Thèse n° 10 522

Secure and Efficient Cryptographic Algorithms in a Quantum World

Présentée le 19 avril 2024

Faculté informatique et communications
Laboratoire de sécurité et de cryptographie
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Loïs Evan HUGUENIN-DUMITTAN

Acceptée sur proposition du jury

Prof. O. N. A. Svensson, président du jury
Prof. S. Vaudenay, directeur de thèse
Prof. T. Jager, rapporteur
Prof. P. Schwabe, rapporteur
Prof. C. Troncoso, rapporteuse

École
polytechnique
fédérale
de Lausanne

2024

To my mum, my sister and Jessica.

"And then there's quantum, of course. [...] There's *always* bloody quantum".

— *Night Watch, Terry Pratchett, 2002*

Acknowledgements

First, I want to thank Serge, who supervised this thesis and who has taught me so much. Serge has given me the freedom to choose the topics I wanted to work on and was always supportive of my work. He is also the one who has brought together all the great people in LASEC, and for that I'm really grateful. I also want to thank the members of my jury: Professors Tibor Jager, Peter Schwabe, Ola Svensson, and Carmela Troncoso.

Before beginning my thesis, I was lucky enough to be supervised by several brilliant people who encouraged me to pursue research in cryptography. Among those, I'm most grateful to Sonia, with whom I collaborated at ELCA and who was a fantastic mentor, to Iraklis, with whom I wrote my very first paper and who was kind enough to write me a recommendation letter for my PhD application, and finally to Betül, who helped me take my first steps in the academic world and looked after me at Eurocrypt 2019.

I also want to thank all the students I was entrusted with to supervise and who in turn taught me a lot: Zhendong, Samuel, Jimmy, Marco, Nicolas, Max, Gaétan, Nathan, Marco (another one), Mounir, and Guilhem. During my PhD, the lab hosted many talented Master thesis students, summer interns and research scholars. Whilst I can't mention everyone, I want to thank Phillip for all the nice chats before and during COVID. Many thanks to Burcu as well, who was a great lunch companion last summer.

This brings me to the permanent LASEC members I've been fortunate enough to work with. First, I want to say that I feel privileged to have done my PhD among such bright, kind and interesting people. Starting with the old gen, my thanks to Gwangbae, Fatih, Subhadeep and Hailun. You guys made the lab a welcoming place and I have so many fond memories of the hikes and the moments spent in the old kitchen (I miss the couches).

Now comes the list of people I've spent most of my time over the past few years. I always insisted on calling them *colleagues* but I think it's time to correct this and call them for what they really are: *friends*. Thank you: Khashayar for your edgy jokes and for breathing life into LASEC, Andi for your dark humour and reconciling cryptography with art, Bénédikt for our morning talks and making our life as TAs easier, Abdullah for your kindness, opinions and tricks (but not for bringing the yo-yos), Boris for bringing joy and dad jokes to the lab and for always trying to help, Ritam for the exciting political discussions and tongue-in-cheek comments, and finally Laurane for all our lunch chats, and for standing up for your convictions.

Special thanks to Daniel, with whom I have shared an office these last four years and who has been one of my best friends throughout this journey. Thanks for the endless jokes, laughs, more-or-less serious debates and, with any spare time, the fruitful research discussions. I

also want to express my gratitude to Novak and Aymeric, who brought a bit of LACAL and Kudelski to the lab, and to Martine and Sylvie, for their tremendous help, their kindness and their generosity.
I somehow managed to befriend several fellow PhD students in other labs during my time at EPFL. I'm particularly grateful to Aditya for all the delightful chats we had from day 1, to Simone for the good times sharing an office and beers, and to Jade for our doorway discussions.
I want to thank all my friends, in Neuchâtel and elsewhere. Thanks to Loïc and Fanny for the great times in Saas-Fee and failing escape games together. My heartfelt thanks to the Neuch crew aka *Les Dôleux* aka Antoine, Gaston, Guillaume, Morgan and Philéas. You guys have been there forever, thanks for all the great nights, your support and, most of all, your unwavering sense of humour.
Finally, I want to thank my (extended) family. In particular, I wish to express my utmost gratitude to the three most important women in my life: to my sister Julia for continuously being a role-model and trying to fix the world, to my mum Chantal for sparking and fostering my curiosity, encouraging me and always putting us first, and to Jess for sharing my life, your unfaltering support and for always lifting my spirits during difficult times.

Lausanne, March 21, 2024 L. H.

Abstract

Since the advent of internet and mass communication, two public-key cryptographic algorithms have shared the monopoly of data encryption and authentication: Diffie-Hellman and RSA. However, in the last few years, progress made in quantum physics –and more precisely in quantum computing– has changed the state of affairs. Indeed, since Shor's algorithm was published in 1994, we know that both Diffie-Hellman and RSA could be broken by a quantum computer. This motivated the National Institute of Standards and Technology in the US (NIST) to launch in 2017 a call for Key-Encapsulation Mechanism (KEM) and Signature schemes that resist quantum computers, i.e. *Post-Quantum* schemes.
An important building block that is used in the construction of most Post-Quantum KEMs is the Fujisaki-Okamoto (FO) transform, that compiles a passively secure (IND-CPA) KEM into an actively secure (IND-CCA) one. In short, the transform works by modifying the underlying decryption procedure as follows: the ciphertext is decrypted into some plaintext, which is output only if its re-encryption is equal to the input ciphertext.

In this thesis, we first focus on the security of Post-Quantum KEMs. In particular, we show that it is critical that the FO transform is properly implemented and never leaks information on the decrypted plaintext unless the re-encryption check passes. More precisely, for many of the KEMs proposed to the NIST standardisation process, we demonstrate that it is possible to recover the secret key with a few thousand decryptions if the leakage mentioned above is present. We then prove that schemes based on the rank metric, such as RQC, are somewhat immune to our kind of attacks.

We then focus on combiners, or how to combine several primitives together to obtain a more secure one. We introduce a construction that generalises the FO transform by taking several IND-CPA Public-Key Encryption schemes (PKEs) and outputting one IND-CCA KEM that is secure as long as *one* of the underlying PKEs is secure. This is an interesting property as many of the assumptions Post-Quantum cryptography is based on are relatively new and have been less studied, and are therefore more likely to suffer a devastating cryptanalysis.

Then, based on the observation that the re-encryption step in the FO transform is expensive, we tackle the question of whether this can be improved. It turns out that a previous result by Gertner et al. rules out such a possibility in the classical model, in other words an IND-CPA to IND-CCA black-box transform *must* re-encrypt in the decryption. We generalise this impossibility result to the post-quantum setting.

In a subsequent chapter, we show that if the security requirement can be lowered from IND-CCA to IND-qCCA (i.e. the adversary can only obtain a *constant* number q of decryptions), the re-encryption is actually not needed. We also observe that this security notion is sufficient in many applications, making this result most impactful. Using similar proof techniques, we then solve a theoretical open question and prove that *IND-CPA* KEMs can be used in TLS 1.3 instead of Diffie-Hellman.

Finally, we present K-Waay, a Post-Quantum replacement for the X3DH key-exchange that is notably used in Signal and WhatsApp. Our protocol is faster than previous work and the only non-standard primitive used is a variant of the well-studied Frodo key-exchange.

Résumé

Depuis l'avènement d'Internet, deux systèmes cryptographiques se sont historiquement partagés le monopole du chiffrement des données et de l'authentification : Diffie-Hellman et RSA. Cependant, les récentes avancées en physique quantique – et plus précisément en informatique quantique – ont bouleversé cet état de fait. En effet, on sait depuis la publication de l'algorithme de Shor en 1994 que Diffie-Hellman et RSA peuvent être cassés par un ordinateur quantique. Les progrès dans ce domaine ont incité l'Institut National des Standards et de la Technologie (NIST) aux Etats-Unis à lancer en 2017 un processus afin de trouver un Méchanisme d'Encapsulation de Clé (KEM) et un système de signature digitale qui seraient capables de résister aux ordinateurs quantiques. De tels algorithmes sont appelés "post-quantiques". Un outil important dans la construction des KEMs post-quantiques est la transformée de Fujisaki-Okamato, qui convertit un KEM passivement sûr (IND-CPA) en un KEM activement sûr (IND-CCA).

Dans la première partie de cette thèse, nous nous concentrons sur la sécurité des KEMs post-quantiques. Nous montrons d'abord qu'il est indispensable que la transformée de Fujisaki-Okamoto soit implémentée correctement. Plus précisément, pour la plupart des KEMs proposés à la standardisation, nous démontrons qu'il est possible de retrouver la clé secrète après quelques milliers de déchiffrements si un certain type d'information fuite. Nous prouvons ensuite que des systèmes basés dans la métrique de rang tel que RQC sont plus résistants à ce genre d'attaques.

Nous nous concentrons ensuite sur la notion de combineurs : des algorithmes qui combinent plusieurs primitives cryptographiques ensemble afin d'obtenir un nouveau système plus sûr. Nous présentons une construction qui généralise la transformée de Fujisaki-Okamoto en ce sens qu'elle prend plusieurs systèmes à chiffrement publique (PKEs) et retourne un KEM qui est IND-CCA du moment *qu'un seul* des PKEs sous-jacents est IND-CPA.

Fort du constat que l'étape de rechiffrement dans la transformée de Fujisaki-Okamoto est coûteuse en terme de temps, nous nous attaquons à la question de savoir si ce calcul est réellement nécessaire. Il s'avère qu'un résultat précédent de Gertner et al. confirme que c'est le cas dans le modèle classique. En d'autres mots, les auteurs démontrent qu'une transformée en mode boîte-noire entre un algorithme IND-CPA et un autre IND-CCA nécessite un rechiffrement dans la fonction de déchiffrement. Nous généralisons ce résultat au monde post-quantique.

Dans le chapitre qui suit, nous prouvons que si la sécurité peut être abaissée à IND-qCCA (c'est-à-dire que l'adversaire peut obtenir un nombre *constant q* de textes déchiffrés), cette étape de rechiffrement n'est pas indispensable. Nous soulignons aussi que la notion de IND-qCCA est suffisante dans pléthore d'applications, ce qui rend ce résultat intéressant aussi bien au niveau pratique que théorique. Enfin, en utilisant notre technique de preuve, nous résolvons une question ouverte en prouvant qu'un KEM IND-CPA peut être utilisé dans le protocole TLS 1.3 à la place de Diffie-Hellman.

Enfin, nous présentons K-Waay, une alternative post-quantique au protocole d'échange de clé X3DH notamment utilisé dans Signal et WhatsApp. Notre construction est plus rapide que les protocoles similaires existants et la seule primitive non-standard utilisée est une variante de Frodo.

Contents

1 Introduction

Transmitting gibberish on a channel hoping that *only* the intended recipient will be able to extract meaningful information out of it: that is the original goal of cryptography summed up. From the early primitive ciphers relying on the limited computational capacity of the human brain for security, cryptography evolved into a fully fledged science following the advent of computers in the second half of the 20th century. Along the way, the field started to encompass more concepts than mere confidentiality of information, like authenticity, integrity or, more recently, deniability. A significant milestone in the history of cryptography is the discovery of public-key cryptography in the 1970s, and in particular the invention of the Diffie-Hellman (DH) key exchange and the RSA encryption and signature schemes. The great strength of these constructions is that they can be proven secure assuming the computational hardness of a problem. This concept is central to the present thesis and is known today as *provable security*.

Meanwhile, in the seemingly unrelated field of physics, progress in quantum mechanics resulted in the formalisation of *quantum computing* in the 1980s. Instead of conveying and storing information in electrical voltage as classical computers do, their quantum counterparts process quantum particles like photons to perform computations. The link with cryptography was established in the early 1980s, when Bennett and Brassard [BB84] proposed their quantum key distribution (QKD) algorithm proven secure under the mere laws of physics. However, it is only in 1994 that quantum computers became a serious cause of concern for cryptographers, with the publication by Shor [Sho94] of a quantum algorithm that could solve both the discrete logarithm and factoring problems, which underpin the security of Diffie-Hellman and RSA.

Post-Quantum cryptography & NIST standardisation process. While only theoretical at first, the risk posed by Shor's algorithm was seen by many as an opportunity to develop cryptosystems based on computational problems that could resist quantum computers and offer interesting properties. The study of these quantum resistant schemes is what is known as *post-quantum* (PQ) cryptography and is the topic of this thesis. Among these post-quantum assumptions, the most famous one is probably the *learning with error* (LWE) hardness assumption proposed by Regev in 2005 [Reg05], which postulates that, given $(a, a\cdot s+e)$ where

$a, s \in \mathbb{Z}_q^n, e \in \mathbb{Z}_q$, and s, e are "small", recovering s is hard. Since its introduction, the LWE problem or variants of it have been used in countless applications, from public-key encryption to multi-party computation.

In the 2010s, efforts and breakthroughs in quantum computing started to make part of the security and cryptography community feel uneasy. This led the US National Institute of Standards and Technology (NIST), in 2017, to launch a call for standardisation of post-quantum public-key schemes. More than 60 proposals of key-encapsulation mechanisms (KEMs), public-key encryption (PKEs) and signature schemes were received. In 2022, the NIST decided to standardise one KEM and three signature schemes, namely CRYSTALS-Kyber, CRYSTALS-Dilithium, Falcon and SPHINCS$^+$. Several other KEMs (BIKE, Classic McEliece, HQC and SIKE) were also labelled as "alternate candidates" and will be considered for later standardisation, except SIKE that was later broken by Castryck and Decru [CD23]. It is worth noting that out of the four algorithms selected, only one (SPHINCS$^+$) is not based on a problem involving lattices. This drove the NIST to issue another call for post-quantum signatures based on different assumptions, which is ongoing at the time of writing.

Overall, the NIST standardisation process brought a spotlight on the field of post-quantum cryptography and induced a massive effort by researchers from all over the world; this thesis is a modest contribution to this endeavour. Quantum computers powerful enough to break cryptography might never exist, but PQ cryptosystems will be deployed and thus need to be secure.

IND-CPA/CCA KEMs and Fujisaki-Okamoto. Let's now dive into the details of post-quantum key-encapsulation mechanisms (KEMs), which are central to this thesis. The notion of a KEM was first proposed by Shoup in 2001 [Sho01] for an ISO standard and can be seen as the formalisation of a public-key encryption (PKE) system that always encrypts a random symmetric key. That is, in a PKE, the encryption algorithm takes the receiver's public key and a message, and outputs a ciphertext, whereas in a KEM, the encapsulation procedure takes the receiver's public key only, and outputs a random key and a ciphertext that contains the key. Both the decryption of a PKE and the decapsulation of a KEM work similarly, except the KEM outputs the key that was encrypted in the ciphertext instead of the message. An illustration of the working flow of both primitives is given in Figure 1.1. Cramer and Shoup [CS03] then proved that combining a KEM with a block cipher (using the symmetric key output by the encapsulation function to encrypt symmetrically the message) was equivalent to building a PKE. This construction, known today as the "KEM/DEM paradigm", turned out to be extremely popular as it reduces the problem of designing a PKE to the problem of designing a KEM. It also gives a simple and efficient recipe for building PKEs for arbitrary long messages.

Two notions of security are usually considered for KEMs: security against passive adversaries also called *security against chosen plaintexts attacks* (CPA) and security against active adversaries, also *called security against chosen ciphertexts attacks* (CCA). Assuming a suitable

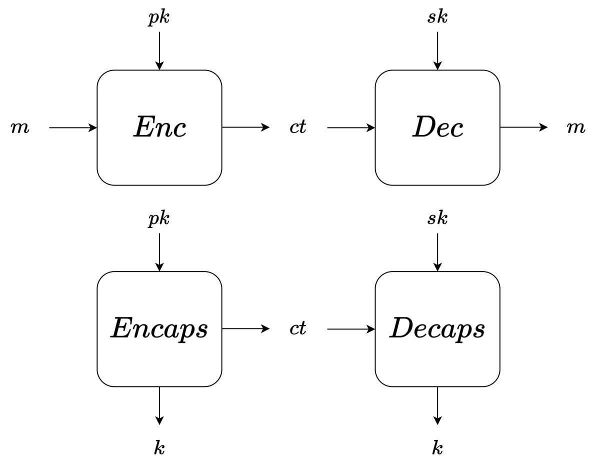


Figure 1.1: Illustration of the difference between a PKE (top) and a KEM (bottom).

block cipher is used, the KEM/DEM paradigm implies that a CPA (resp. CCA) KEM can be used to build a CPA (resp. CCA) PKE. As a consequence, most of the NIST PQ proposals were CCA-secure KEMs and not PKEs. More precisely, the way a vast majority of these schemes are built is as follows:

1. A CPA-secure PKE is built from a post-quantum hardness assumption (e.g. LWE).
2. The CPA-secure PKE is compiled into a CCA-secure KEM using a technique known as the Fujisaki-Okamoto transform.

Thus, in short, a weak PKE is transformed into a strong KEM that can itself be used to build a strong PKE through the KEM/DEM paradigm.

The core component of the recipe given above is the Fujisaki-Okamoto (FO) transform or variations of it (called *FO-like transforms* in this dissertation) that take a CPA-secure scheme and output a CCA-secure one. The original construction was proposed by Fujisaki and Okamoto in 1999 [FO99; FO13] and was building a strong PKE out of a weak one. Newer variants like Hofheinz et al.'s [HHK17] build a KEM out of a PKE and these are the ones used by the NIST candidates. At a high level, these transforms work as follows:

1. The encapsulation function of the KEM samples a random key k and encrypts it with the underlying PKE using the hash of the key k as the source of randomness: $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, k; H(k))$, with H a hash function modelled as a *random oracle*, a concept we discuss in more details in the next paragraph. The ciphertext is then simply ct and the key is k.
2. In order to decapsulate, the KEM first decrypts the ciphertext ct to get the key k': $k' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$. Then, it re-encrypts k' into a ciphertext ct': $\mathsf{ct}' \leftarrow \mathsf{Enc}(\mathsf{pk}, k'; H(k'))$. Finally, it outputs the key k' if $\mathsf{ct} = \mathsf{ct}'$, otherwise nothing (or an error).

Informally, we see that due to the de-randomisation step in the encapsulation, each key is associated to a unique ciphertext. Then, in the decapsulation, the KEM checks that the ciphertext corresponds to the decrypted key and aborts otherwise.

ROM. One thing that might seem off in the FO transform presented above is the use of $H(k)$ as the source of randomness. Indeed, one could argue that $\mathsf{Enc}(\mathsf{pk}, k; H(k))$ might not be secure even though the underlying encryption function is when used with random coins instead of $H(k)$. This intuition is actually correct and the security of the FO construction can only be proven in a model where the hash function H is assumed to be perfectly random. That is, H returns a value sampled uniformly at random on each *fresh* query and can only be accessed as an external oracle by the different parties. This abstraction is called the *Random Oracle Model* (ROM) and, like many cryptosystems used in practice, most of the constructions presented in this thesis are proven secure in this ideal model only.

The ROM was first introduced by Bellare and Rogaway in 1993 [BR93] as a way to prove the security of protocols that are much more efficient than their counterparts in the so-called *standard model* (in opposition to the ROM). In order to understand why the ROM is so powerful, we need to understand how security is proved: if we want to prove that a primitive Q is secure, we first assume that another primitive or problem P is hard to break/solve. Then, we show that if some algorithm $\mathscr{A}$ can break Q, one can build another algorithm $\mathscr{B}$ that breaks or solve P, where typically $\mathscr{B}$ uses $\mathscr{A}$ as a subroutine. Now, in the ROM, $\mathscr{A}$ will typically query the random oracle H on some "important" values that might help $\mathscr{B}$ break its own primitive. Thus, $\mathscr{B}$ can observe these queries and exploit them, whereas in the standard model H would simply be a hash function that can be implemented directly by $\mathscr{A}$, making the computation of hash values invisible to $\mathscr{B}$. The FO transform example mentioned above perfectly illustrates how the ROM can be helpful: by the uniform distribution of H, $\mathsf{Enc}(\mathsf{pk}, k'; H(k'))$ looks exactly the same as $\mathsf{Enc}(\mathsf{pk}, k'; \mathsf{random\ coins})$ to any party, unless the latter queries $H(k')$. In turn, such a query would mean that k' is known and that would break the security of the encryption scheme.

The idealised properties of the ROM have sparked fierce debates among cryptographers throughout the years on whether security in this model is meaningful or not (see e.g. [KM15] for a summary). The detractors would say that a random oracle has little to do with a real hash function, which must be simply collision and preimage resistant. In addition, it was proven that ROM security does not imply standard security by Canetti et al. [CGH04]. That is, they showed that there exists a scheme that is secure in the ROM but insecure when the random oracle is instantiated with *any* hash function. Since then, several other works demonstrated similar results [GK03; BBP04].

On the other hand, advocates of the ROM would argue that these counterexamples are contrived and "unnatural". In addition, no real-world protocol or cryptosystem proven secure in the ROM has ever been broken unless the underlying assumption turned out to be insecure in the first place. This supports the idea that the ROM is a good heuristic. The dispute has been somewhat settled by practitioners, as an overwhelming majority of the schemes deployed in practice are only proven secure in the ROM or other similar ideal model. Among these we can mention RSA-OAEP, (EC)DSA, signatures and NIZKs based on the Fiat-Shamir transformation, key-exchange schemes based on the PRF-ODH assumption, and consequently all protocols that integrate one or several of these primitives, like X3DH and TLS 1.3. Last but not least, post-quantum KEMs can also be added to the list.

In conclusion, as this thesis is concerned with efficient schemes that are meant to be widely adopted by the public, the ROM is somewhat unavoidable and we will use it extensively.

QROM. The rationale behind the ROM is that hash functions are used by parties as black-boxes that output random-looking strings. However, in a quantum world, it is conceivable that quantum algorithms could access theses black-boxes (i.e. the hash functions) in superposition. That is, instead of querying the random oracle $H(x)$ for some value x, the quantum parties would have access to a unitary that computes the operation: $|x, y\rangle \mapsto |x, y \oplus H(x)\rangle$. Informally, for the reader unfamiliar with quantum notation, this means that an algorithm could, with one query, store all possible values of the random oracle in a state; then, in a later stage, it could extract a random value out of it. This new model, called *Quantum Random Oracle Model* (QROM) was introduced by Boneh et al. in 2011 [Bon+11] and it is now customary for post-quantum schemes to be proven secure in the QROM.

It turns out that translating existing ROM proofs to the QROM setting is challenging. One of the main reasons is that quantum queries made by an algorithm $\mathscr{A}$ cannot be observed by $\mathscr{B}$: all $\mathscr{B}$ can see is a quantum state that cannot be measured at the risk of $\mathscr{A}$ noticing it. Coming back to the FO example, one cannot argue in the QROM that "$\mathscr{A}$ cannot distinguish between $H(k')$ and a uniform value unless $H(k')$ was queried and is observable by other parties". Indeed, $H(k')$ might have been queried in the superposition and thus is "hidden" from external observers' view. Several other subtle issues we will not detail here can also arise in this model. However, many techniques to remedy these problems have been proposed, among those we can cite the One-Way to Hiding lemma (OW2H) [Unr15] and the compressed oracle technique [Zha19]. Leveraging these, we prove most of the constructions presented in this thesis secure in the QROM.

Outline of the Thesis

The core of this thesis is divided in eight chapters, including the present introduction and the conclusion. We briefly summarise each of them below.

First, in Chapter 2, we introduce the notation used throughout this document and we recall useful primitives and concepts. In particular, we formally define PKE, KEMs and the corresponding security notions. We also present the random oracle model and FO-like transforms in detail. Finally, we give a short introduction to the QROM and we state several related lemmas.

In Chapter 3, we develop misuse attacks against several PQ KEMs submitted to the NIST standardisation process. The threat model is as follows: we assume public keys are reused multiple times by parties and that the adversary has access to an oracle that, on input ct, k, returns whether $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = k$, where Dec is the decryption function of the underlying PKE (which is used to build the KEM through a FO-like transform). This corresponds to a real-life scenario where the FO transform is badly implemented and leaks the decrypted key –or simply whether it is equal to some other key– regardless of the result of the re-encryption check. Alternatively, such a leakage could be obtained through side-channels (e.g. time of execution, power consumption, electromagnetic radiations, etc.). We show that in this setting, the secret key can be recovered with a few thousand queries to the oracle in most of the schemes considered, completing the full picture of misuse attacks against PQ KEMs that had passed to the second round of the NIST process. The high-level strategy behind our attacks is always the same, it consists of learning the noise that is induced by the encryption in these schemes. Then, solving an equation is enough to recover the secret key. The only algorithm that seems to somewhat resist this tactic (i.e. $\approx 2^{38}$ queries needed to recover the key) is RQC [Mel+19a], a KEM based on the hardness of the syndrome decoding problem in the rank metric, that was discarded by NIST after the second round. We prove that resistance to the noise learning technique is inherent to rank-based schemes, hinting that constructions based in such a metric are more robust against misuse attacks than others.

In Chapter 4, we look to strengthen the security of post-quantum cryptosystems. One simple way to achieve this goal is to combine several schemes together into another one that is secure as long as *one* (or more) of the underlying algorithms is secure. The way the underlying blocks are merged into one is called a *combiner* in the literature. CCA-secure KEM combiners have been proposed before [GHP18; Bin+19a] and these constructions all work in a similar fashion: if $\mathscr{C}$ is the combiner and $\mathsf{KEM}_1, \dots, \mathsf{KEM}_n$ are KEMs, then $\mathscr{C}[\mathsf{KEM}_1, \dots, \mathsf{KEM}_n]$ outputs a CCA-secure KEM if there exists $i \in \{1, \dots, n\}$ s.t. KEM_i is CCA-secure. We propose another kind of combiner that generalises the concept of FO-like transforms as follows: let $\mathscr{F}$ be our type of combiner and $\mathsf{PKE}_1, \dots, \mathsf{PKE}_n$ be PKEs, then $\mathscr{F}[\mathsf{PKE}_1, \dots, \mathsf{PKE}_n]$ is a CCA-secure KEM if there exists $i \in \{1, \dots, n\}$ s.t. PKE_i is CPA-secure. In the context of the NIST post-quantum proposals, this has the advantage of being simpler and more efficient as *one* transform can be applied on the underlying PKEs to get a combined KEM, instead of applying a FO-transform n times and then applying a KEM combiner. We present several of these *FO-like combiners*, some proven in the ROM and others in the QROM, and we thoroughly formalise the theory underlying our constructions. In a second part of the chapter, we investigate which of the second round NIST algorithms should be combined together to maximise security and/or efficiency.

In Chapter 5, we study the efficiency of FO-like transforms and whether it is possible to do better. In particular, the main overhead of these transforms compared to the underlying CPA-secure PKE they take as input comes from the re-encryption step at decryption. A natural question is therefore whether this extra computation can be removed. It turns out that Gertner et al. [GMM07] proved it is not possible in the standard model (their result readily extends to the ROM as well). More precisely, they showed that there is no black-box construction of CCA-secure scheme from a CPA-secure one, under the restriction that the decryption function of the former does not call the encryption function of the latter, i.e. FO-like transforms are evidently not covered by this result. Removing this limitation and proving a general separation between CCA and CPA *in the standard model* is still a major open problem in theoretical cryptography. On our side, we generalise Gertner et al.'s result to the post-quantum setting, i.e. the primitive must still be computable classically but the adversaries/the reduction can be quantum. Following the original work, our proof uses the two-oracle techniques by Hsiao and Reyzin [HR04], which for us boils down to showing that there are two oracles (O, R) s.t. O can be used to build a PQ CPA-secure PKE, but there exists an adversary that uses O and R than can break any PQ CCA construction (with the restriction mentioned above). The main part of the demonstration is to show that O is a CPA-secure PKE even if adversaries have *quantum* black-box access to R. To do so, we reduce to several (information-theoretically) hard quantum problems.

In Chapter 6, motivated by the previous negative result, we explore the potential use-cases of post-quantum KEMs and identify several of them that do not require full CCA security but some weaker security. Among these applications, we can cite TLS 1.3, a variant of it called KEMTLS designed by Schwabe et al. [SSW20a], and ratcheting primitives. These protocols require only 1CCA, or more generally qCCA-secure KEMs, where q is a constant that denotes the number of decryptions the adversary is allowed to know (note that in normal CCA, the number of decryption queries the adversary can do is not fixed a priori). We introduce two very simple transforms that take a CPA-secure PKE/KEM and outputs a qCCA secure KEM. In particular, our constructions do not induce a re-encryption step, offering a $\approx 2\times$ speed-up at decryption compared to the CCA secure KEM obtained through a FO-like transformation. Both our designs are proven secure in the ROM and QROM. Then, using similar proof techniques, we solve an open question raised in previous work (e.g. [PST20; Dow+20]), and prove that replacing Diffie-Hellman in TLS 1.3 with *CPA-secure* KEMs is sound in the ROM. This result is mainly theoretical, as our proof incurs a large security loss which would not offer any security guarantees when instantiated with practical parameters. However, when replacing CPA-secure KEMs with 1CCA-secure ones, the security bound becomes similar to the one of (classical) TLS 1.3 [Dow+20]. Also, thanks to our transforms, 1CCA-secure KEMs can offer performances similar to their CPA-secure counterparts.

In Chapter 7, we present K-Waay, a post-quantum variant of X3DH, the key-exchange algorithm used in the Signal protocol. The challenge when designing a X3DH-like scheme is that it must fulfil two properties: *deniability*, which means parties can plausibly deny having completed key exchange and *asynchronicity*, which means parties can immediately derive keys after uploading them to a central server. Without the first requirement, X3DH could trivially be made post-quantum using KEMs and signatures only; without the second requirement, KEMs would suffice. In order to satisfy both, K-Waay uses at its core a split-KEM, a primitive introduced by Brendel et al. [Bre+21] that we augment with two security properties (deniability and unforgeability) that are needed to prove our key-exchange protocol secure and deniable. Then, we instantiate a split-KEM based on Frodo key exchange [Bos+16], which itself relies on the LWE assumption: our proofs might be of independent interest as we show it satisfies our novel unforgeability and deniability security notions. Compared to existing PQ X3DH proposals [Has+22; Bre+22], K-Waay does not use ring signatures, which are generally not proven secure in the QROM unlike our split-KEM, and are slower than standard primitives like KEMs. Then, we provide a thorough benchmark of both K-Waay and existing X3DH protocols. Our results show that, even when using plain LWE and a conservative choice of parameters, K-Waay is significantly faster than previous work.

Personal Bibliography

Below is a list of all the papers that were published before or during the making of this thesis. Entries in bold are included in this dissertation. Note that some content from [11] is used as part of Chapter 3 for the sake of explanation but it should not be considered novel as it was previously included in a Master thesis.

[1] **Loïs Huguenin-Dumittan and Serge Vaudenay. Impossibility of Post-Quantum Shielding Black-Box Constructions of CCA from CPA. *Communications in Cryptology, Volume 1. IACR, 2024. [HV24] (to appear)***

[2] **Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures. *USENIX Security'24.*** [Col+ar]

[3] Khashayar Barooti, Alex B. Grilo, Loïs Huguenin-Dumittan, Giulio Malavolta, Or Sattath, and Quoc-Huy Vu. Public-Key Encryption with Quantum Keys. In Guy Rothblum and Hoeteck Wee, editors. *Theory of Cryptography – TCC 2023, Lecture Notes in Computer Science, Volume 14372. Springer, 2023.* [Bar+23b]

[4] Khashayar Barooti, Daniel Collins, Simone Colombo, Loïs Huguenin-Dumittan and Serge Vaudenay. On Active Attack Detection in Messaging with Immediate Decryption. In Helena Handschuh and Anna Lysyanskaya, editors. *Advances in Cryptology – CRYPTO 2023, Lecture Notes in Computer Science, Volume 14084. Springer, 2023.* [Bar+23a]

[5] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real World Deniability in Messaging. *Extended abstract of a talk given at RWC 2023. https://eprint.iacr.org/2023/403.pdf*. [CCH23]

[6] **Loïs Huguenin-Dumittan and Serge Vaudenay. On IND-qCCA Security in the ROM and Its Applications. In Orr Dunkelman and Stefan Dziembowski, editors. *Advances in Cryptology – EUROCRYPT 2022, Lecture Notes in Computer Science, volume 13277. Springer, 2022.*** [HV22]

[7] **Loïs Huguenin-Dumittan and Serge Vaudenay. FO-like Combiners and Hybrid Post-Quantum Cryptography. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors. *Cryptology and Network Security – CANS 2021, Lecture Notes in Computer Science, volume 13099. Springer, 2021.*** [HV21]

[8] Loïs Huguenin-Dumittan and Iraklis Leontiadis. A Message Franking Channel. In Yu Yu and Moti Yung, editors. *Information Security and Cryptology – Inscrypt 2021, Lecture Notes in Computer Science, volume 13099. Springer, 2021.* [HL21]

[9] F. Betül Durak, Loïs Huguenin-Dumittan, and Serge Vaudenay. BioLocker: A Practical Biometric Authentication Mechanism Based on 3D Fingervein. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors. *Applied Cryptography and Network Security – ACNS 2020, Lecture Notes in Computer Science, volume 12147. Springer, 2020.* [DHV20]

[10] **Loïs Huguenin-Dumittan and Serge Vaudenay. Classical Misuse Attacks on NIST Round 2 PQC. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors. *Applied Cryptography and Network Security – ACNS 2020, Lecture Notes in Computer Science, volume 12146. Springer, 2020.*** [HV20]

[11] Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse Attacks on Post-quantum Cryptosystems. In Yuval Ishai and Vincent Rijmen, editors. *Advances in Cryptology – EUROCRYPT 2019, Lecture Notes in Computer Science, volume 11477. Springer, 2022.* [Băe+19]

2 Preliminaries

In this chapter, we introduce the notation and concepts used throughout this thesis.

