Faster Lattice-Based KEMs via a Generic Fujisaki-Okamoto Transform Using Prefix Hashing

Julien Duman
Ruhr Universität Bochum
Germany
julien.duman@rub.de

Kathrin Hövelmanns
TU Eindhoven
The Netherlands
kathrin@hoevelmanns.de

Eike Kiltz
Ruhr Universität Bochum
Germany
eike.kiltz@rub.de

Vadim Lyubashevsky
IBM Research Europe, Zurich, Switzerland
vad@zurich.ibm.com

Gregor Seiler
ETH Zürich and IBM Research Europe, Zurich
Switzerland
gseiler@inf.ethz.ch

ABSTRACT

Constructing an efficient CCA-secure KEM is generally done by first constructing a passively-secure PKE scheme, and then applying the Fujisaki-Okamoto (FO) transformation. The original FO transformation was designed to offer security in a single user setting. A stronger notion, known as multi-user security, considers the attacker's advantage in breaking one of many user's ciphertexts. Bellare et al. (EUROCRYPT 2020) showed that standard single user security implies multi-user security with a multiplicative tightness gap equivalent to the number of users.

To obtain even more confidence in the security of KEMs in the multi-user setting, it is a common design paradigm to also "domain separate" the random oracles of each user by including his public key as an input to the hash function. We are not aware of any formal analysis of this technique, but it was at least informally thought to be a computationally cheap way to add security. This design principle was carried over into the FO transformations used by several schemes in the NIST post-quantum standardization effort – notably the lattice-based schemes Kyber and Saber, which are two of the four KEM finalists.

In this work, we formally analyze domain separation in the context of the FO transformation in the multi-user setting. We first show that including the public key in the hash function is indeed important for the tightness of the security reductions in the ROM and the QROM. At the same time, we show that including the *entire* public key into the hash function is unnecessarily wasteful – it is enough to include just a small (e.g. 32 byte) unpredictable part of the key to achieve the same security. Reducing the input of the hash function results in a very noticeable improvement in the running time of the lattice-based KEMs. In particular, using this generic transform results in a 2X - 3X speed-up over the current (Round 3) key generation and encapsulation procedures in Kyber, and up to a 40% improvement in the same functions in Saber.

KEYWORDS

FO Transform; QROM; Key Exchange; Lattice Cryptography; Implementation

1 INTRODUCTION

Security definitions for public key encryption (PKE) schemes are generally stated in the *single-user* setting. In this setting, one party publishes its public key, which allows other parties to send it encrypted messages of their choice. For practical applications, however, this was shown to not be enough. In particular, Håstad showed [20] that if there are multiple receivers, each with a different public key, and a sender encrypts the same message to all of them, then for certain RSA parameter settings everyone will be able to recover the message.

Håstad's simple attack against the basic RSA cryptosystem demonstrated that schemes can be secure in the single user setting, but completely broken in the multi-user one. In the multi-user and multi-challenge (multi-user/challenge) setting, it is furthermore required that the encryptions of many plaintexts, possibly encrypted to different public keys, remain hidden secure. Furthermore, NIST lists resistance to multi-key attacks as a "desirable property" in their Call for Proposals for the post-quantum standardization process [31, Section 4.A.6]. Luckily, Bellare et al. [6] showed that for CPA and CCA-secure schemes, security in a single-user and single-challenge setting implies security (with a multiplicative loss in the number of parties times the number of challenges) in the multi-user/challenge setting as well.

Constructing an efficient Key Encapsulation Mechanism (KEM) with security against chosen-ciphertext attacks (CCA) is generally done by first constructing a passively-secure PKE scheme, and then applying the Fujisaki-Okamoto (FO) transformation [16, 17, 21]. To achieve multi-user security in the random oracle model, it is a common design paradigm to use "domain separation" so that in the random oracle model, the parties all appear to be using different random functions. The simple way of achieving this in the context of the FO transformation is to always include the public key as an additional input to the cryptographic hash function that is being modeled as a random oracle. The other stated reason for including the public key in the hash is an informally-defined notion of making the KEM "contributory" – that is, both parties affect the shared key.

For classical schemes based on the hardness of the discrete logarithm problem, including the public key as an extra input to the hash function has virtually no effect on the running time of the full protocol. The reason is that the key is short (e.g. 32 bytes), and the hash function contributes a negligible amount of computation compared to the much more expensive group operations such as exponentiation. So even if unnecessary for practical security, adding the public key into the hash function does not have any measurable negative effects on the scheme.

The above-mentioned domain separation technique was carried over to schemes in the ongoing NIST post-quantum standardization process. For example, Kyber [12] and Saber [14], which are two of the four KEM finalists in the NIST quantum-safe competition, use this method with the explicit purpose of protecting against multi-user attacks. Even though it was at least informally thought to be more secure, we are not aware of any formal security analysis of it.[1]

1.1 Our Results

In this paper we formally analyze domain separation in the context of the FO transformation. Our results are twofold. First, we observe that there are good reasons for including the public key into the hash. We show that hashing the public key results in a tighter reduction in a multi-user/multi-challenge setting when converting a CPA-secure scheme into a CCA-secure one, than if one were to directly apply the hybrid argument from [6] to the FO transformation. And for schemes that additionally have a small correctness error, the tightness in the reduction is potentially even more improved. We additionally give a proof in the QROM which is significantly tighter that what one would trivially obtain from the hybrid approach of [6]. This proof also appears to use (but in a different way) the fact that each party uses a different random function.

Our second, and main, result is that even though there are reasons to hash the public key, hashing the *entire* public key (as is currently done) is unnecessarily wasteful. The sizes of public keys in lattice-based schemes ($\approx$ 1KB) are noticeably larger than the 32 byte keys used in the discrete log setting; so the hash function that takes the public key as input is now approximately an order of magnitude slower. At the same time, the underlying lattice-based CPA-secure scheme is significantly *more* computationally efficient than its discrete logarithm counterpart. When compounded, these two properties result in the hash function being a very significant contributor to the total running time of the resulting CCA-secure scheme.

FO Transformation with Prefix Hashing. Our new variant of the FO transformation, $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$, transforms a passively secure PKE scheme into an actively secure KEM. It uses "implicit rejection" and "prefix hashing": Implicit rejection (indicated by the $\not\perp$ symbol in $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$) means that decapsulation of an invalid ciphertext deterministically outputs a pseudoranom key; prefix hashing (indicated by the $_{\mathsf{ID}(\mathrm{id}),m}$ in $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$) means that we feed a small (e.g. 32 byte), unpredictable part of the public key into the hash function. $\mathsf{FO}^{\not\perp}_{m}$ also uses implicit rejection but does not include any part of pk in the hash function. Compared to feeding the whole public-key into the hash function, our new transformation significantly reduces the running time of the scheme. In the ROM we prove that multi-user/challenge CPA-security of PKE tightly implies multi-user/challenge CCA-secure KEM. As an additional result, we give a reduction for multi-user/challenge security in the QROM which is tighter than the previously-known results which stem from the generic Bellare et al. result [6]. We remark that ThreeBears [18] (a Round 2 candidate of the NIST post-quantum competition) implicitly uses an FO transformation with prefix hashing and proves multi-user/challenge CCA-security in the QROM [19]. It is unclear, however, if the specific proof carries over to the general FO transformation.

[1]This is in contrast to digital signatures where domain separation provably helps improving security in the multi-user setting. See, e.g., [7, 9].

Impact on Kyber. Instantiating our transformation $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ with the CPA-secure Kyber PKE scheme, the key generation and encryption of Kyber is reduced by 30-56% and 47-66%, respectively; and the security of the CCA-scheme in the ROM and QROM is improved as in Fig. 1. So we are now in the same situation with lattice schemes as we are in the discrete logarithm setting. There is no longer a theory vs. practice trade-off required for achieving multi-user/challenge security – hashing the (partial) public key leads to tighter security reductions and is computationally very cheap to implement. Since Kyber is currently the most efficient – as measured by the running time of key generation, encapsulation, and decapsulation procedures [8] – of all KEMs (quantum-safe and non-quantum-safe), our new transformation also improves on how fast a secure KEM can be.

1.2 Impact on Concrete Security

In the Fujisaki-Okamoto transform, a hash function F (modeled as a random oracle) is mainly used to derive the PKE randomness r and the symmetric KEM key K from a random message m. In the following table, we define three variants of the "implicit rejection" Fujisaki-Okamoto transformation, depending on which parts of pk it includes in the hash function F.

Transformation	Use of hash F
$\mathsf{FO}^{\not\perp}_{m}$ [21]	$(K,r)=\mathsf{F}(m)$
$\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ (§3)	$(K,r)=\mathsf{F}(\mathsf{ID}(pk),m)$
$\mathsf{FO}^{\not\perp}_{pk,m}$ (§3)	$(K,r)=\mathsf{F}(pk,m)$

The formal definitions of $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ and $\mathsf{FO}^{\not\perp}_{pk,m}$ are given in Fig. 4 of Section 3. $\mathsf{FO}^{\not\perp}_{m}$ was formally analyzed in [21]. We remark that [21] also considered a variant (called $\mathsf{FO}^{\not\perp}_{m,c}$) where the ciphertext c is included in F. Follow-up work [10] showed that hashing c is not necessary so we do not further consider it here.

Fig. 1 compares multi-user/challenge security bounds in the ROM/QROM of $\mathsf{FO}^{\not\perp}_{m}$, $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$, and $\mathsf{FO}^{\not\perp}_{pk,m}$. The ROM bounds for $\mathsf{FO}^{\not\perp}_{m}$ are new bounds obtained in this work. They improve on the naive "hybrid bound" $\mathsf{Adv}^{(n,q_{\mathsf{C}})\text{-}\mathsf{IND}\text{-}\mathsf{CCA}}_{\mathsf{KEM}} \le nq_{\mathsf{C}} \cdot \mathsf{Adv}^{(n,q_{\mathsf{C}})\text{-}\mathsf{IND}\text{-}\mathsf{CCA}}_{\mathsf{KEM}}$ $\le nq_{\mathsf{C}} \cdot (\mathsf{Adv}^{\mathsf{IND}\text{-}\mathsf{CPA}}_{\mathsf{PKE}} + q_{\mathsf{F}}\delta(1))$ obtained by applying the Bellare et al. hybrid argument to the single user/challenge CCA-bounds of [21]. Unfortunately, we were not able to improve the hybrid bounds in the QROM setting for $\mathsf{FO}^{\not\perp}_{m}$. The bounds for $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ and $\mathsf{FO}^{\not\perp}_{pk,m}$ are new bounds obtained in this work, again improving on the hybrid bounds. Our proofs in the ROM rely on techniques of [21], while the ones in the QROM also rely on techniques of [10, 23, 26].

First we note that the bounds for $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ and $\mathsf{FO}^{\not\perp}_{pk,m}$ are exactly the same (for prefixes with sufficient min-entropy l). Hence from a security perspective, it does not seem to make much sense to include the *entire* public key in the hash function. Hence in what follows we will only consider the prefix-hash variant $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$. How the bounds for $\mathsf{FO}^{\not\perp}_{m}$ and $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ compare to each other, depends on the relation between $\delta(n)$ and δ and $\mathsf{Adv}^{(n,q_{\mathsf{C}})\text{-}\mathsf{IND}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}$ and $\mathsf{Adv}^{\mathsf{IND}\text{-}\mathsf{CPA}}_{\mathsf{PKE}}$, respectively.

Figure 1: Multi-user/challenge security advantages $\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}$ in the ROM/QROM (simplified) for the Fujisaki-Okamoto variants $\mathsf{FO}^{\not\perp}_{m}$, $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$, and $\mathsf{FO}^{\not\perp}_{pk,m}$, as described in the text. Here $\delta(n)$ is the n-user correctness error of PKE, q_C is the number of challenges, q_F is the number of (Q)ROM queries, and l is the min-entropy of $\mathsf{ID}(pk)$.

FO variant	$\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}$ (ROM)	$\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}$ (QROM)
$\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ (Th. 3.1+3.2)	$\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} + q_\mathsf{F}\delta(n) + \frac{n^2}{2^\ell}$	$\sqrt{q_\mathsf{F}\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}} + q_\mathsf{F}^2\delta(n) + \frac{n^2}{2^\ell}$
$\mathsf{FO}^{\not\perp}_{pk,m}$ (Th. 3.1+3.2)	$\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} + q_\mathsf{F}\delta(n)$	$\sqrt{q_\mathsf{F}\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}} + q_\mathsf{F}^2\delta(n)$
$\mathsf{FO}^{\not\perp}_{m}$ (Th. 3.1+[6, 21])	$\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} + n \cdot q_\mathsf{F}\delta(1)$	$nq_\mathsf{C} \cdot (\sqrt{q_\mathsf{F}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}} + q_\mathsf{F}^2\delta(1))$

The n-user correctness error $\delta(n)$. $\delta(n)$ is defined as the n-user correctness error, i.e., the probability that one message induces a decryption error for one of n independent public-keys. Informally (see Section 2.1 for a formal definition), the adversary, who is given n public and *secret* keys, chooses a message and one of the public keys under which the message is to be encrypted; he wins if the ciphertext then induces a decryption failure. The trivial bounds are $\delta(1) \leq \delta(n) \leq n\delta(1)$.

