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

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

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

Abstract. In this work, we 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 larger protocols. Recently, Cremers et al. (ePrint'24) introduced a framework for binding notions, encompassing previously existing but also new ones. While implicitly-rejecting 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 are among the round-4 candidates in NISTs PQC standardization process. Through this, we close a second gap as our results finish the analysis of the binding notions for the NIST round-4 KEMs.

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 [NIST17].

* Work of Juliane Krämer was funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) – 505500359. Patrick Struck was supported by the Hector Foundation II. Maximiliane Weishäupl was funded by the German Federal Ministry of Education and Research (BMBF) under the project Quant-ID (16KISQ111).

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 [NIST17]. 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 $\mathsf{H}(\sigma, c)$, where σ is the implicit rejection value contained in the secret key. Another variant (also covered in this work) is $\mathsf{FO}^{\not\perp}_m$, 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. [CDM23] showed that the authentication protocol presented in [BDK+18] is vulnerable to what is called a "re-encapsulation attack". 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.

To deal with this 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 of finding ciphertexts that decrypt under more than one key, for which there is already a long line of research [DGRW18, BH22, MLGR23, CR22, BH24, BCC+24, SW24, KSW23, NSS23, CFGI+23, DFG23, DMVA23]. 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 [NIST24, NIST22]. Very recently, Cremers et al. [CDM23]—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, 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. [CDM23] 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 decryption oracles for the secret keys. In total, Cremers et al. [CDM23] give 18 different security notions. 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 $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$.

There is a number of results regarding the security of implicitly-rejecting KEMs with respect to the binding notions. Firstly, Cremers et al. [CDM23] show that the ciphertext cannot bind any other value, i.e., the six notions X-BIND-CT-PK and X-BIND-CT-K with $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$ 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 [CDM23, 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 $\mathrm{X} \in \{\mathrm{MAL}, \mathrm{LEAK}, \mathrm{HON}\}$. 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. [CDM23] show that it achieves LEAK-BIND-K,PK-CT security (hence HON-BIND-K,PK-CT) while it is neither X-BIND-K-PK nor X-BIND-K-CT secure (for any $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}, \mathrm{MAL}\}$).[3] This leaves several notions open for $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. Fig. 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 NISTs standardization process: Grubbs et al. [GMP22] analyzed the security of Classic-McEliece [ABC+22], (the round-3 version of) Hqc [AAB+20], 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 [NIST23] was extended by Cremers et al. [CDM23] 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][4], however, are yet to be analyzed with respect to *any* of the binding properties.[5]

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

[3] We excluded the results that are based on a conjecture.

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

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

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 Fig. 1. Further, we complete the binding analysis of the NIST round-4 KEMs[6] with the results being depicted in Table 2.

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 (generally weaker) notion HON-BIND-K,CT-PK.[7] 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-regidity, 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.

We further show that these results can be applied to the code-based KEM 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 the HON variants of the notions under consideration, but none of the LEAK or MAL ones, i.e., only 4 out of 12 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. On the positive side, we show that the PKE schemes underlying Bike and Hqc* achieve SCFR-CPA security which—by [GMP22, Theorem 6]—yields that the resulting KEMs achieve HON-BIND-K,CT-PK. As we prove equivalence between HON-BIND-K-PK and HON-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}$-KEMs, this entails that Bike and Hqc* achieve both HON notions. On the negative side, we show that they do not achieve any of the remaining four notions (MAL-BIND-K-PK, MAL-BIND-K,CT-PK, LEAK-BIND-K-PK, and LEAK-BIND-K,CT-PK). This is the case as the underlying PKEs fulfill restricted non-rigidity, hence our generic attack for schemes with that property is applicable.

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

[6] Additionally, we provide results for Ml-Kem regarding the two open binding notions.
[7] 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 ransforms. Positive results are marked with ✓ while negative ones are marked with ✗; an asterix, i.e., ✓*/✗*, indicates that an result (either positive or negative) relies on some mild assumptions. Results from this work are marked blue while results from prior works, i.e., Theorem 6, Theorem 7, Theorem 8, and Theorem 9, are marked gray.

General Notion	Notion	$\mathsf{FO}^{\not\perp}$	$\mathsf{FO}^{\not\perp}_m$
X-BIND-K-PK	MAL-BIND-K-PK	✗ Cor. 19	✗ Thm. 9
	LEAK-BIND-K-PK	✗* Cor. 13	✗ Thm. 9
	HON-BIND-K-PK	✓* Cor. 15	✗ Thm. 9
X-BIND-K,CT-PK	MAL-BIND-K,CT-PK	✗ Thm. 18	✗ Thm. 18
	LEAK-BIND-K,CT-PK	✗* Thm. 12	✓* Thm. 20
	HON-BIND-K,CT-PK	✓* Thm. 6	✓* Cor. 21
X-BIND-K-CT	MAL-BIND-K-CT	✓ Thm. 16	✗ Thm. 9
	LEAK-BIND-K-CT	✓ Thm. 7	✗ Thm. 9
	HON-BIND-K-CT	✓ Thm. 7	✗ Thm. 9
X-BIND-K,PK-CT	MAL-BIND-K,PK-CT	✓ Cor. 17	✓ Thm. 24
	LEAK-BIND-K,PK-CT	✓ Thm. 7	✓ Thm. 9
	HON-BIND-K,PK-CT	✓ Thm. 7	✓ Thm. 9

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.

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) and one alternate candidate (FrodoKem [NAB+20]); the latter analyzed the anonymity for all round-3 KEMs.

Cremers et al. [CDM23] provide a framework covering various binding notions for KEMs which subsumes existing robustness properties. Similar properties were introduced in [BCDD+24] and [AHK+22] which are covered by the

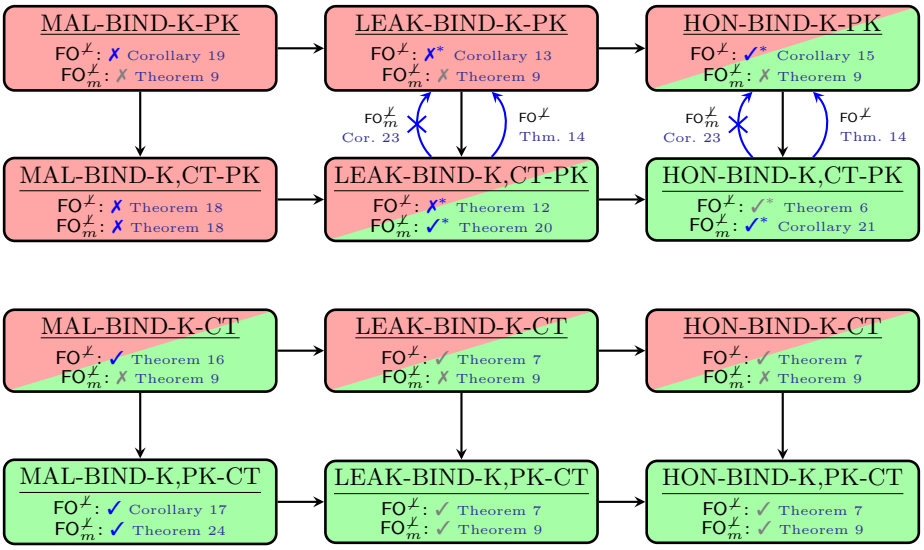

Fig. 1: Overview of the 12 binding notions for key-encapsulation mechanisms with their implications. For each notion, we state whether it is achieved (✓) or not achieved (✗) by $\mathsf{FO}^{\not\perp}$ or $\mathsf{FO}^{\not\perp}_m$. If the result relies on an extra consition, it is indicated by an asterix (✓*/✗*). Results from this work are marked in blue while results from prior works, i.e., Theorem 6, Theorem 7, Theorem 8, and Theorem 9 are marked in gray. Arrows indicate implications while crossed arrow denote separations.

general framework from [CDM23]. Besides the introduction of the framework, Cremers et al. [CDM23] 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.

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 ist

Table 2: Binding security of the round-4 KEMs Bike, Hqc, and Classic-McEliece, as well as Ml-Kem.

