B = e_0 + e_1 h$, where $(e_0, e_1) = \mathsf{H}_{\mathsf{T}_{\not\perp}}(m)$ and

[23] Remember that for $\mathsf{T}_{\$}$, the ciphertext contains a salt.

[24] Note that for HQC, we could also argue over the salt value included in each ciphertext, which is chosen randomly during encapsulation and hence differs for different ciphertexts.

$h = pk$. (2) For HQC, consider $r_H = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$, where $(\mathbf{r}_1, \mathbf{r}_2) = \mathsf{H}_{\mathsf{T}_\$}(m, pk, salt)$ and h is part of the public key. To check the property, we consider honestly generated public keys $pk_B, \overline{pk}_B$ for BIKE and $pk_H, \overline{pk}_H$ for HQC as well as two messages $m, \overline{m}$. Denote the first ciphertext components resulting from encrypting m with pk_B (and pk_H) by r_B (and r_H), and the ones resulting from encrypting $\overline{m}$ with $\overline{pk}_B$ (and $\overline{pk}_H$) by $\overline{r}_B$ (and $\overline{r}_H$). Then, we obtain

$$r_B = e_0 + e_1 h \neq \overline{e}_0 + \overline{e}_1 \overline{h} = \overline{r}_B$$
$$r_H = \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \neq \overline{\mathbf{r}}_1 + \overline{\mathbf{h}}\overline{\mathbf{r}}_2 = \overline{r}_H$$

with overwhelming probability as h, $\overline{h}$, $\mathbf{h}$, and $\overline{\mathbf{h}}$ are randomly chosen values for honestly generated keys, and (e_0, e_1) and $(\mathbf{r}_1, \mathbf{r}_2)$ are derived using random oracles. This finishes the proof. □

C.7 Proof of Theorem 14

Proof. We cover the proof for both notions simultaneously, emphasizing the relevant differences when necessary. In both games, adversary $\mathcal{A}$ receives two honestly generated secret keys $sk^{\not\perp} = (sk, \sigma), \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})$—note that we assume the corresponding public keys $(pk/\overline{pk})$ and their hash values $(h_{pk}/h_{\overline{pk}})$ to be included in the secret keys—and has to output two ciphertexts $c, \overline{c}$ that decapsulate to the same shared key.

Depending on the validity of the ciphertexts (wrt PKE), we can distinguish between three cases. For each of them, we show that for $\mathcal{A}$ to be successful, there would have to be a random oracle collision, which can only happen with negligible probability.

Case 1: $m = \perp \wedge \overline{m} = \perp$ (both ciphertexts are invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(\sigma, c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, c)$. Since the keys are honestly generated, we have $\sigma \neq \overline{\sigma}$ with overwhelming probability which entails that $\mathcal{A}$ has found a collision for H_{U}. (In case of LEAK-BIND-K-CT we get a collision for H_{U} with certainty as the ciphertexts output by $\mathcal{A}$ have to be distinct.)

Case 2: $m \neq \perp \wedge \overline{m} = \perp$ (one ciphertext is invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(m, h_{pk}) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, c)$. This yields a collision as $(m, h_{pk}) \neq (\overline{\sigma}, c)$ due to the second inputs differing (one is a hash value, the other a ciphertext).

Case 3: $m \neq \perp \wedge \overline{m} \neq \perp$ (both ciphertexts are valid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(m, h_{pk}) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{m}, \overline{h}_{pk})$. Since the keys are honestly generated, we have $pk \neq \overline{pk}$ with overwhelming probability and thus a collision for H_{U} or H.

This concludes the proof. □

C.8 Proof of Theorem 15

Proof. We give the proof for MAL-BIND-K-PK and MAL-BIND-K-CT for scenario (I) DECAPS-DECAPS. At the end of the proof, we argue why it also covers the other two cases: (II) ENCAPS-DECAPS and (III) ENCAPS-ENCAPS.

Adversary $\mathcal{A}$ outputs two secret keys $sk^{\not\perp} = (sk, \sigma), \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})$ (which contain the corresponding public keys pk, $\overline{pk}$) and two ciphertexts $c, \overline{c}$ that decapsulate to the same shared keys. Depending on the validity of the ciphertexts (wrt PKE), we can distinguish between three cases. For all of them (except the last one), $\mathcal{A}$ can only win if it finds a random oracle collision, which we assume to be impossible.

Case 1: $m = \bot \wedge \overline{m} = \bot$ (both ciphertexts are invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(\sigma, \mathsf{H}(pk), c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \mathsf{H}(\overline{pk}), \overline{c})$. Depending on the notion the public keys (MAL-BIND-K-PK) or the ciphertexts (MAL-BIND-K-CT) have to be different. In both cases, we can therefore conclude that the adversary needs to find a collision in the hash function H_{U} (or H in case of MAL-BIND-K-PK where also $\mathsf{H}(pk) = \mathsf{H}(\overline{pk})$ can hold for $pk \neq \overline{pk}$).

Case 2: $m \neq \bot \wedge \overline{m} = \bot$ (one ciphertext is invalid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(m, \mathsf{H}(pk)) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \mathsf{H}(\overline{pk}), \overline{c})$. Clearly, this yields a collision as $(m, \mathsf{H}(pk)) \neq (\overline{\sigma}, \mathsf{H}(\overline{pk}), \overline{c})$.

Case 3: $m \neq \bot \wedge \overline{m} \neq \bot$ (both ciphertexts are valid)

In this case, we have $\mathsf{H}_{\mathsf{U}}(m, \mathsf{H}(pk)) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{m}, \mathsf{H}(\overline{pk}))$ and need to further distinguish between the two notions:

Case 3.1: MAL-BIND-K-PK

Since the output by the adversary for MAL-BIND-K-PK must satisfy $pk \neq \overline{pk}$, this yields that $\mathcal{A}$ has found a collision for H_{U} or H.

Case 3.2: MAL-BIND-K-CT

Assuming that the adversary does not find collisions for any of the hash functions H_{U}, H, and H_{T}, equality of the shared keys k and $\overline{k}$ yields $pk = \overline{pk}$ and $m = \overline{m}$ (for the messages resulting from decrypting c and $\overline{c}$ using sk and $\overline{sk}$, respectively). Since both ciphertexts are valid, the re-encryption succeeds for both, i.e.,

$$c = \textsc{Pke.Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m)) = \textsc{Pke.Enc}(\overline{pk}, \overline{m}; \mathsf{H}_{\mathsf{T}}(\overline{m})) = \overline{c}\,.$$

Then, however, $\mathcal{A}$ cannot win MAL-BIND-K-CT, as the game requires different ciphertexts.

The above cases also cover the Encaps-Decaps scenario and Encaps-Encaps scenario. Here, at least one ciphertext is honestly generated, meaning that it will be a valid ciphertext as encryption does not fail. This means that certain cases cannot occur. More precisely, only the last two cases can appear for Encaps-Decaps while only the last case can for Encaps-Encaps.

By the hierarchy of the notions, the above also shows security wrt to the other binding notions. This concludes the proof. □