There is also very strong evidence that for most natural lattice-based schemes, $\delta(n) < n\delta(1)$. The decryption error of a basic LWE-based encryption scheme encrypting one bit with secret $\vec{s}$ and encryption randomness $\vec{r}$ is

$$\Pr_{\vec{s},\vec{r}\leftarrow\psi^m}[|\vec{s}\cdot\vec{r}| > \lfloor q/4 \rfloor - 1] .$$

If the coefficients of randomness $\vec{r}$ were continuous gaussians, rather than e.g. discrete binomials, then one could prove that the optimal strategy for the adversary is to "attack" the public key whose corresponding secret key has the largest norm. Then the probability that we are interested in is

$$\Pr_{\vec{s}_1,\ldots,\vec{s}_n,\vec{r}\leftarrow\psi^m}\left[|\vec{s}_i\cdot\vec{r}| > \lfloor q/4 \rfloor - 1 \colon i := \arg\max_{1\leq j\leq n} \|\vec{s}_j\|\right].$$

In schemes where the randomness is discrete, this is almost certainly still the best strategy. So, for example, in schemes where all the secret keys are prescribed to have the same norm, it makes no difference which public key to attack, and therefore we would have $\delta(n) \approx \delta(1)$. In Kyber and Saber, where each coefficient is chosen independently, there will be secret keys with larger norms, and so $\delta(n) > \delta(1)$. Nevertheless, because the norm of the secret keys has a tight concentration when the coefficients follow the binomial distribution, we will still have $\delta(n) < n \cdot \delta(1)$.

In Table 1, we give the exact values for $\delta(n)$ illustrating this phenomenon for an LWE scheme that has secret vectors consisting of $-1/0/1$ coefficients . For example, note that for $n = 2^{30}$, we have $n\delta(1) = 2^{-148}$, whereas $\delta(n) = 2^{-161}$. Computing exact $\delta(n)$ for Kyber and Saber would be more computationally exhaustive because there are many more possibilities for how the vector with the largest norm looks like when the coefficients are larger. But the secret key norm will still be tightly concentrated around its expected value and we will thus still have $\delta(n) < n\delta(1)$.

The n-user CPA advantage $\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$. By a hybrid argument [6] one obtains the trivial bounds

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \leq \mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \leq nq_\mathsf{C} \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} .$$

Table 1: $\delta(n)$ for a basic LWE encryption scheme over $\mathbb{Z}_q$, for $q = 1409$, where the secret $\vec{s} \leftarrow \psi^{2048}$, where $\psi(0) = 1/2, \psi(\pm 1) = 1/4$, and the public key is $\mathbf{A} \leftarrow \mathbb{Z}_q^{1024\times1024}, \vec{t} = [\mathbf{A}\ \mathbf{I}]\cdot\vec{s} \bmod q$. The ciphertext is constructed as $\vec{u} = [\mathbf{A}^T\ \mathbf{I}]\cdot\vec{r}^T, v = [\vec{t}^T\ 0^{1024}]\cdot\vec{r} + e + \lfloor q/2 \rceil \mu$, where $\vec{r} \leftarrow \psi^{2048}, e \leftarrow \psi$.

n	1	2^{10}	2^{20}	2^{30}	2^{40}	2^{50}
$\log(\delta(n))$	-178	-170	-165	-161	-158	-155

For schemes based on prime-order groups (eg., ElGamal) we actually have $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \approx \mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$ by the well known random self reducibility of Diffie-Hellman tuples.

For lattice-based schemes, we do not have self-reductions that allow us to directly conclude that $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \approx \mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$ because we do not know of a way to create extra samples from the same distribution (i.e. either the MLWE one or uniform) even if these samples are allowed to have completely distinct secrets. Nevertheless, if we assume the hardness of MLWE as originally defined for the purpose of worst-case to average-case reductions [28, 30, 32] where the number of samples (using the same secret) is unlimited, then we can show that $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \approx \mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$. In particular, if we assume that distinguishing $\{(\mathbf{A}_i, \mathbf{A}_i\cdot\vec{\mathbf{s}}_i + \vec{\mathbf{e}}_i)\}_{1\leq i\leq k}$ from uniform is hard for $k = \max(n, q_\mathsf{C})$,[2] then using the transformation from [2], one can argue that distinguishing the n public keys from uniform ones is as hard as MLWE. After replacing the public keys with random values, simulating the q_C queries to the n public keys can again be done with access to q_C samples of an MLWE problem and the transformation from [2]. And thus we have $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}} \leq 2\cdot\mathsf{Adv}^{\mathsf{MLWE}}$, which is also what we based the hardness of the $\mathsf{IND\text{-}CPA}$ scheme on. In short, we believe that in practice the MLWE problem with k samples is no easier than with 1 sample, and then the bound in our reduction is significantly tighter than the previously known one. And independently of the relationship between the number of MLWE samples, the new QROM bound for $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ is also noticeably better than the one for $\mathsf{FO}^{\not\perp}_{m}$.

[2]Unless k is so large that the Arora-Ge [3] attack applies, it is not known that seeing more samples makes the MLWE problem any easier in practice. It should also be noted that even if such an attack existed, it would still not necessarily apply to increasing $\mathsf{Adv}^{(n,q_\mathsf{C})\text{-}\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}$. This is because the secrets in the (n, q_C)-IND-CPA game are all *distinct*, and so it is a plausibly much harder problem than MLWE with many samples for the same secret. In particular, we do not know of any better algorithm for solving MLWE with many samples if the secrets are small and different for every sample.

Prefix collisions. We note that given a target public-key pk, it might be possible for an adversary to efficiently compute a different public key pk' and a matching sk' such that $\mathsf{ID}(pk) = \mathsf{ID}(pk')$. This "attack" does not violate any of our security claims, nor do we see any practical attack that could exploit this property.

But we stress that $\mathsf{ID}(pk)$ should never be used as a fingerprint. Collision-prone public-key fingerprints are known to be insecure and have been successfully exploited in practice, for example in the context of PGP [29].

1.3 Impact on Efficiency

We now measure the effect of replacing the original FO transform used in the Kyber and Saber KEMs with our new transform $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ (see Figure 4). We implemented $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$ on top of the underlying CPA-secure encryption schemes in the official AVX2-optmized implementations of Kyber and Saber, and performed benchmarks on an Intel Skylake CPU. Concretely, the numbers in Tables 2 and 3 are the medians of the cycle counts of 10000 executions of the key generation (**K**), encapsulation (**E**), and decpasulation (**D**) operations for either the original FO transform or the new one from this work.

For the 32-byte prefix $\mathsf{ID}(pk)$ of the public key in Kyber and Saber one can take the seed ρ that is already of size 32 bytes and uniformly random in these schemes. Alternatively, for Kyber, the first 33 bytes of the bitpacked representation of the polynomial vector $\vec{\mathbf{t}}$ in the public key can also be used. This is sufficient since $\vec{\mathbf{t}}$ is given in the NTT basis and contains the independently random short error vector $\vec{\mathbf{e}}$ as an additive term. It follows from a simple Fourier analysis computation as in [4, 13] that the first few NTT coefficient of $\vec{\mathbf{e}}$ are close to uniform modulo $q = 3329$. Concretely, the first 22 coefficients have more than 256 bit of entropy and they occupy 33 bytes in the bitpacking that uses 12 bits per coefficient. Taking the prefix from $\vec{\mathbf{t}}$ instead of ρ has the advantage that this is still secure in the slightly modified but fully compatible variant of Kyber where users re-use ρ and hence the MLWE matrix $\mathbf{A}$.

We observe that one obtains significant speed improvements throughout all parameter sets and variants. For example, the key generation and encapsulation of Kyber-512-90s are 56% and 66% faster, respectively, when using $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$.[3] There is no speed-up in decryption since in the original Kyber and Saber CCA transforms avoid the expensive full public key hash needed for the re-encryption during decapsulation by pre-computing this hash during key generation and storing it in the secret key. But there is still an expensive hashing operation in decapsulation that is responsible for a significant portion of the runtime. Namely, the hash for $\tilde{K}$ in line 11 of the decapsulation algorithm in Figure 4, which includes the full ciphertext as input. The fake key $\tilde{K}$ is only needed when re-encryption fails, so one could achieve a significant speed-up also in decapsulation by only computing $\tilde{K}$ in the case of decryption failure. The resultion FO transform would effectively be a middle ground between implicit and explicit rejection as it would leak rejection only via a timing side-channel. So in an application where timing side-channels are absolutely of no concern, this faster and fully compatible transformation can be used. It is well known that explicit rejection is secure in the ROM [21]. One can also use the techniques from this paper to show improved bounds in the multi-user/challenge setting. Moreover, Theorem 6.1 in [15] now gives a security proof for explicit rejection in the QROM, but unfortunately only with much worse security bounds. Given the recent progress in this area, it is natural to think that there will be better bounds in the future, decreasing the gap between implicit and explicit rejection. For these reasons we have also benchmarked this soft variant of explicit rejection. The resulting cycle counts are included in brackets in Tables 2 and 3 and we see that one would for example achieve a speed-up of 45% for the Kyber512-90s decapsulation runtime.

Our new transform also has a noticeably larger effect on Kyber than Saber because the CPA-secure scheme underlying Kyber is more efficient than its Saber counterpart. Thus the running time of Kyber with the original FO transform was much more dominated by hashing.

[3]It should be mentioned again that another reason that encapsulation has a larger increase is that we removed an additional hash of the ciphertext. This was already shown to be intuitively unnecessary in [10], and in this work we show that it is also unnecessary in the multi-user setting.

2 PRELIMINARIES

For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. For a set S, $|S|$ denotes the cardinality of S. For a finite set S, we denote the sampling of a uniform random element x by $x \xleftarrow{\$} S$. The min entropy of a discrete random variable X is defined as $H_\infty(X) = -\log(\max_x \Pr[X = x])$.

2.1 Cryptographic Definitions

Public-Key Encryption. A public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms, and a finite message space $\mathcal{M}$. The key generation algorithm Gen outputs a key pair (pk, sk), where pk also defines a finite randomness space $\mathcal{R} = \mathcal{R}(pk)$ as well as a ciphertext space C. The encryption algorithm Enc, on input pk and a message $m \in \mathcal{M}$, outputs an encryption $c \xleftarrow{\$} \mathsf{Enc}(pk, m)$ of m under the public key pk. If necessary, we make the used randomness of encryption explicit by writing $c := \mathsf{Enc}(pk, m; r)$, where $r \in \mathcal{R}$. The decryption algorithm Dec, on input sk and a ciphertext c, outputs either a message $m = \mathsf{Dec}(sk, c) \in \mathcal{M}$ or a special symbol $\perp \notin \mathcal{M}$ to indicate that c is not a valid ciphertext.

PKE has n-user correctness error $\delta(n)$ if

$$\mathbb{E}\left[\max_{j\in[n]} \max_{m\in\mathcal{M}} \Pr\left[\mathsf{Dec}(sk_j, \mathsf{Enc}(pk_j, m)) \neq m\right]\right] \leq \delta(n),$$

where the expectation is taken over $((pk_1, sk_1), \ldots, (pk_n, sk_n)) \xleftarrow{\$} (\mathsf{Gen})^n$. For $n = 1$ we obtain the single-user correctness definition $\delta := \delta(1)$ of [21]. Note that $\delta \leq \delta(n) \leq n\delta$. Furthermore, there exists schemes for which either $\delta(n) = \delta$ or $\delta(n) = n\delta$. PKE is weakly γ-spread [15] if

$$\mathbb{E}\left[\max_{m,c} \Pr\left[\mathsf{Enc}_{pk}(m) = c\right]\right] \leq 2^{-\gamma},$$

where the expectation is taken over $(pk, sk) \xleftarrow{\$} \mathsf{Gen}$.

We define n-user/q_C-challenges IND-CPA ("passive") security for PKE in terms of the advantage function of an adversary $\mathcal{A}$:

$$\mathsf{Adv}^{(n,q_C)\text{-IND-CPA}}_{\mathsf{PKE}}(\mathcal{A}) \quad := \quad \left|\Pr[(n, q_C)\text{-IND-CPA}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|,$$

where game (n, q_C)-IND-CPA is defined in Fig. 2.

Table 2: Median Skylake cycle counts of 10000 executions of Kyber and Saber using either the original CCA transform or the improved transform from this work. The "original" version of Kyber and Saber are the Round 3 submissions to the NIST post-quantum standardization process. The decapsulation cycle counts in brackets are for the "soft" explicit rejection variant that leaks rejection via a timing side-channel.

		Kyber			Saber		
NIST Level		Original	This Work	Speed-up	Original	This Work	Speed-up
1	**K**	23562	12883	45%	42169	36220	14%
	E	37144	16981	54%	57831	39232	32%
	D	28595	28529	0%	57780	57806	0%
			(18332)	(36%)		(47516)	(18%)
3	**K**	40487	25272	38%	74577	64180	14%
	E	55726	27624	50%	95958	69304	28%
	D	43553	43442	0%	95388	95301	0%
			(29660)	(32%)		(80847)	(15%)
5	**K**	55770	38815	30%	116178	102101	12%
	E	77011	40692	47%	142034	109203	23%
	D	61470	61473	0%	142957	143090	0%
			(43194)	(30%)		(125589)	(12%)

Table 3: Median Skylake cycle counts of 10000 executions of the 90's variants of Kyber and Saber using either the original CCA transform or the improved transform from this work. The "original" version of Kyber and Saber are the Round 3 submissions to the NIST post-quantum standardization process. The decapsulation cycle counts in brackets are for the "soft" explicit rejection variant that leaks rejection via a timing side-channel.