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	✓	✓ / ✓	✓	✓

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 $y \leftarrow \text{Alg}(x; r)$). Throughtout this work, we consider the security parameter λ implicit 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 give the definitions for 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.*

$\textsc{Dec}(sk, c)$ *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)* $\textsc{Kem}$ *is a triple of three algorithms* $(\textsc{KeyGen}, \textsc{Encaps}, \textsc{Decaps})$ *where*

$\textsc{KeyGen}()$ *is the key generation algorithm that outputs a key pair consisting of a public key* pk *and a secret key* sk*.*

$\textsc{Encaps}(pk)$ *is the encapsulation algorithm which takes as input a public key* pk*, and outputs a ciphertext* c *and an encapsulated key* k*.*

$\textsc{Decaps}(sk, c)$ *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 randomenss space of a key-encapsulation mechanism. As mentioned above, we will typically be explicit about the random coins. This entails that we write $c \leftarrow \textsc{Enc}(pk, m; r)$ to indicate that c is deterministically computed via the encryption algorithm using r as random coins. Likewise, we write $(c, k) \leftarrow \textsc{Encaps}(pk; r)$ to indicate that (c, k) is deterministically computed via the encapsulation algorithm using random coins r.

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, $\textsc{Dec}$ and $\textsc{Decaps}$ 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 transform 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: T and U. The former (transform T) 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 latter (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

Table 3: Overview about the computation of the shared key for the different variants of the FO transform.

	Implicit		Explicit	
	$\mathsf{FO}^{\not\perp}$	$\mathsf{FO}^{\not\perp}_m$	$\mathsf{FO}^{\perp}$	$\mathsf{FO}^{\perp}_m$
$m \neq \perp$	$\mathsf{H}_\mathsf{U}(m, c)$	$\mathsf{H}_\mathsf{U}(m)$	$\mathsf{H}_\mathsf{U}(m, c)$	$\mathsf{H}_\mathsf{U}(m)$
$m = \perp$	$\mathsf{H}_\mathsf{U}(\sigma, c)$	$\mathsf{H}_\mathsf{U}(\sigma, c)$	$\perp$	$\perp$

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 $\textsc{Kem.Encaps}(pk; r)$. 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. The differences between these variants, i.e., how the shared key k is computed conditioned on decryption succeeding or failing, can be seen in Table 3. Composing the T 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[8], which we will consider in Section 4. This variant (decomposed into the two underlying transforms T and $\mathsf{U}^{\not\perp}$) is shown in Fig. 2 along with the $\mathsf{FO}^{\not\perp}_m$ transform (decomposed into the two underlying transforms T 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. 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. [CDM23] 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.[9] The generic binding security game X-BIND-P-Q is shown

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

[9] A fourth case was introduced by Fiedler and Günther [FG24]. In this case, dubbed LEAK$^+$, the adversary is not provided with the key-pairs but the randomness used to generate those; this puts it between LEAK and MAL.

$\underline{\text{PKE}^{\not\perp}.\text{KeyGen}()}$

$(pk, sk) \leftarrow \text{KeyGen}()$
return (pk, sk)

$\underline{\text{PKE}^{\not\perp}.\text{Enc}(pk, m)}$

$c \leftarrow \text{Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m))$
return c

$\underline{\text{PKE}^{\not\perp}.\text{Dec}(sk, c)}$

$m \leftarrow \text{Dec}(sk, c)$
$\overline{c} \leftarrow \text{Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m))$
if $\overline{c} \neq c$
  return $\perp$
return m

$\underline{\text{KeyGen}^{\not\perp}()}$

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

$\underline{\text{Encaps}(pk)}$

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

$\underline{\text{Decaps}(sk^{\not\perp}, c)}$

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

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

in Fig. 3.[10] For implicitly-rejecting KEMs, there are twelve binding notions: X-BIND-K-PK, and X-BIND-K,CT-PK (cf. Fig. 4) as well as X-BIND-K-CT and X-BIND-K,PK-CT (cf. Fig. 5) for X ∈ {HON, LEAK, MAL}. The relations between the notions are described in Fig. 1. Below we define security wrt the various binding notions.

Definition 3. *Let* KEM *be a key-encapsulation mechanism. Further consider* X ∈ {HON, LEAK, MAL}, P ∈ {{K}, {K,CT}}*, and the game* X-BIND-P-PK *defined in Fig. 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.*

Remark 4. In the formalization of the MAL notions by Cremers et al. [CDM23], the adversary has to output two key pairs (pk, sk) and $(\overline{pk}, \overline{sk})$ and either (I) two ciphertexts c and $\overline{c}$ or (II) a randomness r and a ciphertext $\overline{c}$ or (III) two randomnesses r and $\overline{r}$. Depending on the outputs, either Encaps or Decaps is used by the game. Schmieg [Sch24] has shown a generic attack against this formalization which exploited the fact that the public keys output by $\mathcal{A}$ were not actually used by the game when the adversary outputs two ciphertext. More precisely, the game made use of the public key that is contained in the secret key whereas the actual public key is essentially obsolete. In an updated version, Cremers et al. [CDM23] formulated an explicit assumption stating that the decapsulation algorithm does not ignore the public key and use the provided public

[10] SCFR-CCA and SROB-CCA from [GMP22] correspond to HON-BIND-K,CT-PK and HON-BIND-CT-PK, respectively, in the framework from [CDM23].

key if needed, e.g., for re-encryption as done for FO-KEMs. We opt for a different, yet equivalent formalization: The adversary outputs either the secret key *or* the public key of a key pair, but never both at the same time. More precisely, for the different variants in the MAL games (see Fig. 3), the outputs are (I) sk, $\overline{sk}$; (II) pk, $\overline{sk}$; and (III) pk, $\overline{pk}$. Since we assume that the secret key contains the public key, the re-encryption step during decapsulation is still possible. At the same time Schmieg's attack cannot be applied anymore as the adversary is not able to provide a secret key (containing a public key) and a contradicting second public key.

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.[11]

Definition 5. *Consider the games* SCFR-CPA*,* SCFR-CCA*, and* SCFR-LEAK *as defined in Fig. 6. 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. [CDM23].

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

Theorem 7 (Adapted from [CDM23, Theorem D.1]). *Let* PKE *be a PKE scheme and* H_T *and* H_U *be random oracles. Then* $\text{KEM}^{\not\perp} = \mathsf{FO}^{\not\perp}[\text{PKE}, \mathsf{H}_\mathsf{T}, \mathsf{H}_\mathsf{U}]$ *is* LEAK-BIND-K-CT *secure.*

Theorem 8 ([CDM23, Theorem 4.11]). *An implicitly-rejecting KEM* KEM *cannot be* HON-BIND-CT-PK *or* HON-BIND-CT-K *secure.*

Theorem 9 (Adapted from [CDM23, Section B.4]). *Let* PKE *be a PKE scheme and* H_T *and* H_U *be random oracles. Then* $\text{KEM}_m^{\not\perp} = \mathsf{FO}_m^{\not\perp}[\text{PKE}, \mathsf{H}_\mathsf{T}, \mathsf{H}_\mathsf{U}]$ *is* LEAK-BIND-K,PK-CT *secure, but insecure with respect to* HON-BIND-K-CT *and* HON-BIND-K-PK*.*

Remark 10. Note that [CDM23, Section B.4] also contains a positive result for the LEAK-BIND-K,CT-PK security of $\text{FO}_m^{\not\perp}$-KEMs. However, a part of the proof

[11] Note that the PKE schemes underlying BIKE and HQC fulfill the new notion SCFR-LEAK (see Remark 32).

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 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 $g = 1$: // Variant (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 $g = 2$: // Variant (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 $g \notin \{1, 2\}$: // Variant (III): Encaps-Encaps
  $(pk, \overline{pk}, r, \overline{r}) \leftarrow \mathcal{A}()$
  $(k, c) \leftarrow \textsc{Encaps}(pk; r)$
  $(\overline{k}, \overline{c}) \leftarrow \textsc{Encaps}(\overline{pk}; \overline{r})$
$v_{\mathrm{p}} \leftarrow (\forall \mathrm{p} \in \mathrm{P} : x_{\mathrm{p}} = \overline{x}_{\mathrm{p}})$
$v_{\mathrm{q}} \leftarrow (\exists \mathrm{q} \in \mathrm{Q} : x_{\mathrm{q}} \neq \overline{x}_{\mathrm{q}})$
return $v_{\mathrm{p}} \wedge v_{\mathrm{q}}$

Fig. 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 variant (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 4 for more details. The decryption oracles Dec and $\overline{\mathsf{Dec}}$ in case of X = HON are omitted and work in the obvious way (without any restrictions) using sk and $\overline{sk}$, respectively.

is based on a conjecture and left open for future work—this is why we do not include the result in Theorem 9. Further details on this are provided in Section 3.5.