2.1 Notation

Sets and sampling. We denote by $[n]$ (resp. $[n]_-$) the set $\{1,\ldots,n\}$ (resp. $\{0,\ldots,n-1\}$). For $\mathscr{A}$ a randomised algorithm, we write $b \leftarrow\!\!\$\, \mathscr{A}$ to indicate b is set to the value output by $\mathscr{A}$. Similarly, if Ψ (resp. $\mathscr{X}$) is a distribution (resp. a set), then $x \leftarrow\!\!\$\, \Psi$ (resp. $x \leftarrow\!\!\$\, \mathscr{X}$) means that x is sampled from Ψ (resp. uniformly at random from $\mathscr{X}$). If x is a vector of dimension n or a polynomial of degree $n-1$, we write $x \leftarrow\!\!\$\, \Psi^n$ to say that each component/coefficient of x is sampled independently from Ψ. For f any function, $\mathsf{Im}(f)$ denotes its image.

Multiplication. The multiplication in multiplicative groups, rings, and fields is denoted by $\times$, $\cdot$, or even nothing.

Algorithms and oracles. We denote by 1_P the indicator function which returns 1 if the predicate P is fulfilled and 0 otherwise. For any algorithm Γ that takes an input x, we write "$\Gamma(x) \Rightarrow b$" to denote the event $\Gamma(x)$ *outputs* b. Also, when it is clear from the context, we write that same event "$\Gamma \Rightarrow b$" or even "Γ" when $b = 1$. In a game, we write **abort** to mean that the algorithm is stopped (i.e. the adversary "loses" the game).

For any classical algorithm $\mathscr{A}$ and $\mathscr{O}_1,\ldots,\mathscr{O}_n$, we write $\mathscr{A}^{\mathscr{O}_1,\ldots,\mathscr{O}_n}$ to denote the fact that $\mathscr{A}$ has *black-box* access to $\mathscr{O}_1,\ldots,\mathscr{O}_n$. When $\mathscr{A}$ computes $\mathscr{O}_i(x)$ for some input x, we say $\mathscr{A}$ *queries* $\mathscr{O}_i$. For any quantum algorithm $\mathscr{A}$ and unitaries $\mathscr{O}_1,\ldots,\mathscr{O}_n$, we write $\mathscr{A}^{\mathscr{O}_1,\ldots,\mathscr{O}_n}$ to denote the fact that $\mathscr{A}$ can use the unitaries $\mathscr{O}_1,\ldots,\mathscr{O}_n$ as *black-boxes*. When $\mathscr{A}$ uses $\mathscr{O}_i$, we say $\mathscr{A}$ *queries* $\mathscr{O}_i$. Sometimes, a quantum adversary has *classical* access to an oracle (e.g. a decryption oracle) but *quantum* access to another (e.g. a random oracle). In this case, we write $\mathscr{A}^{\mathscr{O},|H\rangle}$ to denote that $\mathscr{A}$ can only query $\mathscr{O}$ classically but has quantum access to the oracle H, or simply $\mathscr{A}^{O,H}$ when it is clear from the context.

$\Gamma^{1\text{-}2}$

1 : $x \leftarrow 0$ // Γ^1
2 : $x \leftarrow 1$ // Γ^2
3 : **return** x

Figure 2.1: Pseudocode example.

If $\mathscr{A}(x)$ is a randomised algorithm running on input x, it is assumed that enough random coins are implicitly passed to $\mathscr{A}$. We sometimes write $\mathscr{A}(x;r)$ to denote the fact that $\mathscr{A}$ is run with random coins r.

We say an algorithm is *efficient* if it is a probabilistic polynomial-time (ppt) or quantum polynomial-time (qpt) algorithm.

Pseudocode and error symbol. Errors are denoted with the symbol $\perp$. In pseudocode, several algorithms/games are often compressed into one with comments specifying which line is executed in which game. We give an example in Figure 2.1, where two games Γ^1 and Γ^2 are made explicit. In Γ^1, only lines 1 and 3 are executed (0 is returned) and in Γ^2 only lines 2 and 3 are executed (1 is returned).

Vectors and rounding. For some vector or polynomial x, x_i is the i-th coefficient and $(x)_i$ is the subset composed of the i-th first coefficients of x. For $x \in \mathbb{Z}_q$, we write $x' = \langle x \rangle_q$ for the unique integer $x' \in (-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor]$ s.t. $x' \equiv x \pmod{q}$. We denote by $\lceil x \rfloor$ rounding x to the nearest integer, with ties rounded up. If f is a function defined on a component of a vector (or polynomial) v, we write $f(v)$ to denote the function being applied to each component of v.

Negligible function. We denote by $\mathsf{negl}(\lambda)$ any negligible function in a given parameter λ. When it is clear from the context, we sometimes shorten the notation to negl. We recall that a function $f(\lambda)$ is negligible in λ iff $\forall c \in \mathbb{Z}\ \exists \lambda_c$ s.t. $\forall \lambda > \lambda_c\ |f(\lambda)| < \frac{1}{\lambda^c}$.

Advantage of an adversary. Security is often defined in terms of the probability of an adversary winning an experiment, called *game*. We refer to this probability as the *advantage* of the adversary. This quantity is denoted by $\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A})$, where sec, Π, and $\mathscr{A}$ stands for the security definition, the primitive, and the adversary considered, respectively.

2.2 Primitives (PKE/KEM/Signatures/PRF)

In this section, we introduce the main cryptographic primitives employed in this dissertation as well as the corresponding security definitions. All security definitions are valid in both the

$\text{CORR}_{\mathsf{PKE}}(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$\mathsf{pt} \leftarrow \mathscr{A}^H(\mathsf{pk}, \mathsf{sk})$
$\mathsf{ct} \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})$
return $1_{\mathsf{Dec}(\mathsf{sk},\mathsf{ct}) \neq \mathsf{pt}}$

Figure 2.2: Correctness game with a random oracle H.

classical and quantum model of computation, depending on whether we let adversaries to be quantum algorithms or not.

2.2.1 Public-Key Encryption scheme

Definition 2.2.1 (Public-Key Encryption)**.** *A Public-Key Encryption scheme (PKE) is composed of three efficient algorithms* Gen, Enc, Dec *and is associated to a message space* $\mathscr{M}$*:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes the security parameter* λ *as input and outputs the public key* pk *and the secret key* sk.
- $\mathsf{ct} \leftarrow\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})$*: The encryption algorithm takes as inputs the public key* pk *and a plaintext* $\mathsf{pt} \in \mathscr{M}$*, and it outputs a ciphertext* ct.
- $\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: The decryption procedure takes as inputs the secret key* sk *and the ciphertext* $\mathsf{ct} \in \mathscr{C}$*, and it outputs a plaintext* $\mathsf{pt}' \in \mathscr{M} \cup \{\perp\}$.

Gen *and* Enc *are probabilistic algorithms that can be made deterministic by adding random coins as inputs. The decryption procedure is deterministic.*

Correctness. We define correctness as follows.

Definition 2.2.2 (Correctness)**.** *We consider the game CORR defined in Figure 2.2. We say a PKE scheme is* $\delta(q_H)$ *correct if for any efficient adversary* $\mathscr{A}$ *making at most* q_H *adversary to the random oracle* H*, we have*

$$\Pr[CORR_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1] \leq \delta(q_H, \lambda) \;,$$

where λ *is the security parameter. Note that we omit* λ *from now on for the sake of simplicity. The correctness in the standard model is defined similarly except* δ *does not depend on* q_H.

Intuitively, correctness means that no adversary can find with probability greater than $\delta(q_H)$ a plaintext such that its encryption does not decrypt to the original plaintext.

$\mathsf{IND\text{-}ATK}_{\mathsf{PKE}}(\mathscr{A})$

$b \leftarrow\!\$ \{0,1\}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$ \mathsf{Gen}(1^\lambda)$
define $\mathsf{ct}^* \leftarrow \emptyset$
$\mathsf{pt}_0, \mathsf{pt}_1 \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK1}}}(\mathsf{pk})$
$\mathsf{ct}^* \leftarrow\!\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_b)$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK2}}}(\mathsf{pk}, \mathsf{ct}^*)$
return $1_{b=b'}$

$\mathsf{IND\text{-}ATK'}^{b}_{\mathsf{PKE}}(\mathscr{A})$

$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$ \mathsf{Gen}(1^\lambda)$
define $\mathsf{ct}^* \leftarrow \emptyset$
$\mathsf{pt}_0, \mathsf{pt}_1 \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK1}}}(\mathsf{pk})$
$\mathsf{ct}^* \leftarrow\!\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}_b)$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK2}}}(\mathsf{pk}, \mathsf{ct}^*)$
return b'

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
$\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
return pt'

Figure 2.3: Equivalent indistinguishability games and the decryption oracle.

Table 2.1: Oracles for IND and OW games.

ATK	CPA	CCA1	CCA
$\mathscr{O}^{\mathsf{ATK1}}$	$\perp$	$\mathscr{O}^{\mathsf{Dec}}$	$\mathscr{O}^{\mathsf{Dec}}$
$\mathscr{O}^{\mathsf{ATK2}}$	$\perp$	$\perp$	$\mathscr{O}^{\mathsf{Dec}}$

ATK	CPA	PCA	VCA	PVCA
$\mathscr{O}^{\mathsf{ATK}}$	$\perp$	$\mathscr{O}^{\mathsf{PCO}}$	$\mathscr{O}^{\mathsf{VCO}}$	$\mathscr{O}^{\mathsf{PCO}}, \mathscr{O}^{\mathsf{VCO}}$

γ-spreadness. In some of the proofs, we need the ciphertexts of a PKE to be well-spread. That is, the probability to obtain a given ciphertext when encrypting should be negligible, or at least upper-bounded by some value. This idea is formalised in the following definition.

Definition 2.2.3 (γ-spreadness)**.** *For any public key* pk *and plaintext* pt*, we define the min-entropy of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt})$ *as*

$$\gamma(\mathsf{pk}, \mathsf{pt}) = -\log\left(\max_{\mathsf{ct}\in\mathscr{C}} \Pr\left[\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})\right]\right),$$

where the probability is taken over the randomness of Enc*, the logarithm is in base 2, and* $\mathscr{C}$ *is the ciphertext domain. Then, we say that a PKE scheme is* γ*-spread if for any public key* pk *and plaintext* pt*, we have* $\gamma(\mathsf{pk}, \mathsf{pt}) \geq \gamma$*. This implies that* $\Pr[\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})] \leq 2^{-\gamma}$.

Rigidity. When introducing transforms from Hofheinz et al. [HHK17], we will need the notion of rigidity, that states that either a ciphertext does not decrypt, or the decrypted message re-encrypts to the same ciphertext. Note that this property can hold only if the scheme has deterministic encryption.

Definition 2.2.4. *We say a PKE* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* rigid *if for all* $(\mathsf{pk}, \mathsf{sk})$ *output by* Gen *and for all* $\mathsf{ct} \in \mathscr{C}$*, either* $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \perp$ *or* $\Pr[\mathsf{Enc}(\mathsf{pk}, \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})) = \mathsf{ct}] = 1$.

Indistinguishability.

$\mathsf{OW\text{-}ATK}_{\mathsf{PKE}}(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$
$\mathsf{pt}^* \leftarrow\!\!\$\ \mathscr{M}$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}^*)$
$\mathsf{pt}' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK}}}(\mathsf{pk}, \mathsf{ct}^*)$
return $1_{\mathsf{pt}'=\mathsf{pt}^*}$

Oracle $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, \mathsf{ct})$

$\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
return $1_{\mathsf{pt}'=\mathsf{pt}}$

Oracle $\mathscr{O}^{\mathsf{VCO}}(\mathsf{ct} \neq \mathsf{ct}^*)$

$\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
return $1_{\mathsf{pt}' \in \mathscr{M}}$

Figure 2.4: One-Wayness games.

Definition 2.2.5 (PKE IND-CPA/CCA/CCA1)**.** *We consider the games induced by the pseudocode given on the left of Figure 2.3, where the oracles given in each game are defined as in the left of Table 2.1. A PKE scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-ATK *for* ATK ∈ {CPA, CCA, CCA1} *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ind\text{-}atk}}_{\mathscr{A},\mathsf{PKE}} := \left|\Pr\left[\mathsf{IND\text{-}ATK}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda).$$

Equivalently, we can consider the games induced by the pseudocode given in the middle of Figure 2.3, where the oracles given in each game are defined as in the left of Table 2.1. Then, a PKE scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-ATK *for* ATK ∈ {CPA, CCA, CCA1} *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ind\text{-}atk'}}_{\mathscr{A},\mathsf{PKE}} := \left|\Pr\left[\mathsf{IND\text{-}ATK'}^{1}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1\right] - \Pr\left[\mathsf{IND\text{-}ATK'}^{0}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1\right]\right| = \mathsf{negl}(\lambda).$$

Informally, these definitions state that no adversary should be able to distinguish between the encryption of two different messages. In IND-CCA, the adversary has further access to a decryption oracle that returns the decryption of any ciphertext except the challenge one ct^*. In IND-CCA1, access to the decryption oracle is only granted before the challenge ciphertext is generated. We stress that the two definitions of indistinguishability we gave for each notion are equivalent, and we use them interchangeably in the rest of this thesis.

One-Wayness. We also recall four security definitions of one-wayness: One-Wayness under Chosen-Plaintext Attacks (OW-CPA), One-Wayness under Plaintext-Checking Attacks (OW-PCA), One-Wayness under Validity Checking Attacks (OW-VA), and One-Wayness under Plaintext and Validity Checking Attacks (OW-PVCA).

Definition 2.2.6 (One-Wayness and Plaintext/Validity Checking)**.** *Let* $\mathscr{M}$ *be the message space,* PKE *a PKE scheme, and we consider the games defined in Figure 2.4 with the different oracles as defined on the right in Table 2.1. Then,* PKE *is* OW-ATK*, for* ATK ∈ {CPA, PCA, VCA, PVCA}*, if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ow\text{-}atk}}_{\mathsf{PKE}}(\mathscr{A}) := \Pr\left[\mathsf{OW\text{-}ATK}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1\right] = \mathsf{negl}(\lambda)\,,$$

$\text{KR-PCA}_{\mathsf{PKE}}(\mathscr{A})$	**Oracle** $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt},\mathsf{ct})$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$	$\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk},\mathsf{ct})$
$\mathsf{sk}' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{PCO}}}(\mathsf{pk})$	**return** $1_{\mathsf{pt}'=\mathsf{pt}}$
return $1_{\mathsf{sk}'=\mathsf{sk}}$	

Figure 2.5: KR-PCA game.

where $\Pr[\text{OW-}\mathsf{ATK}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1]$ *is the probability that the adversary wins the* OW-ATK *game.*

These notions model the intuition that no adversary should be able to decrypt a ciphertext. Note that one-wayness is weaker than indistinguishability as one can distinguish if one can decrypt. Both the $\mathscr{O}^{\mathsf{PCO}}$ and $\mathscr{O}^{\mathsf{VCO}}$ oracles model *reaction attacks*, where the adversary is able to check if a decrypted ciphertext matches a certain plaintext (PCA) or whether the decryption is successful (VCA). Bleichenbacher's attack is a famous example of a VCA [Ble98].

Remark. Any perfectly correct and deterministic OW-CPA PKE system is OW-PCA. Indeed, since the encryption is deterministic, an adversary can always compute $1_{\mathsf{Enc}(\mathsf{pk},\mathsf{pt})=\mathsf{ct}}$, which returns the same result as the plaintext-checking oracle in the perfect correctness case, making the latter superfluous.

IND-CPA $\Rightarrow$ OW-CPA. The following folklore result states that IND-CPA implies OW-CPA if the message space $\mathscr{M}$ is large enough.

Lemma 2.2.1. *Let* PKE *be any PKE. Then, for all adversaries* $\mathscr{A}$*, there exists an adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{PKE}}(\mathscr{A}) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathscr{B}) + \frac{1}{|\mathscr{M}|} .$$

KR-PCA. Finally, we define the notion of key-recovery under plaintext-checking attack (KR-PCA), where the adversary has access to the $\mathscr{O}^{\mathsf{PCO}}$ oracle and must recover the secret key sk, given a public key pk. The game is given in Figure 2.5.

Definition 2.2.7 (Key-Recovery under Plaintext-Checking Attack)**.** *Let* PKE *be a PKE scheme. We say* PKE *is* KR-PCA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{kr-pca}}_{\mathsf{PKE}}(\mathscr{A}) = \Pr[\text{KR-PCA}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1] = \mathsf{negl}(\lambda) ,$$

where $\Pr[\text{KR-PCA}_{\mathsf{PKE}}(\mathscr{A}) \Rightarrow 1]$ *is the probability that the adversary wins the* KR-PCA *game given in Figure 2.5.*

$\mathsf{IND\text{-}ATK}_{\mathsf{KEM}}(\mathscr{A})$	$\mathsf{IND\text{-}ATK'}^{b}_{\mathsf{KEM}}(\mathscr{A})$	**Oracle** $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct})$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$	$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$	**if** $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
$st \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK1}}}(\mathsf{pk})$	$st \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK1}}}(\mathsf{pk})$	$K' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$
$b \leftarrow\!\!\$\ \{0,1\}$	$\mathsf{ct}^*, K_0 \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$	**return** K'
$\mathsf{ct}^*, K_0 \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$	$K_1 \leftarrow\!\!\$\ \mathscr{K}$	
$K_1 \leftarrow\!\!\$\ \mathscr{K}$	$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK2}}}(st,\mathsf{pk},\mathsf{ct}^*,K_b)$	
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK2}}}(st,\mathsf{pk},\mathsf{ct}^*,K_b)$	**return** b	
return $1_{b'=b}$		

Figure 2.6: Equivalent indistinguishability games and the decapsulation oracle.

2.2.2 Key Encapsulation Mechanism (KEM)

Definition 2.2.8 (Key Encapsulation Mechanism)**.** *A KEM is a tuple of three algorithms* Gen, Encaps, Decaps*:*

- $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes as input the security parameter, and it outputs the public key* pk *and the secret key* sk.
- $\mathsf{ct}, K \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$*: The encapsulation algorithm takes as inputs the public key* pk*, and it outputs a ciphertext* $\mathsf{ct} \in \mathscr{C}$ *and a key* $K \in \mathscr{K}$*.*
- $K' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$*: The decapsulation procedure takes as inputs the secret key* sk *and the ciphertext* $\mathsf{ct} \in \mathscr{C}$*, and it outputs a key* K*. If the KEM allows explicit rejection, the output is a key* $K \in \mathscr{K}$ *or the rejection symbol* $\perp$*. If the rejection is implicit, the output is always a key* $K \in \mathscr{K}$*.*

The Gen *and* Encaps *are probabilistic algorithms that can be made deterministic by adding random coins as inputs. The decapsulation function is deterministic.*

Correctness. We define correctness for KEMs as follows.

Definition 2.2.9 (KEM Correctness)**.** *We say a KEM* $(\mathsf{Gen},\mathsf{Encaps},\mathsf{Decaps})$ *is* δ*-correct if*

$$\Pr\left[K \neq K' : \begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda); \\ \mathsf{ct}, K \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk}); \\ K' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct}) \end{array}\right] \leq \delta\,.$$

Indistinguishability. We now recall the different notions of indistinguishability for KEMs.

Definition 2.2.10 (KEM IND-CPA/CCA1/CCA)**.** *We consider the games induced by the pseudocode on the left in Figure 2.6, where the oracles given in each game are defined as in the left of*

$\mathsf{OW\text{-}ATK_{KEM}}(\mathscr{A})$	**Oracle** $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct})$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$	**if** $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
$\mathsf{ct}^*, K^* \leftarrow \mathsf{Enc}(\mathsf{pk})$	$K' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$
$K' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{ATK}}}(\mathsf{pk},\mathsf{ct}^*)$	**return** K'
return $1_{K'=K^*}$	

Figure 2.7: One-wayness games for KEM.

Table 2.1. A KEM scheme $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* IND-ATK *for* $\mathsf{ATK} \in \{\text{CPA}, \text{CCA}, \text{CCA1}\}$ *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ind\text{-}atk}}_{\mathscr{A},\mathsf{KEM}} := \left|\Pr[\mathsf{IND\text{-}ATK_{KEM}}(\mathscr{A}) \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{negl}(\lambda).$$

Equivalently, we can consider the games induced by the pseudocode given in the middle in Figure 2.6, where the oracles given in each game are defined as in the left of Table 2.1. Then, a KEM scheme $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-ATK *for* $\mathsf{ATK} \in \{\text{CPA}, \text{CCA}, \text{CCA1}\}$ *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ind\text{-}atk}'}_{\mathscr{A},\mathsf{KEM}} := \left|\Pr\left[\mathsf{IND\text{-}ATK'}^{1}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] - \Pr\left[\mathsf{IND\text{-}ATK'}^{0}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right]\right| = \mathsf{negl}(\lambda).$$

As in the case of PKE, KEM indistinguishability means that an adversary, given a ciphertext and a key, should not be able to tell whether the key is encapsulated in the ciphertext or is a random key. The different flavours (CPA, CCA1, CCA) are analogous to the PKE case.

One-Wayness. We also recall the definition of one-wayness for KEMs.

Definition 2.2.11 (KEM One-Wayness)**.** *Let* $\mathscr{K}$ *be the message space,* KEM *a KEM scheme and we consider the games defined in Figure 2.7 with the different oracles as defined on the left in Table 2.1. Then,* KEM *is OW-*ATK*, for* $\mathsf{ATK} \in \{\text{CPA}, \text{CCA}\}$*, if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{ow\text{-}atk}}_{\mathsf{KEM}}(\mathscr{A}) = \Pr[\mathsf{OW\text{-}ATK_{KEM}}(\mathscr{A}) \Rightarrow 1] = \mathsf{negl}(\lambda)\ ,$$

where $\Pr[\mathsf{OW\text{-}ATK_{KEM}}(\mathscr{A}) \Rightarrow 1]$ *is the probability that the adversary wins the* OW-ATK *game.*

Similarly to the case of PKE, one-wayness means that an adversary cannot recover a key that is encapsulated in a given ciphertext. In the CCA variant, the adversary has further access to a decapsulation oracle, with the restriction that the challenge ciphertext cannot be queried.

As in the PKE case, it is easy to show that if the key space is large enough, IND-CPA security implies OW-CPA in the KEM setting.

Lemma 2.2.2. *Let* KEM *be any KEM. Then, for all adversaries* $\mathscr{A}$*, there exists an adversary* $\mathscr{B}$

$\text{SUF-CMA}_{\mathsf{Sig}}(\mathscr{A})$	$\text{SIGN}(m)$
$L \leftarrow \emptyset$	$\sigma \leftarrow\$ \mathsf{Sign}(\mathsf{sk}, m)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$	$L \leftarrow L \cup \{m, \sigma\}$
$m^*, \sigma^* \leftarrow\$ \mathscr{A}^{\text{SIGN}}(\mathsf{pk})$	**return** σ
if $\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*)$ **and** $(m^*, \sigma^*) \notin L$	
return 1	
return 0	

Figure 2.8: SUF-CMA game.

s.t.

$$\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{KEM}}(\mathscr{A}) \le \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM}}(\mathscr{B}) + \frac{1}{|\mathscr{K}|} .$$

2.2.3 Signature

We recall here the notion of a digital signature scheme.

Definition 2.2.12. *A signature scheme is a tuple of three efficient algorithms* (Gen,Sign,Vrfy)*:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$*: The key generation function outputs a pair of keys.*
- $\sigma \leftarrow\$ \mathsf{Sign}(\mathsf{sk}, m)$*: The signing function takes as inputs a secret key* sk *and the message to sign m, and it outputs a signature* σ*.*
- $0/1 \leftarrow \mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$*: The verification function takes as inputs a public key* pk*, the signed message m, and the signature* σ*, and it outputs either* 0 *or* 1 *(for failure and success, respectively).*

Finally, we say a signature scheme is δ*-correct if for all messages m:*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 0 : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda); \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m) \end{array}\right] \le \delta$$

Definition 2.2.13 (SUF-CMA security)**.** *We consider the game shown in Figure 2.8. We say a signature scheme* Sig *is* SUF-CMA *if for all efficient adversaries* $\mathscr{A}$*, we have*

$$\mathsf{Adv}^{\text{suf-cma}}_{\mathsf{Sig}}(\mathscr{A}) := \Pr[\text{SUF-CMA}_{\mathsf{Sig}}(\mathscr{A}) \Rightarrow 1] = \mathsf{negl} .$$

2.2.4 Pseudorandom Function (PRF)

We also recall the notion of Pseudorandom Function (PRF).

$\mathrm{PRF}_{\mathsf{PRF}}(\mathscr{A})$	$\mathscr{O}_{\mathsf{prf}}(x)$
$b \leftarrow\!\!\$\ \{0,1\}$	**if** $b = 0$: **return** $\mathsf{PRF}_K(x)$
$K \leftarrow\!\!\$\ \mathscr{K}$	**if** $b = 1$: **return** $F(x)$
$b' \leftarrow\!\!\$\ \mathscr{A}^{\mathscr{O}_{\mathsf{prf}}}$	
return $1_{b=b'}$	

Figure 2.9: PRF game, where $F : \mathscr{K} \times \mathscr{M} \mapsto \mathscr{R}$ is a random function.

Definition 2.2.14. *We say* $\mathsf{PRF} : \mathscr{K} \times \mathscr{M} \mapsto \mathscr{R}$ *is a pseudorandom function (PRF) if for all efficient adversaries* $\mathscr{A}$*, we have*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}}(\mathscr{A}) := \left|\Pr[\mathrm{PRF}_{\mathsf{PRF}}(\mathscr{A}) \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{negl}\ ,$$

where PRF is the game given in Figure 2.9.

2.2.5 Game-based proofs.

All theorems establishing the security of a construction in this thesis are proven using the well-known game-playing framework formalised by Shoup and Bellare and Rogaway [Sho04; Bel06a].

In these proofs, we usually start from the game defining the security we want to prove (e.g. one of the indistinguishability games defined above) instantiated with the primitive we want to prove secure. Then, the starting game is modified into a succession of *hybrids*, where each of these is shown to be negligibly close in terms of adversarial advantage to the previous one. This is formalised in the following facts.

Fact 1 (Hybrid argument for search-type game)**.** *Let* Π, sec *and* $\mathscr{A}$ *be any primitive, security notion, and adversary, respectively, s.t. the adversary's advantage can be written as*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) := \Pr[\Gamma^0(\mathscr{A}) \Rightarrow 1]\ ,$$

where Γ^0 *is some game. I.e. the security notion considered is defined with a search-type game. Let* $\Gamma^1, \ldots, \Gamma^n$ *be s.t.* $|\Pr[\Gamma^i(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma^{i-1}(\mathscr{A}) \Rightarrow 1]| = \mathsf{negl}$ *for all* $i \in [n]$*, and* $\Pr[\Gamma^n(\mathscr{A}) \Rightarrow 1] = \mathsf{negl}$*. Then,*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) = \mathsf{negl}\ .$$

Proof. It follows from the fact that $\Pr[\Gamma^0]$ can be written as

$$\left|\left(\Pr[\Gamma^0] - \Pr[\Gamma^1]\right) + \left(\Pr[\Gamma^1] - \Pr[\Gamma^2]\right) + \ldots + \Pr[\Gamma^n]\right|$$

and the triangle inequality. □

Fact 2 (Hybrid argument for decision-type game of type I)**.** *Let* Π, sec *and* $\mathscr{A}$ *be any primitive, security notion, and adversary, respectively, s.t. the adversary's advantage can be written as*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) := \Pr[\Gamma^0(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma^n(\mathscr{A}) \Rightarrow 1] \ ,$$

where Γ^0 *and* Γ^n *are some games. Let* $\Gamma^1, \ldots, \Gamma^{n-1}$ *be s.t.* $|\Pr[\Gamma^i(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma^{i-1}(\mathscr{A}) \Rightarrow 1]| = \mathsf{negl}$ *for all* $i \in [n]$*. Then,*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) = \mathsf{negl} \ .$$

Proof. It follows from the fact that $|\Pr[\Gamma^0] - \Pr[\Gamma^n]|$ can be written as

$$\left|\left(\Pr[\Gamma^0] - \Pr[\Gamma^1]\right) + \left(\Pr[\Gamma^1] - \Pr[\Gamma^2]\right) + \ldots + \left(\Pr[\Gamma^{n-1}] - \Pr[\Gamma^n]\right)\right|$$

and the triangle inequality. □

Fact 3 (Hybrid argument for decision-type game of type II)**.** *Let* Π, sec *and* $\mathscr{A}$ *be any primitive, security notion, and adversary, respectively, s.t. the adversary's advantage can be written as*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) := \Pr[\Gamma^0(\mathscr{A}) \Rightarrow 1] - c \ ,$$

where Γ^0 *and* Γ^n *are some games and* $c \in \mathbb{Q}$ *is a constant. Let* $\Gamma^1, \ldots, \Gamma^{n-1}$ *be s.t.* $|\Pr[\Gamma^i(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma^{i-1}(\mathscr{A}) \Rightarrow 1]| = \mathsf{negl}$ *for all* $i \in [n]$*, and* Γ^n *be s.t.* $\Pr[\Gamma^n(\mathscr{A}) \Rightarrow 1] = c$*. Then,*

$$\mathsf{Adv}^{\mathsf{sec}}_{\Pi}(\mathscr{A}) = \mathsf{negl} \ .$$

Proof. It follows from the fact that $|\Pr[\Gamma^0] - c|$ can be written as $|\Pr[\Gamma^0] - \Pr[\Gamma^n]|$ and Fact 1. □

In order to show that two games Γ, Γ' are indistinguishable (i.e. $\Pr[\Gamma] - \Pr[\Gamma'] = \mathsf{negl}$) the following lemma, sometimes called the *difference lemma* [Sho04] or the *fundamental lemma of game-playing* [Bel06a], is most useful.

Lemma 2.2.3 (Difference lemma)**.** *Let* Γ *and* Γ' *be two games that are identical unless some event* bad *occurs. Then, for any adversary* $\mathscr{A}$*,*

$$\left|\Pr[\Gamma(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma'(\mathscr{A}) \Rightarrow 1]\right| \leq \Pr[\mathsf{bad}] \ .$$

Proof. Since both games are identical unless bad occurs, we have $\Pr[\Gamma|\overline{\mathsf{bad}}] = \Pr[\Gamma'|\overline{\mathsf{bad}}]$. Also, note that the probability $\overline{\mathsf{bad}}$ (or bad) occurs is the same in both games. Hence,

$$\begin{aligned}
\Pr[\Gamma] - \Pr[\Gamma'] &= \Pr[\Gamma|\overline{\mathsf{bad}}]\Pr[\overline{\mathsf{bad}}] + \Pr[\Gamma|\mathsf{bad}]\Pr[\mathsf{bad}] - \Pr[\Gamma'|\overline{\mathsf{bad}}]\Pr[\overline{\mathsf{bad}}] - \Pr[\Gamma'|\mathsf{bad}]\Pr[\mathsf{bad}] \\
&= \left(\Pr[\Gamma|\mathsf{bad}] - \Pr[\Gamma'|\mathsf{bad}]\right)\Pr[\mathsf{bad}] \\
&\leq \Pr[\mathsf{bad}]
\end{aligned}$$

$$
\begin{array}{l}
\underline{H(x)} \\
\textbf{if } L_H[x] \text{ does not exist}: \\
\quad L_H[x] \leftarrow\!\$\, \{0,1\}^n \\
\textbf{return } L_H[x]
\end{array}
$$

Figure 2.10: Random oracle implemented with lazy sampling. The parameter n should correspond to the output size of the hash function the RO represents.

where the last inequality follows from the fact that the difference between two probabilities is smaller or equal to 1. □

2.2.6 ROM and game-based proofs

The Random Oracle. As briefly explained in the introduction, the random oracle model (ROM) is an abstraction where hash functions are assumed to be perfectly random functions that are accessed in a black-box way by the different parties in a security experiment. More formally, a random oracle can be seen as an oracle that performs lazy sampling and keeps track of values that have already been queried. Such a behaviour is illustrated in Figure 2.10, where the list $\mathscr{L}_H$ is assumed to be empty at the beginning. This representation of the random oracle (RO) highlights the fact that a RO can be efficiently simulated with lazy sampling. This is important as in security reductions we want to show that an *efficient* adversary $\mathscr{B}$ can simulate $\mathscr{A}$'s environment and use it as a black-box to solve some hard problem.

We show in the following simple example how both the game-playing technique and the ROM can be used together to prove the security of a construction. More precisely, we demonstrate that, in the ROM, a OW-CPA PKE can be transformed into an IND-CPA KEM by simply hashing a random plaintext and encrypting it.