		Kyber90s			uSaber90s		
NIST Level		Original	This Work	Speed-up	Original	This Work	Speed-up
1	**K**	13994	6224	56%	24557	17294	30%
	E	23069	7894	66%	36544	22363	39%
	D	16917	16959	0%	38156	38014	0%
			(9233)	(45%)		(30579)	(20%)
3	**K**	21783	10995	50%	37511	29881	20%
	E	33534	13137	61%	55436	36282	35%
	D	25014	24957	0%	58395	58405	0%
			(14893)	(40%)		(47889)	(18%)
5	**K**	31576	18834	40%	59169	50796	14%
	E	46881	21404	54%	81187	57920	29%
	D	36190	36165	0%	86142	86359	0%
			(22605)	(38%)		(72879)	(15%)

```
$(n, q_C)$-IND-CPA
for $j \in [n]$
    $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
$\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
$b \xleftarrow{\$} \{0, 1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Chall}}(\vec{pk})$
return $[\![b' = b]\!]$

$\mathsf{Chall}(j, m_0, m_1)$                    / max. $q_C$ queries
return $\mathsf{Enc}(pk_j, m_b)$
```

Figure 2: Game (n, q_C)-IND-CPA for PKE in the n-user/q_C-challenges setting. Set $\mathcal{L}_{M_j}$ describes the set of plaintext messages encrypted to user j.

Key Encapsulation Mechanisms. A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of three algorithms and a finite key space $\mathcal{K}$ similar to a PKE scheme, but Encaps does not take a message as input. The key generation algorithm Gen outputs a key pair (pk, sk), where pk also defines a finite randomness space $\mathcal{R} = \mathcal{R}(pk)$ as well as a ciphertext space C. The encapsulation algorithm Encaps takes as input a public-key pk and outputs a key encapsulation ciphertext c and a key k, that is $(c, k) \xleftarrow{\$} \mathsf{Encaps}(pk)$. The decapsulation algorithm Decaps, on input sk and a ciphertext c, outputs either a key $k = \mathsf{Decaps}(sk, c) \in \mathcal{K}$ or a special symbol $\perp \notin \mathcal{K}$ to indicate that c is not a valid ciphertext.

In terms of KEM's security, we consider the n-user/q_C-challenges IND-CCA advantage function of an adversary $\mathcal{A}$:

$$\mathrm{Adv}_{\mathsf{KEM}}^{(n,q_C)\text{-}\mathsf{IND\text{-}CCA}}(\mathcal{A}) \quad := \quad \left|\Pr[(n, q_C)\text{-}\mathsf{IND\text{-}CCA}_{\mathsf{KEM}}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|$$

```
$(n, q_C)$-IND-CCA
01 for $j \in [n]$
02     $(pk_j, sk_j) \xleftarrow{\$}$ Gen
03 $\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
04 $b \xleftarrow{\$} \{0,1\}$
05 $b' \xleftarrow{\$} \mathcal{A}^{\text{Decaps},\text{Chall}}(\vec{pk})$
06 return $[\![b = b']\!]$

Chall($j$)          / max. $q_C$ queries
07 $(c, K_0) \xleftarrow{\$}$ Encaps$(pk_j)$
08 $K_1 \xleftarrow{\$} \mathcal{K}$
09 $\mathcal{L}_{C_j} := \mathcal{L}_{C_j} \cup \{c\}$
10 return $(c, K_b)$

Decaps$(j, c \notin \mathcal{L}_{C_j})$
11 return Decaps$(sk_j, c)$
```

Figure 3: Game (n, q_C)-IND-CCA for KEM in the n-user/q_C-challenges setting. Set $\mathcal{L}_{C_j}$ describes the set of challenge ciphertexts for user j.

where game (n, q_C)-IND-CCA is defined in Fig. 3.

3 FUJISAKI-OKAMOTO TRANSFORMATION WITH PREFIX HASHING

Let $k, \lambda \in \mathbb{N}$ be integers. Let PKE be a public-key encryption scheme with message space $\mathcal{M}$, public-key space $\mathcal{PK}$, randomness space $\mathcal{R}$, and ciphertext space C. We assume that the two spaces $\mathcal{M}$ and $\{0,1\}^k \times C$ are disjoint. Let ID : $\mathcal{PK} \to \{0,1\}^\gamma$ be a fixed-output length function and let F : $\{0,1\}^* \to \{0,1\}^k \times \mathcal{R}$ be a hash function, where $\mathsf{F}_1(X)$ is defined as the first k bits of $\mathsf{F}(X)$. To PKE, ID, and F we associate the key encapsulation mechanism $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$[PKE, ID, F] as described in Fig. 4. Our transformation is essentially $\mathsf{FO}^{\not\perp}_{m}$ of [21] with the difference that we feed ID(pk) into the hash function F for domain separation. Note that with the identity function ID$(pk) = pk$, we recover $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m} = \mathsf{FO}^{\not\perp}_{pk,m}$; with ID$(pk) = \varepsilon$, we recover $\mathsf{FO}^{\not\perp}_{m}$.

Note that computing the pseudo-random key in line 11 of Decaps(sk, c) is only required in case c is inconsistent. We still recommend to always compute $\tilde{K}$ because otherwise the system might be prone to a simple side-channel attack.

We now state the our main theorems about $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$[PKE, ID, F]'s security in the ROM and QROM, respectively. In the concrete security statements we will use the following terms

- n-user correctness error $\delta(n)$
- (weak) γ-spreadness
- Min entropy ℓ of ID(pk), i.e., $\ell := H_\infty(pk)$, where $(pk, sk) \xleftarrow{\$}$ Gen
- Bit-length λ of the secret seed $s \in \{0,1\}^\lambda$
- Max. number of (Q)ROM queries q_F
- Max. number of decapsulation queries q_D
- Max. number of challenge queries q_C

Theorem 3.1 ((n, q_C)-IND-CPA of PKE $\overset{\text{ROM}}{\Longrightarrow}$ (n, q_C)-IND-CCA of KEM). *For any adversary $\mathcal{A}$ against the (n, q_C)-IND-CCA security of KEM := $FO^{\not\perp}_{ID(pk),m}$[PKE, ID, F] there exist adversaries $\mathcal{B}$ and C against (n, q_C)-IND-CPA of PKE (with roughly the same running time) such that* $\mathrm{Adv}^{(n,q_C)\text{-}IND\text{-}CCA}_{KEM}(\mathcal{A}) \leq$

$$2\mathrm{Adv}^{(n,q_C)\text{-}IND\text{-}CPA}_{PKE}(\mathcal{B}) + \frac{2(q_F + q_C)q_C}{|\mathcal{M}|} + \frac{q_F}{2^\lambda} + (q_F + q_D)\cdot\delta(n) + \frac{n^2}{2^\ell} \tag{1}$$

$$2\mathrm{Adv}^{(n,q_C)\text{-}IND\text{-}CPA}_{PKE}(C) + \frac{2(q_F + q_C)q_C}{|\mathcal{M}|} + \frac{n(q_F + n)}{2^\lambda} + (q_F + q_D)\cdot n\delta + n \cdot q_D 2^{-\gamma}. \tag{2}$$

Equation (1) of Theorem 3.1 will be proved in Section 4, equation (2) in Section 7. The proofs follow essentially the ones from [21], where we have to take extra care to allow for a reduction from multi-user/challenge IND-CPA security.

Note that bound (1) is meaningless for small values ℓ. Bound (2) is slightly weaker but independent of ℓ and therefore also holds for ID$(pk) = \varepsilon$.

Theorem 3.2 ((n, q_C)-IND-CPA of PKE $\overset{\text{QROM}}{\Longrightarrow}$ (n, q_C)-IND-CCA of KEM). *For any quantum adversary $\mathcal{A}$ against the (n, q_C)-IND-CCA security of KEM := $FO^{\not\perp}_{ID(pk),m}$[PKE, ID, F] there exists a quantum adversary $\mathcal{B}$ against (n, q_C)-IND-CPA of PKE (with roughly the same running time) such that* $\mathrm{Adv}^{(n,q_C)\text{-}IND\text{-}CCA}_{KEM}(\mathcal{A}) \leq$

$$2\sqrt{q\mathrm{Adv}^{(n,q_C)\text{-}IND\text{-}CPA}_{PKE}(\mathcal{B})} + 4q\sqrt{\frac{q_C \cdot n}{|\mathcal{M}|} + \frac{n^2}{2^\ell}} + 4(q_F + 1)\sqrt{\frac{n}{2^\lambda}} + 16q^2\delta(n) + \frac{{q_C}^2}{|\mathcal{M}|},$$

where $q := q_F + q_D + 1$.

The proof of Theorem 3.2 is given in Section 5.

4 PROOF OF THEOREM 3.1

In this section we prove (1) of Theorem 3.1. The proof of (2) is similar and can be found in Section 7.

Proof. Let $\mathcal{A}$ be an adversary and consider the games given in Fig. 5.

Game G_0. This is the original (n, q_C)-IND-CCA game.

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \frac{1}{2}\right| = \mathrm{Adv}^{(n,q_\mathrm{C})\text{-IND-CCA}}_{\mathrm{KEM}}(\mathcal{A}).$$

Game G_1. In game G_1 an abort condition COLL is introduced, which aborts if there is a collision in the public-key identifiers ID(pk_j). In case there no collision in the identifiers, we are able to identity each identifier ID(pk_j) with a unique index j pointing to pk_j. This allows us to internally simulate

$$\mathsf{F}(\mathsf{ID}(pk_j)), A) := \begin{cases}(\mathsf{H}_j(m), \mathsf{G}_j(m)) & A = m\\ \mathsf{K}_j(s, c) & A = (s, c)\end{cases},$$

where G_j and H_j are internal random oracles, i.e., perfect random functions not accessible by $\mathcal{A}$. (This works because the two spaces $\mathcal{M}$ and $\{0,1\}^k \times C$ are disjoint.) Since the two games are identical until COLL happens, we have by the birthday bound

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{COLL}] \leq \frac{n^2}{2^\ell},$$

where $l = H_\infty(\mathsf{ID}(pk))$ is the min-entropy of ID(pk) for $(pk, sk) \xleftarrow{\$}$ Gen.

```
Gen'
01 $(pk, sk) \xleftarrow{\$} \mathsf{Gen}$
02 $s \xleftarrow{\$} \{0,1\}^\lambda$
03 $sk' := (sk, s)$
04 **return** $(pk, sk')$

Encaps(pk)
05 $m \xleftarrow{\$} \mathcal{M}$
06 $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk), m)$
07 $c \leftarrow \mathsf{Enc}(pk, m; r)$
08 **return** $(K, c)$

Decaps((sk, s), c)
09 $m' \leftarrow \mathsf{Dec}(sk, c)$
10 $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk), m')$
11 $\tilde{K} := \mathsf{F}_1(\mathsf{ID}(pk), s, c)$
12 **if** $m' = \bot$ **or** $\mathsf{Enc}(pk, m'; r) \neq c$
       **return** $\tilde{K}$
13 **else return** $K$
```

Figure 4: KEM = $\mathsf{FO}^{\not\perp}_{\mathsf{ID}(pk),m}$[PKE, ID, F] with "implicit rejection" and "partial key hashing" built from PKE, F, and ID.

```
GAMES $G_0$ - $G_5$
01 for $j \in [n]$
02     $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
03     $s_j \xleftarrow{\$} \{0,1\}^\lambda$
04 $\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
05 $\exists i, j \in [n]$ s.t. $\mathsf{ID}(pk_i) = \mathsf{ID}(pk_j)$      / $G_1$-$G_5$
06     COLL := true                                  / $G_1$-$G_5$
07     abort                                         / $G_1$-$G_5$
08 $b \xleftarrow{\$} \{0,1\}$
09 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Decaps},\mathsf{Chall},\mathsf{F}}(\vec{pk})$
10 return $[\![b' = b]\!]$

Chall(j)
11 $m \xleftarrow{\$} \mathcal{M}$
12 $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$                   / $G_2$-$G_5$
13 $(K_0, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m)$
14 $\mathcal{L}_{M_j} := \mathcal{L}_{M_j} \cup \{m\}$
15 $r \xleftarrow{\$} \mathcal{R}; K_0 \xleftarrow{\$} \{0,1\}^k$              / $G_5$
16 $K_1 \xleftarrow{\$} \{0,1\}^k$
17 $c \leftarrow \mathsf{Enc}(pk_j, m; r)$
18 $\mathcal{L}_{C_j} := \mathcal{L}_{C_j} \cup \{c\}$
19 return $(c, K_b)$

Decaps($j, c \notin \mathcal{L}_{C_j}$)                             / $G_0$-$G_3$
20 $m' := \mathsf{Dec}(sk_j, c)$
21 $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m')$
22 if $m' = \bot$ or $\mathsf{Enc}(pk_j, m'; r) \neq c$
23     $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), s_j, c)$          / $G_0$-$G_1$
24     $K := \mathsf{K}'_j(c)$                               / $G_3$
25 return $K$

Decaps($j, c \notin \mathcal{L}_{C_j}$)                             / $G_4$-$G_5$
26 if $\exists K$ s. th. $(c, K) \in \mathcal{L}_{D_j}$
27     return $K$
28 $K \xleftarrow{\$} \mathcal{K}$
29 $\mathcal{L}_{D_j} := \mathcal{L}_{D_j} \cup \{(c, K)\}$
30 return $K$

F(id, A)
31 if $\exists j \in [n] : \mathrm{id} = \mathsf{ID}(pk_j)$                / $G_1$-$G_5$
32     if $A \in \mathcal{M}$                                / $G_1$-$G_5$
33         $m := A$                                      / $G_1$-$G_5$
34         $(K, r) := (\mathsf{H}_j(m), \mathsf{G}_j(m))$              / $G_1$-$G_5$
35     if $A \in \{0,1\}^\lambda \times C$                      / $G_1$-$G_5$
36         $(s, c) := A$                                 / $G_1$-$G_5$
37         $K := \mathsf{K}_j(s, c)$                           / $G_1$-$G_5$
38 if $K$ undefined: $K \xleftarrow{\$} \mathcal{K}$
39 if $r$ undefined: $r \xleftarrow{\$} \mathcal{R}$
40 return $(K, r)$

$\mathsf{G}_j(m)$                                 / Internal random oracle
41 $r \xleftarrow{\$} \mathcal{R}$
42 if $m \in \mathcal{L}_{M_j}$                                    / $G_5$
43     QUERY := true                                 / $G_5$
44     abort                                         / $G_5$
45 return $r$

$\mathsf{K}_j(s, c)$                              / Internal random oracle
46 $K \xleftarrow{\$} \mathcal{K}$
47 if $s = s_j$                                      / $G_3$-$G_5$
48     BAD := true                                   / $G_3$-$G_5$
49     abort                                         / $G_3$-$G_5$
50 return $K$

$\mathsf{H}_j(m)$                                 / Internal random oracle
51 $K \xleftarrow{\$} \mathcal{K}$
52 if $m \in \mathcal{L}_{M_j}$                                    / $G_5$
53     QUERY := true                                 / $G_5$
54     abort                                         / $G_5$
55 $c' := \mathsf{Enc}(pk_j, m; \mathsf{G}_j(m))$                    / $G_4$-$G_5$
56 if $\exists K'$ such that $(c', K') \in \mathcal{L}_{D_j}$          / $G_4$-$G_5$
57     $K := K'$                                     / $G_4$-$G_5$
58 else                                              / $G_4$-$G_5$
59     $\mathcal{L}_{D_j} := \mathcal{L}_{D_j} \cup \{(c', K)\}$            / $G_4$-$G_5$
60 return $K$
```