3 General Analysis of Implicitly-Rejecting FO

In this section, we give general results about the binding security of implicitly-rejecting FO-KEMs, i.e., $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. Section 3.1 and Section 3.2 cover the notions of the form X-BIND-K,CT-PK and X-BIND-K-PK. Section 3.3 completes the picture by showing MAL-BIND-K-CT and MAL-BIND-K,PK-CT security for $\mathsf{FO}^{\not\perp}$. Section 3.4 presents several MAL-BIND-P-Q attacks for both $\mathsf{FO}^{\not\perp}$ and $\mathsf{FO}^{\not\perp}_m$. Section 3.5 covers notions of the form X-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}_m$ and Section 3.6 notions of the form X-BIND-K,PK-CT.

Game HON-BIND-K,CT-PK

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

Game LEAK-BIND-K,CT-PK

$(pk, sk) \leftarrow\$ \textsc{KeyGen}()$
$(\overline{pk}, \overline{sk}) \leftarrow\$ \textsc{KeyGen}()$
$(c, \overline{c}) \leftarrow \mathcal{A}(pk, sk, \overline{pk}, \overline{sk})$
$k \leftarrow \textsc{Decaps}(sk, c)$
$\overline{k} \leftarrow \textsc{Decaps}(\overline{sk}, \overline{c})$
return $k = \overline{k} \wedge pk \neq \overline{pk} \wedge c = \overline{c}$

Game MAL-BIND-K,CT-PK

$g \leftarrow \mathcal{A}()$
if $g = 1$: // Variant (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 $g = 2$: // Variant (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 $g \notin \{1, 2\}$: // Variant (III): Encaps-Encaps
  $(pk, \overline{pk}, r, \overline{r}) \leftarrow \mathcal{A}()$
  $(k, c) \leftarrow \textsc{Encaps}(pk; r)$
  $(\overline{k}, \overline{c}) \leftarrow \textsc{Encaps}(\overline{pk}; \overline{r})$
return $k = \overline{k} \wedge pk \neq \overline{pk} \wedge c = \overline{c}$

Fig. 4: 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.

3.1 LEAK-BIND-K,CT-PK Attack for 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 T transform underlying the FO transform. This transformation derandomized 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 T transform but also a modified version used by Hqc. Hence, in the following we define the property for an abstract transformation but one can simply think of the T transform for now.

Definition 11. *Let* Pke *be a PKE scheme and* X *be a transformation. We say that* Pke *fulfills* X-restricted non-rigidity *(or* Pke *is* X-restricted non-rigid*) if for two honestly generated key pairs* $(pk, sk), (\overline{pk}, \overline{sk})$*, a randomly chosen mes-*

Game HON-BIND-K,PK-CT

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

Game LEAK-BIND-K,PK-CT

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

Game MAL-BIND-K,PK-CT

$g \leftarrow \mathcal{A}()$
if $g = 1$: // Variant (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 $g = 2$: // Variant (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 $g \notin \{1, 2\}$: // Variant (III): Encaps-Encaps
  $(pk, \overline{pk}, r, \overline{r}) \leftarrow \mathcal{A}()$
  $(k, c) \leftarrow \textsc{Encaps}(pk; r)$
  $(\overline{k}, \overline{c}) \leftarrow \textsc{Encaps}(\overline{pk}; \overline{r})$
return $k = \overline{k} \wedge c \neq \overline{c} \wedge pk = \overline{pk}$

Fig. 5: Security games HON-BIND-K-CT, HON-BIND-K,PK-CT, LEAK-BIND-K-CT, LEAK-BIND-K,PK-CT, MAL-BIND-K-CT, and MAL-BIND-K,PK-CT. The variants involving the public key include the highlighted statements, the other do not.

sage m, and $c \leftarrow\!\!\$\ \mathsf{X}[\textsc{Pke}].\textsc{Enc}(pk, m)$, we have

$$\mathsf{X}[\textsc{Pke}].\textsc{Dec}(\overline{sk}, c) = \bot$$

with overwhelming probability.

The theorem below shows that applying $\text{FO}^{\not\perp}$ to a PKE scheme that is X-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 decapsulates to $\mathsf{H}_{\mathsf{U}}(\sigma, c)$ (under the key pair which was used to honestly generate c). By non-rigidness 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 12. *Let* Pke *be a public-key encryption scheme that is* T*-restricted non-rigid and* $\textsc{Kem}^{\not\perp}$ *be the key-encapsulation mechanism obtained from applying* $\text{FO}^{\not\perp}$ *to* Pke*. Then,* $\textsc{Kem}^{\not\perp}$ *is not* LEAK-BIND-K,CT-PK *secure.*

Proof. We construct a LEAK-BIND-K,CT-PK adversary $\mathcal{A}$ against $\textsc{Kem}^{\not\perp}$ as follows. As input, $\mathcal{A}$ obtains honestly generated $(pk, sk^{\not\perp} = (sk, \sigma)), (\overline{pk}, \overline{sk}^{\not\perp} =$

Game SCFR-LEAK	Game SCFR-CCA	Oracle $\mathsf{Dec}(c)$
$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$	$(pk, sk) \leftarrow\!\!\$\ \textsc{KeyGen}()$	$k \leftarrow \textsc{Decaps}(sk, c)$
$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$	$(\overline{pk}, \overline{sk}) \leftarrow\!\!\$\ \textsc{KeyGen}()$	**return** k
$c \leftarrow \mathcal{A}(pk, \overline{pk}, sk, \overline{sk})$	$c \leftarrow \mathcal{A}^{\mathsf{Dec}, \overline{\mathsf{Dec}}}(pk, \overline{pk})$	Oracle $\overline{\mathsf{Dec}}(c)$
$m \leftarrow \textsc{Dec}(sk, c)$	$m \leftarrow \textsc{Dec}(sk, c)$	$k \leftarrow \textsc{Decaps}(\overline{sk}, c)$
$\overline{m} \leftarrow \textsc{Dec}(\overline{sk}, c)$	$\overline{m} \leftarrow \textsc{Dec}(\overline{sk}, c)$	**return** k
return $m = \overline{m} \neq \bot$	**return** $m = \overline{m} \neq \bot$	

Fig. 6: Security games SCFR-LEAK and SCFR-CCA for PKEs. By removing the decryption oracles in game SCFR-CCA, we obtain the game SCFR-CPA as defined in [GMP22] and required for Theorem 6.

$(\overline{sk}, \overline{\sigma})) \leftarrow\!\!\$\ \textsc{Kem}^{\not\perp}.\textsc{KeyGen}()$ 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 $k = \mathsf{H}_{\mathsf{U}}(m, c)$ which follows by definition of $\textsc{Pke.Encaps}$. 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 $\bot$.[12] Next, we show that c decapsulates to k under $\overline{sk}^{\not\perp}$ as well: As the underlying PKE scheme is T-restricted non-rigid, we have that

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

with overwhelming probability, for $\overline{m} = \textsc{Pke.Dec}(\overline{sk}, c)$. Therefore, we get $\mathsf{T}[\textsc{Pke}, \mathsf{H}_{\mathsf{T}}].\textsc{Dec}(\overline{sk}, c) = \bot$ and hence $\textsc{Kem}^{\not\perp}.\textsc{Decaps}(\overline{sk}^{\not\perp}, c) = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, c)$. This leads to

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

which shows that $\mathcal{A}$ wins the game LEAK-BIND-K,CT-PK. □

Corollary 13. *Let* $\textsc{Pke}$ *be a public-key encryption scheme that is* T*-restricted non-rigid and* $\textsc{Kem}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\textsc{FO}^{\not\perp}$ *to* $\textsc{Pke}$*. Then* $\textsc{Kem}^{\not\perp}$ *is neither* MAL-BIND-K-PK *nor* LEAK-BIND-K-PK *nor* MAL-BIND-K,CT-PK *secure.*

[12] Note that the decryption for $\textsc{Kyber.Pke}$ never returns $\bot$, while for $\textsc{Bike.Pke}$ and $\textsc{Hqc.Pke}$ this is possible due to decoding failures.