Example 2.2.1. *Let* $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ *be any PKE and* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be the KEM built out of* PKE *as shown in Figure 2.11, where* H *is a random oracle. Then, for any IND-CPA adversary* $\mathscr{A}$ *against* KEM*, there exists a OW-CPA adversary* $\mathscr{B}$ *against* PKE *s.t.*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM}}(\mathscr{A}) \leq q_H \cdot \mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) ,$$

where q_H *is the number of queries* $\mathscr{A}$ *can make to the random oracle* H*. Therefore, if* PKE *is OW-CPA, then* KEM *is IND-CPA.*

Proof. We start with game Γ, which is the IND-CPA game instantiated with KEM and $b = 0$. The game is detailed in Figure 2.11.
We then modify Γ into a game Γ', where the real challenge key K_0 is picked at random (see Figure 2.11). As long as $\mathscr{A}$ does not query $H(\mathsf{pt}^*)$, both Γ and Γ' are identical as K^* is picked uniformly at random the first time $H(\mathsf{pt}^*)$ is queried by the game. Let bad be the event that $\mathscr{A}$

$\mathsf{Gen}(1^\lambda)$

return $\mathsf{Gen}'(1^\lambda)$

$\mathsf{Encaps}(\mathsf{pk})$

$\mathsf{pt} \leftarrow\$ \mathcal{M}$
$K \leftarrow H(\mathsf{pt})$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt})$
return ct, K

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

$\mathsf{pt}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
return $H(\mathsf{pt}')$

$\Gamma(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$\mathsf{pt}^* \leftarrow\$ \mathcal{M}$
$K_0 \leftarrow H(\mathsf{pt}^*)$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}^*)$
$b' \leftarrow \mathscr{A}^H(\mathsf{pk}, \mathsf{ct}^*, K_0)$
return b'

$\Gamma'(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$\mathsf{pt}^* \leftarrow\$ \mathcal{M}$
$K_1 \leftarrow\$ \mathcal{K}$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}^*)$
$b' \leftarrow \mathscr{A}^H(\mathsf{pk}, \mathsf{ct}^*, K_1)$
return b'

$\mathscr{B}(\mathsf{pk}, \mathsf{ct}^*)$

$K \leftarrow\$ \mathcal{K}$
run $\mathscr{A}^H(\mathsf{pk}, \mathsf{ct}^*, K)$
$(\mathsf{pt}', h) \leftarrow\$ \mathscr{L}_H$
return pt'

Figure 2.11: From left to right: OW-CPA PKE to IND-CPA KEM transform, games Γ and Γ' for the proof in Example 2.2.1, and adversary $\mathscr{B}$ for the same proof. .

queries $H(\mathsf{pt}^*)$. Then, by Lemma 2.2.3, we have

$$\Pr[\Gamma] - \Pr[\Gamma'] \leq \Pr[\mathsf{bad}] \ .$$

Now, we show that if bad happens, one can build an adversary $\mathscr{B}$ that retrieves pt^*. The OW-CPA adversary $\mathscr{B}$ receives its own challenge ciphertext that encrypts pt^*, which is a valid KEM ciphertext. Therefore, $\mathscr{B}$ can simply run $\mathscr{A}$ with ct^* and a random key K. It also manages $\mathscr{A}$'s call to H by simulating a random oracle by lazy sampling and managing a list $\mathscr{L}_H$, as shown in Figure 2.10. Note that this perfectly simulates Γ' for $\mathscr{A}$. Then, if bad happens, then pt^* is in the list $\mathscr{L}_H$ managed by $\mathscr{B}$, which can then simply sample a random query pt out of it and output it as its answer to the OW-CPA game. The pseudocode of $\mathscr{B}$ is given in Figure 2.11. Overall, $\mathscr{B}$ wins when its guess is correct and bad occurred. Thus,

$$\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) = \frac{1}{q_H} \Pr[\mathsf{bad}] \ .$$

Hence,

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM}}(\mathscr{A}) = \Pr[\Gamma] - \Pr[\Gamma'] \leq \Pr[\mathsf{bad}] = q_H \cdot \mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) \ .$$

□

2.3 Quantum Computing and QROM

2.3.1 Quantum computation

Providing a complete introduction to quantum information theory and computation is obviously out of the scope of this thesis and we present only the main concepts below.

Hilbert space and quantum states. Quantum states are vectors in a Hilbert space $\mathscr{H}$, which in our case will be the complex space $\mathbb{C}^{2^n}$ for some n indicating the number of qbits in our system. These vectors are represented with a *ket* $|\cdot\rangle$ and their conjugate transpose with a *bra* $\langle\cdot|$. Moreover, $\langle x|y\rangle$ expresses the inner product between two vectors (i.e. states) $|x\rangle$ and $|y\rangle$.

Qbits. A qbit is simply a state in $\mathbb{C}^2$, the smallest space we are interested in (i.e. $n = 1$). If we consider the following basis: $\{|0\rangle, |1\rangle\}$, where $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$, then each qbit can be expressed as a linear combination of these vectors of norm 1. I.e. any qbit $|\psi\rangle$ can be expressed as $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ for $\alpha_0, \alpha_1 \in \mathbb{C}$ s.t. $|\alpha_0|^2 + |\alpha_1|^2 = 1$. If $\alpha_0 \neq 0$ and $\alpha_1 \neq 0$, we say the qbit is in *superposition*.

We can generalise the notion above to handle n qbits. That is, one can combine n systems of 1 qbit to form a system with n qbits via the tensor product $\otimes$. More precisely, we write $|\psi_1, \dots, \psi_n\rangle := |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \in \mathbb{C}^{2^n}$ for any qbits $|\psi_i\rangle \in \mathbb{C}^2,\ i \in [n]$.

Quantum states. In general, any (pure) quantum state of n qbits $|\phi\rangle \in \mathbb{C}^{2^n}$ can be expressed as a linear combination of a basis of the vector space, i.e. $|\phi\rangle = \sum_x \alpha_x |x\rangle$, where $\{|x\rangle\}_x$ is a basis of the Hilbert space and $\alpha_x \in \mathbb{C}$ with $\sum_x |\alpha_x|^2 = 1$. We will say the state is in superposition if there exist x, x' s.t. $x \neq x'$, $\alpha_x \neq 0$ and $\alpha_{x'} \neq 0$. In this thesis, we will use only the *computational basis* of $\mathbb{C}^{2^n}$, that is $\{|x\rangle\}_{x\in\{0,1\}^n}$.

Unitaries and computation. Quantum computation can be performed on quantum states through unitary transformations described as unitary matrices. A unitary matrix U is s.t. its conjugate transpose U^* is its inverse, i.e. $UU^* = U^*U = I$, for I the identity. In particular, every unitary computation is *invertible* and *linear* (i.e. it is applied to every state in the superposition).

A famous example of a unitary is the quantum CNOT gate which, given two qbits $|b_1, b_2\rangle$ with $b_1, b_2 \in \{0,1\}$, outputs $|b_1, b_1 \oplus b_2\rangle$. More generally, applying the CNOT gate on a quantum state $\sum_{b_1,b_2\in\{0,1\}} \alpha_{b_1,b_2}|b_1, b_2\rangle$ results in the state being transformed to $\sum_{b_1,b_2\in\{0,1\}} \alpha_{b_1,b_2}|b_1, b_1 \oplus b_2\rangle$. This illustrates the power of quantum computing: a unitary/gate can be applied on all states in the superposition at once.

In general, any function $f : \{0,1\}^i \mapsto \{0,1\}^o$ that can be computed classically can be implemented quantumly with a unitary that performs the map $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$, for $x \in \{0,1\}^i$ and $y \in \{0,1\}^o$.

Measurements. In practice, a state in superposition must be observed before any useful value can be extracted from it, this is where *measurement* comes into play. A state is measured according to a basis, which is the computational basis in our case. At measurement, a state

$|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ *collapses* into a state $|x'\rangle$ with probability $|\alpha_{x'}|^2$ for any $x' \in \{0,1\}^n$. In addition, the measurement outputs the value x'. Informally, this means that even if unitaries can be applied to all the states in the superposition at once, only one value can be retrieved at the end of some quantum computation.

Commutation. Let A and B be two quantum operations (i.e. unitaries or measurements). We say A and B *commute* if executing A and then B or B and then A in an algorithm does not affect the final result. We say they ϵ*-almost commute* if executing A and B in a different order affects the probability of obtaining a given result by at most ϵ. That is, let $\mathscr{A}$ be any algorithm where A is followed by B and let $\mathscr{A}'$ be the same as $\mathscr{A}$ except B is executed right before A. Then,

$$\left|\Pr[\mathscr{A} \Rightarrow x] - \Pr[\mathscr{A}' \Rightarrow x]\right| \le \epsilon$$

for any output x.

2.3.2 QROM

As stated above, any classically computable function f can be implemented by a unitary and that includes hash functions. This led to the definition of the Quantum Random Oracle Model (QROM) [Bon+11], where parties have *quantum* access to a random oracle. Formally, it means that participants are given black-box access to a unitary

$$U_H : |x, y\rangle \mapsto |x, y \oplus H(x)\rangle \ ,$$

where x (resp. y) is in the domain of x (resp. co-domain) and H is a random oracle. We often write $\mathscr{A}^{|H\rangle}$ (or even $\mathscr{A}^H$ when it is clear from the context) to denote the fact that an algorithm $\mathscr{A}$ has black-box access to the unitary U_H defined above.

As mentioned in the introduction, access to such a unitary makes QROM proofs more involved than their classical counterparts, and we present below several results that help overcome these difficulties.

One-Way to Hiding Lemma (OW2H). We first recall a variant of the well-known one-way to hiding lemma (OW2H) of Unruh [Unr15] as stated by Hofheinz et al. [HHK17]. Informally, this lemma states that if an adversary having quantum access to a RO H can distinguish with high probability between $H(x)$ and a uniform value, then one can extract the value x with high probability as well.

Lemma 2.3.1 (OW2H [HHK17]). *Let $\mathscr{A}$ be a quantum adversary making at most q_H queries to the quantum random oracle $|H\rangle$ with $H : \{0,1\}^\ell \mapsto \{0,1\}^n$ and outputting* 0 *or* 1. *Let* $\mathsf{Ext}_{q_H}^{\mathscr{A},|H\rangle}$ *be*

$\mathsf{Ext}^{\mathscr{A},H}(inp)$

```
1: i ←$ {1,...,q_H}
2: run A^H(inp) until i-th query |QUERY_i⟩
3: x' ← measure input register of |QUERY_i⟩
4: if A did not make i queries : return ⊥
5: return x'
```

Figure 2.12: Extractor Ext for the AOW2H lemma.

the algorithm in Figure 2.12. Then, for any algorithm F *that does not call* H*,*

$$\Big|\Pr\left[\mathscr{A}^{|H\rangle}(inp) \Rightarrow 1 | \sigma^* \leftarrow\!\!\$\, \{0,1\}^\ell; inp \leftarrow \mathsf{F}(\sigma^*, H(\sigma^*))\right]$$
$$- \Pr\left[\mathscr{A}^{|H\rangle}(inp) \Rightarrow 1 | (\sigma^*, K) \leftarrow\!\!\$\, \{0,1\}^{n+\ell}; inp \leftarrow \mathsf{F}(\sigma^*, K)\right]\Big|$$
$$\leq 2q_H \sqrt{\Pr\left[\sigma^* \leftarrow \mathsf{Ext}^{\mathscr{A},|H\rangle}(inp) | (\sigma^*, K) \leftarrow\!\!\$\, \{0,1\}^{n+\ell}; inp \leftarrow \mathsf{F}(\sigma^*, K)\right]} \quad .$$

Extractable random oracle. We also recall the notion of extractable RO-simulator introduced by Don et al. [Don+22], which is based on Zhandry's compressed oracle [Zha19].

The definition below is slightly simplified compared to the original version but it will be sufficient for the proofs presented in this dissertation; we refer the reader to Don et al.'s paper for more details.

Definition 2.3.1 (Theorem 4.3, Don et al. [Don+22])**.** *We say two quantum queries are* independent *if the input of one does not depend on the output of the other. An extractable RO-simulator is a tuple* $\mathsf{S} = (\mathsf{S.RO}, \mathsf{S.Ext})$*, where* $\mathsf{S.RO} : \{0,1\}^\ell \mapsto \{0,1\}^n$ *is a compressed RO and* $\mathsf{S.Ext}$ *is the extractor. Then,* S *satisfies the following properties:*

1. *As long as* $\mathsf{S.Ext}$ *is never called,* $\mathsf{S.RO}$ *is indistinguishable from a (standard) RO.*
2. *Subsequent independent queries to* $\mathsf{S.RO}$ *commute.*
3. *Subsequent independent queries to* $\mathsf{S.Ext}$ *commute.*
4. *Subsequent independent queries to* $\mathsf{S.RO}$ *and* $\mathsf{S.Ext}$ $8\sqrt{2/2^n}$*-almost commute.*
5. *Making multiple* identical *classical queries to* $\mathsf{S.RO}$ *(resp.* $\mathsf{S.Ext}$*) has the same effect on the state of* $\mathsf{S.RO}$ *as making* one *of these queries.*
6. *Let* $\widehat{x} \leftarrow \mathsf{S.Ext}(t)$ *for some* t*, and* $\widehat{t} \leftarrow \mathsf{S.RO}(\widehat{x})$ *be two subsequent classical queries. Then,*

$$\Pr[\widehat{t} \neq t \wedge \widehat{x} \neq \perp] \leq 2/2^n \; .$$

$\mathsf{COLL}(\mathscr{A})$

$(x_1, t_1), \ldots, (x_m, t_m) \leftarrow \mathscr{A}^{\mathsf{S.RO}}$
for $i \in \{1, \ldots, m\}$: $t'_i \leftarrow \mathsf{S.RO}(x_i)$
for $i \in \{1, \ldots, m\}$: $\widehat{x}_i \leftarrow \mathsf{S.Ext}(t_i)$
if $\exists i : \widehat{x}_i \neq x_i$ **and** $t_i = t'_i$:
  return 1
return 0

Figure 2.13: Collision game for Definition 2.3.1.

$\Gamma_f(\mathscr{A})$

$(t, o) \leftarrow \mathscr{A}^{\mathsf{S.RO}}$
$x \leftarrow f(o)$
$h \leftarrow \mathsf{S.RO}(x)$
return $1_{t=h}$

$\Gamma'_f(\mathscr{A})$

$(t, o) \leftarrow \mathscr{A}^{\mathsf{S.RO}}$
$x^\star \leftarrow \mathsf{S.Ext}(t)$
$x \leftarrow f(o)$
$h \leftarrow \mathsf{S.RO}(x)$
return $1_{t=h \wedge x=x^\star}$

Figure 2.14: Early extraction games for Lemma 2.3.2.

7. *Let* $t \leftarrow \mathsf{S.RO}(x)$ *for some* x, *and* $\widehat{x} \leftarrow \mathsf{S.Ext}(t)$ *be two subsequent classical queries. Then,*

$$\Pr[\widehat{x} = \perp] \leq 2/2^n\ .$$

8. *We consider the collision game in Figure 2.13. Then, for any* $\mathscr{A}$ *making at most* q *queries to* $\mathsf{S.RO}$ *and outputting* m *tuples we have*

$$\Pr[\mathsf{COLL}(\mathscr{A}) \Rightarrow 1] \leq \frac{40e^2(q+m+1)^3 + 2}{2^n}\ .$$

Finally, the following lemma will be useful.

Lemma 2.3.2 (Early Extraction)**.** *Let* Γ *and* Γ' *be the games described in Figure 2.14. Then,*

$$\Pr[\Gamma \Rightarrow 1] - \Pr[\Gamma' \Rightarrow 1] \leq \frac{2}{2^n} + 8\sqrt{2/2^n} + \frac{40e^2(q_H+2)^3 + 2}{2^n}.$$

Proof. This follows from Corollary 4.7 in Don et al. [Don+22] and the fact that if $h = t$, where $h = \mathsf{S.RO}(x)$, then $\Pr[x^\star = \perp] \leq \frac{2}{2^n}$. □

2.4 FO-like Transforms

Fujisaki and Okamoto introduced one of the first generic IND-CPA to IND-CCA transforms for PKE [FO99; FO13] in 1999, which is detailed in Figure 2.15. This construction illustrates the

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Enc}(\mathsf{pk},\mathsf{pt})$	$\mathsf{Dec}(\mathsf{sk},(\mathsf{ct}_1,\mathsf{ct}_2))$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{Gen}'(1^\lambda)$	$k \leftarrow\$ \mathcal{K}$	$k' \leftarrow \mathsf{Dec}'(\mathsf{sk},\mathsf{ct}_2)$
return $(\mathsf{pk},\mathsf{sk})$	$\mathsf{ct}_1 \leftarrow E_k(\mathsf{pt})$	**if** $\mathsf{ct}_2 \neq \mathsf{Enc}'(\mathsf{pk},k';G(k',\mathsf{ct}_1))$:
	$\mathsf{ct}_2 \leftarrow \mathsf{Enc}'(\mathsf{pk},k;G(k,\mathsf{ct}_1))$	**return** $\perp$
	return $(\mathsf{ct}_1,\mathsf{ct}_2)$	$\mathsf{pt}' \leftarrow D_{k'}(\mathsf{ct}_1)$
		return pt'

Figure 2.15: The original Fujisaki-Okamoto (FO) transform, where $(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$ is the underlying PKE, G is a hash function modelled as a random oracle, and $\mathcal{K}$ is the keyspace of the underlying symmetric cipher (E,D). The resulting PKE is $(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$.

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Enc}(\mathsf{pk},\mathsf{pt})$	$\mathsf{Dec}(\mathsf{sk},\mathsf{ct})$
return $\mathsf{Gen}'(1^\lambda)$	$\mathsf{coins} \leftarrow G(\mathsf{pt})$	$\mathsf{pt}' \leftarrow \mathsf{Dec}'(\mathsf{sk},\mathsf{ct})$
	return $\mathsf{Enc}'(\mathsf{pk},\mathsf{pt};\mathsf{coins})$	**if** $\mathsf{pt}' = \perp$ **or** $\mathsf{Enc}(\mathsf{pk},\mathsf{pt}') \neq \mathsf{ct}$
		return $\perp$
		return pt'

Figure 2.16: T transform, where $(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$ is the underlying PKE, G is a hash function modelled as a random oracle, and $(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ is the resulting PKE.

KEM/DEM paradigm, where a PKE transformed into a KEM is used to transport a symmetric key, which is itself used to encrypt a message of arbitrary length.

Other transforms were introduced in the following years (e.g. [Jea+02; OP01]) but the topic only took off recently, due to the heavy use of CPA-to-CCA transforms in the NIST PQ proposals. In particular, several variants of the Fujisaki-Okamoto (FO) transform that build IND-CCA KEMs out of CPA-secure PKEs have been proposed [SXY18; TU16; Bin+19b]. We recall here four constructions of Hofheinz et al. [HHK17], which generalise and decompose FO-like transforms in smaller parts.

T. The first one is the T transform (presented in Figure 2.16), which takes an OW/IND-CPA PKE $\mathsf{PKE}' = (\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$ and outputs a *rigid* OW-PVCA PKE scheme $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$, where $G : \{0,1\}^* \mapsto \{0,1\}^\lambda$ is a hash function modelled as a RO. Informally, the transform derandomises the underlying scheme by computing the random coins as the hash of the message. Then, the decryption function checks that the ciphertexts are well-formed by re-encrypting the decrypted message.

Then, the following theorems formally state the security of the T transform in the ROM and QROM, respectively.

Theorem 2.4.1 (OW-CPA $\xrightarrow{\text{ROM}}$ OW-PVCA, Theorem 3.1 [HHK17])**.** *Let G be a hash function modelled as a random oracle,* PKE' *a γ-spread and $\delta(q_G)$-correct PKE scheme, and* PKE *the* *resulting PKE after applying* T. *Then, for any OW-PVCA adversary* $\mathscr{A}$ *issuing at most* q_G, q_V, q_P *queries to G,* $\mathscr{O}^{\mathsf{VCO}}$*, and* $\mathscr{O}^{\mathsf{PCO}}$*, respectively, there exists an OW-CPA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pvca}}(\mathscr{A}) \leq (q_G + q_P)\cdot\delta + q_V \cdot 2^{-\gamma} + (q_G + q_P + 1)\cdot \mathsf{Adv}_{\mathsf{PKE}'}^{\mathrm{ow-cpa}}(\mathscr{B}) ,$$

where the running time and the number of queries of $\mathscr{B}$ *are similar to the ones of* $\mathscr{A}$. *Moreover,* PKE *is rigid (and deterministic).*

Theorem 2.4.2 (OW-CPA $\overset{\text{QROM}}{\Longrightarrow}$ OW-PCA, Theorem 3.1 [HHK17])**.** *Let G be a hash function modelled as a quantumly accessible random oracle,* PKE' *a* $\delta(q_G)$*-correct PKE scheme, and* PKE *the resulting PKE after applying* T. *Then, for any OW-PVCA adversary* $\mathscr{A}$ *issuing at most* q_G, q_P *queries to G and* $\mathscr{O}^{\mathsf{PCO}}$*, respectively, there exists an OW-CPA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{A}) \leq 8\cdot(q_G + q_P + 1)^2\cdot\delta + (1 + 2q_G)\cdot\sqrt{\mathsf{Adv}_{\mathsf{PKE}'}^{\mathrm{ow-cpa}}(\mathscr{B})} ,$$

where the running time and the number of queries of $\mathscr{B}$ *are similar to the ones of* $\mathscr{A}$. *Moreover,* PKE *is rigid (and deterministic).*

In addition, the following theorem gives an upper bound on the correctness of the resulting PKE in the QROM.

Theorem 2.4.3 (Lemma 4.3, [HHK17])**.** *Let* PKE' *be* δ'*-correct and* PKE *be the PKE obtained from applying* T *to* PKE'. *Then,* PKE *is* δ*-correct with*

$$\delta \leq 8\cdot\delta'\cdot(q_G + 1)^2 ,$$

where q_G *is the number of quantum queries one can make to the random oracle.*

$\mathsf{U}^{\perp}$ **and** $\mathsf{U}^{\not\perp}$**.** We present now two other transforms called $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$. These transforms convert an OW-PVCA (resp. OW-PCA) PKE into an IND-CCA KEM. The transforms are shown in Figure 2.17. The only difference between both transforms is that in $\mathsf{U}^{\not\perp}$ the rejection is implicit, i.e. when an error occurs during decryption a random key is returned instead of the error symbol. The ROM security of these constructions is formally stated in the following theorems.

Theorem 2.4.4 (PKE OW-PVCA $\overset{\text{ROM}}{\Longrightarrow}$ KEM IND-CCA, Theorem 3.3 [HHK17])**.** *Let H be a random oracle,* PKE *a* δ*-correct PKE scheme, and* KEM *the resulting KEM after applying* $\mathsf{U}^{\perp}$ *on* PKE. *Then, for any IND-CCA adversary* $\mathscr{A}$ *issuing at most* q_H, q_D *queries to H and* $\mathscr{O}^{\mathsf{Dec}}$*, respectively, there exists an OW-PVCA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-cca}}(\mathscr{A}) \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pvca}}(\mathscr{B}) ,$$

where $\mathscr{B}$ *has roughly the same running time as* $\mathscr{A}$ *and* $\mathscr{B}$ *makes at most* q_H *queries to both* $\mathscr{O}^{\mathsf{PCO}}$ *and* $\mathscr{O}^{\mathsf{VCO}}$.

$\mathsf{Gen}(1^\lambda)$

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}'(1^\lambda)$
- **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Encaps}(\mathsf{pk})$

- $\mathsf{pt} \leftarrow\$ \mathcal{M}$
- $\mathsf{ct} \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{pt})$
- $K \leftarrow H(\mathsf{pt}, \mathsf{ct})$
- **return** ct, K

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

- $\mathsf{pt}' \leftarrow \mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$
- **if** $\mathsf{pt}' = \perp$: **return** $\perp$
- **return** $H(\mathsf{pt}', \mathsf{ct})$

$\mathsf{Gen}(1^\lambda)$

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}'(1^\lambda)$
- $s \leftarrow\$ \mathcal{M}$
- $\mathsf{sk} \leftarrow (\mathsf{sk}, s)$
- **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Encaps}(\mathsf{pk})$

- $\mathsf{pt} \leftarrow\$ \mathcal{M}$
- $\mathsf{ct} \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{pt})$
- $K \leftarrow H(\mathsf{pt}, \mathsf{ct})$
- **return** ct, K

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

- parse $\mathsf{sk}, s \leftarrow \mathsf{sk}$
- $\mathsf{pt}' \leftarrow \mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$
- **if** $\mathsf{pt}' = \perp$: **return** $H(s, \mathsf{ct})$
- **return** $H(\mathsf{pt}', \mathsf{ct})$

$\mathsf{Gen}(1^\lambda)$

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}'(1^\lambda)$
- **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Encaps}(\mathsf{pk})$

- $\mathsf{pt} \leftarrow\$ \mathcal{M}$
- $\mathsf{ct} \leftarrow \mathsf{Enc}'(\mathsf{pk}, \mathsf{pt})$
- $\mathsf{tag} \leftarrow H'(\mathsf{pt})$
- $\mathsf{ct} \leftarrow (\mathsf{ct}, \mathsf{tag})$
- $K \leftarrow H(\mathsf{pt})$
- **return** ct, K

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

- parse $\mathsf{ct}, \mathsf{tag} \leftarrow \mathsf{ct}$
- $\mathsf{pt}' \leftarrow \mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$
- **if** $\mathsf{pt}' = \perp$ **or** $H'(\mathsf{pt}') \neq \mathsf{tag}$:
  - **return** $\perp$
- **return** $H(\mathsf{pt}')$

Figure 2.17: $\mathsf{U}^\perp$ (top), $\mathsf{U}^{\not\perp}$ (middle), and $\mathsf{QU}_m^\perp$ (bottom) transforms from Hofheinz et al. [HHK17]. $\mathsf{PKE}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ is the underlying PKE that is transformed into a KEM $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$, and H, H' are hash functions modelled as random oracles.

Theorem 2.4.5 (PKE OW-PCA $\overset{\text{ROM}}{\Longrightarrow}$ KEM IND-CCA, Theorem 3.4 [HHK17])**.** *Let H be a random oracle,* PKE *a δ-correct PKE scheme, and* KEM *the resulting KEM after applying* $\mathsf{U}^{\not\perp}$ *on* PKE*. Then, for any IND-CCA adversary $\mathscr{A}$ issuing at most q_H queries to H, there exists an OW-PCA adversary $\mathscr{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-cca}}(\mathscr{A}) \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{B}) + \frac{q_H}{|\mathscr{M}|},$$

where the running time of $\mathscr{B}$ is roughly the same as the one of $\mathscr{A}$ and $\mathscr{B}$ makes at most q_H queries to $\mathscr{O}^{\mathsf{PCO}}$.

$\mathsf{QU}_m^\perp$**.** The last transform we present is the one called $\mathsf{QU}_m^\perp$, presented at the bottom of Figure 2.17. The difference with $\mathsf{U}^\perp$ and $\mathsf{U}^{\not\perp}$ is that $\mathsf{QU}_m^\perp$ attaches a confirmation hash to the ciphertext, which is then checked at decapsulation. This allows for an easier proof in the QROM, and the security of the transform in this model is stated in the following theorem. We note that $\mathsf{QU}_m^\perp$ needs the underlying PKE to be rigid.

Theorem 2.4.6 (PKE OW-PCA $\overset{\text{QROM}}{\Longrightarrow}$ KEM IND-CCA, Theorem 4.5 [HHK17])**.** *Let H, H' be quantumly accessible random oracles,* PKE *a δ-correct rigid PKE scheme, and* KEM *the resulting*

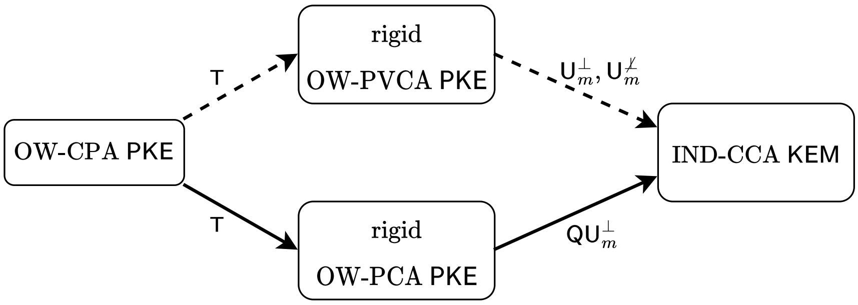


Figure 2.18: Transforms and results from Hofheinz et al. [HHK17]. Solid arrows denote QROM (thus ROM) security, dashed arrows denote ROM security.

KEM after applying $\mathsf{QU}^{\perp}_{m}$ *on* PKE*. Then, for any IND-CCA adversary* $\mathscr{A}$ *issuing at most* q_H *(resp.* $q_{H'}$*) queries to* H *(resp.* H'*), there exists an OW-PCA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathscr{A}) \leq (2q_{H'} + q_H + 2q_D) \cdot \sqrt{\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{B})} + \delta \ ,$$

where the running time of $\mathscr{B}$ *is roughly the same as the one of* $\mathscr{A}$ *and* $\mathscr{B}$ *makes at most* $q_D q_{H'}$ *queries to* $\mathscr{O}^{\mathsf{PCO}}$.

Summary. We present an illustration of the different transforms in Figure 2.18. In short, the T construction builds a rigid OW-P(V)CA PKE from a OW-CPA PKE in both the ROM and the QROM. Then, in the ROM, both $\mathsf{U}^{\perp}$ and $\mathsf{U}^{\not\perp}$ can be used to build an IND-CCA KEM from the PKE output by T, while in the QROM the $\mathsf{QU}^{\perp}_{m}$ transform can be employed.

3 Classical Misuse Attacks on NIST Round 2 PQC: The Power of Rank-based Schemes

In this chapter, we study the resistance against KR-PCA attacks of the CPA-secure PKEs underlying several NIST proposals. That is, most PQ IND-CCA KEMs are built applying a FO-like transform to a IND/OW-CPA PKE; however, while the CPA scheme is not meant to be secure if the secret key is used more than once, it is usually simpler and more efficient than its strongly secure counterpart. Therefore, the threat of misuse of the weaker construction by non-experts in the implementation stage is high. For instance, if the CPA-secure PKE is part of a key-exchange protocol and the secret key is reused, an adversary might be able to mount attacks by sending carefully crafted ciphertexts to the server and observe its behaviour while trying to establish a shared secret (e.g. the server might return errors when the shared secret on the server side does not match the adversary's). This type of attacks is sometimes called *reaction attacks* in the literature, and the plaintext-checking attack (PCA) model somewhat captures these.

Another concern is the mis-implementation of the FO transform. For example, it was mentioned by Lepoint [Lep18] that badly implemented KEMs could leak information about the underlying CPA construction via side channels. More precisely, these implementations leaked whether the decryption of a ciphertext was correct or not, and several timing attacks exploiting this flaw were subsequently proposed (e.g. [DAn+19b; Bet+19]). Again, these side-channel attacks can be abstracted as plaintext-checking attacks.

The content of this chapter is a joint work with Serge Vaudenay and was published at ACNS 2020 [HV20]. The technique used to mount our KR-PCA attacks is inspired by the one we developed in a previous work published at EUROCRYPT 2019 [Băe+19]. At the time of publication, the NIST had announced the proposals that passed to the second round of the standardisation process, and these are the schemes considered in this chapter. It is worth noting that parameters of the algorithms that passed to the third round and beyond (e.g. Kyber) might have changed, and therefore the attacks presented below might not apply *as is* to these newer versions.

Table 3.1: KR-PCA on NIST round 2 post-quantum cryptosystems. For each attack, we report the number of unknowns in the key, the number of oracle calls to recover the private key, and the expected number of oracle calls, respectively. Values are rounded to the closest power of 2. The results presented in this thesis are highlighted.

Schemes	Unknowns	max. #queries	$\mathbb{E}$[#queries]
CRYSTALS-Kyber-512	2^{10}	2^{11}	2^{10}
Frodo-640 [Băe+19]	2^{12}	2^{16}	-
HQC-128	2^{15}	2^{16}	2^{16}
LAC-128	2^{9}	2^{11}	2^{11}
NewHope1024 [QCD19a]	2^{10}	-	2^{20}
Round5 (HILA5) [Ber+18]	2^{10}	-	2^{13}
RQC-I	2^{13}	2^{67}	$\leq 2^{38}$
SABER (LightSaber)	2^{9}	2^{11}	2^{11}

3.1 Contributions

We present several key-reuse attacks in the KR-PCA model (see Definition 2.2.7). More precisely, we design KR-PCA attacks against the following NIST round 2 proposals: HQC, LAC, CRYSTALS-Kyber, SABER, and RQC. In our attacks (except the one against RQC), only a few thousands queries to the oracle are needed to recover the private key. Moreover, the complexity is polynomial in the size of the parameters. The only exception is RQC [Mel+19a], a rank-metric proposal, for which our best attack is exponential (but still practical for the proposed parameters). We report our and other existing results against round 2 candidates in Table 3.1. We included external results only when the attack was in the same model as ours and targeted explicitly a version of a cryptosystem submitted to the NIST process. This does not mean that other round 2 candidates are not vulnerable to existing reaction attacks. Actually, apart from the schemes targeted in this thesis, nearly all round 2 candidates have existing reaction attacks against them or similar schemes (e.g. attacks against ROLLO [Sam+19], LEDACrypt [FHZ18], NTRU [How+03], the attack by Guo et al. [GJS16] probably works on BIKE, etc.).
For each scheme, we indicate the number of unknowns in the secret key in $\mathbb{Z}_q$, the maximal and expected number of queries necessary to recover the key. Concretely, the number of oracle calls can be seen as the number of times the key must be reused before the adversary can recover it. As a proof-of-concept, we also implemented the attacks against CRYSTALS-Kyber and SABER.