Figure 5: Games G_0 - G_5 for the proof of Theorem 3.1. The internal random oracles $\mathsf{K}'_j, \mathsf{H}_j, \mathsf{G}_j, \mathsf{K}_j$ are not accessible by the adversary. We assume wlog that F is only queried once on each value (id, A)**.**

Game G_2 In game G_2 the challenge oracle Chall samples m from the set $\mathcal{M} \setminus \mathcal{L}_{M_j}$ instead of $\mathcal{M}$. This is necessary since $m_i = m_j$ implies $c_i = c_j$. Therefore we have $K_i = K_j$ in the "real world", but the KEM keys in the "random world" are independent. Consequently, an adversary would be able distinguish between the "real" from "random" keys. Since there are at most q_C challenge queries we have (by the birthday bound)

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \frac{{q_C}^2}{|\mathcal{M}|}.$$

Game G_3. In game G_3 we modify the Decaps oracle in lines 23 and 24 such that for an invalid ciphertext the key is defined as $\mathsf{K}'_j(c)$, where K'_j is an independent internal random oracle. This remains unnoticed to adversary $\mathcal{A}$ unless it queries $\mathsf{K}_j(s_j, \cdot)$ for some $j \in [n]$. Since the seeds $s_j \in \{0,1\}^\lambda$ are uniformly random and information-theoretically hidden from the adversary, we have by the union bound

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \frac{q_{\mathsf{F}}}{2^{\lambda}}\,.$$

Game G_4. In game G_4 we simulate the decapsulation oracle $\mathsf{Decaps}(j,\cdot)$ without knowledge of the secret key by patching the random oracles H_j in lines 55-59. Note that if PKE was perfectly correct, then the random oracle patching is also perfectly correct and therefore the two games would look identical in $\mathcal{A}$'s view.

The only bad case happens if G_j is queried on some m which induces a correctness error, that is $\mathsf{Dec}(sk_j, \mathsf{Enc}(pk_j; m; \mathsf{G}_j(m))) \neq m$. More concretely, define the sets

$$\mathrm{BAD}_j := \left\{ m \in \mathcal{M} \;\middle|\; \begin{array}{l} m \neq m', \text{where } c \leftarrow \mathsf{Enc}(pk_j; m; \mathsf{G}_j(m)); \\ m' \leftarrow \mathsf{Dec}(sk_j, c) \end{array} \right\}$$

Define the event CORR to be the event that $\mathcal{A}$ makes an (implicit) query to $\mathsf{G}_j(m)$ for some $m \in \mathrm{BAD}_j$. Since there are at most $(q_{\mathsf{F}} + q_{\mathsf{D}})$ explicit and implicit queries to G_j, we have

$$\Pr[\mathsf{CORR}] \leq (q_{\mathsf{F}} + q_{\mathsf{D}})\delta(n)\,.$$

We now claim that

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{CORR}]\,.$$

Let us analyze why G_3 and G_4 are identical conditioned on $\neg\mathsf{CORR}$. Consider a query $\mathsf{Decaps}(j, c)$ and define $m' := \mathsf{Dec}(sk_j, c)$ and $c' := \mathsf{Enc}(pk_j, m'; \mathsf{G}_j(m'))$.

- **Case 1:** $m' = \bot$. H_j cannot be called on $m' = \bot$ and hence the KEM key of $K = \mathsf{Decaps}(sk_j, c) = \mathsf{K}'_j(c)$ in G_3 is identically distributed as $(c, K) \in \mathcal{L}_{D_j}$ in G_4.
- **Case 2:** $m' \neq \bot \wedge c \neq c'$. Both games return a uniform random key K. The only way for $\mathcal{A}$ to detect a difference between the two games is if it makes a query $\mathsf{H}_j(m)$ such that $\mathsf{Enc}(pk_j, m; \mathsf{G}_j(m)) = c$. (In G_4, $\mathsf{H}_j(m)$ would return the same key K as in $\mathsf{Decaps}(j, c)$, whereas in G_3 the two keys would be independent.) Since $c \neq c'$ we also have $m \neq m'$. But such a query $\mathsf{H}_j(m)$ would internally involve the query $\mathsf{G}_j(m)$ for $m \in \mathrm{BAD}_j$.
- **Case 3:** $m' \neq \bot \wedge c = c'$. In game G_3, $\mathsf{Decaps}(j, c)$ returns $K = \mathsf{H}_j(m')$, whereas G_4 first picks a uniform K and patches $\mathsf{H}_j(m)$ to match K for all m that deterministically encrypt to the same c, i.e., all m satisfying $\mathsf{Enc}(pk_j, m; \mathsf{G}_j(m)) = c$. The only way for $\mathcal{A}$ to detect a difference between the two games is to query H_j on some value $m \neq m'$ that also deterministically encrypts to the same c. But this also implies that $\mathsf{G}_j(m)$ was queried for some $m \in \mathrm{BAD}_j$.

Game G_5. In game G_5 we abort on queries of the form $\mathsf{H}_j(m)$ or $\mathsf{G}_j(m)$ for some challenge message $m \in \mathcal{L}_{M_j}$, in which case the event QUERY holds true. We have by the difference lemma,

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{QUERY}]\,.$$

Note that in G_5 bit b is independent of the view of the adversary. We thus have

$$\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] = \frac{1}{2}\,.$$

We claim that

$$\Pr[\mathsf{QUERY}] \leq 2 \cdot \left(\mathsf{Adv}_{\mathsf{PKE}}^{(n,q_{\mathsf{C}})\text{-IND-CPA}}(\mathcal{B}) + \frac{q_{\mathsf{F}} q_{\mathsf{C}}}{|\mathcal{M}|}\right)\,. \qquad (3)$$

Summing up the inequalities yields the claimed bound, concluding the proof of the theorem.

We show (3) by giving an adversary $\mathcal{B}$ against the (n, q_{C})-IND-CPA security of PKE. Adversary $\mathcal{B}$ from the (n, q_{C})-IND-CPA challenger receives n public-keys and has access to a challenge encryption oracle. It runs $\mathcal{B}$ on the public keys simulating the decapsulation oracle Decaps and the random oracle F as in G_4 and G_5. On $\mathcal{A}$'s challenge query $\mathsf{Chall}(j)$, $\mathcal{B}$ proceeds as follows. It picks random $m_0 \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_{j,0}}$ and $m_1 \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_{j,1}}$ and adds m_0 to list $\mathcal{L}_{M_{j,0}}$ and m_1 to list $\mathcal{L}_{M_{j,1}}$. (If it is the first challenge query on input j it initializes the lists $\mathcal{L}_{M_{j,0}}$ and $\mathcal{L}_{M_{j,1}}$ to be empty.) Next, $\mathcal{B}$ queries its own challenge oracle to obtain $c \leftarrow \mathsf{Enc}(pk_j, m_b)$, where b is the (n, q_{C})-IND-CPA's challenge bit. Finally, it returns (c, K) to $\mathcal{A}$, for a random key $K \in \{0,1\}^k$. Note that this perfectly simulates the challenge oracle as in games G_4 and G_5.

If, during the simulation of F, $\mathcal{B}$ detects a query $\mathsf{H}_j(m)$ or $\mathsf{G}_j(m)$ for some $m \in \mathcal{L}_{M_{j,b'}}$, it returns b' and terminates. If no such query happens and $\mathcal{A}$ terminates, $\mathcal{B}$ returns a uniform bit b'.

Since all messages in the set $\mathcal{L}_{M_{j,1-b}}$ are information-theoretically hidden from $\mathcal{A}$, the probability that it queries $\mathsf{G}_j(m)$ or $\mathsf{H}_j(m)$ on some $m \in \mathcal{L}_{M_{j,1-b}}$ is bounded by $q_{\mathsf{F}} q_{\mathsf{C}}/|\mathcal{M}|$. Assume that this is not the case. We have

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PKE}}^{(n,q_{\mathsf{C}})\text{-IND-CPA}}(\mathcal{B}) + \frac{q_{\mathsf{F}} q_{\mathsf{C}}}{|\mathcal{M}|} &\geq \left|\Pr\left[b = b'\right] - \frac{1}{2}\right| \\ &= \left|\Pr[\mathsf{QUERY}] + \frac{1}{2}\Pr[\neg\mathsf{QUERY}] - \frac{1}{2}\right| \\ &= \frac{1}{2}\Pr[\mathsf{QUERY}]\,,\end{aligned}$$

which proves (3). □

5 PROOF OF THEOREM 3.2

We refer to Section 6 for standard quantum notation. Our notation and the presentation of known results closely follows [23] and [22, Section 1.3].

We will now recall some QROM theorems that we will use during our proof of Theorem 3.2.

Lemma 5.1 (Generic Distinguishing Problem with Bounded Probabilities [23]). *Let X be a finite set, and let $\lambda \in [0,1]$. For any (unbounded, quantum) algorithm $\mathcal{A}$ issuing at most q quantum queries to F,*

$$|\Pr[\mathsf{GDPB}_{\lambda,0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{GDPB}_{\lambda,1}^{\mathcal{A}} \Rightarrow 1]| \leq 8 \cdot \lambda \cdot (q+1)^2\,, \quad (4)$$

where games $\mathsf{GDPB}_{\lambda,b}$ (for bit $b \in \{0,1\}$) are defined in Fig. 6.

Oneway to Hiding with semi-classical oracles. In [1], Ambainis et al. defined semi-classical oracles that return a state that was measured with respect to one of the input registers. To any subset $S \subset X$, one can associate the following "semi-classical" oracle $\mathsf{O}_S^{\mathsf{SC}}$: Intuitively, $\mathsf{O}_S^{\mathsf{SC}}$ collapses states taken from $\mathcal{H}_{X\times Y}$ to a state that contains only elements of either S or $X \setminus S$. To be more precise, $\mathsf{O}_S^{\mathsf{SC}}$ takes as input a quantum state $|\psi, 0\rangle$ such that $|\psi\rangle \in \mathcal{H}_X \otimes \mathcal{H}_Y$.

```
GAME GDPB_{λ,b}
(λ_x)_{x∈X} ← 𝒜_1
if ∃x ∈ X s.t. λ_x > λ
    return 0
if b = 1
    for all x ∈ X
        O(x) ← B_{λ_x}
else
    O := 0
b′ ← 𝒜_2^{|O⟩}
return b′
```

Figure 6: Generic distinguishing games $\mathsf{GDPB}_{\lambda,b}$ with bounded maximal Bernoulli parameter $\lambda \in [0,1]$.

$\mathsf{O}_\mathsf{S}^\mathsf{SC}$ first measures the X-register with respect to the projectors $M_1 := \sum_{x\in S} |x\rangle\langle x|$ and $M_0 := \sum_{x\notin S} |x\rangle\langle x|$, and then initialises the last register to $|b\rangle$ for the measured bit b. Consequently, $|\psi, 0\rangle$ collapses to either a state $|\psi', 1\rangle$ such that the X-register of $|\psi'\rangle$ only contains elements of S, or a state $|\psi', 0\rangle$ such that the X-register of $|\psi'\rangle$ only contains elements of $X \setminus S$.

To a quantum-accessible oracle O and a subset $S \subset X$, one can furthermore associate oracle $\mathsf{O} \setminus \mathsf{S}$ which first queries $\mathsf{O}_\mathsf{S}^\mathsf{SC}$ and then O. Let FIND denote the event that $\mathsf{O}_\mathsf{S}^\mathsf{SC}$ ever returns a state $|\psi', 1\rangle$. Unless FIND occurs, the outcome of $\mathsf{O} \setminus \mathsf{S}$ is independent of the values $\mathsf{O}(x)$ for all $x \in S$, which is why $\mathsf{O} \setminus \mathsf{S}$ is also called a "punctured" oracle.