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 6 also yields HON-BIND-K-PK security.

The following theorem establishes equivalence between HON-BIND-K-PK and HON-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}$-KEMs.

Theorem 14. *Let* $\textsc{Pke}$ *be a public-key encryption scheme and* $\textsc{Kem}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}^{\not\perp}$ *to* $\textsc{Pke}$*. 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* $\mathrm{X} \in \{\mathrm{HON}, \mathrm{LEAK}\}$.

Proof. We give the proof in terms of X = HON and argue at the relevant parts what changes for X = LEAK. We prove the claim by showing 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 decryption oracles for $\mathcal{A}$ using its own ones. When $\mathcal{A}$ outputs $(c, \overline{c})$, $\mathcal{B}$ in turn outputs 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. [CDM23] showed a separation between the notions for $\mathsf{FO}^{\perp}_{m}$. Looking ahead, our results for $\mathsf{FO}^{\not\perp}_{m}$ (cf. Corollary 23) 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 $\text{FO}^{\not\perp}$-KEMs in the above theorem, we can leverage Theorem 6 to show HON-BIND-K-PK security: 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 $\text{FO}^{\not\perp}$ KEM. This is formalized in the following corollary.

Corollary 15. *Let* PKE *be a public-key encryption scheme that has negligible decryption failures and* $\text{KEM}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}^{\not\perp}$ *to* PKE. *If* $\text{PKE}^{\not\$} = \mathsf{T}[\text{PKE}, \mathsf{H}_\mathsf{T}]$ *is* SCFR-CPA *secure, then* $\text{KEM}^{\not\perp}$ *is* HON-BIND-K,CT-PK *secure.*

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

We show that KEMs obtained via the $\mathsf{FO}^{\not\perp}$ transform are MAL-BIND-K-CT and MAL-BIND-K,PK-CT secure.

Theorem 16. *Let* PKE *be a public-key encryption scheme and* $\text{KEM}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}^{\not\perp}$ *to* PKE. *Then* $\text{KEM}^{\not\perp}$ *is* MAL-BIND-K-CT *secure.*

Proof. Let $\mathcal{A}$ be an adversary againg 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 obtain 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. □

The following corollary follows from the above theorem via the trivial implications.

Corollary 17. *Let* PKE *be a public-key encryption scheme and* $\text{KEM}^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}^{\not\perp}$ *to* PKE. *Then* $\text{KEM}^{\not\perp}$ *is* MAL-BIND-K,PK-CT *secure.*

While these two results were claimed in an earlier version of [CDM23][13], the current version claims that result only for 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.

[13] 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.

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 does not achieve MAL-BIND-K-PK security 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 an thus breaks MAL-BIND-K,CT-PK security. This is formally stated in the theorem below.

Theorem 18. *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 not* MAL-BIND-K,CT-PK *secure.*

Proof. We give an 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, 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. □

The following corollary follows directly from the above theorem. For ML-KEM the corollary is exactly what is shown in [Sch24].

Corollary 19. *Let* PKE *be a public-key encryption scheme and* KEM *be the KEM resulting from applying the implicitly-rejecting* FO *transform (regardless of which of the two variants) to* PKE*. Then* KEM *is not* MAL-BIND-K-PK *secure.*

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

In the following section, we show that $\mathsf{FO}^{\not\perp}_m$ 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 [CDM23]: In [CDM23, Appendix B.4] LEAK-BIND-K,CT-PK security of $\mathsf{FO}^{\not\perp}_m$ is reduced to LEAK-BIND-K,CT-PK security of the explicitly-rejecting variant $\mathsf{FO}^{\perp}_m$. The latter is handled in [CDM23, Appendix B.2], which traces LEAK-BIND-K,CT-PK security back to LEAK-BIND-CT-PK security of $\mathsf{FO}^{\perp}_m$. However, so far it is only conjectured that $\mathsf{FO}^{\perp}_m$ 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 and we fill this gap resulting in Theorem 20. Further, we make the robustness assumption on the PKE more precise (SCFR-LEAK) and show that it is not only a sufficient but also a necessary condition for achieving LEAK-BIND-K,CT-PK security (Theorem 22).

Theorem 20. *Let* PKE *be a public-key encryption scheme,* $\text{PKE}^{\$}$ *its derandomized variante, and* $\text{KEM}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}_m^{\not\perp}$*. If* $\text{PKE}^{\$}$ *is* SCFR-LEAK *secure, then* $\text{KEM}_m^{\not\perp}$ *is* LEAK-BIND-K,CT-PK *secure.*

Proof. Assume for a contradiction that there is a successful adversary $\mathcal{A}$ against LEAK-BIND-K,CT-PK: given two honestly generated key pairs $(pk, sk^{\not\perp} = (sk, \sigma)), (\overline{pk}, \overline{sk}^{\not\perp} = (\overline{sk}, \overline{\sigma})) \leftarrow \text{KEM}_m^{\not\perp}.\text{KEYGEN}()$, $\mathcal{A}$ outputs a ciphertext c such that $k = \text{KEM}_m^{\not\perp}.\text{DECAPS}(sk^{\not\perp}, c) = \text{KEM}_m^{\not\perp}.\text{DECAPS}(\overline{sk}^{\not\perp}, c) = \overline{k}$. For $m = \text{PKE}^{\$}.\text{DEC}(sk, c)$ and $\overline{m} = \text{PKE}^{\$}.\text{DEC}(\overline{sk}, c)$, 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)[14]
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 $\text{PKE}^{\$}$ as it validly decrypts to the same message under two different secret keys.

This concludes the proof. □

The following corollary follows directly from Theorem 20. Note, however, that for the weaker notion HON-BIND-K,CT-PK, also 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.

Corollary 21. *Let* PKE *be a public-key encryption scheme,* $\text{PKE}^{\$}$ *its derandomized variante, and* $\text{KEM}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}_m^{\not\perp}$*. If* $\text{PKE}^{\$}$ *is* SCFR-CCA *secure, then* $\text{KEM}_m^{\not\perp}$ *is* HON-BIND-K,CT-PK *secure.*

Theorem 22. *Let* PKE *be a public-key encryption scheme,* $\text{PKE}^{\$}$ *its derandomized variante, and* $\text{KEM}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}_m^{\not\perp}$*. If* $\text{KEM}_m^{\not\perp}$ *is* LEAK-BIND-K,CT-PK *secure, then* $\text{PKE}^{\$}$ *is* SCFR-LEAK *secure.*

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

Proof. Assume for a contradiction that $\text{PKE}^{\not\perp}$ 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. $\text{PKE}^{\not\perp}.\text{DEC}(sk, c) = m = \overline{m} = \text{PKE}^{\not\perp}.\text{DEC}(\overline{sk}, c) \neq \perp$. As $m = \overline{m} \neq \perp$, we obtain that $k = \mathsf{H}_{\mathsf{U}}(m) = \mathsf{H}_{\mathsf{U}}(\overline{m}) = \overline{k}$. Then we can construct an adversary $\mathcal{B}$ against $\text{KEM}_m^{\not\perp}$ 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 $\text{KEM}_m^{\not\perp}$ 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 from our positive result for X-BIND-K,CT-PK (Theorem 20) and the the negative result for X-BIND-K,CT-PK (Theorem 9).

Corollary 23. *There is a KEM* KEM *that is* X-BIND-K,CT-PK *secure but not* X-BIND-K-PK *secure, for* $X \in \{HON, LEAK\}$.

3.6 MAL-BIND-K,PK-CT Security for $\mathbf{FO}_m^{\not\perp}$

We show that the $\mathsf{FO}_m^{\not\perp}$ transform achieves X-BIND-K,PK-CT security for $\text{X} \in \{\text{HON}, \text{LEAK}, \text{MAL}\}$ which, together with Theorem 7, 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.