In addition, we show that the learning problem is hard in the rank-metric for some parameters. As most key-reuse attacks solve an instance of the learning problem in order to recover the key, this result demonstrates that such a strategy is not applicable to rank-based schemes. We stress that this result does not prove that efficient KR-PCA are impossible in the rank-metric but that common techniques are not applicable, which is still significant. From a more information-theoretical point of view, this confirms the intuition that the rank distance between a secret and a given value leaks much less information on the secret than other distances such as Hamming.

3.2 Related Work

Reaction attacks is an old topic in cryptography and one of the most famous examples is Bleichenbacher's attack against RSA published in 1998 [Ble98]. The term *reaction attack* was probably first mentioned by Hall et al. in 1999 [HGS99]. In that paper, the authors showed that in the McEliece scheme, an adversary can recover a plaintext by observing decryption results of erroneous ciphertexts. In 2003, Howgrave-Graham et al. presented a reaction attack against the NTRU cryptosystem, which recovers the secret key [How+03]. More recently, several key-reuse and reaction attacks against post-quantum cryptosystems were published (e.g. attacks against QC-MDPC [GJS16], LEDApkc [FHZ18], NewHope [Bau+19], HILA5 [Ber+18], etc.). In 2016, Fluhrer [Flu16] and Ding et al. [Din+17] showed how key-reuse can be exploited against Ring-LWE based schemes.

In 2019, we introduced a framework capturing the similar structure shared by lattice-based proposals [Băe+19]. In the same paper, the notion of key-recovery under plaintext-checking attack (KR-PCA) was presented, which formalised the concept of reaction attacks. More notably, we designed several misuse attacks against NIST candidates. It was shown that with a few thousand queries, many proposals can be broken if the secret key is reused. The algorithms attacked were (R.)EMBLEM, Frodo, KINDI, LIMA, LOTUS and Titanium. However, results against several NIST round 2 candidates were missing and we complete the picture in this chapter.

In that same paper [Băe+19], we also introduced the concept of *learning problem* where an adversary tries to recover a secret value, having access to an oracle that returns whether the distance between the secret and a given input is below some threshold. It was shown that an efficient learning algorithm was sufficient to design a practical KR-PCA attack in most cases. Interestingly, many key-reuse attacks solve an instance of the learning problem in one way or another in order to recover the key (e.g. [Băe+19; Bau+19; Din+17]).

In an independent and concurrent work, D'Anvers et al. [DAn+19b] introduced a timing attack against LAC similar to our KR-PCA attack. Finally, in another independent and concurrent work, Qin et al. [QCD19b] presented a reaction attack against Kyber similar to ours. The performance of their best attack is similar to ours, even if our algorithm seems to perform slightly better on average, at least for Kyber512.

Subsequent work. Our results proved to be useful for designing side-channel attacks against the schemes we targeted. For instance, Ueno et al. [Uen+22] used our key-recovery algorithm against Lightsaber to perform a side-channel attack against that same algorithm. Another example is the fault-injection attack against Kyber by Xagawa et al. [Xag+21], which is inspired by our KR-PCA attack presented in Section 3.5.

$\text{LEARN}_{\Psi,\rho,\|\cdot\|}(\mathscr{A})$	**Oracle** $\mathscr{O}^{\mathsf{learn}}(x)$
1 : $\delta \leftarrow\!\!\$\ \Psi$	1 : **return** $1_{\|\delta+x\|\leq\rho}$
2 : $\delta' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{learn}}}$	
3 : **return** $1_{\delta'=\delta}$	

Figure 3.1: Learning game.

3.3 The Learning Problem

The learning problem [Băe+19] is defined by the game detailed in Figure 3.1, which is parametrised by a threshold ρ, a secret value distribution Ψ, and a norm $\|\cdot\|$. The adversary has access to the public parameters and to the oracle $\mathscr{O}^{\mathsf{learn}}$ and tries to guess the secret δ.

We showed [Băe+19] that for most of the lattice-based schemes of the NIST competition, the KR-PCA game reduces to the LEARN game. In addition, for most common norms (e.g. Hamming, L_1 in $\mathbb{Z}_q$, ...) the learning game can be solved in a logarithmic number of queries in the size of the secret domain D (i.e. $O(\log_2(|D|))$ for $\delta \in D$). This led to the design of several efficient KR-PCA attacks.

3.4 KR-PCA Attack against LAC

3.4.1 The LAC-CPA algorithm

We start by explaining how the LAC cryptosystem [Lu+19] works. Let $\mathscr{R}_q := \mathbb{Z}_q[X]/(X^n+1)$ for some parameter n. For $v \in \mathscr{R}_q, x \in \mathbb{Z}_q$, let $h(v,x) := |\{i : v_i = x, i \in [n]_-\}|$ be the function that counts the number of coefficients set to x in v. Then, let $S_w := \{v : v \in \mathscr{R}_q, h(v,-1) = h(v,1) = \frac{w}{2}\}$ for some even parameter w be the set of polynomials in $\mathscr{R}_q$ that contains exactly $\frac{w}{2}$ 1s and -1s. In addition, we consider a centered binary distribution ψ_σ on $\{-1,0,1\}$ with variance σ, a BCH code [Hoc59; BR60] of error-correcting capacity t with codewords in $\mathbb{Z}_q^{\ell_v}$, and a message space $\mathscr{M} := \{0,1\}^k$, where σ, t, ℓ_v, and k are given as parameters.

The scheme then works as follows:

- Gen: Sample $(\mathsf{sk}, d) \leftarrow\!\!\$\ S_w^2$ and $A \leftarrow\!\!\$\ \mathscr{R}_q$. Set $\mathsf{pk} = (A, B = A \times \mathsf{sk} + d)$.

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt} \in \{0,1\}^k)$: Sample $(t, e, f) \leftarrow\!\!\$\ S_w^2 \times \Psi_\sigma^{\ell_v}$ and output

$$(U, V) \leftarrow \left(t \times A + e, (t \times B)_{\ell_v} + f + \left\lceil \frac{q}{2} \right\rfloor \times \mathsf{encode}_{\text{BCH}}(\mathsf{pt})\right).$$

- $\mathsf{Dec}(\mathsf{sk}, U, V)$: Compute $W \leftarrow V - (U \times \mathsf{sk})_{\ell_v}$ and output $\mathsf{decode}(W)$, where decode computes W' with

$$W'_i = \begin{cases} 1, & \text{if } \lceil \frac{q}{4} \rfloor \le W_i < \lceil \frac{3q}{4} \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{3.1}$$

and then outputs $\mathsf{decode}_{\mathrm{BCH}}(W')$.

3.4.2 KR-PCA

W.l.o.g. we use $\mathsf{pt} = 0^k$ in the KR-PCA attack. Then, we have

$$\mathsf{encode}_{\mathrm{BCH}}(\mathsf{pt}) = 0^{\ell_v} \in \mathbb{Z}_q^{\ell_v}.$$

Now, since the BCH code can correct up to t errors, the decryption of some ciphertexts (U, V) will be incorrect (i.e. $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V)) = 0$) iff for at least t of the components of W we have $W_i \in [\lceil \frac{q}{4} \rfloor, \lceil \frac{3q}{4} \rfloor)$ by Eq. (3.1). Therefore, we can consider the following plaintext-checking attack (see Figure 3.2 for the detailed pseudocode).

First, we set $U := -(\lceil \frac{q}{4} \rfloor - 1) \in \mathcal{R}_q$ (i.e. a constant polynomial). Then, we observe that

$$1 + (-U \times \mathsf{sk})_i \notin \left[-\lceil \frac{q}{4} \rfloor, \lceil \frac{q}{4} \rfloor\right) \Leftrightarrow \mathsf{sk}_i = 1 \tag{3.2}$$

$$-2 + (-U \times \mathsf{sk})_i \notin \left[-\lceil \frac{q}{4} \rfloor, \lceil \frac{q}{4} \rfloor\right) \Leftrightarrow \mathsf{sk}_i = -1. \tag{3.3}$$

Next, let $V = \mathbf{1} \in \mathbb{Z}_q^{\ell_v}$ be the vector with 1 in every component. By Eq. (3.2), if there are more than t ones in sk, $V - (U \times \mathsf{sk})_{\ell_v}$ will decode incorrectly and $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V))$ will return a failure. Then, by iteratively cutting the number of 1s in V by half and querying the oracle, one can perform a binary search to find $\tilde{V} = (\tilde{V}_0, \ldots, \tilde{V}_{\ell_v}), \tilde{V}_i \in \{0, 1\}$ s.t. $\tilde{V} - (U \times \mathsf{sk})_{\ell_v}$ contains exactly t errors. Finally, given this vector $\tilde{V}$, one can perform the following algorithm.

1. Let $V := \tilde{V}$ and $\mathcal{J} := \{i : \tilde{V}_i \neq 1\}$ be the subset of indices i for which $\tilde{V}_i$ $(= V_i)$ is not 1. Then, let's pick some $i \in \mathcal{J}$ and set $V_i = 1$. If the plaintext-checking oracle returns an error on $(\mathsf{pt}, (U, V))$, it means that $t + 1$ errors have been detected and thus the decoding of the ith component failed. In turn, that implies that condition in Eq. (3.2) is fulfilled. Hence, we know that $\mathsf{sk}_i = 1$. Otherwise, if the oracle returns no error, we set $V_i = -2$ and query again. If an error is returned it means $\mathsf{sk}_i = -1$ by Eq. (3.3), otherwise $\mathsf{sk} = 0$. One can iterate for every $i \in \mathcal{J}$. Thus, at the end of this step, we recovered all sk_i s.t. $i \in \mathcal{J}$.
2. To get the other components of sk, we set $V := \tilde{V}$ as in the beginning of step 1 but we add an extra error such that $V - (U \times \mathsf{sk})_{\ell_v}$ contains $t + 1$ errors (we can do it easily since we know some values sk_i). Then, for each i s.t. $V_i = 1$ (i.e. $i \notin \mathcal{J}$), we proceed as follows. We set $V_i = 0$ and query the oracle. If the oracle does not return an error, it means the ith component was part of the $t + 1$ errors (i.e. Eq. (3.2) was fulfilled) and therefore $\mathsf{sk}_i = 1$.

Otherwise, if the oracle returns an error, we thus know $\mathsf{sk}_i \in \{-1,0\}$. Let $\mathscr{I}$ be the indices of such components.

3. Set $V := \tilde{V}$ (i.e. $V - (U \times \mathsf{sk})_{\ell_v}$ contains t errors). For each $i \in \mathscr{I}$, set $V_i = -2$. If the oracle returns an error, it means that Eq. (3.3) is fulfilled and thus $\mathsf{sk}_i = -1$, otherwise $\mathsf{sk}_i = 0$. Hence, we can recover all components sk_i for $i \in \{1, \ldots, \ell_v\}$.

3.4.3 Remarks and results

Note that we assumed that $(\mathsf{sk})_{\ell_v}$ contained more than t ones for the binary search to succeed in finding $\tilde{V}$. If this is not the case, we can still perform the attack by first looking for $\tilde{V}, \tilde{V}_i \in \{-1,0\}$ s.t. the decryption contains t errors and modify the signs in the attack. Note that for the parameters considered by LAC authors, it is very unlikely that sk contains less than t 1s (same for -1s). For example, for LAC128 ($n = 512, w = 256, \ell_v = 400, t = 16, \sigma = 1$), the probability to have less than t ones and minus ones in $(\mathsf{sk})_{\ell_v}$ if we assume each component i.i.d. with $\Pr[\mathsf{sk}_i = 0] = \Pr[\mathsf{sk}_i \in \{-1,1\}] = \frac{1}{2}$ is

$$\Pr\left[|\{i : \mathsf{sk}_i = 0, \mathsf{sk}_i \in (\mathsf{sk})_{\ell_v}\}| > \ell_v - t\right] = \sum_{i=\ell_v - t+1}^{\ell_v} \frac{1}{2^{\ell_v}} \binom{\ell_v}{i} \approx 2^{-311}.$$

In the worst case, we perform the binary search and query the oracle 2 times for each component, thus the total number of queries is $\log_2(\ell_v) + 2 \times \ell_v$. Hence, since $\ell_v = 400$, we can recover 400 unknowns of sk in at most $\log_2(400) + 2 \times 400 \approx 2^{10}$ queries.

Now, the attack presented above can recover the ℓ_v leftmost coefficients. We can recover the $n - \ell_v$ remaining coefficients by applying the same attack using $U = (\lceil \frac{q}{4} \rceil - 1) \times X^{n-\ell_v}$. This will shift the $n - \ell_v$ coefficients to the leftmost positions (note that $-X^n = 1$ in $\mathscr{R}_q$). Hence, we need to apply at most two times the attack, resulting in a total number of queries smaller than 2^{11}.

Also, in the round 2 specifications of LAC [Lu+19], each component of V has its 4 least significant bits dropped after encryption. At decryption, each component is thus multiplied by 2^4. This does not impact our attack as Eq. (3.2)-(3.3) still hold with $\pm 2^4$ instead of $1, -2$.

3.5 Misuse Attack against CRYSTALS-Kyber

3.5.1 Kyber-CPA

We first describe the CPA-secure PKE underlying Kyber, which is called Kyber-CPA. As in LAC, the scheme works in $\mathscr{R}_q := \mathbb{Z}_q[X]/(X^n + 1)$ for some prime parameter q. Elements are sampled from a distribution Ψ_η which is computed as

```
LAC_KR_PCA(pk)
(A, B) ← pk
pt ← 0^k
U ← −(⌈q/4⌋ − 1) ∈ ℛ_q
Find Ṽ s.t. decode(Ṽ − U × sk) detects t errors.
𝒥 ← {i : Ṽ_i ≠ 1}
for i ∈ 𝒥 :
  V ← Ṽ; V_i ← 1
  r ← 𝒪^PCO(pt, (U, V))
  if r = 0 :
    sk_i ← 1; continue
  V_i ← −2; r ← 𝒪^PCO(pt, (U, V))
  if r = 0 :
    sk_i ← −1; continue
  sk_i ← 0
Set Ṽ′ s.t. decode(V − U × sk) detects t+1 errors
ℐ = ∅
for i ∈ [ℓ_v]_− \ 𝒥 :
  V ← Ṽ′; V_i ← 0
  r ← 𝒪^PCO(pt, (U, V))
  if r = 1 :
    sk_i ← 1; continue
  ℐ ← ℐ ∪ {i}    // sk_i ∈ {−1, 0}
for i ∈ ℐ :
  V ← Ṽ; V_i ← −2
  r ← 𝒪^PCO(pt, (U, V))
  if r = 0 :
    sk_i ← −1; continue
  sk_i ← 0
return sk
```

Figure 3.2: KR-PCA adversary against LAC-CPA.

Ψ_η

$\{(a_i, b_i)\}_{i\in[\eta]} \leftarrow\$ \{0,1\}^{2\times\eta}$

return $\sum_{i=1}^{\eta}(a_i - b_i)$

with $\eta = 2$. Thus, Ψ_η returns a value in $\{-2,-1,0,1,2\}$. For a polynomial $P \in \mathscr{R}_q$, we write $P \leftarrow\$ \Psi_\eta$ to denote that each component of P is sampled independently from Ψ_η. Moreover, we define

$$\begin{aligned}\mathsf{compress}(x,d) &:= \left\lceil \frac{2^d}{q} \times x \right\rfloor \bmod 2^d\\ \mathsf{decompress}(x,d) &:= \left\lceil \frac{q}{2^d} \times x \right\rfloor.\end{aligned}$$

Such functions guarantee that for any $x \in \mathbb{Z}_q$, we have

$$\left|\langle x - \mathsf{decompress}(\mathsf{compress}(x,d),d)\rangle_q\right| \le \left\lceil \frac{q}{2^{d+1}} \right\rfloor.$$

When we apply these functions to vectors or polynomials in $\mathscr{R}_q$, we assume they are applied to each coefficient. Then, Kyber-CPA works as follows, where $n, k, d_u, d_V \in \mathbb{Z}$ are parameters.

- Gen: Sample $A \leftarrow\$ \mathscr{R}_q^{k\times k}$ and $(\mathsf{sk}, d) \leftarrow\$ (\Psi_\eta^k)^2$. Set $\mathsf{pk} \leftarrow (A, B) = (A, A\times\mathsf{sk} + d)$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{pt} \in \{0,1\}^n)$: Sample $(t, e, f) \leftarrow\$ (\Psi_\eta^k)^2 \times \Psi_\eta$. Compute $(U, V) \leftarrow (t\times A + e, t\times B + f + \lceil \frac{q}{2} \rfloor \times \mathsf{pt}) \in \mathscr{R}_q^k \times \mathscr{R}_q$. Output $(\mathsf{compress}(U, d_U), \mathsf{compress}(V, d_V))$.
- $\mathsf{Dec}(\mathsf{sk}, U', V')$: Compute $(U, V) \leftarrow (\mathsf{decompress}(U', d_U), \mathsf{decompress}(V', d_V))$. Return $\mathsf{compress}(V - U\times\mathsf{sk}, 1)$.

We note that with the parameters proposed by the authors, we have

$$\mathsf{compress}(x,1) = \begin{cases} 0, & \text{if } -\lceil \frac{q}{4} \rfloor \le \langle x\rangle_q \le \lceil \frac{q}{4} \rfloor \\ 1, & \text{otherwise}\end{cases}. \tag{3.4}$$

Finally, we define δ as $V - U \times \mathsf{sk} = \delta + \mathsf{encode}(\mathsf{pt})$.

3.5.2 KR-PCA

From now on, we consider the parameters proposed by the authors for the round 2 version of Kyber512, namely $n = 256, q = 3329, \eta = 2, d_U = 10$, and $d_V = 3$. Let's also assume $k = 1$ for now. In the plaintext-checking attack, we use the message with all components set to 0 (i.e. $\mathsf{pt} = 0 \in \mathscr{R}_q$) for the sake of simplicity, although some minor changes would allow the attack to work for any pt. In addition, we let $\rho := \lceil \frac{q}{4} \rfloor$. Then, by the definition of Dec and Eq. (3.4), we know the plaintext-checking oracle (PCO) will return 1 (i.e. success) iff $|\langle \delta_i \rangle_q| \le \rho, \forall i \in [n]$. First, we state the following lemma.

Lemma 3.5.1. *Let* $U := -\lceil \frac{q}{4} \rfloor /2 = -\rho/2$ *be a constant polynomial and* $U' := \mathsf{compress}(U, d_U)$. *Given* $k_i \in \{-3, \dots, 4\}$, $i \in [n]$, *let* $V' := (0, \dots, k_i, \dots, 0)$ *be the polynomial with* k_i *in the* i*-th coefficient and* 0 *elsewhere. Then, for* $\mathsf{pt} = 0$ *and the parameters of Kyber512, we have*

$$\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U', V')) = 1 \Leftrightarrow \left| \left\langle \mathsf{sk}_i \times \frac{\rho}{2} + k_i \times \frac{\rho}{2} \right\rangle_q \right| \le \rho.$$

Proof. First, we observe that for the given parameters, $\mathsf{decompress}(U', d_U) = U$.

Then, for $V' = (0, \dots, k_i, \dots, 0)$, $k_i \in \{-3, \dots, 4\}$ we have $V := \mathsf{decompress}(V', d_V) = (0, \dots, k_i \times \frac{\rho}{2}, \dots, 0)$ because

$$\mathsf{decompress}(k_i, d_V) = \left\lceil \frac{q}{8} \times k_i \right\rfloor \stackrel{*}{=} k_i \times \left\lceil \frac{q}{4} \right\rfloor /2 = k_i \times \frac{\rho}{2}\,, \tag{3.5}$$

where the $*$ equality holds with the parameters $q = 3\,329$ and $k_i \in \{-3, \dots, 4\}$.

Let $\delta = V - U \times \mathsf{sk}$. Then, for all $j \in [n], j \neq i$

$$\delta_j = 0 - \mathsf{sk}_j \times U = \mathsf{sk}_j \times \frac{\rho}{2} \in [-\rho, \rho]$$

since $\mathsf{sk}_j \in \{-2, \dots, 2\}$ and $U = -\rho/2$ is a constant polynomial. For $j = i$ we have $\delta_i = k_i \times \frac{\rho}{2} + \mathsf{sk}_i \times \frac{\rho}{2}$. Now, since $\delta_j \in [-\rho, \rho]$ for all $j \neq i$, an error in the decoding can only happen in the i-th component. Hence, querying $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U', V'))$ is equivalent to querying some oracle $\mathcal{O}^{\mathsf{learn}}(k_i) = \mathbf{1}_{\left|\langle \alpha_i + k_i \times \frac{\rho}{2} \rangle_q\right| \le \rho}$, where $\alpha_i = \mathsf{sk}_i \times \frac{\rho}{2} \in [-\rho, \rho]$. □

Note that the oracle $\mathcal{O}^{\mathsf{learn}}(k_i)$ in the proof above is similar to the one in the learning game defined in Figure 3.1. Now, we set $k_i := -(k_i' + 2)$ for some $k_i' \in \{-2, \dots, 1\}$, $\alpha_i := \mathsf{sk}_i \times \frac{\rho}{2}$ and (U', V') as in Lemma 3.5.1. Then, if the condition

$$|\alpha_i + k_i| = \left| \alpha_i - \rho - k_i' \times \frac{\rho}{2} \right| \le \lceil q/2 \rfloor \tag{3.6}$$

holds, then

$$\begin{gathered}\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U', V')) = 1 \Leftrightarrow |\langle \alpha_i - \rho - k_i' \times \frac{\rho}{2} \rangle_q| \le \rho \stackrel{(3.6)}{\Leftrightarrow} \\ |\alpha_i - \rho - k_i' \times \frac{\rho}{2}| \le \rho \Leftrightarrow -\rho \le \alpha_i - \rho - k_i' \times \frac{\rho}{2} \le \rho \Leftrightarrow \\ k_i' \times \frac{\rho}{2} \le \alpha_i \le 2\rho + k_i' \times \frac{\rho}{2} \Leftrightarrow k_i' \times \frac{\rho}{2} \le \alpha_i \Leftrightarrow k_i' \le \mathsf{sk}_i\end{gathered}$$

where the first equivalence follows from Lemma 3.5.1, the second to last equivalence follows from $\alpha_i \leq \rho$ and $k'_i \times \frac{\rho}{2} \leq \rho$ (hence the upper bound on α_i always holds), and the last because $\alpha_i = -\mathsf{sk}_i \times U = \mathsf{sk}_i \times \frac{\rho}{2}$. Hence, by setting $k_i = -(k'_i + 2)$ and (U', V') as in Lemma 1, one can perform a binary search and recover sk_i by querying $\mathcal{O}^{\mathsf{PCO}}(0, (U', V'))$ and varying k'_i. In order for condition (3.6) to hold, we start with $k'_i = 0$. Then, in the further iterations the condition holds for any $\alpha_i, k'_i \times \rho/2 \in [-\rho, 0]$ or $\alpha_i, k'_i \times \rho/2 \in [0, \rho]$.

The last difficulty is in the case where the final interval is $[1,2]$ (i.e. we know $\mathsf{sk}_i \in \{1,2\}$ after some iterations). In this case, we would need to pick $k'_i = 2$ and set $V'_i = -(k'_i + 2) = -4$. However, in this case the $*$ equality in Equation (3.5) of the proof of Lemma 3.5.1 does not hold. A solution is to set $V'_i = -1$ and $U' = \mathsf{compress}(\frac{\rho}{2}, d_U)$ before querying $\mathcal{O}^{\mathsf{PCO}}(0^n, (U', V'))$. Then, for $\mathsf{sk}_i \in \{1,2\}$ we have

$$\left| -\frac{\rho}{2} - \mathsf{sk}_i \times \frac{\rho}{2} \right| \leq \rho \Leftrightarrow \mathsf{sk}_i = 1.$$

Hence, if the query returns a success we can set $\mathsf{sk}_i \leftarrow 1$, otherwise $\mathsf{sk}_i \leftarrow 2$.

Finally, in the general case where $k > 2$, one can simply iterate the attack k times, moving U' around the vector in $\mathcal{R}^k$.

We give the full KR-PCA adversary in Figure 3.3.

3.5.3 Efficiency and implementation

Since we do 1 binary search with at most 3 queries and the total number of unknowns is $n \times k = 256 \times 2 = 512$ in Kyber512, one can recover sk in at most $3 \times 512 = 1536$ queries. In addition, the number of queries in the binary search is only 2 when $\mathsf{sk}_i \in \{-2, -1, 0\}$. The probability that happens given $\mathsf{sk}_i \xleftarrow{\$} \Psi_\eta$ is $\Pr[\mathsf{sk}_i \in \{-2, 1, 0\}] = \frac{11}{16}$. Hence, $\mathbb{E}[\#\text{queries}] = 512 \times \left(\frac{11}{16} \times 2 + \frac{5}{16} \times 3\right) = 1\,184$. We implemented a proof of concept of the attack in Sage for $k = 1$, which confirms these numbers.

Finally, we note that the only differences between Kyber512 and the more secure versions are the parameter k and the compression factors d_U, d_V. For the higher security levels, the compression is less aggressive thus does not impact our attack and the number of queries required increases linearly with k.

3.6 Misuse Attack against SABER

3.6.1 SABER-CPA

SABER [DAn+19a] works with vectors and matrices where components are polynomials in $\mathcal{R}_q$ for some integer q, as in Kyber. Components of the secret key are sampled from a centered binomial distribution Ψ_η, where the sampled elements are in the range $[-\eta/2, \eta/2]$. We apply our attack to the weaker version of SABER, namely LightSaber. In this version, the parameters

$\mathsf{KR\_PCA\_KYBER}(\mathsf{pk})$

$(A, B) \leftarrow \mathsf{pk}; \rho \leftarrow \left\lceil \frac{q}{4} \right\rfloor$
$\mathsf{pt} \leftarrow 0 \in \mathcal{R}_q$
$U' \leftarrow \mathsf{compress}(-\rho/2, d_U);\ U'_2 \leftarrow \mathsf{compress}(\rho/2, d_U)$
for $\ell \in [k]$
  We write U' (resp. U'_2) for the vector in $\mathcal{R}_q^k$ with polynomial U' (resp. U'_2)
  at position ℓ and $0 \in \mathcal{R}_q$ elsewhere.
  for $i \in [n]$:
    $V' \leftarrow 0 \in \mathcal{R}_q$
    $a \leftarrow -2;\ b \leftarrow 2$
    while $b > a$: // Binary search to find sk_i
      $c \leftarrow \left\lceil \frac{b+a}{2} \right\rceil;\ V'_i \leftarrow -2 - c$ // $\mathsf{decompress}(V, d_V) = -\rho - c \times \frac{\rho}{2}$
      if $c = 2$: // special case
        $V'_i \leftarrow -1;\ r \leftarrow \mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U'_2, V'))$
        if $r = 1$: $a \leftarrow 1$
        else : $a \leftarrow 2$
        continue
      $r \leftarrow \mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U', V'))$
      if $r = 1$: // $\mathsf{sk}_i \geq c$
        $a \leftarrow c$
      else :
        $b \leftarrow c - 1$
    $\mathsf{sk}_{\ell,i} \leftarrow a$
return sk

Figure 3.3: KR-PCA adversary against Kyber-CPA.

are $e_q = 13, e_p = 10, e_T = 3, q = 2^{e_q}, p = 2^{e_p}, T = 2^{e_T}, \eta = 10, n = 256$ and $k = 2$. We also define the polynomial $h \in \mathscr{R}_p$ with all coefficients equal to $2^{e_p-2} + 2^{e_p-e_T-1} + 2^{e_q-e_p-1} = 196$ and the polynomial $h' \in \mathscr{R}_p$ with all coefficients set to $2^{e_q-e_p-1} = 4$. The $\times$ operation here is the standard vector/matrix multiplication with component-wise polynomial multiplication (most elements are matrices or vectors of polynomials). The CPA-secure PKE underlying SABER (that we call SABER-CPA) works as follows.

- Gen: Sample $\mathsf{sk} \leftarrow\!\!\$ \, (\Psi_\eta^n)^k \in \mathscr{R}_q^k$, $A \leftarrow\!\!\$ \, \mathscr{R}_q^{k\times k}$ and set $d \in \mathscr{R}_q^k$ as the vector with each coefficient set to h'. Then, compute $B \leftarrow (A \times \mathsf{sk} + d) \gg (e_q - e_p) \in \mathscr{R}_p^k$ where $\gg$ is the component-wise bitshift operation. Then, set $\mathsf{pk} := (A, B)$.

- $\mathsf{Enc}(\mathsf{pk}, m \in \{0,1\}^n)$: Sample $t \leftarrow\!\!\$ \, (\Psi_\eta^n)^k$, set $e \in \mathscr{R}_q^k$ as the vector with each coefficient set to h', and compute $U \leftarrow (A \times t + e) \gg (e_q - e_p) \in \mathscr{R}_p^k$. Set $V \leftarrow (B^T \times t + h - 2^{e_p-1} m) \gg (e_p - e_T) \in \mathscr{R}_T$ and output (U, V).

- $\mathsf{Dec}(\mathsf{sk}, (U, V))$: Output $(U^T \times \mathsf{sk} - 2^{e_p-e_T} V + h) \gg (e_p - 1) \in \mathscr{R}_2$.

Let $W_i := (U \times \mathsf{sk})_i - 128 \times V_i + 196$. Then, a decrypted component can be written as

$$\mathsf{Dec}(\mathsf{sk}, (U, V))_i = \begin{cases} 0, & \text{if } W_i < 2^{e_p-1} = 2^9 \\ 1, & \text{if } W_i \geq 2^{e_p-1} = 2^9 \end{cases}.$$

3.6.2 KR-PCA

The idea of the Plaintext-Checking attack is similar to the one used in the previous section. However, here we have to deal with the addition of the polynomial $h = 196 + \ldots + 196 \cdot X^{n-1}$. Moreover, the domain of the components of the secret key is $\{-5, \ldots, 5\}$, which is much larger than in Kyber.

First, we consider $k = 1$, $\mathsf{pt} = 0^n$ and $V = 0 \in \mathscr{R}_T$. Then, for any constant polynomial $U \in [-\lfloor \frac{196}{5} \rfloor, \lfloor \frac{196}{5} \rfloor]$ and $\mathsf{sk}_i \in \{-5, \ldots, 5\}$, we have

$$W_i = (U \times \mathsf{sk})_i + 196 < 2^9 \;\; \forall i \in [n]_- \iff \mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V)) = 1.$$

This means that if we set $V = v_i \cdot X^i$ (i.e. only the i-th term is non-null), we have the following equivalence

$$\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V)) = 0 \iff (U \times \mathsf{sk})_i - 2^{e_p-e_T} v_i + 196 \geq 2^9.$$

In other words, an error can occur only in the i-th component. Let $v_i = 2$, then $-2^{e_p-e_T} v_i + 196 \pmod p = 964$. Now for $c \in \{2, 3, 4, 5\}$, we have

$$\mathcal{O}^{\mathsf{PCO}}\left(\mathsf{pt}, \left(\frac{60}{c}, 2X^i\right)\right) = 1 \iff 964 + \mathsf{sk}_i \times \frac{60}{c} \pmod p < 512 \iff \mathsf{sk}_i \geq c.$$

similarly, for $c \in \{-5, \ldots, -2\}$

$$\mathcal{O}^{\mathsf{PCO}}\left(\mathsf{pt}, \left(\frac{60}{c}, 2X^i\right)\right) = 1 \iff 964 + \mathsf{sk}_i \times \frac{60}{c} \mod p < 512 \iff \mathsf{sk}_i \leq c.$$

Hence, by querying $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, v_i \cdot X^i))$ with $U = \frac{60}{c}$ one can perform a binary search to find all sk_i s.t. $\mathsf{sk}_i \in \{-5, \ldots, -2, 2, \ldots, 5\}$. Let $\mathcal{I}$ be the set of indices of such components.

In a second step, we want to find all $\mathsf{sk}_i \in \{-1, 0, 1\}$. As in the previous step, we can set $U = \pm\frac{60}{1}, V = 2X^i$. The problem is that in this case $U \notin \left[-\left\lfloor\frac{196}{5}\right\rfloor, \left\lfloor\frac{196}{5}\right\rfloor\right]$ and therefore it is not guaranteed that an error will occur only in the i-th component. However, since we know every $\mathsf{sk}_j, j \in \mathcal{I}$, we can find two vectors $\tilde{V}^{\pm} = \sum_{j \in \mathcal{I}} v_j^{\pm} \cdot X^j$ s.t. $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (\pm 60, \tilde{V}^{\pm})) = 1$. Hence, by setting $U = \pm 60$ and $V = \tilde{V}^{\pm} + 2X^i$, one can find the remaining $\mathsf{sk}_i \in \{-1, 0, 1\}$. Finally, for $k > 1$, we can simply shift the polynomial U in a vector of size k and apply the same algorithm k times. The full algorithm is given in Figure 3.4.

3.6.3 Efficiency and implementation

The binary search for one secret component takes at most $\lceil \log(\eta) \rceil$ queries and there are $k \times n$ components. For LightSaber, it means that one can recover sk in at most $4 \times 512 = 2^{11}$ queries. The higher security levels for SABER require a less aggressive compression (as in Kyber) and a smaller domain for the components of the secret key. It means that a similar attack can be applied. For Saber and FireSaber, $3 \times 768 \approx 2^{11}$ and $3 \times 1024 = 3072$ queries would be needed, respectively. Interestingly, the maximal number of queries required for Saber would be roughly the same as for LightSaber. As a proof of concept, we implemented the attack against LightSaber using the reference implementation in C.

Finally, we leave as a future improvement the optimisation of the way the value c is picked in the binary search. Following the results presented in [Băe+19], it should be feasible to design a binary search algorithm with an expected number of queries close to $H(\mathsf{sk}_i)$, where $H(\cdot)$ is the Shannon entropy. For instance, in LightSaber we have $H(\mathsf{sk}_i) \approx 2.7$.