We will now restate several "semi-classical one-way to hiding" theorems from [1]. While [1] consider adversaries that might execute parallel oracle invocations, and therefore differentiate between query depth d and number of queries q, we use the upper bound $q \geq d$ for the sake of simplicity. Lemma 5.2 is a simplification of [1, Thm. 1: "Semi-classical O2H"], and Eq. (6) (Eq. (7)) of Lemma 5.3 is a simplification of [1, Thm. 2: "Search in semi-classical oracle"] ([1, Cor. 1]).

Lemma 5.2. *Let $S \subset X$ be random. Let $\mathsf{O}_1, \mathsf{O}_2 \in Y^X$ be random functions such that $\mathsf{O}_1(x) = \mathsf{O}_2(x)$ for all $x \in X \setminus S$, and let z be a random bitstring. (S, O_1, O_2, and z may have an arbitrary joint distribution.) For $i \in \{1, 2\}$, let*

$$p_i := \Pr[1 \leftarrow \mathsf{A}^{|\mathsf{O_i}\rangle}(z)],$$

and let

$$p_\mathsf{FIND} := \Pr[b \leftarrow \mathsf{A}^{|\mathsf{O_1 \setminus S}\rangle}(z) : \mathsf{FIND}].$$

For all quantum algorithms A with binary output, issuing at most q queries, we have that

$$|p_1 - p_2| \leq 2 \cdot \sqrt{(q+1) \cdot p_\mathsf{FIND}}. \tag{5}$$

Lemma 5.3. *Let $S \subset X$ be random, let O be a random function, and let z be a random bitstring. (S, O, and z may have an arbitrary joint distribution.) Let*

$$p_\mathsf{FIND} := \Pr[b \leftarrow \mathsf{A}^{|\mathsf{O \setminus S}\rangle}(z) : \mathsf{FIND}].$$

Then, for all quantum algorithms A with binary output issuing at most q queries, we have that

$$p_\mathsf{FIND} \leq 4q \cdot \Pr[x \leftarrow \mathsf{B}(z) : x \in S], \tag{6}$$

where B is the algorithm that, on input z, chooses $i \xleftarrow{\$} \{1, \cdots, q\}$, runs $\mathsf{A}^{|\mathsf{O}\rangle}$ until (just before) the i-th query, measures its query input register in the computational basis and outputs the measurement outcome.

If $S := \{x_1, \ldots, x_n\}$ for $x_1, \ldots, x_n \xleftarrow{\$} X$, and S and z are independent, we have that

$$p_\mathsf{FIND} \leq \frac{4q|S|}{|X|}. \tag{7}$$

Furthermore, if $S := \{(y_1, x_1) \ldots, (y_n, x_n)\}$ where $x_i \xleftarrow{\$} X \setminus \{x_1, \ldots, x_{i-1}\}$ and $\{y_1, \ldots, y_n\} \subset Y$ independent of $U := \{x_1, \ldots, x_n\}$ we have for U independent of z

$$p_\mathsf{FIND} \leq \frac{4q|S|}{|X|}. \tag{8}$$

We will now prove an additional helper lemma.

Lemma 5.4 (QROM Multi-User PRF). *For $i \in [n]$, let $p_i \in \{0,1\}^Y$ arbitrarily subject to $p_i \neq p_j$, when $i \neq j$. Define $\vec{p} := (p_1, \ldots, p_n)$. Let $\mathsf{H}, \mathsf{H}_1, \ldots, \mathsf{H}_n$ be independent random oracles with $\mathsf{H}\colon \{0,1\}^Y \times \{0,1\}^\lambda \times \mathcal{X} \to \mathcal{Y}'$ and $\mathsf{H}_i : \mathcal{X} \to \mathcal{Y}'$, then for all quantum algorithms $\mathcal{A}$ issuing at most q quantum queries to H and arbitrarily many queries to H_i with $i \in [n]$, we have*

$$\left|\Pr\left[\mathcal{A}^{|\mathsf{H}\rangle, \mathsf{H}(p_1,s_1,\cdot),\ldots,\mathsf{H}(p_n,s_n,\cdot)}(\vec{p}) = 1\right] - \Pr\left[\mathcal{A}^{|\mathsf{H}\rangle, \mathsf{H}_1,\ldots,\mathsf{H}_n}(\vec{p}) = 1\right]\right| \leq 4(q+1)\sqrt{\frac{n}{2^\lambda}},$$

where the probabilities are taken over $\mathsf{H}, \mathsf{H}_1, \ldots, \mathsf{H}_n$ and $s_1, \ldots s_n \xleftarrow{\$} \{0,1\}^\lambda$, and the internal randomness of $\mathcal{A}$.

Proof. The proof follows from combining Lemma 5.2 with Lemma 5.3 and the overall proof idea is very similar to the one of [10, Corollary 1]. The adversary's goal is to distinguish quantum access to H and additional classical access to $(\mathsf{H}(p_1, s_1, \cdot), \ldots, \mathsf{H}(p_n, s_n, \cdot))$ from quantum access to H and additional classical access to a collection $(\mathsf{H}_1, \ldots, \mathsf{H}_n)$ of independent random oracles. This is the same as distinguishing $(\mathsf{H}, \mathsf{H}^{(p_1,s_1,\cdot)\mapsto\mathsf{H}_1(\cdot)}, \ldots, \mathsf{H}^{(p_n,s_n,\cdot)\mapsto\mathsf{H}_n(\cdot)})$ from $(\mathsf{H}, \mathsf{H}_1, \ldots, \mathsf{H}_n)$, and the set of differences between these oracles is $S := \{(p_j, s_j) \mid j \in [n]\} \times \mathcal{X}$. According to Lemma 5.2, the distinguishing advantage is at most $2\sqrt{(q+1) \cdot p_\mathsf{FIND}}$, and we can apply Eq. (8) from Lemma 5.3 to obtain that $p_\mathsf{FIND} \leq 4q \cdot \frac{n}{2^\lambda}$. It remains to note that for $i \neq j$ we have $(p_i, s_i) \neq (p_j, s_j)$.

□

5.1 Proof

With these results at hand, we can finally proceed to the proof of Theorem 3.2. Its overall structure is very similar to recent QROM proofs of IND-CCA security [10, 23–27, 33]: we first change the game such that the decapsulation oracle can be simulated without knowledge of the collection of secret keys, which is achieved as usual by plugging encryption into the random oracle (in our case for multiple users), and then apply one-way to hiding to argue key indistinguishability.

Proof of Theorem 3.2. Let $\mathcal{A}$ be an adversary and consider the games given in Fig. 7 and Fig. 9.

```
GAMES $G_0$ - $G_7$
01 for $j \in [n]$
02     $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
03     $s_j \xleftarrow{\$} \{0,1\}^\lambda$
04 $\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
05 $\exists i, j \in [n]$ s.t. $\mathsf{ID}(pk_i) = \mathsf{ID}(pk_j)$        / $G_1$-$G_7$
06     $\mathsf{COLL} := \mathbf{true}$                                 / $G_1$-$G_7$
07     abort                                                        / $G_1$-$G_7$
08 $b \xleftarrow{\$} \{0,1\}$
09 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Decaps},\mathsf{Chall},|\mathsf{F}\rangle}(\vec{pk})$
10 return $[\![b' = b]\!]$

Chall($j$)
11 $m \xleftarrow{\$} \mathcal{M}$
12 $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$                           / $G_2$-$G_7$
13 $\mathcal{L}_{M_j} := \mathcal{L}_{M_j} \cup \{m\}$
14 $(K_0, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m)$
15 $K_1 \xleftarrow{\$} \{0,1\}^k$
16 $c \leftarrow \mathsf{Enc}(pk_j, m; r)$
17 $\mathcal{L}_{C_j} := \mathcal{L}_{C_j} \cup \{c\}$
18 return $(c, K_b)$

Decaps($j$, $c \notin \mathcal{L}_{C_j}$)                                  // $G_0$-$G_5$
19 $m' := \mathsf{Dec}(sk_j, c)$
20 $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m')$
21 if $m' = \perp$ or $\mathsf{Enc}(pk_j, m'; r) \neq c$
22     $(K, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), s_j, c)$
23     $K := \mathsf{K}'_j(c)$                                           / $G_3$-$G_5$
24 return $K$

F(id, $A$)
25 if $A \in \mathcal{M}$
26     $m := A$
27     return $(\mathsf{H}(\mathrm{id}, m), \mathsf{G}(\mathrm{id}, m))$
28 else if $A \in \{0,1\}^\lambda \times C$
29     $(s, c) := A$
30     return $(\mathsf{K}(\mathrm{id}, s, c), \mathsf{L}(\mathrm{id}, s, c))$
31 return $\mathsf{L}'(\mathrm{id}, A)$

H(id, $m$)                                          / Internal Random Oracle
32 if $\exists j \in [n] : \mathrm{id} = \mathsf{ID}(pk_j)$                    / $G_1$-$G_7$
33     return $\mathsf{H}_j(m)$                                         / $G_1$-$G_7$
34 return $\mathsf{H}_0(\mathrm{id}, m)$

$\mathsf{H}_j(m)$                                           / Internal Random Oracle
35 return $\mathsf{H}^1_j(m)$                                            / $G_0$-$G_4$
36 return $\mathsf{H}^2_j(\mathsf{Enc}(pk_j, m; \mathsf{G}'_j(m)))$                  / $G_5$-$G_6$
37 return $\mathsf{H}^2_j(\mathsf{Enc}(pk_j, m; \mathsf{G}_j(m)))$                   / $G_7$

G(id, $m$)                                          / Internal Random Oracle
38 if $\exists j \in [n] : \mathrm{id} = \mathsf{ID}(pk_j)$                    / $G_1$-$G_7$
39     return $\mathsf{G}_j(m)$                                   / $G_1$-$G_3$,$G_7$
40     return $\mathsf{G}'_j(m)$                                        / $G_4$-$G_6$
41 return $\mathsf{G}_0(\mathrm{id}, m)$

$\mathsf{G}'_j(m)$                                          / Internal Random Oracle
42 return $\mathsf{Sample}(\mathcal{R} \setminus \mathcal{R}_{bad}(pk_j, sk_j, m); \mathsf{R}(j, m))$

Decaps($j$, $c \notin \mathcal{L}_{C_j}$)                                  // $G_6$-$G_7$
43 return $\mathsf{H}^2_j(c)$
```

Figure 7: Games G_0 - G_7 for the proof of Theorem 3.2. The internal random oracles H, H_0, G, G_0, G_j, H^1_j, H^2_j, K and K'_j, L', R are not directly accessible to the adversary. The notation $|\mathsf{F}\rangle$ denotes that the adverary $\mathcal{A}$ has quantum access to the random oracle F. The output range of F is $\{0,1\}^k \times \mathcal{R}$.

Game G_0. This is the original (n, q_C)-IND-CCA game. We have

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{KEM}}^{(n,q_C)\text{-IND-CCA}}(\mathcal{A}) .$$

Game G_1. In game G_1 we abort on a public-key identifier collision (line 05-line 07). Like in the ROM proof, we can now identify every public-key identifier $\mathsf{ID}(pk_j)$ with its index j. This allows us to simulate the random oracles, when called on such an identifier and $m := A \in \mathcal{M}$, via

$$\mathsf{F}(\mathsf{ID}(pk_j), m) := (\mathsf{H}_j(m), \mathsf{G}_j(m)) , \tag{9}$$

where H_j and G_j are independent internal random oracles. We stress the importance of this step. First, it will allow us in later steps of the proof to replace G_j with a random oracle G'_j which only samples "good randomness" with respect to the public-key pk_j. Second, it is essential in order to simulate the $\mathsf{Decaps}(j, \cdot)$ oracle without knowledge of the secret-key. Since the identification of Eq. 9 is only a conceptual change, by the birthday bound and the ℓ-bit min-entropy of $\mathsf{ID}(pk)$ we have

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{COLL}] \leq \frac{n^2}{2^\ell} .$$

Game G_2 In game G_2 the challenge oracle samples m from $\mathcal{M} \setminus \mathcal{L}_{M_j}$ instead of $\mathcal{M}$, since on a message collision an adversary can easily differentiate between the real or random world. By the birthday bound we have

$$\left|\Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \frac{{q_C}^2}{|\mathcal{M}|} .$$

Game G_3. In game G_3 we substitute $\mathsf{F}(\mathsf{ID}(pk_j), s_j, c)$ in line 22, whose first k bits evaluate to $\mathsf{K}(\mathsf{ID}(pk_j), s_j, c)$ in line 30, with $\mathsf{K}'_j(c)$ in line 23, where K'_j is an internal independent random oracle for every $j \in [n]$. With a straightforward reduction, we can apply Lemma 5.4. Concretely, we identify H with K, H_j with K'_j, p_j with $\mathsf{ID}(pk_j)$ and the uniform secrets $s_j \in \{0,1\}^\lambda$ from Lemma 5.4 with the scheme's secret seeds s_j used in the game. We obtain

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 4(q_F + 1)\sqrt{\frac{n}{2^\lambda}} .$$