Theorem 24. *Let* PKE *be a public-key encryption scheme and* $\text{KEM}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\text{FO}_m^{\not\perp}$ *to* PKE. *Then* $\text{KEM}_m^{\not\perp}$ *is* MAL-BIND-K,PK-CT *secure.*

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 distinct, i.e., $pk = \overline{pk}$) two distinct ciphertexts $c \neq \bar{c}$ 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 $m \leftarrow \text{PKE}^{\not\perp}.\text{DEC}(sk, c)$ and $\overline{m} \leftarrow \text{PKE}^{\not\perp}.\text{DEC}(\overline{sk}, \bar{c})$:

Case 1: $m = \perp \wedge \overline{m} = \perp$ (both ciphertexts are invalid)
In this case, we have $\mathsf{H}_{\mathsf{U}}(\sigma, c) = k = \overline{k} = \mathsf{H}_{\mathsf{U}}(\overline{\sigma}, \bar{c})$. Even thought $\mathcal{A}$ has full control over σ and $\overline{\sigma}$, the fact that $c \neq \bar{c}$ 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)[15]
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 H_{T}. However, in this case, validity of both ciphertexts yield

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

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

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 yield 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 $c \leftarrow \textsc{Kem}_m^{\not\perp}.\textsc{Encaps}(pk; m)$ is derived. By correctness of $\textsc{Kem}_m^{\not\perp}$, 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 $c \leftarrow \textsc{Kem}_m^{\not\perp}.\textsc{Encaps}(pk; m)$ and $\overline{c} \leftarrow \textsc{Kem}_m^{\not\perp}.\textsc{Encaps}(\overline{pk}; \overline{m})$, respectively. We thus get that both ciphertexts will be valid by correctness, which means we merely need the final case from the three above. This concludes the proof. □

The corollary below states that $\mathsf{FO}_m^{\not\perp}$-KEMs achieve LEAK-BIND-K,PK-CT and HON-BIND-K,PK-CT.

Corollary 25. *Let* Pke *be a public-key encryption scheme and* $\textsc{Kem}_m^{\not\perp}$ *the key-encapsulation mechanism obtained from applying* $\mathrm{FO}_m^{\not\perp}$ *to* Pke. *Then* $\textsc{Kem}_m^{\not\perp}$ *is* LEAK-BIND-K,PK-CT *and* HON-BIND-K,PK-CT *secure.*

4 Application to BIKE and HQC

In this section, we analyze the binding properties of the round-4 KEMs Bike and Hqc which are both based on codes. 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 introduced by us 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 give an overview of the completed results for the round-4 KEMs and Ml-Kem.

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

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 that is described in Fig. 7; the resulting KEM Bike is depicted in Fig. 8.

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 underlying Hqc, Hqc.Pke, is displayed in Fig. 9; the key-encapsulation mechanism Hqc that results from applying *a variant of* the $\mathsf{FO}^{\not\perp}$ transform to Hqc.Pke is shown in Fig. 10. 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 $\mathsf{H}_{\mathsf{T}}(m \parallel pk \parallel salt)$ instead of $\mathsf{H}_{\mathsf{T}}(m)$. We describe the modified transformation in Fig. 11.

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 Fig. 10)—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. 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. As same is the case for Bike, 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 the proposed parameter sets for Bike [ABB+22, Section C.4] and for Hqc [AAB+24, Section 5.3]—using techniques from 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.

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 Fig. 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 key) is used to derive the randomness for the re-encryption approach. While the general attacks we give

BIKE.KEYGEN()	BIKE.ENC(pk, m)	BIKE.DEC(sk, c)
$(h_0, h_1) \leftarrow\!\!\$\ \mathcal{H}_w$	$(e_0, e_1) \leftarrow\!\!\$\ \mathcal{E}_t$	$\overline{e} \leftarrow \text{DECODER}(c_0 h_0, h_0, h_1)$
$h \leftarrow h_0 h_1^{-1}$	$c_0 \leftarrow e_0 + e_1 h$	**if** $\overline{e} = \bot$
$pk \leftarrow h$	$c_1 \leftarrow m \oplus \mathsf{H}_L(e_0, e_1)$	$\quad m \leftarrow \bot$
$sk \leftarrow (h_0, h_1)$	$c \leftarrow (c_0, c_1)$	**else**
return (pk, sk)	**return** c	$\quad m \leftarrow c_1 \oplus \mathsf{H}_L(\overline{e})$
		return m

Fig. 7: Public-key encryption scheme BIKE.PKE. 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 $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$ and the BIKE parameters w, r and t.

BIKE.KEYGEN()	BIKE.ENCAPS(pk)	BIKE.DECAPS(sk, c)
$(h_0, h_1) \leftarrow\!\!\$\ \mathcal{H}_w$	$m \leftarrow\!\!\$\ \mathcal{M}$	$\overline{e} \leftarrow \text{DECODER}(c_0 h_0, h_0, h_1)$
$h \leftarrow h_0 h_1^{-1}$	$(e_0, e_1) \leftarrow \mathsf{H}_H(m)$	$\overline{m} \leftarrow c_1 \oplus \mathsf{H}_L(\overline{e})$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$	$c_0 \leftarrow e_0 + e_1 h$	**if** $\overline{e} = \mathsf{H}_H(\overline{m})$
$pk \leftarrow h$	$c_1 \leftarrow m \oplus \mathsf{H}_L(e_0, e_1)$	$\quad k \leftarrow \mathsf{H}_U(\overline{m}, c)$
$sk \leftarrow (h_0, h_1, \sigma)$	$c \leftarrow (c_0, c_1)$	**else**
return (pk, sk)	$k \leftarrow \mathsf{H}_U(m, c)$	$\quad k \leftarrow \mathsf{H}_U(\sigma, c)$
	return (c, k)	**return** k

Fig. 8: Key-encapsulation mechanism BIKE.

for $\mathsf{FO}^{\not\perp}$-KEMs are still applicable[16], the general proofs from Section 2 and Section 3 cannot be transferred this easily. In fact, HQC fulfills the notions only in the HON setting, while it is insecure with respect to any of the notions in the LEAK/MAL setting. 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.

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 proposition.

Proposition 26. *The key-encapsulation mechanism* HQC *as shown in Fig. 10 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*.*

[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 are still applicable by choosing the salts contained in the ciphertexts output by the adversary to be equal.

HQC.KEYGEN()	HQC.ENC(pk, m)	HQC.DEC(sk, c)
$\mathbf{h} \leftarrow\!\!\$\ \mathcal{R}$	$(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \leftarrow\!\!\$\ \mathcal{R}_{w_e} \times \mathcal{R}_{w_r} \times \mathcal{R}_{w_r}$	$m \leftarrow \text{DECODER}(\mathbf{v} - \mathbf{uy})$
$\mathbf{G} \leftarrow\!\!\$\ \mathbb{F}_2^{k \times n}$	$\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$	**return** m
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_w \times \mathcal{R}_w$	$\mathbf{v} \leftarrow \text{TRUNC}(m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}, \ell)$	
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$	$c \leftarrow (\mathbf{u}, \mathbf{v})$	
$pk \leftarrow (\mathbf{h}, \mathbf{s})$	**return** c	
$sk \leftarrow (\mathbf{x}, \mathbf{y})$		
return (pk, sk)		

Fig. 9: Public-key encryption scheme HQC.PKE. 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.

HQC.KEYGEN()	HQC.ENCAPS(pk)	HQC.DECAPS(sk, c)
$\mathbf{h} \leftarrow\!\!\$\ \mathcal{R}$	$m \leftarrow\!\!\$\ \mathbb{F}_2^k$	$\overline{m} \leftarrow \text{DECODER}(\mathbf{v} - \mathbf{uy})$
$\mathbf{G} \leftarrow\!\!\$\ \mathbb{F}_2^{k \times n}$	$salt \leftarrow\!\!\$\ \mathbb{F}_2^{128}$	$(\overline{\mathbf{e}}, \overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2) \leftarrow \mathsf{H}_\mathsf{T}(\overline{m}, pk, salt)$
$\mathbf{x}, \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_w \times \mathcal{R}_w$	$(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) \leftarrow \mathsf{H}_\mathsf{T}(m, pk, salt)$	$\overline{\mathbf{u}} \leftarrow \overline{\mathbf{r}}_1 + \mathbf{h}\overline{\mathbf{r}}_2$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$	$\mathbf{u} \leftarrow \mathbf{r}_1 + \mathbf{hr}_2$	$\overline{\mathbf{v}} \leftarrow \text{TRUNC}(\overline{m}\mathbf{G} + \mathbf{s}\overline{\mathbf{r}}_2 + \overline{\mathbf{e}}, \ell)$
$\mathbf{s} \leftarrow \mathbf{x} + \mathbf{hy}$	$\mathbf{v} \leftarrow \text{TRUNC}(m\mathbf{G} + \mathbf{sr}_2 + \mathbf{e}, \ell)$	**if** $(\overline{\mathbf{u}}, \overline{\mathbf{v}}) = (\mathbf{u}, \mathbf{v})$
$pk \leftarrow (\mathbf{h}, \mathbf{s})$	$k \leftarrow \mathsf{H}_\mathsf{U}(m, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$	$\quad k \leftarrow \mathsf{H}_\mathsf{U}(\overline{m}, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$
$sk \leftarrow (\mathbf{x}, \mathbf{y}, \sigma)$	$c \leftarrow (\mathbf{u}, \mathbf{v}, salt)$	**else**
return (pk, sk)	**return** (c, k)	$\quad k \leftarrow \mathsf{H}_\mathsf{U}(\sigma, (\mathbf{u}, \mathbf{v}, \mathit{salt}))$
		return k