3.7 Misuse Attack against HQC

We briefly explain here how the attack against Lepton presented in our previous work [Băe+19] can be applied to HQC. The HQC [Mel+19b] scheme works mainly in $\mathcal{R}_2$ and with the Hamming weight $\|x\| = |\{i : x_i \neq 0\}|$. In addition, let w_{sk}, w_t, w_f be some integers given as parameters and $S_w = \{v : v \in \mathcal{R}_2, \|v\| = w\}$ be the set of polynomials in $\mathcal{R}_2$ with Hamming weight w. Then, the CPA-secure PKE underlying HQC is defined as follows.

- Gen: Sample $(\mathsf{sk}, d) \leftarrow\$ S^2_{w_{sk}}$ and $A \leftarrow\$ \mathcal{R}_2$. Set $\mathsf{pk} = (A, B = A \times \mathsf{sk} + d)$.

- $\mathsf{Enc}(\mathsf{pk}, m \in \{0,1\}^k)$: Sample $(t, e, f) \leftarrow\$ S^2_{w_t} \times S_{w_f}$. Then, the ciphertext is $(U, V) = (t \times$

KR_PCA_SABER(pk)

$(A, B) \leftarrow \mathsf{pk};\ \rho \leftarrow \left\lceil \frac{q}{4} \right\rfloor$
$\mathsf{pt} \leftarrow 0^{256}$
for $\ell \in [k]$:
  We write U for the vector in $\mathscr{R}_p^k$ with polynomial U
  at position ℓ and $0 \in \mathscr{R}_p$ elsewhere.
  $\mathscr{I} \leftarrow \emptyset$
  for $i \in [n]_-$:
    $V \leftarrow 2 \cdot X^i \in \mathscr{R}_T$
    $a \leftarrow -5;\ b \leftarrow 5$
    if $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (30, V)) = 1$: $a \leftarrow 2$
    else :
      if $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (-30, V)) = 1$:
        $b \leftarrow -2$
      else :
        $\mathscr{I} \cup \{i\}$; **continue**
    while $b > a$: // Binary search to find $\mathsf{sk}_{\ell,i}$
      $c \leftarrow \mathsf{sgn}(a+b) \left\lceil \frac{|b+a|}{2} \right\rceil;\ U \leftarrow \frac{60}{c}$ // $c|60$ for all $c \in \{-5, \ldots, 5\}$
      if $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V)) = 1$:
        if $c > 0$: $a \leftarrow c$
        else : $b \leftarrow c$
      else :
        if $c > 0$: $b \leftarrow c - 1$
        else : $a \leftarrow c + 1$
    $\mathsf{sk}_{\ell,i} \leftarrow a$
  find two vectors $\tilde{V}^{\pm}$ s.t. $\mathscr{O}^{\mathsf{PCO}}(0^{256}, (\pm 60, \tilde{V}^{\pm})) = 1$
  for $i \in \mathscr{I}$
    $V \leftarrow \tilde{V}^{+} + 2X^i$
    if $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (60, V)) = 1$:
      $\mathsf{sk}_{\ell,i} \leftarrow 1$; **continue**
    $V \leftarrow \tilde{V}^{-} + 2X^i$
    if $\mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (-60, V)) = 1$:
      $\mathsf{sk}_{\ell,i} \leftarrow -1$; **continue**
    $\mathsf{sk}_{\ell,i} \leftarrow 0$
return sk

Figure 3.4: KR-PCA adversary against SABER-CPA.

$A+e, t\times B+f+mG)$ where G is a generator matrix in $\mathbb{Z}_2^{k,n}$ for some linear $[k,n]$-code $\mathscr{C}$.

- $\mathsf{Dec}(\mathsf{sk}, U, V)$: output $\mathsf{decode}(V - U\times \mathsf{sk})$ where decode is the decoding function of the code $\mathscr{C}$ generated by G.

Now, we have $V - U\times \mathsf{sk} = mG + \delta$ with $\delta = t\times d + f - e\times \mathsf{sk}$. Thus, the decoding (hence the decryption) is correct iff $\|\delta\| = \|t\times d + f - e\times \mathsf{sk}\| < \rho$ for some ρ. The goal is to recover δ and use the known relation $B = A\times \mathsf{sk} + d$. Then, $(t\times A + e)\times \mathsf{sk} = t\times B + f - \delta$ gives n linear equations in n unknowns in $\mathbb{Z}_q$ and we can solve for sk.

The code used in HQC is a composition of a d-repetitions code and BCH code. Namely,

$$\mathsf{decode} = \mathsf{decode}_{\mathrm{BCH}}(\mathsf{decode}_{\mathrm{REP}}(c)).$$

This is the same decoding function as the one in Lepton [YZ17] and therefore one can use the same learning algorithm [Băe+19] to deduce δ and thus obtain n linear equations in sk. For parameters of HQC-128, it requires

$$n + \frac{n}{d}\log_2 d + \frac{n}{d} + \log_2 \frac{n}{d} \approx 2^{15}$$

oracle queries to recover sk, with $n = 24677$ and $d = 31$. In the revised version of HQC for the second round of the NIST standardisation process, the polynomial V is truncated to fit into n_c coefficients at the end of the encryption, where n_c is the length of the code. Similarly, it is expanded by ℓ coefficients set to 0 before decryption, with $\ell = n - n_c$. As n is picked as the least prime larger than n_c, the value ℓ is typically very small (e.g. $\ell = 1$ for HQC-128). Still, this implies that we can only get n_c equations for n unknowns at the end of the attack. However, one can run twice the attack to obtain enough equations or use a bruteforce technique (if ℓ is small) to recover the full key sk.

3.8 RQC: Misuse Attack and Impossibility Result

3.8.1 Rank-based cryptography

The RQC cryptosystem [Mel+19a] is similar to HQC [Mel+19b] but uses the rank metric instead of the Hamming distance. Let q be a prime and consider the finite field $\mathbb{F}_{q^m}$ for some parameter $m \in \mathbb{Z}$. Let $g \in \mathbb{F}_q[X]$ be an irreducible polynomial of degree m. Then, we have $\mathbb{F}_{q^m} \simeq \mathbb{F}_q[X]/\langle g\rangle \simeq \mathbb{F}_q^m$. Now, let $\mathbb{F}_{q^m}^n$ be the vector space over the finite field $\mathbb{F}_{q^m}$ for some parameter $n \in \mathbb{Z}$. Each element of this vector space can be seen as a polynomial in $\mathbb{F}_{q^m}[X]/\langle f\rangle$ where $f \in \mathbb{F}_q[X]$ is an irreducible polynomial of degree n, using the trivial isomorphism

$$\phi\colon\ v \in \mathbb{F}_{q^m}^n \mapsto \sum_{i=0}^{n-1} v_i X^i \pmod{f}.$$

For elements in $\mathbb{F}_{q^m}^n$, the multiplication $\times$ is defined as the polynomial multiplication in $\mathbb{F}_{q^m}[X]/\langle f\rangle$. More formally, for any $a, b \in \mathbb{F}_{q^m}^n$

$$a \times b := \phi^{-1}(\phi(a) \cdot \phi(b)),$$

where $\cdot$ denotes the multiplication in $\mathbb{F}_{q^m}[X]/\langle f\rangle$. Similarly, the multiplication in $\mathbb{F}_{q^m}$ is defined as the polynomial multiplication in $\mathbb{F}_q[X]/\langle g\rangle$. In RQC-I, as $m = 97$ and $n = 67$, the two polynomials are $f = X^{67} + X^5 + X^2 + X + 1$ and $g = X^{97} + X^6 + 1$.

Rank metric and support. Let $v = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_{q^m}^n$ and $\{\beta_i\}_{i\in[m]}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then, each component $v_i \in \mathbb{F}_{q^m}$ can be written as a vector in $\mathbb{F}_q^m$ using the basis representation. Hence, v can be represented as a $m \times n$ matrix with elements in $\mathbb{F}_q$. We denote this matrix by $\mathcal{M}(v)$, which is of the form

$$\mathcal{M}(v) := \begin{pmatrix} v_{0,0} & \cdots & v_{n-1,0} \\ \vdots & \ddots & \vdots \\ v_{0,m-1} & \cdots & v_{n-1,m-1} \end{pmatrix}$$

with $v_{i,j} \in \mathbb{F}_q$ s.t. $v_i = \sum_{j\in[m]} v_{i,j}\beta_j$. While not important, the choice of basis of $\mathbb{F}_{q^m}$ impacts the matrix representation. In what follows, we consider the canonical basis. That is, we consider $v \in \mathbb{F}_{q^m}$ as a polynomial in $\mathbb{F}_q[X]/\langle g\rangle$ and take the trivial representation of this polynomial as a vector in $\mathbb{F}_q^m$.

Definition 3.8.1 (Rank in $\mathbb{F}_{q^m}^n$)**.** *Let $v \in \mathbb{F}_{q^m}^n$ be a vector and $\mathcal{M}(v) \in \mathbb{F}_q^{m\times n}$ be its matrix representation as defined above. Then, we define the rank of v as*

$$\|v\| := \mathsf{rank}\,(\mathcal{M}(v))$$

that is, the rank of the matrix representation of v. Then, the distance between $v, w \in \mathbb{F}_{q^m}^n$ is defined as

$$\|v - w\| = \mathsf{rank}\,(\mathcal{M}(v) - \mathcal{M}(w))\,.$$

For an arbitrary matrix A, let $\mathsf{span}(A)$ be the vector space spanned by the columns of A. Then, the support of a vector is defined as follows.

Definition 3.8.2 (Support in $\mathbb{F}_{q^m}^n$)**.** *Let $v \in \mathbb{F}_{q^m}^n$. Then, the support is*

$$\mathsf{supp}(v) := \mathsf{span}(\mathcal{M}(v))$$

i.e. the vector space spanned by the columns of $\mathcal{M}(v)$. Similarly, we write $\mathsf{supp}(v^T)$ for the vector space spanned by the rows of $\mathcal{M}(v)$. Finally, by the definition of the rank of a matrix, we have $\mathsf{dim}(\mathsf{supp}(v)) = \mathsf{dim}(\mathsf{supp}(v^T)) = \|v\|$.

A useful tool when dealing with vector subspaces is the q-binary coefficient (also called

Gaussian coefficient), which counts the number of subspaces of dimension r in a vector space of dimension n over a field of cardinality q. It is defined as

$$\begin{bmatrix} n \\ r \end{bmatrix}_q := \prod_{i=0}^{r-1} \frac{q^n - q^i}{q^r - q^i}.$$

3.8.2 RQC scheme

Let $w, w', k \in \mathbb{Z}$ be parameters, $S_w^n := \{v \in \mathbb{F}_{q^m}^n : \|v\| := w\}$, and $S_{1,w}^n := \{v \in \mathbb{F}_{q^m}^n : \|v\| = w,\ 1 \in \mathsf{supp}(v)\}$. RQC uses a random Gabidulin code [Gab85] defined by a generating matrix $G \in \mathbb{F}_{q^m}^{k\times n}$ and with decoding capacity $\rho = \lfloor \frac{n-k}{2} \rfloor$. We denote the corresponding decoding algorithm by $\mathsf{decode}_{\mathrm{gab}}$. Then, RQC-CPA, the PKE underlying RQC, works as follows.

- Gen: Sample $(\mathsf{sk}, d) \leftarrow\$ S_{1,w}^{2n}$ and $A \leftarrow\$ \mathbb{F}_{q^m}^n$. Set $B \leftarrow A \times \mathsf{sk} + d$. Pick a random generating matrix $G \in \mathbb{F}_{q^m}^{k\times n}$ for some Gabidulin code. Output $(\mathsf{pk} := (A, B, G), \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m \in \{0,1\}^k)$: Sample $(t, e, f) \leftarrow\$ S_{w'}^{3n}$. Compute $U \leftarrow A \times t + e$ and $V \leftarrow B \times t + mG + f$. Output (U, V).
- $\mathsf{Dec}(\mathsf{sk}, U, V)$: Output $\mathsf{decode}_{\mathrm{gab}}(V - U \times \mathsf{sk})$.

Correctness. Let $\delta := t \times d + f - e \times \mathsf{sk}$. Then, for any honestly generated ciphertext (U, V) (i.e. $(U, V) = \mathsf{Enc}(\mathsf{pk}, m)$ for some pk, m) we have $V - U \times \mathsf{sk} = mG + \delta$. Since the decoding capacity of the code is ρ, we assume $\mathsf{Dec}(\mathsf{sk}, U, V) = m \iff \|\delta\| \le \rho$ thus,

$$\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, U, V) = 1 \iff \|\delta\| \le \rho\ .$$

3.8.3 KR-PCA against RQC-CPA

We give a Key-Recovery under Plaintext-Checking attack that works with $O(wq^{\min\{m,n\}-\rho+1})$ queries in expectation. As $q = 2, w = 5, m = 97, n = 67$ and $\rho = 31$ for RQC-I, we obtain a complexity of $O(2^{39})$.

First, we state a useful theorem and two lemmas.

Theorem 3.8.1 (Lemma 1, [Car75] or Theorem 11, [MS74])**.** *Let $X, Y \in \mathbb{F}^{m\times n}$ be two $m \times n$ matrices over an arbitrary field $\mathbb{F}$. Then,*

$$\mathsf{rank}(Y + X) = \mathsf{rank}(Y) + \mathsf{rank}(X)$$

iff

$$\mathsf{span}(Y) \cap \mathsf{span}(X) = \{0\}\ \textit{and}\ \mathsf{span}(Y^T) \cap \mathsf{span}(X^T) = \{0\}.$$

In other words, for two matrices over a field, the rank of their sum is equal to the sum of their rank iff their column space (resp. their row space) trivially intersect.

Lemma 3.8.1. *We consider the RQC-CPA. Let* $B = A \times \mathsf{sk} + d$, $\mathsf{sk}, d \in \mathbb{F}_{q^m}^n$, $\mathsf{supp}(\mathsf{sk}) = \mathsf{supp}(d)$ *and* $\|\mathsf{sk}\| = \|d\| = w$. *Then, finding a subspace* $F \subset \mathbb{F}_{q^m}$ *s.t.* $z = \mathsf{dim}(F) \leq \frac{m}{2}$ *and* $\mathsf{supp}(\mathsf{sk}) = \mathsf{supp}(d) \subseteq F$ *is sufficient to recover* sk *and* d. *Similarly, let* $z = \mathsf{dim}(F)$, $z' = \mathsf{dim}(F')$, *then finding* $F, F' \subset \mathbb{F}_q^n$ *s.t.* $z + z' \leq n$, $\mathsf{supp}(\mathsf{sk}^T) \subseteq F$ *and* $\mathsf{supp}(d^T) \subseteq F'$ *is sufficient to recover* sk *and* d.

Proof sketch. We give here an informal argument. A complete discussion can be found in a paper by Aragon et al. [Ara+18]. If one can find a subspace F s.t. the support of sk (and d) is contained in it, one can compute a basis $\{\beta_i\}_{i\in[z]_-}$ for the subspace F. Then, one can write $\mathsf{sk}_i = \sum_{j=0}^{z-1} a_{i,j}\beta_j$ and $d_i = \sum_{j=0}^{z-1} b_{i,j}\beta_j$, where the $2nz$ coefficients $a_{i,j}, b_{i,j}$ are unknown. Then, $B = (A, 1)\cdot(\mathsf{sk}, d)^T \in \mathbb{F}_{q^m}^n$ can be seen as a system of nm linear equations in $\mathbb{F}_q$ with $2nz$ unknown coefficients. Hence, as long as $nm \geq 2nz \iff z \leq \frac{m}{2}$, one can solve the system of equations to recover sk, d.

Similarly, if one can find a basis for a subspace containing the row space of $\mathcal{M}(\mathsf{sk})$ and another for the row space of $\mathcal{M}(d)$, one can write the system of mn equations in $\mathbb{F}_q$ given by B as a system with $m(z+z')$ unknown coefficients. In this case, the system is solvable for $z+z' \leq n$. □

Lemma 3.8.2. *Let* $p_{k,w}^n$ *the probability that a random subspace of dimension* k *non-trivially intersects a given subspace of dimension* w *in* $\mathbb{F}_q^n$, *with* $k + w \leq n$. *Then,*

$$p_{k,w}^n \leq (q^k - 1)\frac{(q^w - 1)}{(q^n - 1)} \leq q^{w+k-n}.$$

Proof. The proof of the first inequality is a simple union bound. The probability that a random non-zero random vector is in the subspace of dimension w is $\frac{(q^w-1)}{(q^n-1)}$ (i.e. the number of non-zero vectors in the subspace over the number of non-zero vectors in $\mathbb{F}_q^n$). Then, the probability that at least one of the $q^k - 1$ non-zero vectors of the random subspace is in the given subspace is upper bounded by $(q^k - 1)\frac{(q^w-1)}{(q^n-1)}$. The second bound is straightforward analysis: one can compute the following equivalence

$$(q^k - 1)\frac{(q^w - 1)}{(q^n - 1)} \leq q^{w+k-n} \iff q^w + q^k - 1 \geq \frac{q^{w+k}}{q^n}$$

which holds with $k + w \leq n$. □

The attack. Let $V := x$ for some $x \in \mathbb{F}_{q^m}^n$ and $U := -1 \in \mathbb{F}_{q^m}^n$. Then,

$$\mathcal{O}^{\mathsf{PCO}}(0, (U, V)) = 1 \iff \|\mathsf{sk} + x\| \leq \rho.$$

Let's pick $x \in \mathbb{F}_{q^m}^n$ at random s.t. $\|x\| = \rho - w$. Then, by Theorem 3.8.1, we have $\|\mathsf{sk} + x\| = \rho$ iff the column spaces (resp. the row spaces) of sk and x do not intersect (i.e. trivially intersect). By

Lemma 3.8.2, the probability an intersection occurs in the column space or in the row space is upper bounded by $p^m_{\rho-w,w} + p^n_{\rho-w,w} \leq q^{\rho-m} + q^{\rho-n}$. Since $m \geq n$ and $\rho < \frac{n}{2}$ in RQC, this can be further bounded by $O(q^{-n/2})$, which is negligible in n. Hence, we assume this does not occur and $\|\mathsf{sk} + x\| = \rho$. In this case, $\mathsf{supp}(\mathsf{sk}) \subset \mathsf{supp}(\mathsf{sk}+x)$ and $\mathsf{supp}\left(\mathsf{sk}^T\right) \subset \mathsf{supp}\left((\mathsf{sk}+x)^T\right)$. Indeed, each vector in $\mathsf{supp}(\mathsf{sk}+x)$ can be written as a linear combination of vectors in the union of the basis of sk and x. Clearly, the union of the two basis is then a basis for $\mathsf{supp}(\mathsf{sk}+x)$ since $\|\mathsf{sk}+x\| = w + (\rho - w)$. The same argument works for the row space. Hence, the attack consists of finding a basis of $\mathsf{supp}(\mathsf{sk}+x)$ or $\mathsf{supp}\left((\mathsf{sk}+x)^T\right)$ and then finding sk by Lemma 3.8.1. We focus on finding the first one.

Let $u := \mathsf{sk} + x$ with $\|u\| = \rho$ and $y := (\alpha, 0, \ldots, 0) \in \mathbb{F}^n_{q^m}$ for some $\alpha \in \mathbb{F}_{q^m}$. Then,

$$\mathscr{M}(y) = \begin{pmatrix} \alpha_0 & 0 & \cdots & 0 \\ \alpha_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m-1} & 0 & \cdots & 0 \end{pmatrix}.$$

We observe that for $\|y\| = 1$

$$\mathsf{supp}(u) \cap \mathsf{supp}(y) \neq \{0\} \iff \alpha \in \mathsf{supp}(u)\ .$$

Therefore, by Theorem 3.8.1, $\|u + y\| = \rho$ iff $y \in \mathsf{supp}(u)$ or $(1, 0, \ldots, 0) \in \mathsf{supp}(u^T) \subset \mathbb{F}^n_q$. Now, if we consider $\mathsf{supp}(u^T)$ as a random subspace of dimension ρ in $\mathbb{F}^n_q$, the probability that $(1, 0, \ldots 0) \in \mathsf{supp}(u^T)$ can be upper bounded by $q^{\rho+1-n} \leq q^{-n/2+1}$ by Lemma 3.8.2, which is negligible. Hence, one can iterate over all $\alpha \in \mathbb{F}_{q^m}$ and mark α whenever $\|u + y\| \leq \rho$. At the end, all marked α's form the vector space $\mathsf{supp}(u)$. Then, one can find a basis for this subspace and recover the secret key sk by Lemma 3.8.1, since $\rho < \frac{n}{2} < \frac{m}{2}$. In this case, the total number of queries needed is $O(q^m)$. Note that the strategy of querying y with only one non-null component is similar to an independent and concurrent timing attack against RQC [Bet+19].

Improved attack. Now, instead of marking all α's in the vector space $\mathsf{supp}(u)$, one can mark α s.t. α is not in the subspace spanned by the already marked α's. More formally, in the i-th step, if we know that $\alpha^{(1)}, \ldots, \alpha^{(i-1)} \in \mathsf{supp}(u)$, we do not mark $\alpha^{(i)}$ s.t. $\alpha^{(i)} \in \mathsf{span}\left(\alpha^{(1)}, \ldots, \alpha^{(i-1)}\right)$. In that way, the expected number of queries needed is lowered since we recover only a basis of $\mathsf{supp}(u)$ and not the whole subspace. Note that we could check for linear independence of $\alpha^{(i)}$ before querying it, sparing a few queries but increasing the amount of offline work.

The expected number of queries needed can be approximated as follows. Let X_i be the number of queries needed to find a new basis vector in $\mathsf{supp}(u)$, knowing we already found $\alpha^{(1)}, \ldots, \alpha^{(i)} \in \mathsf{supp}(u)$. We refer to the vectors which are not a new basis vector as *bad*. In each step, we assume we did not query any *bad* vectors. Thus, the number of potential basis vectors is $q^\rho - q^i$ and the total number of vectors left to query is $q^m - i$. The expected number of

RQC_KR_PCA(pk)

$(A, B) \leftarrow \mathsf{pk}$
$\mathsf{pt} \leftarrow 0^k$
$U \leftarrow -1$
$x \leftarrow\$\, S^n_{\rho-w}$
compute basis $\{\beta_i\}_{i\in[\rho-w]_-}$ of $\mathsf{span}(x)$
$\mathscr{W} \leftarrow \{\beta_i\}_{i\in[\rho-w]_-}$
for $\alpha \in \mathbb{F}_{q^m}$
  $y \leftarrow (\alpha, 0, \ldots, 0) \in \mathbb{F}^n_{q^m}$
  $V \leftarrow x + y$
  $r \leftarrow \mathscr{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V))$
  if $r = 1$:
    if α not in subspace spanned by the elements of $\mathscr{W}$:
      $\mathscr{W} = \mathscr{W} \cup \{\alpha\}$
    if $|\mathscr{W}| = \rho$: **break**
Set $\mathsf{sk}_i = \sum_{i=0}^{\rho-1} a_{i,j}\gamma_i$ and $d_i = \sum_{i=0}^{\rho-1} b_{i,j}\gamma_i$ with $\gamma_i \in \mathscr{W}$
Solve $B = (A, 1) \cdot (\mathsf{sk}, d)^T$
return sk

Figure 3.5: KR-PCA adversary against RQC-CPA.

draws before getting a *good* vector (i.e. a new basis vector) is therefore $\mathbb{E}[X_i] = \frac{q^m-i+1}{q^\rho-q^{i-1}+1}$. At the beginning, we already know that the basis of x is a set of $\rho - w$ linearly independent elements of $\mathsf{supp}(u)$. Therefore, we set $\alpha^{(1)}, \ldots, \alpha^{(\rho-w)}$ as the basis of x and only w basis vectors need to be found. Hence, the expected total number of queries before getting the ρ basis vectors is approximately

$$\sum_{i=\rho-w}^{\rho-1} \frac{q^m - i + 1}{q^\rho - q^i + 1} \le \sum_{i=\rho-w}^{\rho-1} \frac{q^m}{q^\rho - q^i} \le w q^{m-\rho+1}.$$

Note that this is actually an upper bound on the real expectation, since we made an assumption that worsens the actual performance (i.e. we forget we already queried some *bad* vectors). The full attack is given in Figure 3.5. Hence, the expected total number of queries is $O(wq^{m-\rho+1})$. The success probability of the algorithm is at least $1 - O(q^{-n/2+1})$. Finally, observe that in RQC, sk, d are picked uniformly at random from $\mathbb{F}^n_{q^m}$ s.t. $\|\mathsf{sk}\| = \|d\| = w$, $\mathsf{supp}(\mathsf{sk}) = \mathsf{supp}(d)$ and $1 \in \mathsf{supp}(\mathsf{sk}) = \mathsf{supp}(d)$. The fact that we know one vector of the subspace spanned by sk does not impact the attack but merely decreases the randomness of sk.

Row support recovery. The attack that recovers a vector subspace $\mathsf{supp}(u^T)$ which contains the row space of sk is nearly identical to the one above. The only difference is that we iterate over all $\alpha \in \mathbb{F}_q^n$ by setting $y \in \mathbb{F}_{q^m}^n$ s.t. $y = (\alpha_0 X, \alpha_1 X, \dots, \alpha_{n-1} X)$. We do not set $y = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$, otherwise $1 \in \mathsf{supp}(y)$ and thus $\|u + y\| \leq \rho$ for all α. Now, the row space of the secret key $\mathsf{supp}(\mathsf{sk}^T)$ is not necessarily equal to the row space of d. However, one can recover a subspace containing the latter in the exact same way. Indeed, the only difference is that we set $U := A, V := B + x$ for any $x \in \mathbb{F}_{q^m}^n$ and then $\mathcal{O}^{\mathsf{PCO}}(\mathsf{pt}, (U, V)) = 1 \iff \|V - U \times \mathsf{sk}\| = \|d + x\| \leq \rho$. Note that Lemma 3.8.1 still applies since $\rho < \frac{n}{2}$. The expected number of queries is upper bounded by $wq^{n-\rho+1}$.

Total cost. Hence, the total number of queries needed to recover the key is upper bounded by $wq^{\min\{m,n\}-\rho+1}$. For the CPA version of RQC-I (which targets 128-bit security), this amounts to roughly 2^{39} queries.

3.8.4 Hardness of Learning in the rank metric

As the KR-PCA attack against RQC given above has an exponential complexity, one could wonder whether a polynomial attack would be possible. While not proving the hardness of the KR-PCA game in the RQC setting, we show below that the learning game is hard for small errors.

First, we state useful theorems and lemmas.

Lemma 3.8.3 (Corollary 8.1, [MS74])**.** *Let $X, Y \in \mathbb{F}^{m\times n}$ be two $m \times n$ matrices over a field $\mathbb{F}$, $c := \mathsf{dim}(\mathsf{span}(X) \cap \mathsf{span}(Y))$ and $d := \mathsf{dim}(\mathsf{span}(X^T) \cap \mathsf{span}(Y^T))$. Then,*

$$\mathsf{rank}(X) + \mathsf{rank}(Y) - c - d \leq \mathsf{rank}(X + Y) \leq \mathsf{rank}(X) + \mathsf{rank}(Y) - \max(c, d).$$

Lemma 3.8.3 directly implies the following corollary.

Corollary 3.8.1. *Let $x, y \in \mathbb{F}_{q^m}^n$ s.t. $\|x\| = w$, $\|y\| = z$ and $z \geq w$. Let $c := \mathsf{dim}(\mathsf{supp}(x) \cap \mathsf{supp}(y))$, $d := \mathsf{dim}(\mathsf{supp}(x^T) \cap \mathsf{supp}(y^T))$ and ρ be some positive integer. Then, if $z > \rho + w$*

$$\|x + y\| > \rho.$$

Lemma 3.8.4 (Intersection of subspaces)**.** *Let $w, d, n \in \mathbb{N}$ and W be some random secret subspace of $\mathbb{F}_q^n$ of dimension w. We consider the following game. A participant who does not know W tries to find a subspace X of $\mathbb{F}_q^n$ of dimension d s.t. the intersection $W \cap X$ is non-trivial. The game stops when such a subspace is found. Then, the probability $p_{w,d}^{n,t}$ of success in t trials is*

$$p_{w,d}^{n,t} \leq \frac{t}{q^{n-d-w}}.$$

Proof. By a union bound, the probability of finding an intersection with a subspace of dimension d in a given trial is upper bounded by the probability of finding an intersection with a subspace of dimension 1 (i.e. a vector) in $q^d - 1$ trials. Therefore, we have

$$p^{n,t}_{w,d} \le p^{n,(q^d-1)t}_{w,1} \le p^{n,t'}_{w,1} \tag{3.7}$$

for $t' := q^d t - 1$ (and $t > 0$). Then, in any of the t' trials, the probability that a given vector is in the secret subspace of dimension w is upper bounded by $\frac{q^w-1}{q^n-t'-1}$ (i.e. there are $q^w - 1$ non-zero vectors in W and at most t' non-zero vectors have already been tried). Hence,

$$p^{n,t'}_{w,1} \le t' \frac{q^w - 1}{q^n - t' - 1} \le \frac{t'}{q^{n-w}} = \frac{t}{q^{n-d-w}} \ , \tag{3.8}$$

where the first inequality follows from a union bound and the second holds iff $t' + 1 \le q^{n-w} \iff t \le q^{n-w-d}$. As the theorem clearly holds for $t > q^{n-w-d}$ since $p^{n,t}_{w,d} \le 1$, combining Eq. (3.7) and (3.8) concludes the proof. □

Now we can prove the hardness of the learning game in the rank metric setting.

Theorem 3.8.2 (Hardness of learning in the rank metric)**.** *Let $q := 2, w, \rho, n, m$ and $d := \rho + w$ be some positive integers s.t. $w + d = \rho + 2w < \min\{m, n\}$. In addition, we consider $S^n_w := \{v \in \mathbb{F}^n_{q^m} : \|v\| = w\}$, Ψ the uniform distribution over S^n_w and $\|\cdot\|$ the rank distance. Then, for any learning adversary $\mathscr{A}_t$ restricted to t number of queries with $t < q^{\min\{m,n\}-w-d}$, we have*

$$\mathsf{Adv}^{\text{learn}}_{\Psi,\rho,\|\cdot\|}(\mathscr{A}_t) = \Pr[\mathit{LEARN}_{\Psi,\rho,\|\cdot\|}(\mathscr{A}_t) \Rightarrow 1] \le \frac{t}{q^{n-w-d}} + \frac{t}{q^{m-w-d}} + \mathsf{negl}$$

where $\mathsf{negl} = \left(\begin{bmatrix} n \\ w \end{bmatrix}_q \prod_{i=0}^{w-1}(q^m - q^i)\right)^{-1}$.

Proof. The idea of the proof is to show that the oracle of the learning game is useful only if the adversary can find a non-trivial intersection with the subspace spanned by the columns or the rows of $\mathscr{M}(\delta)$. We proceed by the game hopping technique.

First, consider the learning game of Figure 3.1 but we replace the oracle with the oracle $\mathscr{O}^{\mathsf{G}_0}$ of Figure 3.6. We call this new game G_0. One can see that this game is the same as the learning game. Indeed, by Corollary 3.8.1, the condition in line 2 returns the same result as $1_{\|\delta+x\|\le\rho}$. Then, if the column (and row) spaces of $\mathscr{M}(x)$ and $\mathscr{M}(y)$ trivially intersect, we have $\|\delta + x\| = \|\delta\| + \|x\|$ by Theorem 3.8.1. Hence, line 8 returns the correct result because the condition on lines 5 were not satisfied. Finally, if this condition did hold, the result is obviously the same as in the original oracle. Now, consider the game G_1 which is the same as G_0 except it returns $1_{\|\delta\|+\|x\|\le\rho}$ when both $\|x\| \le \rho + w$ and condition on line 5 holds. Let's call this event int. Clearly, G_0 and G_1 are the same except when int happens.