Game G_4. In game G_4 we switch the random oracle G to an oracle which samples only "good" randomness, meaning that no decryption failure of proper encryptions can possibly occur anymore. To make this more formal, for fixed $(pk, sk) \in \operatorname{supp}(\mathsf{Gen})$ and $m \in \mathcal{M}$, let

$$\mathcal{R}_{\text{bad}}(pk, sk, m) := \{r \in \mathcal{R} \colon \mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r)) \neq m\}$$

in order to define $\delta(pk, sk, m) := |\mathcal{R}_{\text{bad}}(pk, sk, m)| / |\mathcal{R}|$. The modified oracle G can now be defined as follows: We will still let G coincide with G_0 anywhere but on $\{\mathsf{ID}(pk_j)) \mid j \in [n]\} \times \mathcal{M}$. Whereas for any index j, however, $\mathsf{G}(\mathsf{ID}(pk_j), \cdot)$ was defined until this game by as an oracle G_j which has range $\mathcal{R}$ (line 39), we now replace each G_j with a random oracle G'_j that has range $\mathcal{R} \setminus \mathcal{R}_{\text{bad}}(pk_j, sk_j, m)$ instead in line 40. We will now argue that distinguishing game G_3 from G_4 can be reduced to distinguishing the GDPB games (see Lemma 5.1). Consider the quantum distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ given in Fig. 8. After choosing its Bernoulli parameters $\lambda(j, m) := \delta(pk_j, sk_j, m)$, $\mathcal{D}$ is provided with access to oracle O which is either the constant 0 function (if $\mathcal{D}$ is run in game $\mathsf{GDPB}_{\lambda,0}$) or distributed according to the chosen Bernoulli parameters (in game $\mathsf{GDPB}_{\lambda,1}$). Note that $\delta(n)$ serves as an upper bound for these parameters. $\mathcal{D}$ perfectly simulates game G_3 if run in game $\mathsf{GDPB}_{\lambda,1}$, and game G_4 if run in $\mathsf{GDPB}_{\lambda,0}$. Identifying λ with $\delta(n)$ and q with $q_{\mathsf{F}} + q_{\mathsf{D}}$, we can apply Lemma 5.1 to obtain

$$\left|\Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 8(q_{\mathsf{F}} + q_{\mathsf{D}} + 1)^2 \delta(n)\,.$$

Game G_5. In game G_5, we modify the random oracle $\mathsf{H}_j(m)$ such that it returns $\mathsf{H}^2_j(\mathsf{Enc}(pk_j, m; \mathsf{G}'_j(m)))$ in line 36 instead of $\mathsf{H}^1_j(m)$ in line 35. Here, H^1_j and H^2_j are independent internal random oracles. Since we use the modified G that only samples "good" randomness, the mapping $\mathsf{Enc}(pk_j, \cdot; \mathsf{G}'_j(\cdot))$ is injective for all $j \in [n]$. Furthermore, since $\mathsf{ID}(pk_i) \neq \mathsf{ID}(pk_j)$ for all $i \neq j$, we have that the following function h is injective:

$$h(\mathsf{id}, m) := \begin{cases} (\mathsf{id}, \mathsf{Enc}(pk_j, m; \mathsf{G}'_j(m))) & \text{if } \exists j \in [n] \text{ s.t. } \mathsf{id} = \mathsf{ID}(pk_j) \\ (\mathsf{id}, m) & \text{else} \end{cases}. \tag{10}$$

We can now rewrite H in Game G_5 as $\mathsf{H} = \mathsf{H}^2 \circ h$, where we define the internal random oracle H^2 as

$$\mathsf{H}^2(\mathsf{id}, m) := \begin{cases} \mathsf{H}^2_j(m) & \text{if } \exists j \in [n] \text{ s.t. } \mathsf{id} = \mathsf{ID}(pk_j) \\ \mathsf{H}_0(\mathsf{id}, m) & \text{else} \end{cases}.$$

After these changes, H equals $\mathsf{H}^2 \circ h$ for an internal random oracle H^2 and an injective function h. Therefore, H is still a random oracle in G_5 and hence perfectly indistinguishable from the one of G_4. Thus, we have shown

$$\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right].$$

Game G_6. In game G_6, we start to simulate the decapsulation oracle without knowledge of the secret keys. We modify $\mathsf{Decaps}(j, c)$ to return $\mathsf{H}^2_j(c)$ in line 43. Let $m' := \mathsf{Dec}(sk_j, c)$ and consider the following two cases:

- Case 1: The ciphertext c lies in the image of encryption, that is $\mathsf{Enc}(pk_j, m'; \mathsf{G}'_j(m')) = c$. In this case we have
$$\begin{aligned} \mathsf{H}^2_j(c) &= \mathsf{H}^2_j(\mathsf{Enc}(pk_j, m'; \mathsf{G}'_j(m'))) \\ &= (\mathsf{H}^2 \circ h)(\mathsf{ID}(pk_j), m') \\ &= \mathsf{H}(\mathsf{ID}(pk_j), m') \\ &= \mathsf{Decaps}(j, c)\,. \end{aligned}$$
Therefore the oracles in G_5 and G_6 are identical.
- Case 2: The ciphertext c lies not in the image of encryption, that is $\mathsf{Enc}(pk_j, m'; \mathsf{G}'_j(m')) \neq c$. Here the adversary gets to see $\mathsf{K}'_j(c)$ in G_5 and $\mathsf{H}^2_j(c)$ in G_6 for internal random oracles K'_j and H^2_j. In both games the adversary gets access to H^2 indirectly through $\mathsf{H}(\mathsf{ID}(pk_j), m)$ forcing m through the encryption algorithm. Therefore, the adversary can not access H^2 for ciphertexts lying outside the image space of encryption. Therefore no adversary can distinguish between $\mathsf{K}'_j(c)$ in G_5 and $\mathsf{H}^2_j(c)$.

We just have shown

$$\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right].$$

Game G_7. In game G_7, we switch back to using the original random oracle G in lines 39 and 37. With essentially the same argument as in the game-hop from G_3 to G_4, we have

$$\left|\Pr\left[G_6^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 8(q_{\mathsf{F}} + q_{\mathsf{D}} + 1)^2 \delta(n)\,.$$

What we have achieved so far is that the decapsulation oracle is simulatable by a reduction that does not know the secret keys, we can now apply the one-way to hiding theorems in order to reduce key indistinguishability to an attacker against the underlying encryption scheme.

Game G_8. In game G_8 we make the following conceptual change, in order to be able to apply the one-way to hiding theorems: Instead of sampling the challenge plaintexts and computing the corresponding challenge keys $K_{j,i}$ on the fly in the Chall oracle (like in G_7), we now precompute these values before running the adversary, and use them in the Chall oracle. Since this change is purely conceptual, we have that

$$\Pr\left[G_7^{\mathcal{A}} \Rightarrow 1\right] = \Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right].$$

Game G_9. In game G_9, we decouple the oracle values on the challenge plaintexts from the values that are used to compute the challenge ciphertexts and the challenge keys, see Fig. 9. In game G_9, the adversary's view is independent of b and

$$\Pr\left[G_9^{\mathcal{A}} \Rightarrow 1\right] = \frac{1}{2}\,.$$

We have reprogrammed F on the set $\{(\mathsf{ID}(pk_j), m_{j,i}) \mid j \in [n], i \in [q_{\mathsf{C}}]\}$, and we can now first apply Lemma 5.2 in order to argue that

$$\left|\Pr\left[G_8^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_9^{\mathcal{A}} \Rightarrow 1\right]\right| \leq 2\sqrt{(q_{\mathsf{F}} + q_{\mathsf{D}} + 1) \cdot \Pr[\mathsf{FIND}]}\,.$$

Here $\Pr[\mathsf{FIND}]$ denotes the probability that when game G_9 is run with the punctured counterpart $F \setminus S$ of F instead of F, $F \setminus S$ ever collapses the input register of one of the random oracle queries to

```
$\underline{\mathcal{D}_1}$
for $j \in [n]$
    $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
    for $m \in \mathcal{M}$
        $\lambda(j, m) := \delta(pk_j, sk_j, m)$
return $((\lambda(j, m)))_{j \in [n], m \in \mathcal{M}}$

$\underline{\mathsf{G}_j(m)}$
if $\mathsf{O}(j, m) = 0$
    $\mathsf{G}_j(m) := \mathsf{Sample}(\mathcal{R} \setminus \mathcal{R}_{bad}(pk_j, sk_j, m); \mathsf{R}(j, m))$
else
    $\mathsf{G}_j(m) := \mathsf{Sample}(\mathcal{R}_{bad}(pk_j, sk_j, m); \mathsf{R}(j, m))$
return $\mathsf{G}_j(m)$

$\underline{\mathcal{D}_2^{|\mathsf{O}\rangle}}$
for $j \in [n]$
    $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
    $s_j \xleftarrow{\$} \{0, 1\}^\lambda$
$\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
$\exists i, j \in [n]$ s.t. $\mathsf{ID}(pk_i) = \mathsf{ID}(pk_j)$: abort
$b \xleftarrow{\$} \{0, 1\}$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Decaps}, \mathsf{Chall}, |\mathsf{F}\rangle}(\vec{pk})$
return $[\![b' = b]\!]$

$\underline{\mathsf{G}(\mathrm{id}, m)}$
if $\exists j \in [n] : \mathrm{id} = \mathsf{ID}(pk_j)$
    return $\mathsf{G}_j(m)$
return $\mathsf{G}_0(\mathrm{id}, m)$
```

Figure 8: Distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$ for the generic distinguishing problem with bounded probabilities. The internal random oracle R is used to provide the random coins of the sampling procedure. The oracles Decaps, Chall and F are simulated as in G_3 (or equivalently G_4) of Fig. 7.

```
$\underline{G_7\text{-}G_{10}}$
01 for $j \in [n]$
02     $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
03     $s_j \xleftarrow{\$} \{0, 1\}^\lambda$
04 $\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
05 $\exists i, j \in [n]$ s.t. $\mathsf{ID}(pk_i) = \mathsf{ID}(pk_j)$
06     COLL := true
07     abort
08 for $j \in [n]$                                                      / $G_8$-$G_{10}$
09     for $i \in [q_\mathsf{C}]$                                       / $G_8$-$G_{10}$
10         $m_{j,i} \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$                    / $G_8$-$G_{10}$
11         $\mathcal{L}_{M_j} := \mathcal{L}_{M_j} \cup \{m_{j,i}\}$                   / $G_8$-$G_{10}$
12         $(K_{j,i}, r_{j,i}) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m_{j,i})$           / $G_8$-$G_{10}$
13         $\mathsf{F}(\mathsf{ID}(pk_j), m_{j,i}) \xleftarrow{\$} \{0, 1\}^k \times \mathcal{R}$    / $G_9$ -$G_{10}$
14 $b \xleftarrow{\$} \{0, 1\}$
15 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Decaps}, \mathsf{Chall}, |\mathsf{F}\rangle}(\vec{pk})$
16 return $[\![b' = b]\!]$

$\underline{\mathsf{Chall}(j)}$                                                 / $G_7$
17 $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_{M_j}$
18 $\mathcal{L}_{M_j} := \mathcal{L}_{M_j} \cup \{m\}$
19 $(K_0, r) \leftarrow \mathsf{F}(\mathsf{ID}(pk_j), m)$
20 $c \leftarrow \mathsf{Enc}(pk_j, m; r)$
21 $K_1 \xleftarrow{\$} \{0, 1\}^k$
22 $\mathcal{L}_{C_j} := \mathcal{L}_{C_j} \cup \{c\}$
23 return $(c, K_b)$

$\underline{\mathsf{Chall}(j)}$                                                 / $G_8$-$G_{10}$
24 $K_0 := K_{j,\mathsf{ctr}_j}$
25 $c \leftarrow \mathsf{Enc}(pk_j, m_{j,\mathsf{ctr}_j}; r_{j,\mathsf{ctr}_j})$                   / $G_8$-$G_9$
26 $c \xleftarrow{\$} \mathsf{Enc}(pk_j, 0)$                                      / $G_{10}$
27 $\mathcal{L}_{C_j} := \mathcal{L}_{C_j} \cup \{c\}$
28 $\mathsf{ctr}_j := \mathsf{ctr}_j + 1$
29 $K_1 \xleftarrow{\$} \{0, 1\}^k$
30 return $(c, K_b)$
```

Figure 9: Games G_7 - G_{10}. Apart from the reprogramming in line 13, all oracles except of the Chall oracle remain defined as in game G_7 (see Fig. 7). For $j \in [n]$, we initialize the counters ctr_j to 1.

an element of S. In order to upper bound Pr[FIND], we dissect S into the set S_1 of (identifiers and) messages that were actually used by Chall, and the set S_2 of (identifiers and) messages that were not used by Chall. More formally, we define the set

$$S_1 := \left\{(\mathsf{ID}(pk_j), m_{j,i}) \mid j \in [n], i \in [q_{\mathsf{C},j}]\right\},$$

where $q_{\mathsf{C},j}$ denotes the number of queries that were issued to Chall on index j, and we let $S_2 := S \setminus S_1$. Accordingly, we can now also dissect event FIND into the event FIND_1 that a query was collapsed to an element of S_1, and into the event FIND_2 that a query was collapsed to an element of S_2. We have that

$$\Pr[\mathsf{FIND}] \leq \Pr[\mathsf{FIND}_1] + \Pr[\mathsf{FIND}_2].$$

Note that the suffixes of the elements in S_2 are uniformly random (without repetition), and that these elements were used nowhere in the game except for the puncturing of F. In particular, they are independent of $\mathcal{A}$'s input. Since S_2 consists of $q_\mathsf{C} \cdot (n-1)$ many elements, we can now apply Eq. (8) of Lemma 5.3 to argue that

$$\Pr[\mathsf{FIND}_2] \leq \frac{4(q_\mathsf{F} + q_\mathsf{D}) \cdot q_\mathsf{C} \cdot (n-1)}{|\mathcal{M}|}.$$

It remains to upper bound $\Pr[\mathsf{FIND}_1]$, for which we will first give a reduction to (n, q_C)-IND-CPA. A (n, q_C)-IND-CPA reduction $\mathcal{B}$ can sample the challenge plaintexts and the plaintexts from S_2 before running the adversary, it can hence perfectly simulate the puncturing. Whenever our reduction $\mathcal{B}$ receives a challenge query from $\mathcal{A}$ on an index j, it queries its own challenge oracle on $j, m_0 :=$

$m_{j,\mathrm{ctr}_j}, m_1 := 0$. It uses the response of CHAL to simulate either game G_9 with punctured oracles (if the (n, q_C)-IND-CPA bit is 0) or game G_{10} with punctured oracles (if the (n, q_C)-IND-CPA bit is 1), we hence have

$$\begin{aligned}\Pr[\mathsf{FIND}_1] \leq & \mathrm{Adv}_{\mathsf{PKE}}^{(n,q_\mathsf{C})\text{-IND-CPA}}(\mathcal{B}) \\ & + \Pr[\mathsf{FIND}_1 \text{ in } G_{10} \text{ with punctured oracle}] \ .\end{aligned}$$

Since in game G_{10}, the suffixes of the elements in the set S_1 are also independent of A's input, and since S_1 has q_C many elements, we can now apply Eq. (8) of Lemma 5.3 again to argue that

$$\Pr[\mathsf{FIND}_1 \text{ in } G_{10} \text{ with punctured oracle}] \leq \frac{4(q_\mathsf{F} + q_\mathsf{D}) \cdot q_\mathsf{C}}{|\mathcal{M}|} \ .$$

Adding the bounds yields the claimed bound in the theorem, concluding our proof.