Fig. 10: Key-encapsulation mechanism HQC. The modified version HQC* includes the highlighted parts, i.e., the *entire* ciphertext is hashed to compute the key k.

Proof. Insecurity with respect to LEAK-BIND-K,CT-PK follows easily from Theorem 12 using the fact that HQC.PKE fulfills T_{HQC}-restricted non-rigidity. The latter is proven in Proposition 33, which covers BIKE, HQC, and the modified variant HQC*. While Theorem 12 is formulated for T, it directly transfers to T_{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.

The following theorem 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, which is described in the ensuing corollary. The overall attack idea follows the strategy of our attack against LEAK-BIND-K,CT-PK for $\mathsf{FO}^{\not\perp}$-KEMs (cf. Theorem 12): 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.

$\text{PKE}^{\not\perp}.\text{KeyGen}()$	$\text{PKE}^{\not\perp}.\text{Enc}(pk, m)$	$\text{PKE}^{\not\perp}.\text{Dec}(sk, (c, salt))$
$(pk, sk) \leftarrow \text{KeyGen}()$	$salt \leftarrow\!\!\$\ \mathcal{S}$	$m \leftarrow \text{Dec}(sk, c)$
return (pk, sk)	$r \leftarrow \mathsf{H}_{\mathsf{T}}(m, pk, salt)$	$r \leftarrow \mathsf{H}_{\mathsf{T}}(m, pk, salt)$
	$c \leftarrow \text{Enc}(pk, m; r)$	$\overline{c} \leftarrow \text{Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(r))$
	return $(c, salt)$	**if** $\overline{c} \neq c$
		return $\perp$
		return m

$\text{KeyGen}^{\not\perp}()$	$\text{Encaps}(pk)$	$\text{Decaps}(sk^{\not\perp}, (c, salt))$
$(pk, sk) \leftarrow \text{KeyGen}()$	$m \leftarrow\!\!\$\ \mathcal{M}$	$(sk, \sigma) \leftarrow sk^{\not\perp}$
$\sigma \leftarrow\!\!\$\ \mathcal{M}$	$(c, salt) \leftarrow \text{PKE}^{\not\perp}.\text{Enc}(pk, m)$	$m \leftarrow \text{PKE}^{\not\perp}.\text{Dec}(sk, (c, salt))$
$sk^{\not\perp} \leftarrow (sk, \sigma)$	$k \leftarrow \mathsf{H}_{\mathsf{U}}(m, c)$	**if** $m \neq \perp$
return $(pk, sk^{\not\perp})$	**return** $((c, salt), k)$	**return** $\mathsf{H}_{\mathsf{U}}(m, c)$
		return $\mathsf{H}_{\mathsf{U}}(\sigma, c)$

Fig. 11: **Top:** The PKE scheme $\mathsf{T}_{\text{HQC}}[\text{PKE}]$. **Bottom:** The implicitly-rejecting KEM $\mathsf{U}_{\text{HQC}}[\mathsf{T}_{\text{HQC}}[\text{PKE}]]$.

Since the salt is not used to derive the key k, the valid and invalid ciphertexts result in the same key.

Theorem 27. *The key-encapsulation mechanism* HQC *as shown in Fig. 10 is not* LEAK-BIND-K,PK-CT *secure.*

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 $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) = \mathsf{H}_{\mathsf{T}}(m, pk, salt)$. 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 $m = \text{DECODER}(\mathbf{v} - \mathbf{uy})$ 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 $(\overline{\mathbf{e}}, \overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2) = \mathsf{H}_{\mathsf{T}}(m, pk, \overline{salt})$ is used which will be different from $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2) = \mathsf{H}_{\mathsf{T}}(m, pk, salt)$ as otherwise, $\mathcal{A}$ would have found a collision for H_{T}. 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 ciphertexts 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. □

The following corollary follows directly from the above theorem.

Corollary 28. *The key-encapsulation mechanism* HQC *as shown in Fig. 10 is neither* MAL-BIND-K-CT *nor* MAL-BIND-K,PK-CT *nor* LEAK-BIND-K-CT *secure.*

Proving HON Binding Notions for HQC**.** The theorem below shows that HQC achieves both HON-BIND-K-CT and HON-BIND-K-PK.

Theorem 29. *Consider the key-encapsulation mechanism* HQC *as shown in Fig. 10. Then* HQC *is both* HON-BIND-K-CT *and* HON-BIND-K-PK *secure.*

Proof. We will start with HON-BIND-K-CT and subsequently explain how the proof can be extended to also cover HON-BIND-K-PK.

Assume, for sake of contradiction, that there is an adversary $\mathcal{A}$ that wins the game HON-BIND-K-CT, 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 $c \neq \overline{c}$ such that $\text{HQC.DECAPS}(sk^{\not\perp}, c) = k = \overline{k} = \text{HQC.DECAPS}(\overline{sk}^{\not\perp}, \overline{c})$.

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

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

Independent of the fact whether the ciphertexts are rejected or not, the computation of the keys includes $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{u}, \mathbf{v})$, 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 written above cannot occur, as otherwise the adversary would have found a collision for H_{U}.[17] This leaves us with the case that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $salt \neq \overline{salt}$. Then, we can distinguish between the following cases for the ciphertexts output by $\mathcal{A}$ and $m \leftarrow \text{PKE}^{\not\perp}.\text{DEC}(sk, c)$, $\overline{m} \leftarrow \text{PKE}^{\not\perp}.\text{DEC}(\overline{sk}, \overline{c})$:

[17] Note that this argument is essentially the proof of Theorem 7.

Case 1: $m = \bot \wedge \overline{m} = \bot$ (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: $m \neq \bot \wedge \overline{m} = \bot$ (one ciphertext is invalid)[18]
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: $m \neq \bot \wedge \overline{m} \neq \bot$ (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 $\textsc{Pke}^{\$}.\textsc{Dec}(sk, (\mathbf{u}, \mathbf{v})) = \textsc{Pke}^{\$}.\textsc{Dec}(\overline{sk}, (\overline{\mathbf{u}}, \overline{\mathbf{v}})) \neq \bot$, however, this contradicts the fact that Hqc is SCFR-CCA secure, which is proven in Proposition 31.[19]

As we have derived a contradiction in each case, HON-BIND-K-CT security for Hqc is proven.

Next, consider $\mathcal{B}$ an adversary 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 and $\overline{c}$ such that $\textsc{Hqc.Decaps}(sk^{\not\perp}, c) = k = \overline{k} = \textsc{Hqc.Decaps}(\overline{sk}^{\not\perp}, \overline{c})$. Firstly, note that $pk \neq \overline{pk}$ holds with overwhelming probability. We can distinguish two cases based on the ciphertexts output by the adversary: Firstly, if $c \neq \overline{c}$, we are in the situation of the HON-BIND-K-CT proof given above. Secondly, for $c = \overline{c}$ (this case is excluded in the notion HON-BIND-K-CT considered above) we obtain $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $salt = \overline{salt}$. In the proof given above for HON-BIND-K-CT, we reduce to the case that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$ and $salt \neq \overline{salt}$ holds. This part of the proof, however, relies only on the fact that $(\mathbf{u}, \mathbf{v}) = (\overline{\mathbf{u}}, \overline{\mathbf{v}})$—at no point it is used that $salt \neq \overline{salt}$. This allows to apply the same three cases as above here. This finishes the proof for HON-BIND-K-PK. □

The corollary below follows directly using the hierarchy between the binding notions.