We want to compute $\Pr[\mathsf{int}]$, that is, the probability that the adversary finds some x s.t. $\|x\| \le \rho + w$ and a non-trivial intersection with the column or row space of δ in less than t queries. Now, in the learning game, the oracle replies $1_{\|\delta\|+\|x\|\le\rho}$ (which contains no extra information

Oracle $\mathcal{O}^{\mathsf{G}_0}(x)$

```
1: w ← ‖δ‖
2: if ‖x‖ > ρ + w :
3:    return 1_{‖δ‖+‖x‖≤ρ} = 0
4: end if
5: if supp(x) ∩ supp(δ) ≠ {0} or supp(x^T) ∩ supp(δ^T) ≠ {0} :
6:    return 1_{‖δ+x‖≤ρ}
7: end if
8: return 1_{‖δ‖+‖x‖≤ρ}
```

Oracle $\mathcal{O}^{\mathsf{G}_1}(x)$

```
1: return 1_{‖δ‖+‖x‖≤ρ}
```

Figure 3.6: Oracles of games G_0 and G_1.

about δ) as long as int does not occur. Therefore, the probability of int to occur is upper bounded by the probability to find a non-trivial intersection in the row or column space in t tries with $\|x\| = \rho + w$. Hence, by a union bound and Lemma 3.8.4, we have

$$\Pr[\mathsf{int}] \le \frac{t}{q^{n-\rho-2w}} + \frac{t}{q^{m-\rho-2w}}.$$

In G_1, the oracle gives no information to the adversary, as $\|\delta\|$ and $\|x\|$ are known. Therefore, one can remove the oracle and the probability of success of the adversary is simply the probability to guess the correct value δ. The number of vectors in S_w^n is $\begin{bmatrix} n \\ w \end{bmatrix}_q \prod_{i=0}^{w-1}(q^m - q^i)$ (see Gadouleau et al. [GY06] for example). Therefore,

$$\Pr[\mathsf{G}_1(\mathscr{A}_t) \Rightarrow 1] \le \left(\begin{bmatrix} n \\ w \end{bmatrix}_q \prod_{i=0}^{w-1} (q^m - q^i) \right)^{-1}.$$

Hence,

$$\begin{aligned}
\mathsf{Adv}^{\text{learn}}_{\Psi,\rho,\|\cdot\|}(\mathscr{A}_t) &\le |\Pr[\mathsf{G}_0(\mathscr{A}_t) \Rightarrow 1] - \Pr[\mathsf{G}_1(\mathscr{A}_t) \Rightarrow 1]| + \Pr[\mathsf{G}_1(\mathscr{A}_t) \Rightarrow 1] \\
&\le \Pr[\mathsf{int}] + \Pr[\mathsf{G}_1(\mathscr{A}_t) \Rightarrow 1] \\
&\le \frac{t}{q^{n-w-d}} + \frac{t}{q^{m-w-d}} + \left(\begin{bmatrix} n \\ w \end{bmatrix}_q \prod_{i=0}^{w-1} (q^m - q^i) \right)^{-1}.
\end{aligned}$$

□

Discussion. While not proving the hardness of KR-PCA attacks, Theorem 3.8.2 shows that the learning game in the rank metric is difficult for some parameters. As many reaction attacks are based on the capability to solve an instance of the learning game, this result is still significant. Note that when the error weight w is large, $q^{n-\rho-2w} \le 1$ and the bound becomes meaningless.

However, in most settings, the value w is picked small enough. For example, in RQC-I, we have $w = 5, \rho = 31, m = 97$ and $n = 67$. Therefore, the advantage of a t-bounded adversary is roughly bounded by $\frac{t}{2^{26}}$. This means that a number of queries of the order of 2^{26} is necessary to win with good probability. While feasible, the cost is still exponential. More generally, if $\rho + 2w$ is smaller but proportional to n (and m), the learning problem requires an exponential number of queries in the rank metric.

Overall, this result shows that the learning problem is harder in the rank metric than in other norms. Indeed, as we showed [Băe+19], the learning problem for other distances such as the Hamming distance, the L_∞ norm in $\mathbb{Z}_q$ or some variants can be solved with a polynomial number of queries. One explanation is that the learning problem for other metrics can be solved component-wise. That is, by varying one component of x in the query, one can extract information only about the corresponding component in the secret value. Then, it is sufficient to recover the secret component by component. In the rank metric though, this strategy is not possible as varying one entry in the value x does not necessarily give information about a given component. More generally, this confirms the intuition that the rank leaks less information, as flipping one entry in a vector always changes the Hamming weight but not necessarily the rank.

This proof tends to show that the rank metric may be well suited to resist to key misuse and similar attacks.

4 FO-like Combiners and Hybrid Post-Quantum Cryptography

While Fujisaki-Okamoto-like transforms can become security hazards if badly implemented, we show in this chapter that one can use similar constructions to boost security. More precisely, we study here how one can build KEMs that are based on *several* hardness assumptions. Indeed, most of the assumptions the PQ schemes are based on (e.g. learning with errors, syndrome decoding) have been less extensively studied than their classical counterparts. Therefore, combining several systems into one to increase security seems a sound idea. For example, one could combine both a classical PKE/KEM scheme such as RSA with a PQ one, and ideally the resulting cryptosystem should be secure as long as one of the underlying schemes is secure. Such systems have been popularised under the term *hybrid* schemes and the way the underlying systems are combined is called a *combiner*. Moreover, if the resulting hybrid scheme is secure as long as one of the underlying systems is secure, the *combiner* is said to be *robust*.

When it comes to PQ cryptography, hybrid schemes may offer many advantages (depending on how they are implemented), such as:

1. Providing trust regardless of the security of post-quantum assumptions.
2. Fulfilling the standards requirement by combining a standard scheme with another one which is not.
3. Allowing a smooth transition between classical and PQ cryptography in practice, i.e. hybrid cryptography can allow support of both classical and PQ schemes.
4. Combining multiple PQ schemes together might offer better confidence. Such hybrid schemes would come at the cost of efficiency, however combining two efficient schemes might result in a more efficient scheme than one inefficient one. Such ideas and issues were briefly discussed on the NIST PQC forum[1]. We focus mostly on this application of hybrid systems in this work.

[1] https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/msRrR13muS4

Unfortunately, hybrid schemes do not offer much improvement in terms of theoretical security. Indeed, if both underlying schemes require 2^{λ} operations to be broken, the hybrid system would be broken in $2^{\lambda+1}$ operations (i.e. we gain only 1 bit of security). In practice however, the security gain might be better, depending on the underlying schemes. Indeed, one might reasonably argue that the probability of a major breakthrough in two different problems believed to be hard by the community is lower than the probability of one devastating breakthrough. In any case, while the practical security offered by hybrid cryptosystems obviously depends on many parameters, we think that such schemes offer a greater security boost than what can be deduced from the theoretical bounds only.

The results presented in this chapter are part of a joint work with Serge Vaudenay that was published at CANS 2021 [HV21]. As in the previous chapter, this research was conducted during the second round of the NIST PQ standardisation process.

4.1 Contributions

Several authors have considered KEM or signature combiners targeting post-quantum systems in recent years [Bin+17; Bin+19a; GHP18]. However, all the combiners introduced in these papers work in a black-box manner on IND-CCA KEMs. That is, these combiners take two KEMs (or signature schemes) as inputs and output the hybrid construction. Yet, we know that most PQ KEM proposals share a very similar structure: an OW/IND-CPA secure PKE is introduced and then the Fujisaki-Okamato (FO) transform or a variant is applied to give an IND-CCA KEM. Therefore, one could try to directly combine the IND-CPA schemes to give an IND-CCA KEM, hopefully getting better performances. Therefore, we present in this report several hybrid FO-like transforms which combine two OW-CPA PKEs into one IND-CCA KEM. We also generalise these constructions to n schemes (i.e. n PKEs are combined into one KEM).

Compared to previous work, our combiners are simpler as they do not require extra primitives such as special types of PRFs or MACs. As a result, they are slightly more efficient by removing calls to these primitives and by optimising the use of hash functions. Finally, our combiners follow a different paradigm as they replace FO transforms. Thus, they would likely be implemented in cryptographic libraries directly, whereas previous combiners would likely be implemented in applications/protocol libraries (e.g. openssl). Hence, our constructions offer another approach that might be useful to implementors, for example for optimisation or security purposes.

The main disadvantage of FO transforms is that they are only secure in the random oracle model (ROM) and we prove the security of our FO-like hybrid combiner in the ROM as well. However, as all PQ IND-CCA KEM submitted to the NIST process are only proven secure in the ROM, it does not add an extra assumption. We also prove that one of our combiners is secure in the Quantum Random Oracle Model (QROM). The results are summarised in Figure 4.1.

At a high level, our combiners share the same structure as a system that would apply a robust

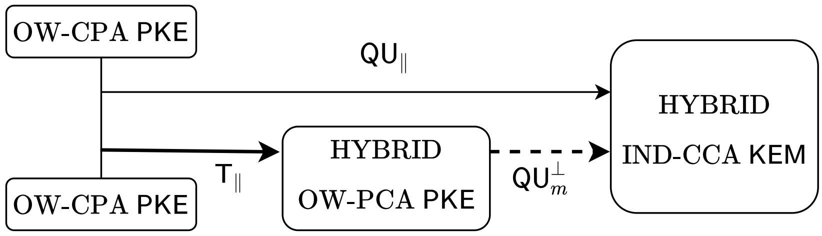


Figure 4.1: Solid arrows indicate results implied by our combiners, bold arrows indicate QROM security. The dashed arrow indicates results from Hofheinz et al. [HHK17].

PKE combiner (e.g. concatenating ciphertexts) followed by a FO-like transform to get a KEM. However, having one scheme for the whole process allows a fine-grained control over the way key derivation and de-randomisation are performed, in turn offering better flexibility. For instance, we study how one can combine hash functions (i.e. random oracles) s.t. our main FO-like combiner is more efficient or secure. More precisely, we define the properties the functions g (used to derive random coins in our construction) and h (used to derive the shared key) should have in order for our construction to be secure. Such theoretical analysis is important, as it was demonstrated that Random Oracles in FO transforms are easily combined in an insecure way [BDG20]. Therefore, by presenting generic n-PKEs-to-KEM combiners with detailed security proofs and several examples of ROs combinations, we hope to offer clear flexibility and security guarantees to implementors.

As a proof of concept, we implemented a hybrid KEM based on the IND-CPA version of HQC and LAC, two round 2 proposals to the NIST PQ standardisation process. We call this hybrid KEM `hqc_lac_128` and we report and analyse how this scheme compares to the other round 2 proposals. In particular, we show that the performance of the hybrid scheme is comparable to the performance of the least efficient underlying scheme (i.e. HQC in this case). Moreover, as our combiner is highly parallelisable, our tests show that a parallelised version of `hqc_lac_128` is as efficient as HQC in terms of speed, excluding a negligible overhead (mainly due to the creation of an additional thread). We think this demonstrates that using a hybrid PQ system in place of a single PQ scheme may increase significantly the security at a small cost.

Finally, we compute the theoretical performance (based on the data from eBACS [Be20]) of every possible hybrid scheme based on two PQ IND-CPA schemes that are based on assumptions of a different type (e.g. a lattice-based scheme with a code-based scheme). We discuss the performance of the most efficient ones in two metrics, namely public key/ciphertext size and encapsulation/decapsulation speed. This analysis shows that a given hybrid scheme struggles to perform as well as an efficient non-hybrid one in both metrics.

4.2 Related Work

Many authors have considered robust combiners for different primitives, like combiners for PKEs [DK05; Zha+16], hash functions [AHV98; FL08; FLP08], commitment schemes [Her05], PQ signatures [Bin+17], AEAD [PR20]. Recently, robust combiners for KEMs have also been considered by Giacon et al. in [GHP18]. In that work, they propose various robust combiners in the standard model and in the random oracle model that take two IND-CCA KEMs and output another IND-CCA KEM. Similarly, Bindel et al. [Bin+19a] propose similar robust KEM combiners which are secure against quantum adversaries. Our combiners differ from these as we aim at building a monolithic IND-CCA KEM based on several IND-CPA (or OW-CPA) PKEs. In a way, we bypass the intermediate KEM constructions, as many KEMs are based on FO-transformed IND-CPA schemes.

4.3 FO-like Combiners

Compared to FO-like transforms, we wish to design constructions that take *two* (or more) IND/OW-CPA schemes instead of one and that output an IND-CCA KEM. Compared to black-box combiners, this approach allows for lower-level combiners, which in turn can be more efficient. As more precise examples, we consider KEM combiners proposed by Bindel et al. [Bin+19a], given in Figure 4.2. These three constructions, namely XtM, $\mathsf{dualPRF}$ and N are based on special kinds of MACs and PRFs (we refer the interested reader to the original paper for the corresponding definitions). In the XtM combiner, the keys must be split and a tag on the ciphertexts is computed. Similarly, in the $\mathsf{dualPRF}$ and N combiners, multiple passes on the keys and ciphertext must be performed to derive the key. All these operations add complexity and/or increase the ciphertext length while being redundant if the underlying KEMs are built using a FO-like transform. Thus, one could hope to remove several superfluous computations and primitives by looking at the actual implementation of the underlying KEMs. We apply this idea to construct several new combiners, which we call *FO-like combiners*. In addition of not being black-box, these combiners differ from other proposals in the fact that they take several PKEs as inputs and output a KEM.

4.3.1 $\mathsf{T}_\parallel$ combiner

For our first construction, the idea is to apply twice the T transform of Hofheinz et al. [HHK17] (see Figure 2.16) to obtain an OW-PCA PKE from two OW-CPA PKEs $\mathsf{PKE}_i = (\mathsf{Gen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ with $i \in \{1, 2\}$. We call this FO-like combiner $\mathsf{T}_\parallel$ and we present it in Figure 4.3. Then, one can apply the $\mathsf{U}^{\not\perp}$ transform (see Figure 2.17) and Theorem 2.4.5 to obtain an IND-CCA KEM. The message space $\mathcal{M}$ of the resulting PKE is $\mathcal{M}_1 \times \mathcal{M}_2$ (i.e. the space product of the two message spaces). This construction is actually a useful intermediary step towards a more general OW-CPA to KEM IND-CCA combiner we present in the next section.

The following theorem shows that $\mathsf{T}_\parallel$ is a robust combiner (as long as one of the two underlying

$\mathsf{Encaps}_{\mathsf{XtM}}(\mathsf{pk}_1,\mathsf{pk}_2)$

$(\mathsf{ct}_1, K_{k,1} \| K_{m,1}) \leftarrow \mathsf{Encaps}_1(\mathsf{pk}_1)$
$(\mathsf{ct}_2, K_{k,2} \| K_{m,2}) \leftarrow \mathsf{Encaps}_2(\mathsf{pk}_2)$
$K_k \leftarrow K_{k,1} \oplus K_{k,2}$
$K_m \leftarrow K_{m,1} \| K_{m,2}$
$\mathsf{ct} \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2)$
$\mathsf{tag} \leftarrow \mathsf{MAC}_{K_m}(\mathsf{ct})$
$\mathbf{return}\ ((\mathsf{ct}, \mathsf{tag}), K_k)$

$\mathsf{Encaps}_{\mathsf{dualPRF}}(\mathsf{pk}_1,\mathsf{pk}_2)$

$(\mathsf{ct}_1, K_1) \leftarrow \mathsf{Encaps}_1(\mathsf{pk}_1)$
$(\mathsf{ct}_2, K_2) \leftarrow \mathsf{Encaps}_2(\mathsf{pk}_2)$
$\mathsf{ct} \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2)$
$K_d \leftarrow \mathsf{dPRF}(K_1, K_2)$
$K \leftarrow \mathsf{PRF}(K_d, \mathsf{ct})$
$\mathbf{return}\ (\mathsf{ct}, K)$

$\mathsf{Encaps}_{\mathsf{N}}(\mathsf{pk}_1,\mathsf{pk}_2)$

$(\mathsf{ct}_1, K_1) \leftarrow \mathsf{Encaps}_1(\mathsf{pk}_1)$
$(\mathsf{ct}_2, K_2) \leftarrow \mathsf{Encaps}_2(\mathsf{pk}_2)$
$\mathsf{ct} \leftarrow (\mathsf{ct}_1, \mathsf{ct}_2)$
$K_p \leftarrow \mathsf{PRF}'(0, K_1)$
$K_d \leftarrow \mathsf{dPRF}(K_p, K_2)$
$K \leftarrow \mathsf{PRF}(K_d, \mathsf{ct})$
$\mathbf{return}\ (\mathsf{ct}, K)$

$\mathsf{Decaps}_{\mathsf{XtM}}((\mathsf{sk}_1,\mathsf{sk}_2),\mathsf{ct})$

parse $(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{tag}) \leftarrow \mathsf{ct}$
$K'_{k,1} \| K'_{m,1} \leftarrow \mathsf{Decaps}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$K'_{k,2} \| K'_{m,2} \leftarrow \mathsf{Decaps}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
$K'_k \leftarrow K'_{k,1} \oplus K'_{k,2}$
$K'_m \leftarrow K'_{m,1} \| K'_{m,2}$
$\mathbf{if}\ \mathsf{Ver}_{K'_m}(\mathsf{ct}) = 0$
  $\mathbf{return}\ \perp$
$\mathbf{return}\ K'_k$

$\mathsf{Decaps}_{\mathsf{dualPRF}}((\mathsf{sk}_1,\mathsf{sk}_2),\mathsf{ct})$

parse $(\mathsf{ct}_1, \mathsf{ct}_2) \leftarrow \mathsf{ct}$
$K'_1 \leftarrow \mathsf{Decaps}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$K'_2 \leftarrow \mathsf{Decaps}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
$K'_d \leftarrow \mathsf{dPRF}(K'_1, K'_2)$
$K' \leftarrow \mathsf{PRF}(K'_d, \mathsf{ct})$
$\mathbf{return}\ K'$

$\mathsf{Decaps}_{\mathsf{N}}((\mathsf{sk}_1,\mathsf{sk}_2),\mathsf{ct})$

parse $(\mathsf{ct}_1, \mathsf{ct}_2) \leftarrow \mathsf{ct}$
$K'_1 \leftarrow \mathsf{Decaps}_1(\mathsf{sk}_1, \mathsf{ct}_1)$
$K'_2 \leftarrow \mathsf{Decaps}_2(\mathsf{sk}_2, \mathsf{ct}_2)$
$K'_p \leftarrow \mathsf{PRF}'(0, K'_1)$
$K'_d \leftarrow \mathsf{dPRF}(K'_p, K'_2)$
$K' \leftarrow \mathsf{PRF}(K'_d, \mathsf{ct})$
$\mathbf{return}\ K'$

Figure 4.2: KEM combiners from Bindel et al. [Bin+19a]. The underlying KEMs are $(\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $(\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$. The key generation function of the resulting KEM is omitted as it is simply the concatenation of Gen_1 and Gen_2.

PKEs is OW-CPA, the resulting PKE is OW-PCA).

Theorem 4.3.1. *Let* PKE *be the PKE resulting from applying* $\mathsf{T}_\|$ *on* PKE_1 *and* PKE_2*, which are respectively* δ_1 *and* δ_2 *correct. In addition, let G be a hash function modelled as a random oracle. Then, for any efficient OW-PCA adversary* $\mathscr{A}$ *making at most* q_G *queries to G and* q_P *queries to the plaintext-checking oracle, there exist adversaries* $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{A}) \le (q_G + q_P)\cdot(\delta_1+\delta_2) + (q_G+1)\cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\}\,,$$

where $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *run in about the same time as* $\mathscr{A}$*.*

Proof. We first show that the trivial PKE combiner C in Figure 4.4 is a robust OW-PCA combiner. Let $\mathsf{PKE} = \mathsf{C}(\mathsf{PKE}_1, \mathsf{PKE}_2)$ be the PKE resulting from applying C on two PKEs PKE_1 and PKE_2. We show w.l.o.g. that the OW-PCA security of PKE reduces to the OW-PCA security of PKE_1. The OW-PCA game against PKE is presented in Figure 4.5. One can see that the plaintext-checking oracle can easily be simulated by an adversary having access to a plaintext-checking oracle for PKE_1 and holding the secret key sk_2. Thus, we can easily build an adversary $\mathscr{B}$ against the OW-PCA security of PKE_1. This adversary generates itself $\mathsf{pk}_2, \mathsf{sk}_2, \mathsf{ct}_2^*$, runs $\mathscr{A}$ and simulates perfectly the PCO oracle with its own oracle and sk_2. When $\mathscr{A}$ returns $(\mathsf{pt}'_1, \mathsf{pt}'_2)$, $\mathscr{B}$

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Enc}(\mathsf{pk},(\mathsf{pt}_1,\mathsf{pt}_2))$	$\mathsf{Dec}(\mathsf{sk},(\mathsf{ct}_1,\mathsf{ct}_2))$
$(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow\$ \mathsf{Gen}_1(1^\lambda)$	parse $(\mathsf{pk}_1,\mathsf{pk}_2) \leftarrow \mathsf{pk}$	parse $(\mathsf{sk}_1,\mathsf{sk}_2) \leftarrow \mathsf{sk}$
$(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow\$ \mathsf{Gen}_2(1^\lambda)$	$r_1 \leftarrow G(\mathsf{pt}_1)$	$\mathsf{pt}'_1 \leftarrow \mathsf{Dec}_1(\mathsf{sk}_1,\mathsf{ct}_1)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1,\mathsf{pk}_2)$	$r_2 \leftarrow G(\mathsf{pt}_2)$	$\mathsf{pt}'_2 \leftarrow \mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1,\mathsf{sk}_2)$	$\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1,\mathsf{pt}_1;r_1)$	**if** $\mathsf{Enc}_1(\mathsf{pk}_1,\mathsf{pt}'_1;G(\mathsf{pt}'_1)) \neq \mathsf{ct}_1$:
return $(\mathsf{pk},\mathsf{sk})$	$\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2,\mathsf{pt}_2;r_2)$	**return** $\perp$
	return $(\mathsf{ct}_1,\mathsf{ct}_2)$	**if** $\mathsf{Enc}_2(\mathsf{pk}_2,\mathsf{pt}'_2;G(\mathsf{pt}'_2)) \neq \mathsf{ct}_2$:
		return $\perp$
		return $(\mathsf{pt}'_1,\mathsf{pt}'_2)$

Figure 4.3: $\mathsf{T}_\parallel$ combiner.

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Enc}(\mathsf{pk},(\mathsf{pt}_1,\mathsf{pt}_2))$	$\mathsf{Dec}(\mathsf{sk},(\mathsf{ct}_1,\mathsf{ct}_2))$
$(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow\$ \mathsf{Gen}_1(1^\lambda)$	parse $(\mathsf{pk}_1,\mathsf{pk}_2) \leftarrow \mathsf{pk}$	parse $(\mathsf{sk}_1,\mathsf{sk}_2) \leftarrow \mathsf{sk}$
$(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow\$ \mathsf{Gen}_2(1^\lambda)$	$\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1,\mathsf{pt}_1)$	$\mathsf{pt}'_1 \leftarrow \mathsf{Dec}_1(\mathsf{sk}_1,\mathsf{ct}_1)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1,\mathsf{pk}_2)$	$\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2,\mathsf{pt}_2)$	$\mathsf{pt}'_2 \leftarrow \mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1,\mathsf{sk}_2)$	**return** $(\mathsf{ct}_1,\mathsf{ct}_2)$	**return** $(\mathsf{pt}'_1,\mathsf{pt}'_2)$
return $(\mathsf{pk},\mathsf{sk})$		

Figure 4.4: Trivial PKE combiner C.

returns pt'_1 and wins with at least the same advantage as $\mathscr{A}$. Hence,

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{A}) \leq \min\{\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\} .$$

To conclude, one can just observe that $\mathsf{T}_\parallel(\mathsf{PKE}_1,\mathsf{PKE}_2) = \mathsf{C}(\mathsf{T}(\mathsf{PKE}_1),\mathsf{T}(\mathsf{PKE}_2))$, where T is the OW-CPA to OW-PCA transform from Hofheinz et al. [HHK17]. Hence, applying Theorem 2.4.5 concludes the proof. □

Corollary 4.3.1. *Let* KEM *be the KEM resulting from applying* $\mathsf{U}^{\not\perp} \circ \mathsf{T}_\parallel$ *onto two PKE schemes* PKE_1 *and* PKE_2*, which are* δ_1*-correct and* δ_2*-correct, respectively. Then, for any IND-CCA adversary* $\mathscr{A}$ *making at most* q_H *and* q_G *queries to the ROs* H *and* G*, respectively, and* q_D *queries to the decapsulation oracle, there exist OW-CPA adversaries* $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathscr{A}) \leq \frac{q_H}{|\mathscr{M}_1||\mathscr{M}_2|} &+ (q_G + q_D)\cdot(\delta_1+\delta_2)\\ &+ (q_G+1)\cdot\min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\} ,\end{aligned}$$

where $\mathscr{M}_i$ *is the message space of* PKE_i *and* $\mathscr{B}_i$ *runs in about the same time as* $\mathscr{A}$.

Proof. This is a simple consequence of Theorems 2.4.5 and 4.3.1. □

$$\begin{array}{l}
\underline{\text{OW-PCA}_{\mathsf{PKE}}(\mathscr{A})} \\
((\mathsf{pk}_1,\mathsf{pk}_2),(\mathsf{sk}_1,\mathsf{sk}_2)) \leftarrow\!\$\, \mathsf{Gen}(1^\lambda) \\
(\mathsf{pt}_1^*,\mathsf{pt}_2^*) \leftarrow\!\$\, \mathscr{M}_1 \times \mathscr{M}_2 \\
\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk},(\mathsf{pt}_1^*,\mathsf{pt}_2^*)) \\
(\mathsf{pt}_1',\mathsf{pt}_2') \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{PCO}}}(\mathsf{pk},\mathsf{ct}^*) \\
\textbf{return } 1_{(\mathsf{pt}_1',\mathsf{pt}_2')=(\mathsf{pt}_1^*,\mathsf{pt}_2^*)}
\end{array}
\qquad
\begin{array}{l}
\underline{\textbf{Oracle } \mathscr{O}^{\mathsf{PCO}}((\mathsf{pt}_1,\mathsf{pt}_2),(\mathsf{ct}_1,\mathsf{ct}_2))} \\
\mathsf{pt}_1' \leftarrow \mathsf{Dec}_1(\mathsf{sk}_1,\mathsf{ct}_1) \\
\mathsf{pt}_2' \leftarrow \mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2) \\
\textbf{return } 1_{(\mathsf{pt}_1,\mathsf{pt}_2)=(\mathsf{pt}_1',\mathsf{pt}_2')}
\end{array}$$

Figure 4.5: OW-PCA game against PKE for the proof of Theorem 4.3.1.

$$\begin{array}{l}
\underline{\mathscr{B}_1^{\mathscr{A},\mathscr{O}^{\mathsf{PCO}_1}}(\mathsf{pk}_1,\mathsf{ct}_1^*)} \\
(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow\!\$\, \mathsf{Gen}_2(1^\lambda) \\
\mathsf{pt}_2^* \leftarrow\!\$\, \mathscr{M}_2 \\
\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk},\mathsf{pt}_2^*) \\
(\mathsf{pt}_1',\mathsf{pt}_2') \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{PCO}}}((\mathsf{pk}_1,\mathsf{pk}_2),(\mathsf{ct}_1^*,\mathsf{ct}_2^*)) \\
\textbf{return } \mathsf{pt}_1'
\end{array}
\qquad
\begin{array}{l}
\underline{\textbf{Oracle } \mathscr{O}^{\mathsf{PCO}}((\mathsf{pt}_1,\mathsf{pt}_2),(\mathsf{ct}_1,\mathsf{ct}_2))} \\
r \leftarrow \mathscr{O}^{\mathsf{PCO}_1}(\mathsf{pt}_1,\mathsf{ct}_1) \\
\mathsf{pt}_2' \leftarrow \mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2) \\
\textbf{return } 1_{r=1 \wedge \mathsf{pt}_2=\mathsf{pt}_2'}
\end{array}$$

Figure 4.6: OW-CPA adversary for the proof of Theorem 4.3.1.

Discussion

Let $\mathsf{UT}_{\|}^{\not\perp}$ be the combiner resulting from composing $\mathsf{U}^{\not\perp}$ and $\mathsf{T}_{\|}$. One could wonder whether combining two PKEs in a trivial way (i.e. encrypting $\mathsf{pt}_1,\mathsf{pt}_2$ as $(\mathsf{Enc}_1(\mathsf{pt}_1),\mathsf{Enc}_2(\mathsf{pt}_2))$ and decrypting both ciphertexts independently) and then applying a FO-like transform would output a robust IND-CCA KEM. In fact, this would give a combiner similar to $\mathsf{UT}_{\|}^{\not\perp}$, except the random coins would be split into two parts $(G(\mathsf{pt}_1,\mathsf{pt}_2))_{\lambda_1}$ and $(G(\mathsf{pt}_1,\mathsf{pt}_2))_{\lambda_2}$ for each encryption procedure, where λ_i is the number of coins needed by the encryption of PKE_i. As G is a RO, both shares would be independent and the result would be similar to the coins $G(\mathsf{pt}_i)$ in our $\mathsf{UT}_{\|}^{\not\perp}$ transform. We preferred the latter solution as it is possible to compute the coins in parallel and we think it makes the separation between both sets of coins clear. One could also wonder whether setting the coins to $G(\mathsf{pt}_1,\mathsf{pt}_2)$ would work. This, in turn, creates a correlation between both ciphertexts, which cannot be dealt with in the security proof.

The choice of computing the deterministic coins for ct_i based on σ_i only (instead of σ_1 and σ_2) has positive and negative impacts on the resulting scheme:

- **Efficiency**: Both ciphertexts are totally independent and can be computed in parallel. In turn, this allows to keep a key share static for a period of time while varying the other one. This could improve consequently the efficiency of hybrid schemes in protocols.

- **Malleability and misuse resistance**: The ciphertext of the resulting KEM $\mathsf{ct}^* = (\mathsf{ct}_1^*,\mathsf{ct}_2^*)$ is somewhat malleable. Indeed, it is easy to modify a ciphertext into another one s.t. the decryption is valid. For instance, $\mathsf{ct}' = (\mathsf{ct}_1^*,\mathsf{ct}_2')$, for a valid ct_2', will decapsulate properly to the key $H(\sigma_1^*,\sigma_2',\mathsf{ct}')$. This has no consequence in the ROM as the RO hides perfectly

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk},(\mathsf{ct}_1,\mathsf{ct}_2))$
$(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow\!\!\$\, \mathsf{Gen}_1(1^\lambda)$	parse $(\mathsf{pk}_1,\mathsf{pk}_2) \leftarrow \mathsf{pk}$	parse $(\mathsf{sk}_1,\mathsf{sk}_2) \leftarrow \mathsf{sk}$
$(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow\!\!\$\, \mathsf{Gen}_2(1^\lambda)$	$(\sigma_1,\sigma_2) \leftarrow\!\!\$\, \mathcal{M}_1 \times \mathcal{M}_2$	$\sigma_1' \leftarrow \mathsf{Dec}_1(\mathsf{sk}_1,\mathsf{ct}_1)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1,\mathsf{pk}_2)$	$\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1;G(1,\sigma_1,\sigma_2))$	$\sigma_2' \leftarrow \mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1,\mathsf{sk}_2)$	$\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2;G(2,\sigma_1,\sigma_2))$	**if** $\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1';G(1,\sigma_1',\sigma_2')) \neq \mathsf{ct}_1$:
return $(\mathsf{pk},\mathsf{sk})$	$K \leftarrow H(\sigma_1 \oplus \sigma_2)$	**return** $\perp$
	return $(\mathsf{ct}_1,\mathsf{ct}_2), K$	**if** $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2';G(2,\sigma_1',\sigma_2')) \neq \mathsf{ct}_2$:
		return $\perp$
		return $H(\sigma_1' \oplus \sigma_2')$

Figure 4.7: $\mathsf{UT}_\parallel$ combiner.

σ_1^*, but this does not necessarily seem like a desired property. In particular, due to this malleability effect, the key must be derived as $H(\sigma_1,\sigma_2,\ldots)$ and other KDFs that would seem intuitive lead to security flaw. For instance, computing the key as $H(\sigma_1) \oplus H(\sigma_2)$ in the transform makes a trivial IND-CCA attack possible.

Efficiency of $\mathsf{T}_\parallel$

One can see that the main cost of the combiner is to compute two hash values on the two plaintexts (i.e. seeds) and then a hash on the two plaintexts and ciphertexts. This already seems slightly more efficient than the XtM (XOR-then-MAC) combiner proposed by Bindel et al. [Bin+19a]. Indeed, XtM doubles the size of the keys returned by the underlying KEMs, splits them, and computes a MAC on the ciphertexts using two halves of the keys.

Now, as the ciphertexts in post-quantum cryptography can be large (usually a few kilobytes), computing a hash on two ciphertexts can be an expensive operation. Our combiner presented in the next section fixes this drawback.

4.3.2 $\mathsf{UT}_\parallel$ combiner

We now propose an FO-like combiner similar to $\mathsf{T}_\parallel$ that combines two OW-CPA PKEs into an IND-CCA KEM. In a way, we skip the $\mathsf{U}^{\not\perp}$ transform to directly get a KEM. The idea is to encrypt two seeds (i.e. plaintexts) σ_1,σ_2 using the PKE resulting from $\mathsf{T}_\parallel$ and then compute the key as $H(\sigma_1 \oplus \sigma_2)$. However, in order to avoid the malleability issue described in the previous section, the deterministic coins are computed as $G(i,\sigma_1,\sigma_2)$. This links both ciphertexts together and makes tampering one of the two more difficult. Note that in order to compute the XOR, we assume that the seeds σ_i are binary strings or that there exists an efficient and unique encoding of these objects as binary strings. Alternatively, one can take the hash of a plaintext to get a binary seed. All these options are compatible with our combiner and the choice of an approach depends on the underlying PKEs. We present the combiner in Figure 4.7.

Now, the following theorem formally states the security of the $\mathsf{UT}_\parallel$ combiner.

Theorem 4.3.2. *Let* KEM *be the KEM resulting from applying* $\mathsf{UT}_\parallel$ *on* PKE_1 *and* PKE_2*, which are respectively* δ_1 *and* δ_2*-correct, and* γ_1 *and* γ_2*-spread. In addition, let* G *and* H *be hash functions modelled as random oracles. Then, for any efficient IND-CCA adversary* $\mathscr{A}$ *making at most* q_G, q_H*, and* q_D *queries to* G, H*, and* $\mathscr{O}^{\mathsf{Dec}}$*, respectively, there exist adversaries* $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathscr{A}) \leq &(q_D + q_G + 1)\cdot(\delta_1+\delta_2) + q_D\cdot(2^{-\gamma_1}+2^{-\gamma_2}) \\ &+ (q_G+q_H)\cdot\min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\}\,,\end{aligned}$$

where $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *run in about the same time as* $\mathscr{A}$ *and make the same number of queries.*

Proof. We first show that if a valid ciphertext is submitted to the decapsulation oracle, then the corresponding plaintexts have been queried to G, and thus one can simulate the decapsulation oracle without the secret key. Then, one can show that the deterministic coins used to compute the challenge ciphertexts look perfectly random until the adversary queries the challenge plaintexts to G. Finally, by the same property of the RO, the challenge key looks perfectly uniform unless the adversary queries $\sigma_1 \oplus \sigma_2$ to H. We proceed by game hopping, the sequence of games is presented in Figure 4.8.