Remark on the running time of the reductions. The running time of the reductions depends on the implementation of data structures. Naively, the running time is about that of the CCA adversary, plus additional quadratic terms in the number of hashing, challenge and decapsulation queries. Using techniques from [19] one can generate for example the plaintexts by means of a (strong) pseudorandom permutation/ideal cipher (e.g., $m_i = \mathsf{PRP}(i)$). Then the reduction only needs to keep track of a counter, which it can use to implement set insertions and set membership tests efficiently. In that case (ignoring logarithmic terms), the running time of the reductions becomes about that of the CCA adversary (assuming that evaluation and inversion of the PRP and other primitives as the encryption algorithm is in O(1)). Similar statements hold for the quantum case, except that a fully-quantum secure PRP or quantum-accesible ideal cipher is needed, since the adversary gets quantum access through the quantum random oracle. Alternatively, using balanced search trees, one should get similar running times as using the PRP (up to logarithmic factors), but larger memory usage. Also note that $\mathcal{B}$ has slightly worse running times (factor n on queries to F) than $\mathcal{A}$ in Theorem 3.1, which is an argument *for* public-key/prefix hashing.

□

ACKNOWLEDGMENTS

We would like to thank Mike Hamburg for his helpful comments. Julien Duman was supported by the EU H2020 PROMETHEUS project 780701. Eike Kiltz was supported by the BMBF iBlockchain project, the EU H2020 PROMETHEUS project 780701, DFG SPP 1736 Big Data, and the DFG Cluster of Excellence 2092 CASA. Vadim Lyubashevsky and Gregor Seiler are supported by the EU H2020 ERC Project 101002845 "PLAZA".

REFERENCES

[1] Andris Ambainis, Mike Hamburg, and Dominique Unruh. 2019. Quantum Security Proofs Using Semi-classical Oracles. In *CRYPTO 2019, Part II (LNCS)*, Alexandra Boldyreva and Daniele Micciancio (Eds.), Vol. 11693. Springer, Heidelberg, 269–295. https://doi.org/10.1007/978-3-030-26951-7_10

[2] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. 2009. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *CRYPTO 2009 (LNCS)*, Shai Halevi (Ed.), Vol. 5677. Springer, Heidelberg, 595–618. https://doi.org/10.1007/978-3-642-03356-8_35

[3] Sanjeev Arora and Rong Ge. 2011. New Algorithms for Learning in Presence of Errors. In *ICALP 2011, Part I (LNCS)*, Luca Aceto, Monika Henzinger, and Jiri Sgall (Eds.), Vol. 6755. Springer, Heidelberg, 403–415. https://doi.org/10.1007/978-3-642-22006-7_34

[4] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. 2020. Practical Product Proofs for Lattice Commitments. In *CRYPTO (2) (Lecture Notes in Computer Science)*, Vol. 12171. Springer, 470–499.

[5] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. 1998. Quantum Lower Bounds by Polynomials. In *39th FOCS*. IEEE Computer Society Press, 352–361. https://doi.org/10.1109/SFCS.1998.743485

[6] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. 2000. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *EUROCRYPT 2000 (LNCS)*, Bart Preneel (Ed.), Vol. 1807. Springer, Heidelberg, 259–274. https://doi.org/10.1007/3-540-45539-6_18

[7] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-Key model and a general forking lemma. In *ACM CCS 2006*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, 390–399. https://doi.org/10.1145/1180405.1180453

[8] Daniel Bernstein. Accessed May 2021. eBACS: ECRYPT Benchmarking of Cryptographic Systems. (Accessed May 2021). `https://bench.cr.yp.to/results-kem.html`.

[9] Daniel J. Bernstein. 2015. Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996. (2015). https://eprint.iacr.org/2015/996.

[10] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. 2019. Tighter Proofs of CCA Security in the Quantum Random Oracle Model. In *TCC 2019, Part II (LNCS)*, Dennis Hofheinz and Alon Rosen (Eds.), Vol. 11892. Springer, Heidelberg, 61–90. https://doi.org/10.1007/978-3-030-36033-7_3

[11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random Oracles in a Quantum World. In *ASIACRYPT 2011 (LNCS)*, Dong Hoon Lee and Xiaoyun Wang (Eds.), Vol. 7073. Springer, Heidelberg, 41–69. https://doi.org/10.1007/978-3-642-25385-0_3

[12] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *EuroS&P*. IEEE, 353–367.

[13] Hao Chen, Kristin E. Lauter, and Katherine E. Stange. 2016. Security Considerations for Galois Non-dual RLWE Families. In *SAC 2016 (LNCS)*, Roberto Avanzi and Howard M. Heys (Eds.), Vol. 10532. Springer, Heidelberg, 443–462. https://doi.org/10.1007/978-3-319-69453-5_24

[14] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. 2018. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. In *AFRICACRYPT (Lecture Notes in Computer Science)*, Vol. 10831. Springer, 282–305.

[15] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2021. Online-Extractability in the Quantum Random-Oracle Model. Cryptology ePrint Archive, Report 2021/280. (2021). https://eprint.iacr.org/2021/280.

[16] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO'99 (LNCS)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, Heidelberg, 537–554. https://doi.org/10.1007/3-540-48405-1_34

[17] Eiichiro Fujisaki and Tatsuaki Okamoto. 2013. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology* 26, 1 (Jan. 2013), 80–101. https://doi.org/10.1007/s00145-011-9114-1

[18] Mike Hamburg. 201. Post-quantum cryptography proposal: ThreeBears. https://www.shiftleft.org/papers/threebears/nist-submission.pdf. (201).

[19] Mike Hamburg. 2019. Security proof of ThreeBears. https://www.shiftleft.org/papers/threebears/proof-round2.pdf. (2019).

[20] Johan Håstad. 1988. Solving Simultaneous Modular Equations of Low Degree. *SIAM J. Comput.* 17, 2 (1988), 336–341. https://doi.org/10.1137/0217019

[21] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. 2017. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *TCC 2017, Part I (LNCS)*, Yael Kalai and Leonid Reyzin (Eds.), Vol. 10677. Springer, Heidelberg, 341–371. https://doi.org/10.1007/978-3-319-70500-2_12

[22] Kathrin Hövelmanns. 2021. *Generic constructions of quantum-resistant cryptosystems.* doctoralthesis. Ruhr-Universität Bochum, Universitätsbibliothek. https://doi.org/10.13154/294-7758

[23] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. 2020. Generic Authenticated Key Exchange in the Quantum Random Oracle Model. In *PKC 2020, Part II (LNCS)*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.), Vol. 12111. Springer, Heidelberg, 389–422. https://doi.org/10.1007/978-3-030-45388-6_14

[24] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. 2018. IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. In *CRYPTO 2018, Part III (LNCS)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10993. Springer, Heidelberg, 96–125. https://doi.org/10.1007/978-3-319-96878-0_4

[25] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. 2019. Key Encapsulation Mechanism with Explicit Rejection in the Quantum Random Oracle Model. In *PKC 2019,*

Part II (LNCS), Dongdai Lin and Kazue Sako (Eds.), Vol. 11443. Springer, Heidelberg, 618–645. https://doi.org/10.1007/978-3-030-17259-6_21

[26] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. 2019. Tighter Security Proofs for Generic Key Encapsulation Mechanism in the Quantum Random Oracle Model. In *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, Jintai Ding and Rainer Steinwandt (Eds.). Springer, Heidelberg, 227–248. https://doi.org/10.1007/978-3-030-25510-7_13

[27] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. 2020. Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security. In *EUROCRYPT 2020, Part III (LNCS)*, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12107. Springer, Heidelberg, 703–728. https://doi.org/10.1007/978-3-030-45727-3_24

[28] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* 75, 3 (2015), 565–599.

[29] Asheesh Laroia. 2011. Short key IDs are bad news (with OpenPGP and GNU Privacy Guard). http://www.asheesh.org/note/debian/short-key-ids-are-bad-news.html. (2011).

[30] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT 2010 (LNCS)*, Henri Gilbert (Ed.), Vol. 6110. Springer, Heidelberg, 1–23. https://doi.org/10.1007/978-3-642-13190-5_1

[31] NIST. 2016. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. (2016).

[32] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, Harold N. Gabow and Ronald Fagin (Eds.). ACM Press, 84–93. https://doi.org/10.1145/1060590.1060603

[33] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. 2018. Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model. In *EUROCRYPT 2018, Part III (LNCS)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.), Vol. 10822. Springer, Heidelberg, 520–551. https://doi.org/10.1007/978-3-319-78372-7_17

6 QUANTUM PRELIMINARIES

Qubits. For simplicity, we will treat a *qubit* as a vector $|\varphi\rangle \in \mathbb{C}^2$, i.e., a linear combination $|\varphi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle$ of the two *basis states* (vectors) $|0\rangle$ and $|1\rangle$ with the additional requirement to the probability amplitudes $\alpha, \beta \in \mathbb{C}$ that $|\alpha|^2 + |\beta|^2 = 1$. The basis $\{|0\rangle, |1\rangle\}$ is called *standard orthonormal computational basis*. The qubit $|\varphi\rangle$ is said to be *in superposition*. Classical bits can be interpreted as quantum bits via the mapping $(b \mapsto 1 \cdot |b\rangle + 0 \cdot |1-b\rangle)$.

Quantum Registers. We will treat a quantum register as a collection of multiple qubits, i.e. a linear combination $|\varphi\rangle := \sum_{x \in \{0,1\}^n} \alpha_x \cdot |x\rangle$, where $\alpha_x \in \mathbb{C}$, with the additional restriction that $\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1$. As in the one-dimensional case, we call the basis $\{|x\rangle\}_{x \in \{0,1\}^n}$ the *standard orthonormal computational basis*. We say that $|\varphi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x \cdot |x\rangle$ *contains the classical query* x if $\alpha_x \neq 0$.

Measurements. Qubits can be measured with respect to a basis. In this paper, we will only consider measurements in the standard orthonormal computational basis, and denote this measurement by $\textsc{Measure}(\cdot)$, where the outcome of $\textsc{Measure}(|\varphi\rangle)$ for a single qubit $|\varphi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle$ will be 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$, and the outcome of measuring a qubit register $|\varphi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x \cdot |x\rangle$ will be x with probability $|\alpha_x|^2$. Note that the amplitudes *collapse* during a measurement, this means that by measuring $\alpha \cdot |0\rangle + \beta \cdot |1\rangle$, α and β are switched to one of the combinations in $\{\pm(1,0), \pm(0,1)\}$. Likewise, in the n-dimensional case, all amplitudes are switched to 0 except for the one that belongs to the measurement outcome and which will be switched to 1.

Quantum oracles and quantum Adversaries. Following [5, 11], we view a quantum oracle $|\mathsf{O}\rangle$ as a mapping

$$|x\rangle|y\rangle \mapsto |x\rangle|y \oplus \mathsf{O}(x)\rangle \ ,$$

where $\mathsf{O} : \{0,1\}^n \to \{0,1\}^m$, and model quantum adversaries A with access to O by a sequence $U_1, |\mathsf{O}\rangle, U_2, \cdots, |\mathsf{O}\rangle, U_N$ of unitary transformations. We write $\mathsf{A}^{|\mathsf{O}\rangle}$ to indicate that the oracles are quantum-accessible (contrary to oracles which can only process classical bits).

Quantum random oracle model. We consider security games in the quantum random oracle model (QROM) as their counterparts in the classical random oracle model, with the difference that we consider quantum adversaries that are given *quantum* access to the (offline) random oracles involved, and *classical* access to all other (online) oracles.

7 PROOF OF THEOREM 3.1 (PART 2)

Proof. Let $\mathcal{A}$ be an adversary and consider the games given in Fig. 10.