Corollary 30. *Consider the key-encapsulation mechanism* Hqc *as shown in Fig. 10. Then* Hqc *is both* HON-BIND-K,CT-PK *and* HON-BIND-K,PK-CT *secure.*

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

[19] More precisely, this proposition proves SCFR-CPA security, however, as is described in Remark 32, this can easily be extended to SCFR-CCA.

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 T-restricted non-rigidity property while security with respect to the latter two, requires the underlying T-transformed PKE scheme to achieve SCFR-CPA. The two propositions below establish these properties for BIKE and HQC*. Note that the T_{HQC}-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.

Proposition 31. *Consider the public-key encryption schemes* BIKE.PKE *and* HQC.PKE *as shown in Fig. 7 and Fig. 9, respectively. The following statements hold:*

1. *The PKE scheme* $\mathsf{T}[\text{BIKE.PKE}, \mathsf{H}_\mathsf{T}]$ *is* SCFR-CPA *secure.*
2. *The PKE scheme* $\mathsf{T}_{\text{HQC}}[\text{HQC.PKE}, \mathsf{H}_\mathsf{T}]$ *is* SCFR-CPA *secure.*

Proof. We start with a general observation regarding the SCFR-CPA security of a transformed PKE scheme. Subsequently, we cover the concrete cases for BIKE.PKE and HQC.PKE. Consider an adversary $\mathcal{A}$ against $\mathsf{X}[\text{PKE}]$ that wins the game SCFR-CPA. That means, given $pk, \overline{pk}$, which stem from honestly generated key pairs (pk, sk) and $(\overline{pk}, \overline{sk})$, $\mathcal{A}$ outputs c such that

$$m := \mathsf{X}[\text{PKE}].\text{DEC}(sk, c) = \mathsf{X}[\text{PKE}].\text{DEC}(\overline{sk}, c) \neq \bot\,. \tag{1}$$

1. Since BIKE deploys the T transform (cf. Fig. 2), Eq. (1) implies

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

This translates to

$$(e_0 + e_1 h, m \oplus \mathsf{H}_L(e_0, e_1)) = (e_0 + e_1 \overline{h}, m \oplus \mathsf{H}_L(e_0, e_1))$$

where $pk = h$, $\overline{pk} = \overline{h}$, and (e_0, e_1) is derived from $\mathsf{H}_\mathsf{T}(m)$. However, $e_0 + e_1 h$ and $e_0 + e_1 \overline{h}$ differ with overwhelming probability as h and $\overline{h}$ are randomly sampled during an honest key generation[20], i.e., $h \neq h$ and thus $e_0 + e_1 h \neq e_0 + e_1 \overline{h}$ holds with overwhelming probability. This yields a contradiction and thus there cannot be a successful SCFR-CPA adversary against BIKE.

2. Since HQC deploys the T_{HQC} transform (cf. Fig. 11) and $c = (\mathbf{u}, \mathbf{v}, salt)$, Eq. (1) implies

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

This translates to:

$$(\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2, \text{TRUNC}(m\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e}, \ell)) = (\mathbf{r}_1 + \overline{\mathbf{h}}\mathbf{r}_2, \text{TRUNC}(m\mathbf{G} + \bar{\mathbf{s}}\mathbf{r}_2 + \mathbf{e}, \ell))$$

[20] This holds by our assumption regarding the distribution D described in Section 4.1.

where $pk = (\mathbf{h}, \mathbf{s})$, $\overline{pk} = (\overline{\mathbf{h}}, \overline{\mathbf{s}})$, and $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2)$ is derived from $\mathsf{H}_\mathsf{T}(m, pk, salt)$. We can distinguish between whether $(\mathbf{r}_1, \mathbf{r}_2) = (\overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2)$ or $(\mathbf{r}_1, \mathbf{r}_2) \neq (\overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2)$. In the first case, $\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \neq \mathbf{r}_1 + \overline{\mathbf{h}}\mathbf{r}_2$ holds with overwhelming probability. Same is true for the second case, as $(\mathbf{r}_1, \mathbf{r}_2) \neq (\overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2)$ are generated using the random oracle H_T and hence the two random values $\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$ and $\mathbf{r}_1 + \overline{\mathbf{h}}\mathbf{r}_2$ differ with overwhelming probability. Note that in both cases we use that $\mathbf{h}$ and $\overline{\mathbf{h}}$ are chosen randomly in honest HQC key generations. □

Remark 32. The above theorem shows SCFR-CPA security of the base PKEs underlying BIKE and HQC/HQC*, however, the proof easily extends to the stronger notion SCFR-LEAK (hence also to SCFR-CCA). The notion SCFR-LEAK differs from SCFR-CPA only in the fact that the adversary is given additionally the secret keys. Note that this does not influence the above proof, as the crucial steps $e_0 + e_1 h \neq e_0 + e_1 \overline{h}$ (for BIKE) and $\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2 \neq \mathbf{r}_1 + \overline{\mathbf{h}}\mathbf{r}_2$ (for HQC/HQC*) only depend on the fact that the public keys are honestly generated and not the choice of ciphertext by the adversary. In particular, the adversary having knowledge of the secret key does not influence the proof.

Proposition 33. *Consider the public-key encryption schemes* BIKE.PKE *and* HQC.PKE *as shown in Fig. 7 and Fig. 9, respectively. The following statements hold:*

1. BIKE.PKE *fulfills* T*-restricted non-rigidity.*
2. HQC.PKE *fulfills* T_{HQC}*-restricted non-rigidity.*

Proof. Consider two honestly generated key pairs (pk, sk) and $(\overline{pk}, \overline{sk})$ and a randomly chosen message m. To prove X-restricted non-rigidity we have to check that for $c \leftarrow\!\!\$\ \mathsf{X}[\text{PKE}].\text{ENC}(pk, m)$, we have

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

with overwhelming probability.

1. For BIKE we have $(pk, sk) = (h, (h_0, h_1))$, $(\overline{pk}, \overline{sk}) = (\overline{h}, (\overline{h}_0, \overline{h}_1))$ and compute

$$c = (c_0, c_1) \coloneqq \text{BIKE.PKE.ENC}(pk, m; \mathsf{H}_\mathsf{T}(m)) = (e_0 + e_1 h, m \oplus \mathsf{H}_L(e_0, e_1))$$

for (e_0, e_1) derived from $\mathsf{H}_\mathsf{T}(m)$. To check T-restricted non-rigidity we decrypt c using $\overline{sk}$, i.e., compute the message

$$\overline{m} = \text{BIKE.PKE.DEC}(\overline{sk}, c) = c_1 \oplus \mathsf{H}_L(\overline{e}_0, \overline{e}_1)\,,$$

for $(\overline{e}_0, \overline{e}_1) = \text{DECODER}(c_0\overline{h}_0, \overline{h}_0, \overline{h}_1)$, re-encrypt $\overline{m}$ under $\overline{pk}$, and compare the result to c. Hence consider

$$\text{BIKE.PKE.ENC}(\overline{pk}, \overline{m}; \mathsf{H}_\mathsf{T}(\overline{m})) = (\overline{e}_0 + \overline{e}_1\overline{h}, \overline{m} \oplus \mathsf{H}_L(\overline{e}_0, \overline{e}_1))$$

for $(\overline{e}_0, \overline{e}_1)$ derived from $\mathsf{H}_\mathsf{T}(\overline{m})$. Note that $(\overline{e}_0 + \overline{e}_1 h, \overline{m} \oplus \mathsf{H}_L(\overline{e}_0, \overline{e}_1))$ differs from c with overwhelming probability as $\overline{e}_0 + \overline{e}_1\overline{h} \neq e_0 + e_1 h$: If $m = \overline{m}$, the values