Game Γ^0: This is the original KEM IND-CCA game for the KEM obtained by applying the $\mathsf{UT}_\parallel$ combiner on two PKEs.

Game Γ^1: In this game, we enforce the correctness of the challenge ciphertexts and the ciphertexts that can be computed by the adversary using the RO G. In particular, we abort if the challenge ciphertexts break the correctness property or if any σ_i in a query m to G is of the form (i, σ_1, σ_2) and is s.t. $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; G(m))$ breaks the correctness property. Now, both challenge queries to G made by the game (i.e. $(i, \sigma_1^*, \sigma_2^*)$) are fresh, hence by the property of the RO and the δ_i-correctness of the underlying schemes PKE_i, the probability there is a correctness error is at most $\delta_1 + \delta_2$. Then, throughout the game, at most $2q_D$ queries is made to G by the game in the decapsulation oracle (q_D of the form $(1, \sigma_1, \sigma_2)$ and q_D of the form $(2, \sigma_1, \sigma_2)$) and q_G by the adversary. Hence, in the worst case all these queries are fresh and the probability there is a correctness error is upper bounded by $(q_D + q_G + 1)\cdot(\delta_1+\delta_2)$. Hence, we have

$$\left|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\right| \leq (q_D + q_G + 1)\cdot(\delta_1+\delta_2)\,.$$

Game Γ^2: We modify the previous game as follows in the decapsulation oracle. We check whether there exist both $((1, \sigma_1, \sigma_2), g_1)$ and $((2, \sigma_1, \sigma_2), g_2)$ in $\mathscr{L}_{\mathscr{A}}$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$. If this is the case, (let's call this event found) we return $H(\sigma_1 \oplus \sigma_2)$, otherwise we return the key K output by the decapsulation function. Now, if found occurs, we return the same key as in game Γ^1. Indeed, by the perfect correctness of the tuples in $\mathscr{L}_{\mathscr{A}}$ enforced in game Γ^1, if we find (σ_1, σ_2) s.t. $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; G(i, \sigma_1, \sigma_2)) = \mathsf{ct}_i$, then $\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct}_i) = \sigma_i$. Hence, we have, $\Pr[\Gamma^1 \Rightarrow 1] = \Pr[\Gamma^2 \Rightarrow 1]$.

Game Γ^3: We modify the previous game as follows. In the decapsulation oracle, we simply return $\perp$ if found does not occur. Hence, game Γ^2 and Γ^3 differ iff the decapsulation oracle successfully decrypts the given ct but the adversary did not query $(1,\sigma_1',\sigma_2')$ or $(2,\sigma_1',\sigma_2')$ to G, where $(\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1'),\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2')) = \mathsf{ct}$ (i.e. at least one tuple is not in $\mathscr{L}_{\mathscr{A}}$). Now, by the perfect correctness of tuples in $\mathscr{L}_{\mathscr{A}}$, this event is equivalent to the decapsulation oracle successfully (i.e. the re-encryption checks pass) recovering the seeds σ_1,σ_2 but either $(1,\sigma_1,\sigma_2)$, or $(2,\sigma_1,\sigma_2)$, or both were not queried to G by the adversary. Let fail be this event and we prove the following lemma.

Lemma 4.3.1.

$$\Pr[\mathsf{fail}] \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \,.$$

Proof. Let fail_k be the event that fail happens at the k-th decapsulation query and $p_k = \Pr[\mathsf{fail}_k]$. By a union bound, it is clear that

$$\Pr[\mathsf{fail}] \leq \sum_{k=1}^{q_D} p_k \,.$$

Now, let's consider an algorithm $\mathscr{B}_k$ as defined in Figure 4.9. This adversary simulates perfectly the view of the adversary in game Γ^3 until the k-th query. In particular, for each decapsulation query $\mathsf{ct} = (\mathsf{ct}_1,\mathsf{ct}_2)$, it checks whether there exist both $((1,\sigma_1,\sigma_2),g_1)$ and $((2,\sigma_1,\sigma_2),g_2)$ in $\mathscr{L}_G$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i) = \mathsf{ct}_i$ for $i \in [2]$. We call this condition cond and if it is fulfilled $\mathscr{B}_k$ outputs $H(\sigma_1 \oplus \sigma_2)$, otherwise it outputs $\perp$.

In the k-th decapsulation query, if cond is fulfilled it aborts. Otherwise, it sets i s.t. there is no $((i,\sigma_1,\sigma_2),g_i) \in \mathscr{L}_G$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i) = \mathsf{ct}_i$. Note that such an i will be found because cond was not fulfilled. Also, this condition might be fulfilled for both $i = 1$ and $i = 2$. If it is the case, the algorithm sequentially performs the remaining of the instructions for both $i = 1$ and $i = 2$. Next, it decrypts ct_1 and ct_2 to both σ_1' and σ_2'. By the definition of i and the perfect correctness of the values σ_i in $\mathscr{L}_G$, we have that $(i,\sigma_1',\sigma_2') \notin \mathscr{L}_G$. In addition, by the perfect correctness of the challenge ciphertexts we have $(\sigma_1',\sigma_2') \neq (\sigma_1^*,\sigma_2^*)$. Finally, $\mathscr{B}_k$ queries $g_i' \leftarrow G(i,\sigma_1',\sigma_2')$ and outputs 1 iff $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i') = \mathsf{ct}_i$. Now, as $g_i' = G(i,\sigma_1',\sigma_2')$ was never queried to G, it is sampled uniformly at random and thus $\Pr[\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i';g_i') = \mathsf{ct}_i] \leq 2^{-\gamma_i}$ by the γ_i-spreadness of PKE_i. In the worst case, the check is performed for both $i = 1$ and $i = 2$ and thus $\Pr[\mathscr{B}_k(\mathscr{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2}$. Now, we simply observe that if fail_k occurs, then $\mathscr{B}_k$ perfectly simulates the decapsulation oracle in Γ^2 and Γ^3 in the first $k-1$ queries and it will output 1 by the definition of fail_k. Thus,

$$p_k \leq \Pr[\mathscr{B}_k(\mathscr{A}) \Rightarrow 1] \leq 2^{-\gamma_1} + 2^{-\gamma_2} \,.$$

Taking the union bound on the p_k's concludes the proof. $\square$

By the previous lemma, we have

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq q_D \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) .$$

<u>Game Γ^4</u>: First, note that the decapsulation oracle does not use the secret key anymore. Then, in Γ^4, we raise a flag chal^1 and abort if the adversary queries $(i, \sigma_1^*, \sigma_2^*)$. In addition, we raise a flag chal^2 and abort if the adversary queries $\sigma_1^* \oplus \sigma_2^*$ to H. Now, if chal^1 or chal^2 happens, one can break the OW-CPA property of both PKEs. We give the reduction $\mathscr{B}_1$ that breaks the one-wayness of PKE_1 in Figure 4.10. More precisely, as long as $\mathsf{chal}^1 \cup \mathsf{chal}^2$ does not happen, the adversary cannot distinguish a game where the coins used to compute the challenge ciphertexts are deterministic from a game where the coins are taken at random. In addition, it cannot distinguish a game where K is random from a game where $K = H(\sigma_1^* \oplus \sigma_2^*)$. Therefore, the probability that $\mathsf{chal}^1 \cup \mathsf{chal}^2$ happens is the same in Γ^4 and in the OW-CPA game played by $\mathscr{B}_1$. Now, if $\mathsf{chal}^1 \cup \mathsf{chal}^2$ happens in a game where the challenge coins and the key are random, one can break the one-wayness of the underlying scheme. Thus, we have

$$|\Pr[\Gamma^3 \Rightarrow 1] - \Pr[\Gamma^4 \Rightarrow 1]| \leq (q_G + q_H) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\} .$$

Note that we have a factor $q_G + q_H$ because in the reduction we cannot check which query m contains the challenge seeds σ_i^* (if we picked a query to G) or $\sigma_1^* \oplus \sigma_2^*$ (if we picked a query to H). Indeed, in the OW-CPA game, the challenge coins are taken at random and are unknown to the adversary. More details are given in the proof of Theorem 4.3.1.

<u>Game Γ^5</u>: Finally, in this last game we replace the challenge key K_0 by a random one. As K_0 and K_1 have the same distribution now, we have $\Pr[\Gamma^5 \Rightarrow 1] = \frac{1}{2}$. In addition, since the adversary cannot query $\sigma_1^* \oplus \sigma_2^*$ anymore, it cannot distinguish between a real key and a random key by the property of the RO H. Hence, we have $|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]| = 0$. Collecting the probabilities and folding the OW-CPA adversaries into one concludes the proof. □

Generalisation to n PKEs

While the $\mathsf{UT}_\parallel$ combiner presented in Figure 4.7 takes two PKEs as input, it is straightforward to generalise it to n PKEs. Each of the n ciphertexts will simply be computed as $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; G(i, \sigma_1, \ldots, \sigma_n))$ and the key as $H(\oplus_i^n \sigma_i)$. Then, the security of such a combiner (we call it $\mathsf{UT}_\parallel^n$) can be stated in the following Theorem, which is a generalisation of Theorem 4.3.2.

Theorem 4.3.3. *Let* KEM *be the KEM resulting from applying* $\mathsf{UT}_\parallel^n$ *on* $\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n$*, which are respectively* $\delta_1, \ldots, \delta_n$*-correct, and* $\gamma_1, \ldots, \gamma_n$*-spread. In addition, let* G *and* H *be hash functions modelled as random oracles. Then, for all efficient IND-CCA adversary* $\mathscr{A}$ *making at most* q_G, q_H *and* q_D *queries to* G, H *and* $\mathcal{O}^{\mathsf{Dec}}$*, respectively, there exist adversaries* $\mathscr{B}_1, \ldots, \mathscr{B}_n$

$\Gamma^i(\mathscr{A})$

$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(\sigma_1^*,\sigma_2^*) \leftarrow\$ \mathscr{M}_1 \times \mathscr{M}_2$
$\mathsf{coins}_1 \leftarrow G(1,\sigma_1^*,\sigma_2^*); \mathsf{coins}_2 \leftarrow G(2,\sigma_1^*,\sigma_2^*)$ $\quad /\!/\ \Gamma^0\text{-}\Gamma^4$
$\mathsf{coins}_1,\mathsf{coins}_2 \leftarrow\$ \mathscr{R}^2$ $\quad /\!/\ \Gamma^5$
$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*;\mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2^*;\mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{Dec}_i(\mathsf{sk}_i,\mathsf{ct}_i^*) \neq \sigma_i^*$: **abort** $\quad /\!/\ \Gamma^1\text{-}\Gamma^5$
$b \leftarrow\$ \{0,1\}$
$K_0 \leftarrow H(\sigma_1^* \oplus \sigma_2^*)$ $\quad /\!/\ \Gamma^0\text{-}\Gamma^4$
$K_0 \leftarrow\$ \mathscr{K}$ $\quad /\!/\ \Gamma^5$
$K_1 \leftarrow\$ \mathscr{K}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}},G,H}(\mathsf{pk},(\mathsf{ct}_1^*,\mathsf{ct}_2^*),K_b)$
return $1_{b'=b}$

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1,\mathsf{ct}_2))$

$\mathsf{flag} \leftarrow$ **false**
if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if $\exists((1,\sigma_1,\sigma_2),g_1) \in \mathscr{L}_{\mathscr{A}}$ s.t. $\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1;g_1) = \mathsf{ct}_1$
    and $\exists((2,\sigma_1,\sigma_2),g_2) \in \mathscr{L}_{\mathscr{A}}$
        s.t. $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2;g_2) = \mathsf{ct}_2$: $\quad /\!/\ \Gamma^2\text{-}\Gamma^5$
  return $H(\sigma_1 \oplus \sigma_2)$ $\quad /\!/\ \Gamma^2\text{-}\Gamma^5$
return $\perp$ $\quad /\!/\ \Gamma^3\text{-}\Gamma^5$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$ $\quad /\!/\ \Gamma^0\text{-}\Gamma^2$
return K' $\quad /\!/\ \Gamma^0\text{-}\Gamma^2$

$H(m)$

if $\exists h$ s.t. $(m,h) \in \mathscr{L}_H$:
  return h
if $m = (\sigma_1^* \oplus \sigma_2^*)$: $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
  $\mathsf{chal}^2 \leftarrow$ **true** $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
  abort $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
$h \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{(m,h)\}$
return h

$G(m)$

if $\exists g$ s.t. $(m,g) \in \mathscr{L}_G$:
  return g
if $m = (1,\sigma_1^*,\sigma_2^*)$ **or** $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
    $m = (2,\sigma_1^*,\sigma_2^*)$: $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
  $\mathsf{chal}^1 \leftarrow$ **true** $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
  abort $\quad /\!/\ \Gamma^4\text{-}\Gamma^5$
$g \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_G \leftarrow \mathscr{L}_G \cup \{(m,g)\}$
if parse $m = (i,\sigma_1,\sigma_2)$ succeeds : $\quad /\!/\ \Gamma^1\text{-}\Gamma^6$
  if $\mathsf{Dec}_i(\mathsf{sk}_i,\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g)) \neq \sigma_i$: $\quad /\!/\ \Gamma^1\text{-}\Gamma^5$
    abort $\quad /\!/\ \Gamma^1\text{-}\Gamma^5$
if m queried by $\mathscr{A}$: $\quad /\!/\ \Gamma^1\text{-}\Gamma^5$
  $\mathscr{L}_{\mathscr{A}} \leftarrow \mathscr{L}_{\mathscr{A}} \cup \{(m,g)\}$ $\quad /\!/\ \Gamma^1\text{-}\Gamma^5$
return g

Figure 4.8: Sequence of games for the proof of Theorem 4.3.2.

such that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{KEM}}(\mathscr{A}) \leq &(q_D + q_G + 1) \cdot \sum_{i=1}^{n} \delta_i + q_D \cdot \sum_{i=1}^{n} 2^{-\gamma_i} \\ &+ (q_G + q_H) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \ldots, \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_n}(\mathscr{B}_n)\}\end{aligned}$$

where $\mathscr{B}_1,\ldots,\mathscr{B}_n$ run in about the same time as $\mathscr{A}$ and make the same number of queries.

Proof. The proof is exactly the same as the one for the security of $\mathsf{UT}_\parallel$ with two PKEs except we consider n schemes. In particular, the probability of having a correctness or spreadness error in some query is upper bounded by $\sum_{i=1}^n \delta_i$ and $\sum_{i=1}^n 2^{-\gamma_i}$, respectively. Also, the reductions $\mathscr{B}_i$ from the OW-CPA security of the PKEs work the same, as an adversary $\mathscr{B}_i$ picks all σ_j^* s.t. $j \neq i$. That is, if $(i,\sigma_1^*,\ldots,\sigma_n^*)$ is queried, $\mathscr{B}_i$ can recover σ_i^*, otherwise we can replace the deterministic coins by random ones. Similarly, if $\sigma^* = \oplus_j^n \sigma_j^*$ is queried by the adversary to H, $\mathscr{B}_i$ can recover σ_i^* by computing $\sigma^* \oplus_{j\neq i} \sigma_j^*$. $\square$

$\mathscr{B}_k(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$ \mathscr{M}_1 \times \mathscr{M}_2$
$\mathsf{coins}_1 \leftarrow G(1, \sigma_1^*, \sigma_2^*); \mathsf{coins}_2 \leftarrow G(2, \sigma_1^*, \sigma_2^*)$
$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort**
$b \leftarrow\$ \{0,1\}$
$K_0 \leftarrow H(\sigma_1^* \oplus \sigma_2^*)$
$K_1 \leftarrow\$ \mathscr{K}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

$G(m)$

if $\exists g$ s.t. $(m, g) \in \mathscr{L}_G$:
  return g
$g \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_G \leftarrow \mathscr{L}_G \cup \{(m, g)\}$
if parse $m = (i, \sigma_1, \sigma_2)$ succeeds :
  if $\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$:
    abort
return g

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathscr{L}_G$ s.t. $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
    and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathscr{L}_G$
      s.t. $\mathsf{Enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$:
  if k-th query : **abort**
  return $H(\sigma_1 \oplus \sigma_2)$
if k-th query :
  $(\sigma_1', \sigma_2') \leftarrow (\mathsf{Dec}_1(\mathsf{sk}_1, \mathsf{ct}_1), \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_2))$
  for i s.t. $\nexists((i, \sigma_1, \sigma_2), g_i) \in \mathscr{L}_G$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g_i) = \mathsf{ct}_i$:
    $g_i' \leftarrow G(i, \sigma_1', \sigma_2')$
    if $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_1'; g_i') = \mathsf{ct}_i$: **return** 1
  abort
return $\perp$

Figure 4.9: Adversary $\mathscr{B}_k$ for the proof of Lemma 4.3.1.

Security in the QROM

As the original FO-transform, our combiner could be made secure in the QROM by adding a hash in the ciphertext, this technique is often called *plaintext confirmation*. For simplicity, here we show that our $\mathsf{T}_\parallel$ transform generalised to n PKEs is secure in the QROM (it outputs an OW-PCA scheme). We call this transform $\mathsf{T}_\parallel^n$ and it is detailed in Figure 4.11. Then, it suffices to combine the $\mathsf{QU}_m^\perp$ transform from Hofheinz et al. [HHK17] (see Figure 2.17) with $\mathsf{T}_\parallel$ to get an IND-CCA secure KEM in the QROM. We show the following theorem.

Theorem 4.3.4. *Let* PKE *be the PKE resulting from applying* $\mathsf{T}_\parallel^n$ *on* $\mathsf{PKE}_1, \ldots, \mathsf{PKE}_n$*, which are respectively* $\delta_1, \ldots, \delta_n$*-correct. In addition, let* G_i *be hash functions modelled as (independent) quantum random oracles. Then, for all quantum OW-PCA adversary* $\mathscr{A}$ *making at most* q_G *queries to all oracles* G_i *and* q_P *queries to the plaintext-checking oracle, there exist adversaries*

$\mathscr{B}_1^{\mathscr{A},G}(\mathsf{pk}_1, \mathsf{ct}_1^*)$

$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\$\ \mathsf{Gen}_2(1^\lambda)$
$\sigma_2^* \leftarrow\!\$\ \mathscr{M}_2$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}, \sigma_2^*)$
$K \leftarrow\!\$\ \mathscr{K}$
run $\mathscr{A}^{\mathscr{O}_2^{\mathsf{Dec}}, G, H}((\mathsf{pk}_1, \mathsf{pk}_2), (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K)$
sample a random query (m, h) from $\mathscr{L}_G \cup \mathscr{L}_H$
if $(m, h) \in \mathscr{L}_G$:
  parse $((i, \mathsf{pt}_1', \sigma_2^*), g) \leftarrow m$
  return pt_1'
if $(m, h) \in \mathscr{L}_H$:
  return $\sigma_2^* \oplus m$

Figure 4.10: OW-CPA adversary for the proof of Theorem 4.3.2. The oracles G and H are simulated by $\mathscr{B}_1$.

$\mathsf{Gen}(1^\lambda)$

for $i \in [n]$:
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{Gen}_i(1^\lambda)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk}, (\mathsf{pt}_1, \ldots, \mathsf{pt}_n))$

parse $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) \leftarrow \mathsf{pk}$
for $i \in [n]$:
  $r_i \leftarrow G_i(\mathsf{pt}_1, \ldots, \mathsf{pt}_n)$
  $\mathsf{ct}_i \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i, \mathsf{pt}_i; r_i)$
return $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$

$\mathsf{Dec}(\mathsf{sk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_n))$

parse $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{sk}$
for $i \in [n]$:
  $\mathsf{pt}_i' \leftarrow \mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct}_i)$
for $i \in [n]$:
  $r_i \leftarrow G_i(\mathsf{pt}_1', \ldots, \mathsf{pt}_n')$
  if $\mathsf{Enc}_i(\mathsf{pk}_i, \mathsf{pt}_i'; r_i) \neq \mathsf{ct}_i$:
    return $\perp$
return $(\mathsf{pt}_1', \ldots, \mathsf{pt}_n')$

Figure 4.11: $\mathsf{T}_\parallel^n$ combiner.

$\mathscr{B}_1, \ldots, \mathscr{B}_n$ such that

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{A}) &\leq \left(8 \cdot (1 + q_G + q_P)^2 + 1\right) \sum_{i \in [n]} \delta_i \\ &+ (1 + 2q_P + 2q_G) \cdot \sqrt{\min\{\mathsf{Adv}_{\mathsf{PKE}_1}^{\mathrm{ow-cpa}}(\mathscr{B}_1), \ldots, \mathsf{Adv}_{\mathsf{PKE}_n}^{\mathrm{ow-cpa}}(\mathscr{B}_2)\}}\,,\end{aligned}$$

where $\mathscr{B}_1, \ldots, \mathscr{B}_n$ run in about the same time as $\mathscr{A}$ and make at most $q_G + q_P$ queries to the QROs.

Proof. We start by recalling a lemma that will be useful for the proof.

The first one, by Zhandry [Zha12], essentially states that a quantum random oracle can be efficiently simulated.

Lemma 4.3.2 (Theorem 3.1, [Zha12])**.** *No adversary limited to q_H quantum queries to an oracle $|H\rangle$ can distinguish between the case where $|H\rangle$ is a QRO and the case where $|H\rangle$ is a $2q_H$-wise* *independent function.*

We now proceed with the proof. The sequence of hybrid games used is detailed in Figure 4.12. The adversary has access to the n different QROs $|G_1,\dots,G_n\rangle$ which can be defined as one oracle $|G\rangle = |G_1,\dots,G_n\rangle$. We assume that the message spaces $\mathcal{M}_i$ are equal to $\{0,1\}^{\ell_i}$ for some integer ℓ_i and that $G_j : \{0,1\}^* \mapsto \{0,1\}^k$.

Game Γ^0: This is the original OW-PCA game in the QROM except we enforce correctness of the challenge ciphertext (i.e. $\mathsf{Dec}(\mathsf{pk},\mathsf{ct}^*) = \mathsf{pt}^*$). As the correctness is broken for ct^* if it is broken for at least one of the components ct_i^*, the probability of that happening is at most $\sum_{i\in[n]}\delta_i$.

Game Γ^1: In this game, we simulate the plaintext-checking oracle by checking whether $\mathsf{Enc}_j(\mathsf{pt}_j; G_j(\mathsf{pt}_1,\dots,\mathsf{pt}_j)) = \mathsf{ct}_j$ for all $j \in [n]$. As seen in the proof of Theorem 4.3.1, the simulation is not perfect iff one of the $(\mathsf{pt}_1,\dots,\mathsf{pt}_n)$ queried is such that the correctness is broken, i.e. $\mathsf{Dec}_j(\mathsf{Enc}_j(\mathsf{pt}_j; G_j(\mathsf{pt}_1,\dots,\mathsf{pt}_n))) \neq \mathsf{pt}_j$ for some $j \in [n]$ at any point in the game. We call this event fail_j. One can see that fail_i occurs if one can find a correctness error in the scheme generated by $\mathsf{T}(\mathsf{PKE}_i)$. By Theorem 2.4.3, this happens with probability at most $8\cdot\delta_i\cdot(q_G+1)^2$, where q_G is the number of calls made to the random oracle, which in our case is $q_G + q_P$.

Overall, we have

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \Pr\left[\cup_{j\in[n]}\mathsf{fail}_j\right] \leq 8\cdot(1+q_G+q_P)^2 \cdot \sum_{j\in[n]}\delta_j\ ,$$

where the second inequality follows from a union bound.

Game Γ^2: In game Γ^2, we replace the deterministic coins $G_j(\mathsf{pt}_1^*,\dots,\mathsf{pt}_n^*)$ by random coins $r_j \leftarrow\$ \{0,1\}^r$ for all $j \in [n]$. We can then use the One-Way to Hiding Lemma (Lemma 2.3.1) to upper bound $|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]|$. First, we consider the RO $G := G_1 \otimes \ldots \otimes G_n$ s.t. $G(m) = (G_1(m),\dots,G_n(m))$ and the function $\mathsf{F}(x,y)$ shown in Figure 4.13 which outputs $inp = (\mathsf{pk},\mathsf{ct}^*)$. In addition, let $\mathcal{A}'$ be the adversary that receives inp, run $\mathcal{A}^{\mathcal{O}_1^{\mathsf{PCO}}}(\mathsf{pk},\mathsf{pt}^*)$ by simulating the plaintext-checking oracle (this is possible since the secret key is not used in $\mathcal{O}_1^{\mathsf{PCO}}$ anymore) and outputs 1 iff $\mathcal{A}$ outputs pt' s.t. $\mathsf{Enc}(\mathsf{pk},\mathsf{pt}') = \mathsf{ct}^*$ (this is equivalent to $\mathsf{pt}' = \mathsf{pt}^*$ by the perfect correctness of the challenge ciphertext). By the AOW2H Lemma (Lemma 2.3.1), one can easily see that

$$\begin{aligned}
&\left|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\right| = \\
&\Big|\Pr[\mathcal{A}'^{|G\rangle}(inp) \Rightarrow 1 | \mathsf{pt}^* \leftarrow\$ \{0,1\}^{\ell_1+\ldots+\ell_n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, G(\mathsf{pt}^*))] \\
&- \Pr[\mathcal{A}'^{|G\rangle}(inp) \Rightarrow 1 | (\mathsf{pt}^*, r^*) \leftarrow\$ \{0,1\}^{\ell_1+\ldots+\ell_n+r\cdot n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, r^*)]\Big| \\
&\leq 2q_{ow2h}\sqrt{\Pr[\mathsf{pt}^* \leftarrow \mathsf{Ext}^{\mathcal{A}',|G\rangle}(inp) | (\mathsf{pt}^*, r^*) \leftarrow\$ \{0,1\}^{\ell_1+\ldots+\ell_n+r\cdot n}; inp \leftarrow \mathsf{F}(\mathsf{pt}^*, r^*)}\ ,
\end{aligned}$$

$\Gamma^i(\mathscr{A})$

$((\mathsf{pk}_1,\ldots,\mathsf{pk}_n),(\mathsf{sk}_1,\ldots,\mathsf{sk}_n)) \leftarrow\!\!\$\; \mathsf{Gen}(1^\lambda)$
$(\mathsf{pt}_1^*,\ldots,\mathsf{pt}_n^*) \leftarrow\!\!\$\; \{0,1\}^{\ell_1+\ldots+\ell_n}$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk},(\mathsf{pt}_1^*,\ldots,\mathsf{pt}_n^*))$
for $i \in [n]$: $\quad /\!\!/\ \Gamma^2$
$\quad (r_1^*,\ldots,r_n^*) \leftarrow\!\!\$\; \{0,1\}^{rn} \quad /\!\!/\ \Gamma^2$
$\quad \mathsf{ct}_i^* \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i,\mathsf{pt}_i^*;r_i^*) \quad /\!\!/\ \Gamma^2$
if $\mathsf{Dec}(\mathsf{sk}^*,\mathsf{ct}^*) \neq (\mathsf{pt}_1^*,\ldots,\mathsf{pt}_n^*)$:
$\quad$ **abort**
$\mathsf{pt}' \leftarrow \mathscr{A}^{|G_1,\ldots,G_n\rangle,\mathscr{O}_0^{\mathsf{PCO}}}(\mathsf{pk},\mathsf{ct}^*) \quad /\!\!/\ \Gamma^0$
$\mathsf{pt}' \leftarrow \mathscr{A}^{|G_1,\ldots,G_n\rangle,\mathscr{O}_1^{\mathsf{PCO}}}(\mathsf{pk},\mathsf{ct}^*) \quad /\!\!/\ \Gamma^1\text{-}\Gamma^2$
return $1_{\mathsf{pt}'=(\mathsf{pt}_1^*,\ldots,\mathsf{pt}_n^*)}$

Oracle $\mathscr{O}_0^{\mathsf{PCO}}((\mathsf{pt}_1,\ldots,\mathsf{pt}_n),(\mathsf{ct}_1,\ldots,\mathsf{ct}_n))$

$(\mathsf{pt}_1',\ldots,\mathsf{pt}_n') \leftarrow \mathsf{Dec}(\mathsf{sk},\mathsf{ct})$
return $1_{(\mathsf{pt}_1,\ldots,\mathsf{pt}_n)=(\mathsf{pt}_1',\ldots,\mathsf{pt}_n')}$

Oracle $\mathscr{O}_1^{\mathsf{PCO}}((\mathsf{pt}_1,\ldots,\mathsf{pt}_n),(\mathsf{ct}_1,\ldots,\mathsf{ct}_n))$

for $j \in [n]$:
$\quad r_j \leftarrow G_j(\mathsf{pt}_1,\ldots,\mathsf{pt}_n)$
$\quad$ **if** $\mathsf{Enc}_j(\mathsf{pk}_j,\mathsf{pt}_j;r_j) \neq \mathsf{ct}_j$:
$\quad\quad$ **return** 0
return 1

Figure 4.12: Sequence of games for the proof of Theorem 4.3.4.

$\mathsf{F}(\mathsf{pt}^*,r^*)$

parse $(r_1^*,\ldots,r_n^*) \leftarrow r^*$
parse $(\mathsf{pt}_1^*,\ldots,\mathsf{pt}_n^*) \leftarrow \mathsf{pt}^*$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\; \mathsf{Gen}(1^\lambda)$
for $i \in [n]$:
$\quad \mathsf{ct}_i^* \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i,\mathsf{pt}_i^*;r_i^*)$
return $(\mathsf{pk},\mathsf{ct}^*)$

Figure 4.13: Function F for applying the AOW2H Lemma in the proof of Theorem 4.3.4.

where Ext is the extractor defined in Figure 2.12 and q_{ow2h} is the number of queries made by $\mathscr{A}'$ to G, which is q_G to answer $\mathscr{A}$'s queries plus q_P to simulate the plaintext-checking oracle (i.e. one can compute the coins $(G_j(\mathsf{pt}_1,\ldots,\mathsf{pt}_n))_{j\in[n]}$ with one quantum query to G). Thus, $q_{ow2h} = (q_P + q_G)$. Now, the probability that the extractor outputs pt^* is precisely the probability that the OW-CPA property of all underlying PKE_i is broken. We provide in Figure 4.14 an adversary $\mathscr{B}_j$ that breaks the OW-CPA security of any PKE_j given $\mathsf{Ext}^{\mathscr{A}'}$. Thus, we have

$$\left|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\right| \le 2(q_P + q_G)\sqrt{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathscr{B}_j)}$$

for any $j \in [n]$. Finally, $\Pr[\Gamma^2 \Rightarrow 1]$ is the probability to win the OW-CPA game against any underlying PKE_j. We provide the given adversary $\mathscr{C}_j$ that breaks PKE_j in Figure 4.14. Hence, $\Pr[\Gamma^2 \Rightarrow 1] \le \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathscr{C}_j) \le \sqrt{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_j}(\mathscr{C}_j)}$. Collecting the bounds and folding adversaries concludes the proof. □

This result then implies that $\mathsf{QU}_m^{\perp} \circ \mathsf{T}_{\|}^{n}$ is a robust FO-like combiner in the QROM by using Theorem 4.5 of Hofheinz et al. [HHK17]. Note that the proof of Theorem 4.3.4 is very similar

$\mathscr{B}_1^{\mathsf{Ext}^{\mathscr{A}'},G}(\mathsf{pk}_j,\mathsf{ct}_j^*)$

$\textbf{for } i \in [n], i \neq j:$
$\quad (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow\$ \mathsf{Gen}_i(1^\lambda)$
$\quad \mathsf{pt}_i^* \leftarrow\$ \{0,1\}^{\ell_i}$
$\quad \mathsf{ct}_i^* \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i,\mathsf{pt}_i^*)$
$\mathsf{pt}' \leftarrow \mathsf{Ext}^{\mathscr{A}'}((\mathsf{pk}_1,\ldots,\mathsf{pk}_n),(\mathsf{ct}_1^*,\ldots,\mathsf{ct}_n^*))$
$\textbf{return } \mathsf{pt}_j'$

$\mathscr{C}_1'^{\mathscr{A},G}(\mathsf{pk}_j,\mathsf{ct}_j^*)$

$\textbf{for } i \in [n], i \neq j:$
$\quad (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow\$ \mathsf{Gen}_i(1^\lambda)$
$\quad \mathsf{pt}_i^* \leftarrow\$ \{0,1\}^{\ell_i}$
$\quad \mathsf{ct}_i^* \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i,\mathsf{pt}_i^*)$
$\mathsf{pt}' \leftarrow \mathscr{A}^{\mathscr{O}_1^{\mathsf{PCO}},|G_1,\ldots,G_n\rangle}((\mathsf{pk}_1,\ldots,\mathsf{pk}_n),(\mathsf{ct}_1^*,\ldots,\mathsf{ct}_n^*))$
$\textbf{return } \mathsf{pt}_j'$

Figure 4.14: OW-CPA adversaries for the proof of Theorem 4.3.4. The oracle $\mathscr{O}_1^{\mathsf{PCO}}$ is defined as in Figure 4.12.

to the proofs of FO security in the QROM. As a result, the bound could much likely be made tighter using recent QROM techniques (e.g. [Bin+19b; Kuc+20; SXY18]). In addition, we conjecture that our main combiner $\mathsf{UT}_\parallel$ could be proven secure in the QROM without the additional hash. We leave these improvements as future work.