Game G_0. This is the original (n, q_{C})-IND-CCA game. Thus,

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \frac{1}{2}\right| = \mathsf{Adv}_{\mathsf{KEM}}^{(n,q_{\mathsf{C}})\text{-IND-CCA}}(\mathcal{A}) \,.$$

Game G_1. In game G_1 the challenge oracle samples m from $\mathcal{M} \setminus \mathcal{L}_M$ instead of $\mathcal{M}$, since on a message collision an adversary can easily distinguish the real KEM key from the random one. By the birthday bound,

$$\left|\Pr\left[G_0^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \frac{{q_{\mathsf{C}}}^2}{|\mathcal{M}|} \,.$$

Game G_2. In game G_2 we modify the Decaps oracle in lines 21 and 22 such that for an invalid ciphertext the key is defined as $\mathsf{K}'_j(c)$, where K'_j is an independent internal random oracle. This remains unnoticed to adversary $\mathcal{A}$ unless it queries

$$\mathsf{K}(s_j, \cdot) \,,$$

for $j \in [1, n]$ or there exists a collision $s_i = s_j$, that is there exists $i, j \in [n]$ s.t. $s_i = s_j$, in which case the event BAD is set to true.

Since the seeds $s_j \in \{0,1\}^\lambda$ are uniformly random and information-theoretically hidden from the adversary, we have by the birthday and union bound

$$\left|\Pr\left[G_2^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_1^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathrm{BAD}] \leq \frac{n^2 + n \cdot q_{\mathsf{F}}}{2^\lambda} \,.$$

Game G_3 In game G_3 we start introducing changes to the decapsulation oracle Decaps$(j, \cdot)$ with the goal of simulating decapsulation queries without the secret-keys by additional changes introduced in game G_4. In more detail, for $m' := \mathrm{Dec}(sk_j, c)$ we return the "implicit reject key" $\mathsf{K}'_j(c)$ if m' was not queried on F, and otherwise return the key K from $(K, r) \leftarrow \mathsf{F}(m')$. In other words, we consider only ciphertexts as valid, where the adversary first queries G (indirectly through F). The probability that a ciphertext is valid without first querying G is bounded by the γ-spreadness. Note that this step is necessary due to not domain separating the random oracle H in order to guarantee the consistency of the decapsulation query lists $\mathcal{L}_{D_j}$ and the random oracle list $\mathcal{L}_{\mathsf{H}}$. To see why this is necessary, consider two ciphertexts c_1 and c_2 which were not derived by making queries first to G (indirectly through F). A priori,

```
GAMES $G_0$ - $G_5$
01 for $j \in [n]$
02     $(pk_j, sk_j) \xleftarrow{\$} \mathsf{Gen}$
03     $s_j \xleftarrow{\$} \{0,1\}^\lambda$
04 if $\exists i, j \in [n]$ s.t. $s_i = s_j$; BAD := true; abort          / $G_2$-$G_5$
05 $\vec{pk} \leftarrow (pk_1, \ldots, pk_n)$
06 $b \xleftarrow{\$} \{0,1\}$
07 $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Decaps},\mathsf{Chall},\mathsf{F}}(\vec{pk})$
08 return $[\![b' = b]\!]$

Chall($j$)
09 $m \xleftarrow{\$} \mathcal{M}$
10 $m \xleftarrow{\$} \mathcal{M} \setminus \mathcal{L}_M$          / $G_1$-$G_5$
11 $(K_0, r) \leftarrow \mathsf{F}(m)$
12 $\mathcal{L}_M := \mathcal{L}_M \cup \{m\}$
13 $r \xleftarrow{\$} \mathcal{R}$; $K_0 \xleftarrow{\$} \{0,1\}^k$          / $G_5$
14 $K_1 \xleftarrow{\$} \{0,1\}^k$
15 $c \leftarrow \mathsf{Enc}(pk_j, m; r)$
16 $\mathcal{L}_{C_j} = \mathcal{L}_{C_j} \cup \{c\}$
17 return $(c, K_b)$

Decaps($j, c \notin \mathcal{L}_{C_j}$)          / $G_0$-$G_2$
18 $m' := \mathsf{Dec}(sk_j, c)$
19 $(K, r) \leftarrow \mathsf{F}(m')$
20 if $m' = \bot$ or $\mathsf{Enc}(pk_j, m'; r) \neq c$
21     $(K, r) \leftarrow \mathsf{F}(s_j, c)$          / $G_0$
22     $K := \mathsf{K}'_j(c)$          / $G_2$
23 return $K$

Decaps($j, c \notin \mathcal{L}_{C_j}$)          / $G_3$
24 $m' := \mathsf{Dec}(sk_j, c)$
25 if $\exists (m, r, K) \in \mathcal{L}_\mathsf{F}$ with $\mathsf{Enc}(pk_j, m; r) = c$ and $m = m'$
26     return $K$
27 $K := \mathsf{K}'_j(c)$
28 return $K$

Decaps($j, c \notin \mathcal{L}_{C_j}$)          / $G_4$-$G_5$
29 if $\exists K$ s. th. $(c, K) \in \mathcal{L}_{D_j}$
30     return $K$
31 $K \xleftarrow{\$} \mathcal{K}$
32 $\mathcal{L}_{D_j} := \mathcal{L}_{D_j} \cup \{(c, K)\}$
33 return $K$

F($A$)
34 if $A \in \mathcal{M}$
35     $m := A$
36     $(K, r) := (\mathsf{H}(m), \mathsf{G}(m))$
37 if $A \in \{0,1\}^\lambda \times C$
38     $(s, c) := A$
39     $K := \mathsf{K}(s, c)$
40 if $K$ undefined: $K \xleftarrow{\$} \mathcal{K}$
41 if $r$ undefined: $r \xleftarrow{\$} \mathcal{R}$
42 return $(K, r)$

G($m$)          / Internal random oracle
43 $r \xleftarrow{\$} \mathcal{R}$
44 if $m \in \mathcal{L}_M$          / $G_5$
45     QUERY := true          / $G_5$
46     abort          / $G_5$
47 return $r$

K($s, c$)          / Internal random oracle
48 $K \xleftarrow{\$} \mathcal{K}$
49 if $s \in \{s_1, \ldots, s_n\}$          / $G_2$-$G_5$
50     BAD := true          / $G_2$-$G_5$
51     abort          / $G_2$-$G_5$
52 return $K$

H($m$)          / Internal random oracle
53 $K \xleftarrow{\$} \mathcal{K}$
54 if $m \in \mathcal{L}_M$          / $G_5$
55     QUERY := true          / $G_5$
56     abort          / $G_5$
57 for $j \in [n]$ :          / $G_4$-$G_5$
58     $c'_j := \mathsf{Enc}(pk_j, m; \mathsf{G}(m))$          / $G_4$-$G_5$
59     $\mathcal{L}_{D_j} := \mathcal{L}_{D_j} \cup \{(c'_j, K)\}$          / $G_4$-$G_5$
60 return $K$
```

Figure 10: Games G_0 - G_4 for the proof of Theorem 3.1. The internal random oracles $\mathsf{H}'_j, \mathsf{H}_j, \mathsf{G}_j, \mathsf{K}_j$ are not accessible by the adversary. We assume wlog that F is only queried once on each value A.

we can not exclude the possibility that c_1 under secret-key sk_1 decrypts to the same m as c_2 under the secret-key sk_2. With this game hop what we are essentially showing is that this does not matter, since by γ-spreadness, the re-encryption check will not pass, and thus the KEM keys will be independent. Without this argument, the simulation would return different KEM keys although we have not excluded that re-encryption might be successful, where the simulation would then need to return the same KEM key, i.e. $\mathsf{H}(m)$. By the weak γ-spreadness we have by a union bound over the n users and q_D decapsulation queries

$$\left|\Pr\left[G_2^\mathcal{A} \Rightarrow 1\right] - \Pr\left[G_3^\mathcal{A} \Rightarrow 1\right]\right| \leq q_\mathrm{D} n 2^{-\gamma} .$$

Game G_4. In game G_4 we simulate the decapsulation oracle $\mathsf{Decaps}(j, \cdot)$ without knowledge of the secret key by patching the random oracle H in lines 57-59 and 29-33. Note that if PKE was perfectly correct, then the random oracle patching is also perfectly correct and therefore two games would look identical in $\mathcal{A}$'s view.

The only bad case happens if G is queried on some m which induces a correctness error, that is $\mathsf{Dec}(sk_j, \mathsf{Enc}(pk_j; m; \mathsf{G}(m))) \neq$

m. More concretely, define the sets $\mathrm{BAD}_j := \{m \in \mathcal{M} \mid m \neq m', \text{where } c \leftarrow \mathsf{Enc}(pk_j; m; \mathsf{G}(m)); m' \leftarrow \mathsf{Dec}(sk_j, c)\}$.

Define the event CORR_j to be the event that $\mathcal{A}$ makes an (implicit) query to $\mathsf{G}(m)$ for some $m \in \mathrm{BAD}_j$. Since there are at most $(q_\mathsf{F} + q_\mathsf{D})$ explicit and implicit queries to G, we have

$$\Pr\left[\mathsf{CORR}_j\right] \leq (q_\mathsf{F} + q_\mathsf{D})\delta\,.$$

Defining $\mathsf{CORR} := \bigcup_{j=1}^{n} \mathsf{CORR}_j$, we obtain by the union bound[4]

$$\Pr[\mathsf{CORR}] \leq n(q_\mathsf{F} + q_\mathsf{D})\delta\,.$$

We now claim

$$\left|\Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_3^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{CORR}]\,.$$

Let us analyze why G_4 and G_3 are identical conditioned on $\neg\mathsf{CORR}$. Consider a query $\mathsf{Decaps}(j, c)$ and define $m'_j := \mathsf{Dec}(sk_j, c)$ and $c'_j := \mathsf{Enc}(pk_j, m'_j; \mathsf{G}(m'_j))$.

- **Case 1:** $m'_j = \bot$. H cannot be called on $m'_j = \bot$ and hence the KEM key of $K = \mathsf{Decaps}(sk_j, c) = \mathsf{K}'_j(c)$ in G_3 is identically distributed as $(c, K) \in \mathcal{L}_{D_j}$ in G_4.
- **Case 2:** $m'_j \neq \bot \wedge c \neq c'_j$. Both games return a uniform random key K. The only way for $\mathcal{A}$ to detect a difference between the two games is if it makes a query $\mathsf{H}(m)$ such that $\mathsf{Enc}(pk_j, m; \mathsf{G}(m)) = c$. But conditioned that $m \notin \mathrm{BAD}_j$, c does not lie in the image space of encryption $\mathsf{Im}(\mathsf{Enc}_{pk_j}(-; \mathsf{G}(-))$, since the re-encryption check failed and therefore such an m cannot exist.
- **Case 3:** $m'_j \neq \bot \wedge c = c'_j$. In game G_3, $\mathsf{Decaps}(j, c)$ returns $K = \mathsf{H}(m'_j)$, whereas G_4 picks a uniform K in $\mathsf{H}(m)$ to match K for all m that deterministically encrypt to the same c, i.e., all m satisfying $\mathsf{Enc}(pk_j, m; \mathsf{G}(m)) = c$ and saves those values in $\mathcal{L}_{D_j}$. Since $c = c'_j$ the re-encryption check had to pass and H was queried before Decaps. The only way for $\mathcal{A}$ to detect a difference between the two games is to query H on some value $m \neq m'_j$ that also deterministically encrypts to the same c. But this also implies that $\mathsf{G}(m)$ was queried for some $m \in \mathrm{BAD}_j$.

In order to see why we had to use γ-spreadness, consider $\mathsf{Decaps}(i, c)$ and define $m'_i := \mathsf{Dec}(sk_i, c_i)$ and $c'_i := \mathsf{Enc}(pk_i, m'_i; G(m'_i))$. Then, there are two additional cases to consider, since the random oracle H is shared between the users.

- Case 1: $m'_j = m'_i$. Then either both c and c_i are valid ciphertexts or both of them are invalid, i.e. do not pass the re-encryption check. If the adversary first queries F (and implicitly H) first and then Decaps the oracles of G_3 and G_4 are consistent, due to the patching in lines 57-59. If the adversary queries first Decaps on c and c_i and then H, they will be invalid ciphertexts and therefore both G_3 and G_4 correctly give out independent "implicit rejection" KEM keys (line 27 in G_3 and line 31 in G_4). This case is the reason we introduced the game hop from G_2 to G_3 where we applied γ-spreadness.
- Case 2: $m'_j \neq m'_i$ Here, there are no new potential incosistency problems due to the multi-user setting, which might occur since $H(m'_j)$ and $H(m'_i)$ are independent.

Game G_5. In game G_5 we abort on queries of the form $\mathsf{H}(m)$ or $\mathsf{G}(m)$ for some challenge message $m \in \mathcal{L}_M$, in which case the event QUERY holds true. We have by the difference lemma,

$$\left|\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] - \Pr\left[G_4^{\mathcal{A}} \Rightarrow 1\right]\right| \leq \Pr[\mathsf{QUERY}]\,.$$

Note that in G_5 bit b is independent of the view of the adversary. We thus have

$$\Pr\left[G_5^{\mathcal{A}} \Rightarrow 1\right] = \frac{1}{2}\,.$$

Analogously to (3), we can again prove that there exists an adversary C with

$$\Pr[\mathsf{QUERY}] \leq 2 \cdot \left(\mathsf{Adv}_{\mathsf{PKE}}^{(n, q_\mathsf{C})\text{-IND-CPA}}(C) + \frac{q_\mathsf{F} q_\mathsf{C}}{|\mathcal{M}|}\right)\,.$$

Summing up the inequalities yields the claimed bound, concluding the proof of the theorem. □

[4]Technically, we would need to consider the δ-correctness for fixed keys pk, sk and then take the expectations to be formally correct. We omitt this for brevity.