of (e_0, e_1) and $(\overline{e}_0, \overline{e}_1)$ agree, however, h and $\overline{h}$ are randomly sampled during an honest key generation, i.e., $h \neq \overline{h}$ and thus $e_0 + e_1 h \neq \overline{e}_0 + \overline{e}_1 \overline{h}$ holds with overwhelming probability. Same is true if $m \neq \overline{m}$, as (e_0, e_1) and $(\overline{e}_0, \overline{e}_1)$ are then derived from the distinct random values $\mathsf{H}_{\mathsf{T}}(m) \neq \mathsf{H}_{\mathsf{T}}(\overline{m})$, i.e., $e_0 + e_1 h$ and $\overline{e}_0 + \overline{e}_1 \overline{h}$ are two random values that hence differ with overwhelming probability. 2. For Hqc we have $(pk, sk) = ((\mathbf{h}, \mathbf{s}), (\mathbf{x}, \mathbf{y}))$, $(\overline{pk}, \overline{sk}) = ((\overline{\mathbf{h}}, \overline{\mathbf{s}}), (\overline{\mathbf{x}}, \overline{\mathbf{y}}))$ and compute $c = (\mathbf{u}, \mathbf{v}, salt)$ for a random salt $salt$ and

$$\begin{aligned}(\mathbf{u}, \mathbf{v}) &:= \textsc{Hqc.Pke.Enc}(pk, m; \mathsf{H}_{\mathsf{T}}(m, pk, salt)) \\ &= (\mathbf{r}_1 + \mathbf{h}\mathbf{r}_2, \textsc{Trunc}(m\mathbf{G} + \mathbf{s}\mathbf{r}_2 + \mathbf{e}, \ell))\end{aligned}$$

for $(\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2)$ derived from $\mathsf{H}_{\mathsf{T}}(m, pk, salt)$. To check $\mathsf{T}_{\textsc{Hqc}}$-restricted non-rigidity we decrypt c using $\overline{sk}$, i.e., compute the message

$$\overline{m} = \textsc{Hqc.Pke.Dec}(\overline{sk}, c) = \textsc{Decoder}(\mathbf{v} - \mathbf{u}\overline{\mathbf{y}})\,,$$

re-encrypt $\overline{m}$ under $\overline{pk}$ (using the salt contained in c), and compare the result to c. Hence consider

$$\textsc{Hqc.Pke.Enc}(\overline{pk}, \overline{m}; \mathsf{H}_{\mathsf{T}}(\overline{m}, \overline{pk}, salt)) = (\overline{\mathbf{r}}_1 + \overline{\mathbf{h}}\overline{\mathbf{r}}_2, \textsc{Trunc}(\overline{m}\mathbf{G} + \overline{\mathbf{s}}\overline{\mathbf{r}}_2 + \overline{\mathbf{e}}, \ell))$$

for $(\overline{\mathbf{r}}_1, \overline{\mathbf{r}}_2)$ derived from $\mathsf{H}_{\mathsf{T}}(\overline{m}, \overline{pk}, salt)$. Just as we did for Bike, we can deduce that $(\overline{\mathbf{r}}_1 + \overline{\mathbf{h}}\overline{\mathbf{r}}_2, \textsc{Trunc}(\overline{m}\mathbf{G} + \overline{\mathbf{s}}\overline{\mathbf{r}}_2 + \overline{\mathbf{e}}, \ell))$ differs from c as $\overline{\mathbf{r}}_1 + \overline{\mathbf{h}}\overline{\mathbf{r}}_2 \neq \mathbf{r}_1 + \mathbf{h}\mathbf{r}_2$. □

Having established Proposition 31 and Proposition 33, 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 for Hqc*.

Theorem 34. *The key-encapsulation mechanisms* Bike *and* Hqc* *as shown in Fig. 8 and Fig. 10 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 16 proves Classic-McEliece to be MAL-BIND-K-CT secure. Further, Ml-Kem is not MAL-BIND-K,CT-PK secure by Theorem 18 (the attack relies exclusively on invalid ciphertexts for which Ml-Kem behaves like $\mathsf{FO}^{\not\perp}$) and MAL-BIND-K,PK-CT secure by Theorem 24 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 a $\mathcal{A}$ 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.

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, National Institute of Standards and Technology, 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, Heidelberg, February 2010.

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.

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 *PQCrypto 2024*, 2024.

AHK+22. Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. Post-quantum multi-recipient public key encryption. Cryptology ePrint Archive, Report 2022/1046, 2022. https://eprint.iacr.org/2022/1046.

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, Heidelberg, December 2001.

BCC+24. Ritam Bhaumik, Bishwajit Chakraborty, Wonseok Choi, Avijit Dutta, Jérôme Govinden, and Yaobin Shen. The committing security of MACs with applications to generic composition. In *CRYPTO 2024*, 2024.

BCDD+24. Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-Wing: The hybrid KEM you've been looking for. *IACR Communications in Cryptology*, 1(1), 2024.

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, EuroS&P 2018, London, United Kingdom, April 24-26, 2018, pages 353–367. IEEE, 2018.

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, Heidelberg, May / June 2022.

BH24. Mihir Bellare and Viet Tung Hoang. Succinctly-committing authenticated encryption. In *CRYPTO 2024*, 2024.

BP18. Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. 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.

CDM23. Cas Cremers, Alexander Dax, and Niklas Medinger. Keeping up with the KEMs: Stronger security notions for KEMs and automated analysis of KEM-based protocols. *IACR Cryptol. ePrint Arch.*, 2023:1933, 2023. v1.1.0.

CFGI+23. Yu Long Chen, Antonio Flórez-Gutiérrez, Akiko Inoue, Ryoma Ito, Tetsu Iwata, Kazuhiko Minematsu, Nicky Mouha, Yusuke Naito, Ferdinand Sibleyras, and Yosuke Todo. Key committing security of AEZ and more. In *ToSC 2023*, 2023.

CR22. John Chan and Phillip Rogaway. On committing authenticated-encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part II*, volume 13555 of *LNCS*, pages 275–294. Springer, Heidelberg, September 2022.

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, Heidelberg, December 2003.

DFF24. Samed Düzlü, Rune Fiedler, and Marc Fischlin. BUFFing FALCON without increasing the signature size. *IACR Cryptol. ePrint Arch.*, 2024:710, 2024.

DFG23. Jean Paul Degabriele, Marc Fischlin, and Jérôme Govinden. The indifferentiability of the duplex and its practical applications. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VIII*, volume 14445 of *LNCS*, pages 237–269. Springer, Heidelberg, December 2023.

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. *IACR Cryptol. ePrint Arch.*, 2024:793, 2024.

DFHS24. Jelle Don, Serge Fehr, Yu-Hsuan Huang, and Patrick Struck. On the (in)security of the BUFF transform. In *CRYPTO 2024*, 2024.

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, Heidelberg, August 2018.

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.

DMVA23. Joan Daemen, Silvia Mella, and Gilles Van Assche. Committing authenticated encryption based on SHAKE. *IACR Cryptol. ePrint Arch.*, 2023:1494, 2023.

FG24. Rune Fiedler and Felix Günther. Security analysis of Signal's PQXDH handshake. *IACR Cryptol. ePrint Arch.*, 2024:702, 2024.

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, Heidelberg, August 1999.

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, Heidelberg, May / June 2022.

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, Heidelberg, November 2017.

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.

KSW23. Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Committing AE from sponges - security analysis of the NIST LWC finalists. *IACR Cryptol. ePrint Arch.*, 2023:1525, 2023.

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.

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, Heidelberg, April 2023.

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.

NIST17. National Institute of Standards and Technology. Post-quantum cryptography standardization process. https://csrc.nist.gov/projects/post-quantum-cryptography, 2017.

NIST22. National Institute of Standards and Technology. 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.

NIST23. National Institute of Standards and Technology. Module-Lattice-based Key-Encapsulation Mechanism Standard. https://doi.org/10.6028/NIST.FIPS.203.ipd, 2023. Draft.

NIST24. National Institute of Standards and Technology. Accordion mode. https://csrc.nist.gov/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd, 2024.

NSS23. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Commiting security of Ascon: Cryptanalysis on primitive and proof on mode. In *ToSC 2023 (4)*, 2023.

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. *IACR Cryptol. ePrint Arch.*, 2024:523, 2024.

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, Heidelberg, November / December 2011.

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

SW24. Patrick Struck and Maximiliane Weishäupl. Constructing committing and leakage-resilient authenticated encryption. In *ToSC 2024 (1)*, 2024.

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, Heidelberg, May / June 2022.