4.4 Other Combiners

It has been shown that the implementation of ROs in FO-like transforms, in particular in the de-randomisation step (i.e. computation of the deterministic coins), is particularly vulnerable to implementation mistakes [BDG20]. Thus, it is of interest to study how these coins can be computed without compromising the security of the resulting scheme. We show in this section how hash functions (i.e. ROs) can be combined s.t. the de-randomisation step is secure and efficient. Many combinations of hash functions are possible and we propose a few of those below, offering flexibility to implementors. Finally, we consider using different hash functions to increase the security at no (or very small) cost. This relates to the notion of hash combiner [FL08; FLP08], which constructs a hash function that fulfils certain security properties as long as one of the underlying hash functions has this property. In our case, we want the hash functions to behave as random oracles, thus we can combine two different functions to make the whole scheme secure as long as *one* of the hash functions is indistinguishable from a RO.

How to combine hash functions. From now on, in order to distinguish (random) functions from random oracles, we denote a function by a small letter and a RO by a capital letter (e.g. $g(x)$ is a function evaluated on x and $G(x)$ is a RO queried on x). Note that in our case, the functions are defined using random oracles (e.g. $g(x) := G(1,x) \oplus G(2,x)$). We consider replacing the RO G in our combiners by such a random function g (but still in the ROM).

One can see from the proofs of security of both $\mathsf{T}_\parallel$ and $\mathsf{UT}_\parallel$ that we want the deterministic coins to be indistinguishable from random ones until we can recover the seeds (or plaintexts) from the list of queries. In addition to this property, one also wants the values $g(i,\sigma_1,\sigma_2)$ to be close to uniform. Indeed, in the proof of Theorem 4.3.2, we extensively use the fact that the correctness and spreadness property hold with probability at least δ and $2^{-\gamma}$, respectively, even when the coins are not random but computed as $g(i,\sigma_1,\sigma_2)$. Obviously, if the values $g(1,\sigma_1,\sigma_2)$ are not sampled uniformly at random, this may not hold anymore. In other words, we want $g(i,\sigma_1,\sigma_2)$ to be either computable by the adversary using its queries to G or distributed uniformly at random. We present below formal definitions (called *Extractable Random Function (ERF)* and *Indistinguishable unless Queried (IUQ)*) capturing these properties.

4.4.1 Extractable Random Functions (ERFs)

We start by introducing the notion of Extractable Random Functions (ERFs), which formalise the fact that the coins computed as $g(i,\sigma_1,\sigma_2)$ should look uniform given the adversary's view, or an extractor can be used to recover both seeds σ_1,σ_2.

We first define the notion of extractor.

Definition 4.4.1 (Extractor)**.** *Let g be a random function defined using a random oracle G. An extractor Ext_g for a function g is a ppt deterministic function that takes a set of tuples $\mathscr{L}_G = \{(x_i,h_i)\}_{i\in[q_G]}$ of cardinality q_G defining the event $\{\wedge_{i\in[q_G]} G(x_i) = h_i\}$ and that outputs a set of q_E tuples $\mathscr{E} = \{((i^j,\sigma_1^j,\sigma_2^j),g^j)\}_{j\in[q_E]}$ s.t.*

1. *(correctness)* $\Pr\left[g(i^j,\sigma_1^j,\sigma_2^j) = g^j \middle| \mathscr{L}_G\right] = 1, \forall j \in [q_E]$.
2. *(initial emptiness)* $\mathsf{Ext}_g(\emptyset) = \emptyset$.
3. *(increasing)* $\mathscr{L}_G \subseteq \mathscr{L}'_G \Rightarrow \mathsf{Ext}_g(\mathscr{L}_G) \subseteq \mathsf{Ext}_g(\mathscr{L}'_G)$.
4. *(initial queries) Let $\mathscr{L}_G^*$ be the set of queries/responses made when computing $g(1,\sigma_1,\sigma_2)$ and $g(2,\sigma_1,\sigma_2)$. Then,*

$$\mathsf{Ext}_g(\mathscr{L}_G^*) = \left\{\left((1,\sigma_1,\sigma_2),g(1,\sigma_1,\sigma_2)\right),\left((2,\sigma_1,\sigma_2),g(2,\sigma_1,\sigma_2)\right)\right\} .$$

   That is, one call to the function $g(i,\sigma_1,\sigma_2)$ (for different i's) does not give away any information on other values of g.

Note that the number of tuples output by the extractor q_E is a function of q_G, that is the number of queries made to the RO G. In addition, we define q_E^1 as the maximum number of tuples of the form (i,σ_1,σ_2) with a fixed σ_1 (or σ_2) output by the extractor.

Now, we can define the notion of extractable random functions.

Definition 4.4.2 (Extractable Random Function (ERF))**.** *Let $g : \{0,1\}^* \mapsto \{0,1\}^n$ be a (random) function defined using a random oracle G. Let $\mathscr{I}_{\sigma_1,\sigma_2} = \{((i,\sigma'_1,\sigma'_2),g(i,\sigma'_1,\sigma'_2)) : \sigma'_1 \neq \sigma_1,\sigma'_2 \neq \sigma_2\}$ be the set of input/output tuples of g for values σ'_1 and σ'_2 different from σ_1 and σ_2,*

$g(i,\sigma_1,\sigma_2)$
$G(\sigma_1 \oplus \sigma_2) \oplus G(i,\sigma_i)$
$G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$

Table 4.1: Different g functions, where G, G_i are ROs.

where each tuple $(x, y) \in \mathscr{J}_{\sigma_1,\sigma_2}$ defines the event $\{g(x) = y\}$. Then, g is an extractable random function (ERF) if there exists an extractor Ext_g s.t. for any $i, \sigma_1, \sigma_2, y, \mathscr{L}_G$ and $\mathscr{J}' \subseteq \mathscr{J}_{\sigma_1,\sigma_2}$ s.t. $\Pr[\mathscr{J}', \mathscr{L}_G] > 0$,

$$\Pr[g(i,\sigma_1,\sigma_2) = y | \mathscr{L}_G, \mathscr{J}'] = \begin{cases} \frac{1}{2^n}, & \textit{if } ((i,\sigma_1,\sigma_2), y) \notin \mathsf{Ext}_g(\mathscr{L}_G) \\ 1, & \textit{if } ((i,\sigma_1,\sigma_2), y) \in \mathsf{Ext}_g(\mathscr{L}_G) \\ 0, & \textit{if } \exists y' \neq y \textit{ s.t. } ((i,\sigma_1,\sigma_2), y) \in \mathsf{Ext}_g(\mathscr{L}_G) \end{cases}$$

In short, as hinted above, this notion captures the fact that either $g(i,\sigma_1,\sigma_2)$ is uniformly distributed, or the extractor can compute it based on the queries made to G. In addition, we require that there is no correlation between different values of g when both inputs are different. Finally, we stress that when a party computes $g(i,\sigma_1,\sigma_2)$, the value of g becomes deterministic. In other words, if we let $\mathscr{L}_G$ be the set of corresponding queries/responses used to compute $g(i,\sigma_1,\sigma_2)$, the list $\mathsf{Ext}_g(\mathscr{L}_G \cup \mathscr{L}'_G)$ will contain $g(i,\sigma_1,\sigma_2)$, for any $\mathscr{L}'_G$.

Example 4.4.1 (ROs are *ERF* functions.)**.** *As an example, we show that ROs are ERFs. More precisely, let $g(i,\sigma_1,\sigma_2) = G(i,\sigma_1,\sigma_2)$ as in the $\mathsf{UT}_{\|}$ combiner. Then, we define the extractor Ext_g as a function that takes all tuples of the form $((i,\sigma_1,\sigma_2), h) \in \mathscr{L}_G$ and outputs them. Clearly, if the extractor does not output a given value (i,σ_1,σ_2), then it was not queried to the RO and it is indistinguishable from a uniform value, as requested. Also, by the property of ROs, a value $G(i,\sigma_1,\sigma_2)$ is mutually independent from any set of values $G(i',\sigma'_1,\sigma'_2)$ with $\sigma_1 \neq \sigma'_1$ and $\sigma_2 \neq \sigma'_2$. Note also that the maximum number of tuples output by the extractor q_E is upper bounded by q_G.*

Other examples of ERFs We give two other examples of functions g satisfying the properties of ERF in Table 4.1.

Proposition 4.4.1. *The two functions $g(i,\sigma_1,\sigma_2)$ presented in Table 4.1 are ERFs.*

Proof.

- $G(\sigma_1 \oplus \sigma_2) \oplus G(i,\sigma_i)$: We first define the extractor Ext_g. We can define $\mathscr{L}_{G_i}$ as the list of query/responses (m, g) s.t. m is of the form (i,σ_i) and $\mathscr{L}_G$ is the list of remaining query responses. The extractor outputs the union of:

1. $\{((1,\sigma_1,\sigma_1 \oplus \sigma), g \oplus g_1) : (\sigma, g) \in \mathscr{L}_G, ((1,\sigma_1), g_1) \in \mathscr{L}_{G_1}\}$. That is for each $(1,\sigma_1)$ that has been queried, it recovers σ_2 from the queries in $\mathscr{L}_G$ and outputs the corresponding value of $g(1,\sigma_1,\sigma_2)$.
2. $\{((2,\sigma_2 \oplus \sigma,\sigma_2), g \oplus g_2) : (\sigma, g) \in \mathscr{L}_G, ((2,\sigma_2), g_2) \in \mathscr{L}_{G_2}\}$.

This is straightforward to see that this function fulfils the properties of an extractor. In particular, if q_G queries are made to G, we have $q_E \le q_G^2$ and $q_E^1 \le q_G$. Finally, we show that if $g(i,\sigma_1,\sigma_2)$ is not in the output of the extractor, then it is indistinguishable from a value sampled uniformly at random. Let $g(i,\sigma_1,\sigma_2) = Y + X$ with $Y := G(\sigma_1 \oplus \sigma_2)$ and $X := G(i,\sigma_i)$. Clearly, by the property of RO, if $\sigma_1 \oplus \sigma_2$ or (i,σ_i) was not queried to G, we have Y, resp. X uniformly distributed. Then, $g(i,\sigma_1,\sigma_2)$ is uniformly distributed as well. Finally, if both are queried, the extractor will be able to compute $g(i,\sigma_1,\sigma_2)$.

- $G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$: We define the extractor as follows. For a given i, let $\mathscr{L}_{G_{ij}}$ be the list of query/answer for queries of the type (i,σ_j) to G_j. For any i (here $i \in [2]$), the extractor considers all pairs of tuples $((i,\sigma_1), g_1), ((i,\sigma_2), g_2) \in \mathscr{L}_{G_{i1}} \times \mathscr{L}_{G_{i2}}$ and for each of them outputs $((i,\sigma_1,\sigma_2), g_1 \oplus g_2)$. Clearly, such an extractor fulfils the necessary properties. Now, we show that $g(i,\sigma_1,\sigma_2)$ is distributed uniformly at random unless the extractor outputs a corresponding tuple. For a given i, let $X_j := G_j(i,\sigma_j)$ and $Z := X_1 + X_2$. Then, following a similar argument as in the previous point, we see that Z is uniform unless (i,σ_1) and (i,σ_2) have been queried to G_1 and G_2, respectively. If that happens, the extractor recovers $g(i,\sigma_1,\sigma_2)$. Finally, as in the previous function, we have $q_E \le q_G^2$ and $q_E^1 \le q_G$.

□

4.4.2 IUQ functions

Now we define a weaker assumption than ERF for the hash function h that derives the key in the encapsulation/decapsulation procedures. Indeed, we notice that the only property we need from this function is to look indistinguishable unless one can recover one challenge seed given the other. We call such property Indistinguishability unless Queried (IUQ) and we define it as follows.

Definition 4.4.3 (IUQ functions)**.** *Let $h(\sigma_1 \in \mathscr{M}_1, \sigma_2 \in \mathscr{M}_2)$ be a (random) function based on a random oracle H where $\mathscr{M}_1$ and $\mathscr{M}_2$ are some message spaces. We consider the* IUQ *game defined in Figure 4.15, where the RO H is defined as shown in the game. Then, if there exists a ppt function* Ext_h *s.t. for any efficient adversary $\mathscr{A}$*

$$\mathsf{Adv}^{\mathrm{iuq}}_{h,H,\mathsf{Ext}_h}(\mathscr{A}) = \left|\Pr\left[\mathrm{IUQ}^1_{h,H,\mathsf{Ext}_h}(\mathscr{A}) \Rightarrow 1\right] - \Pr\left[\mathrm{IUQ}^0_{h,H,\mathsf{Ext}_h}(\mathscr{A}) \Rightarrow 1\right]\right| = 0$$

we say h is IUQ *(Indistinguishable from Uniform unless Queried).*

$\text{IUQ-}^{b}_{h,\mathscr{H},\mathsf{Ext}_h}(\mathscr{A})$	$H(m)$
$\mathsf{chal}^1 \leftarrow \mathbf{false}$	$\mathbf{if}\ \exists x \text{ s.t. } (m,x) \in \mathscr{L}_H:$
$\mathsf{chal}^2 \leftarrow \mathbf{false}$	$\quad \mathbf{return}\ x$
$(\sigma_1,\sigma_2) \leftarrow\$\ \mathscr{M}_1 \times \mathscr{M}_2$	$x \leftarrow\$\ \{0,1\}^n$
$h_0 \leftarrow\$\ \{0,1\}^n$	$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{(m,x)\}$
$h_1 \leftarrow h(\sigma_1,\sigma_2)$	$\mathbf{if}\ \sigma_1 \in \mathsf{Ext}_h(1,\sigma_2,\mathscr{L}_H):\ \mathsf{chal}^1 \leftarrow \mathbf{true}$
$b' \leftarrow \mathscr{A}^H(\sigma_1,\sigma_2,h_b)$	$\mathbf{if}\ \sigma_2 \in \mathsf{Ext}_h(2,\sigma_1,\mathscr{L}_H):\ \mathsf{chal}^2 \leftarrow \mathbf{true}$
$\mathbf{return}\ b'$	$\mathbf{if}\ \mathsf{chal}^1\ \mathbf{and}\ \mathsf{chal}^2: \mathbf{abort}$
	$\mathbf{return}\ x$

Figure 4.15: IUQ game.

While looking cumbersome, this definition simply generalises what we want from the functions that derives the key. Indeed, in the IUQ game, we ask the adversary to distinguish between a uniformly distributed value and $h(\sigma_1,\sigma_2)$ for some random (σ_1,σ_2). However, if there exists some extractor (or parsing function) Ext_h that can recover (σ_1,σ_2) by observing the queries to the random oracles, the game aborts. That captures the fact that either the adversary cannot distinguish, or one can recover the challenge seeds (or plaintexts). Note that the function Ext_h takes the index of the seeds it must recover and the other seed to capture the fact that in a reduction attacking the one-wayness of PKE_1, the adversary can pick σ_2 (and the other way around).

Example 4.4.2 ($\mathsf{H}(\sigma_1) \oplus \mathsf{H}(\sigma_2)$ **is *IUQ*.**)**.** *As an example of a IUQ function, one can consider* $h(\sigma_1,\sigma_2) := H(\sigma_1) \oplus H(\sigma_2)$*. As an extractor, we define* $\mathsf{Ext}_h(i,\sigma,\mathscr{L}_H)$ *as the function that goes through all tuples* $(m,h) \in \mathscr{L}_H$ *and outputs the set of m's. Now, unless* σ_1 *and* σ_2 *are queried, the adversary cannot distinguish a random value from* $h(\sigma_1,\sigma_2)$*. But if both values are queried, the IUQ game will abort because both lists output by the extractor* $\mathsf{Ext}_h(1,\sigma_2,\mathscr{L}_H)$ *and* $\mathsf{Ext}_h(2,\sigma_1,\mathscr{L}_H)$ *will contain* σ_1 *and* σ_2*, respectively. In this case, the advantage of IUQ adversary is* 0.

4.4.3 IUQ and ERF in $\mathsf{UT}_{\parallel}$

Now, based on the IUQ and ERF definitions, we prove the following theorem, which states that the $\mathsf{UT}_{\parallel}$ combiner is still robust if G and H are replaced by ERF and IUQ functions, respectively.

Theorem 4.4.1 ($\mathsf{UT}_{\parallel}$ and ERF/IUQ)**.** *Let* $h(\sigma_1,\sigma_2)$ *and* $g(i,\sigma_1,\sigma_2)$ *be a* IUQ*, resp. ERF function, and H and G be the ROs h and g are based on, respectively. In addition, let* $q_E(|\mathscr{L}_G|)$, $q_{E_h}(|\mathscr{L}_H|)$ *be the maximum number of tuples output by* $\mathsf{Ext}_g(\mathscr{L}_G)$, $\mathsf{Ext}_h(\mathscr{L}_H)$*, respectively (they are a function of the length of the input). We also let* $q^1_E(|\mathscr{L}_G|)$ *be the maximal number of tuples with a fixed* σ *output by* $\mathsf{Ext}_g(\mathscr{L}_G)$ *(see Definition 4.4.1). Finally, let* KEM *be the hybrid KEM built on top of two OW-CPA PKEs using the* $\mathsf{UT}_{\parallel}$ *combiner, where the deterministic coins for encrypting the seed* σ_i *are computed as* $g(i,\sigma_1,\sigma_2)$ *instead of* $G(i,\sigma_1,\sigma_2)$*, and the key is computed as*

$h(\sigma_1, \sigma_2)$ instead of $H(\sigma_1 \oplus \sigma_2)$.

Then, for all efficient IND-CCA adversary $\mathscr{A}$ making at most q_G, q_H, and q_D queries to the oracles G, H, and $\mathcal{O}^{\mathsf{Dec}}$ respectively, there exist adversaries $\mathscr{B}_1$ and $\mathscr{B}_2$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{PKE}}(\mathscr{A}) \leq & q_E(q_g \cdot 2(q_D+1) + q_G) \cdot \max\{\delta_1, \delta_2\} \\ & + (q_D + q^1_E(q_G)) \cdot (2^{-\gamma_1} + 2^{-\gamma_2}) \\ & + (q^1_E(q_G) + q_{E_h}(q_H)) \cdot \min\{\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_1}(\mathscr{B}_1), \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}_2}(\mathscr{B}_2)\},\end{aligned}$$

where q_g is the number of queries to G needed to evaluate g. Both $\mathscr{B}_1$ and $\mathscr{B}_2$ run in about the same time as $\mathscr{A}$ and make the same number of queries.

Proof. The proof is very similar to the proof of security of the $\mathsf{UT}_\parallel$ transform. The only difference is that we use the output of the extractors associated with the ERF/IUQ properties of g/h instead of the list of queries $\mathscr{L}_G, \mathscr{L}_H$. In particular, we argue that if some value is not in these extracted lists, it is uniformly distributed.

Let Ext_g and Ext_h be the functions s.t. g and h are ERF and IUQ, respectively. We also assume that the key space is $\{0,1\}^n$ for some n. We give the sequence of games in Figure 4.16.

Game Γ^1: In this game, we enforce the correctness of all ciphertexts that can be computed using the function g. In particular, we abort if the challenge ciphertexts break the correctness property or if any (i, σ_1, σ_2) output by Ext_g is s.t. $\mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g(i, \sigma_1, \sigma_2))$ breaks the correctness property. Let $\mathscr{L}_G$ collect all tuples of query/responses made throughout the game by the adversary and the game itself, and $\mathscr{L}^k_G$ its state after the k-th query to G is made. Then, we can define the set of new tuples output by the extractor at query k as $\mathscr{T}^k = \mathsf{Ext}_g(\mathscr{L}^k_G) \setminus \mathsf{Ext}_g(\mathscr{L}^{k-1}_G)$. Hence, when submitting the k-th query m to G, the probability a tuple in the corresponding $\mathscr{T}^k = \mathsf{Ext}_g(\mathscr{L}^{k-1}_G \cup (m, G(m))) \setminus \mathsf{Ext}_g(\mathscr{L}^{k-1}_G)$ contains a plaintext that breaks the correctness is

$$\begin{aligned}&\Pr\left[\bigvee_{((i,\sigma_1,\sigma_2), g(i,\sigma_1,\sigma_2)) \in \mathscr{T}^k} \mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g(i,\sigma_1,\sigma_2))) \neq \sigma_i \middle| \mathscr{L}^{k-1}_G\right] \\ &\leq \sum_{((i,\sigma_1,\sigma_2), g(i,\sigma_1,\sigma_2)) \in \mathscr{T}^k} \Pr\left[\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g(i,\sigma_1,\sigma_2))) \neq \sigma_i \middle| \mathscr{L}^{k-1}_G\right] \\ &= \sum_{((i,\sigma_1,\sigma_2), g(i,\sigma_1,\sigma_2)) \in \mathscr{T}^k} \Pr\left[\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; \mathsf{coins})) \neq \sigma_i : \mathsf{coins} \leftarrow\!\!\$\, \{0,1\}^n\right] \\ &\leq |\mathscr{T}^k| \cdot \max\{\delta_1, \delta_2\}\end{aligned}$$

for any query m. The equality follows from the definition of extractable random functions and the last inequality from the δ_i correctness of PKE_i. Then, by a union bound, the probability a correctness error happens for any of the q_E tuples output by Ext_g is upper bounded by $q_E \cdot \max\{\delta_1, \delta_2\}$. Note that q_E is a function of the total number of queries submitted to G, which is $q_g \cdot 2(q_D+1) + q_G$ in this case, where q_g is the number of queries made to G at each evaluation of g (i.e. in total 2 calls to g for the challenge ciphertexts and for each decapsulation query, and q_G queries made by the adversary). Hence, we have

$$\left|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]\right| \leq q_E \cdot \max\{\delta_1, \delta_2\}\,.$$

<u>Game Γ^2</u>: In this game, we consider $\mathcal{L}_G$, which is the transcript of the queries made to G by any party (i.e. game or adversary). We enforce that at no point the extractor $\mathsf{Ext}_g(\mathcal{L}_G)$ contains a tuple $((1,\sigma_1^*,\sigma_2), g_1')$ with $\sigma_2 \neq \sigma_2^*$ or $((2,\sigma_1,\sigma_2^*), g_2')$ with $\sigma_1 \neq \sigma_1^*$ s.t. $\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*; g_1') = \mathsf{ct}_1^*$ or $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2^*; g_2') = \mathsf{ct}_2^*$, respectively. We proceed as in the previous game and let $\mathcal{T}^k = \mathsf{Ext}_g(\mathcal{L}_G^k) \setminus \mathsf{Ext}_g(\mathcal{L}_G^{k-1})$ where $\mathcal{L}_G^k$ is the state of $\mathcal{L}_G^k$ after the k-th query to G. Finally, let $\mathcal{L}_G^0$ be the state of $\mathcal{L}_G$ after computing the challenge ciphertexts. Then, when submitting the k-th ($k \geq 2$) query m to G, the probability a tuple in $\mathcal{T}^k$ breaks the first condition is

$$\begin{aligned}
&\Pr\left[\bigvee_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*; g(1,\sigma_1^*,\sigma_2)) = \mathsf{ct}_1^* \,\middle|\, \mathcal{L}_G^{k-1}\right] \\
&\leq \sum_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \Pr\left[\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*; g(1,\sigma_1^*,\sigma_2))) = \mathsf{ct}_1^* \,\middle|\, \mathcal{L}_G^{k-1}\right] \\
&= \sum_{((1,\sigma_1^*,\sigma_2),g(1,\sigma_1^*,\sigma_2))\in\mathcal{T}^k} \Pr[\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*; \mathsf{coins})) = \mathsf{ct}_1^* : \mathsf{coins} \leftarrow\!\!\$\, \{0,1\}^n] \\
&\leq \left|\left\{\left((1,\sigma_1^*,\sigma_2), g\right) \in \mathcal{T}^k\right\}\right| \cdot 2^{-\gamma_1}
\end{aligned}$$

for any m, where the equality holds by the definition of ERF and the fact that by definition a tuple in $\mathcal{T}^k$ is not in $\mathcal{L}_G^{k-1}$. Now, the equation holds for $k = 1$ as well by the last property of extractors (i.e. $(1,\sigma_1^*,\sigma_2) \notin \mathcal{L}_G^0$ for any $\sigma_2 \neq \sigma_2^*$). Then, it is similar for the second type of failure and the probability it happens for any of the q_E tuples in $\mathsf{Ext}(\mathcal{L}_g)$ is upper bounded by $q_E^1 \cdot (2^{-\gamma_1} + 2^{-\gamma_2})$, where q_E^1 is the maximum number of tuples $g(i,\sigma_1,\sigma_2)$ output by the extractor for a fixed value σ_1 or σ_2. Thus, we have

$$\left|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]\right| \leq q_E^1 \cdot (2^{-\gamma_1} + 2^{-\gamma_2})\,.$$

We also prove the following proposition.

Proposition 4.4.2. *Let* $\mathsf{ct}_i \neq \mathsf{ct}_i^*$. *In game* Γ^2, *if* $\mathcal{A}$ *submits* $(\mathsf{ct}_1, \mathsf{ct}_2^*)$ *or* $(\mathsf{ct}_1^*, \mathsf{ct}_2)$ *to the decapsulation oracle, the latter either returns* $\perp$ *or the game aborts.*

Proof. We assume w.l.o.g. that the adversary submits $(\mathsf{ct}_1, \mathsf{ct}_2^*)$. Let $\sigma_1' := \mathsf{Dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$. By the perfect correctness of ct_2^*, $\mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_2^*) = \sigma_2^*$. If $\sigma_1' = \sigma_1^*$, then $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1'; g(1,\sigma_1',\sigma_2^*)) = \mathsf{ct}_1^* \neq \mathsf{ct}_1$ and the oracle replies $\perp$. Otherwise, in the re-encryption check, the game will compute $g(1,\sigma_1',\sigma_2^*)$ with $\sigma_1' \neq \sigma_1^*$ and thus the extractor will output $((1,\sigma_1',\sigma_2^*), g(1,\sigma_1',\sigma_2^*))$ at some point. By the abort condition in game Γ^2, either $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1^*; g(1,\sigma_1',\sigma_2^*)) = \mathsf{ct}_1^*$ and the game aborts, or $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1^*; g(1,\sigma_1',\sigma_2^*)) \neq \mathsf{ct}_1^*$ and the decapsulation oracle outputs $\perp$. $\square$

<u>Game Γ^3</u>: We make nearly the exact same modifications as in game Γ^2 in the proof of The-

orem 4.3.2. That is, we check whether there exist both $((1,\sigma_1,\sigma_2),g_1)$ and $((2,\sigma_1,\sigma_2),g_2)$ in $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i)=\mathsf{ct}_i$. If this is the case, (let's call this event found) we return $h(\sigma_1,\sigma_2)$, otherwise we return the key K output by the decapsulation function. In other words, this game is the same as Γ^2 of the proof of Theorem 4.3.2 except we check for plaintexts in $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$ instead of $\mathscr{L}_{\mathscr{A}}$. Now, if found occurs, we return the same key as in game Γ^1. Indeed, by the perfect correctness of the tuples in $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})\subseteq\mathsf{Ext}_g(\mathscr{L}_G)$ enforced in game Γ^1, if we find (σ_1,σ_2) s.t. $\mathsf{Enc}_i(\sigma_i;g(i,\sigma_1,\sigma_2))=\mathsf{ct}_i$, then $\mathsf{Dec}_i(\mathsf{sk}_i,\mathsf{ct}_i)=\sigma_i$. Hence, we have, $\Pr[\Gamma^2\Rightarrow 1]=\Pr[\Gamma^3\Rightarrow 1]$.

Game Γ^4: We modify the previous game as follows. As before, this game is the same as game Γ^3 of the proof of Theorem 4.3.2 except we replace $\mathscr{L}_{\mathscr{A}}$ by $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$. In the decapsulation oracle, we simply return $\perp$ if found does not occur. Hence, game Γ^3 and Γ^4 differ iff the decapsulation oracle successfully decrypts ct but the extractor could not find neither $(1,\sigma_1',\sigma_2')$ or $(2,\sigma_1',\sigma_2')$ (i.e. at least one tuple is not in $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$), where $(\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1';g(1,\sigma_1',\sigma_2'))$, $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2';g(2,\sigma_1',\sigma_2')))=\mathsf{ct}$. Now, the ciphertexts corresponding to the seeds in $\mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$ are perfectly correct. Thus, this event is equivalent to the decapsulation oracle successfully (i.e. the re-encryption checks pass) recovering the seeds σ_1,σ_2 but either $(1,\sigma_1,\sigma_2)$ or $(2,\sigma_1,\sigma_2)$ or both were not recovered by the extractor. Let fail be this event and we prove the following lemma.

Lemma 4.4.1.

$$\Pr[\mathsf{fail}]\leq q_D\cdot(2^{-\gamma_1}+2^{-\gamma_2})\,.$$

Proof. The proof is nearly the same as the proof of Lemma 4.3.1. Let fail_k be the event that fail happens at the k-th decapsulation query and $p_k=\Pr[\mathsf{fail}_k]$. By a union bound, we have

$$\Pr[\mathsf{fail}]\leq\sum_{k=1}^{q_D}p_k\,.$$

Then, we consider an algorithm $\mathscr{B}_k$ defined in Figure 4.17, which is the same as the one defined in Figure 4.9 for Lemma 4.3.1, except the calls to G are replaced by invocations of g and the checks for values in $\mathscr{L}_G$ by checks in $\mathsf{Ext}_g(\mathscr{L}_G)$. This adversary simulates perfectly the view of $\mathscr{A}$ in game Γ^4 until the k-th query. In particular, for each decapsulation query $\mathsf{ct}=(\mathsf{ct}_1,\mathsf{ct}_2)$, it checks whether there exist both $((1,\sigma_1,\sigma_2),g_1)$ and $((2,\sigma_1,\sigma_2),g_2)$ in $\mathsf{Ext}(\mathscr{L}_G)$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i)$ for $i\in[2]$. We call this condition cond and if it is fulfilled $\mathscr{B}_k$ outputs $h(\sigma_1,\sigma_2)$, otherwise it outputs $\perp$.

In the k-th decapsulation query, if cond is fulfilled it aborts. Otherwise, it sets i s.t. there is no $((i,\sigma_1,\sigma_2),g_i)\in\mathsf{Ext}(\mathscr{L}_G)$ s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i)=\mathsf{ct}_i$. Next, it decrypts ct_1 and ct_2 to both σ_1' and σ_2'. By the definition of i and the perfect correctness of the values σ_i in $\mathsf{Ext}_g(\mathscr{L}_G)$, we have that $(i,\sigma_1',\sigma_2')\notin\mathsf{Ext}_g(\mathscr{L}_G)$. In addition, by the perfect correctness of the challenge ciphertexts and Proposition 4.4.2 we have $\sigma_1'\neq\sigma_1^*$ and $\sigma_2'\neq\sigma_2^*$. Finally, $\mathscr{B}_k$ computes $g_i'\leftarrow g(i,\sigma_1',\sigma_2')$ and outputs 1 iff $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i')=\mathsf{ct}_i$. Now, as $g_i'=g(i,\sigma_1',\sigma_2')$ is not in $\mathscr{L}_G$ and $\sigma_1'\neq\sigma_1^*,\sigma_2'\neq\sigma_2^*$, it is sampled uniformly at random. More precisely, if we fix all random coins but the ones used by G, only the responses of G and the challenge ciphertexts (which only depend on $g(i,\sigma_1^*,\sigma_2^*)$) are random. Thus, we have

$$\begin{aligned}&\Pr[\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i';g(i,\sigma_1',\sigma_2'))=\mathsf{ct}_i|\mathscr{L}_G,\mathsf{ct}^*]=\\&\Pr[\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i';\mathsf{coins})=\mathsf{ct}_i:\mathsf{coins}\leftarrow\!\!\$\,\{0,1\}^n]\le 2^{-\gamma_i}\end{aligned}$$

by the γ_i-spreadness of PKE_i and the definition of ERF. In the worst case, the check is performed for both $i=1$ and $i=2$ and thus $\Pr[\mathscr{B}_k(\mathscr{A})\Rightarrow 1]\le 2^{-\gamma_1}+2^{-\gamma_2}$. Now, we simply observe that if fail_k occurs, then $\mathscr{B}_k$ perfectly simulates the decapsulation oracle in Γ^3 and Γ^4 in the first $k-1$ queries and it will output 1 by the definition of fail_k. Thus,

$$p_k\le\Pr[\mathscr{B}_k(\mathscr{A})\Rightarrow 1]\le 2^{-\gamma_1}+2^{-\gamma_2}\,.$$

Taking the union bound on the p_k concludes the proof. □

By the previous Lemma, we have

$$\left|\Pr[\Gamma^3\Rightarrow 1]-\Pr[\Gamma^4\Rightarrow 1]\right|\le q_D\cdot(2^{-\gamma_1}+2^{-\gamma_2})\,.$$

Game Γ^5: We replace the deterministic coins used in the computation of the challenge ciphertexts by random coins and we abort if the extractor Ext_g outputs a tuple $(i,\sigma_1^*,\sigma_2^*)$ on an input m to the RO G. Let's call this event chal_g. In addition, we replace the key by a random one when $b=0$ and we raise a flag chal_h when the extractor Ext_h can recover **both** σ_1^* and σ_2^*.

One can see that as long as $\mathsf{chal}_g\cup\mathsf{chal}_h$ does not occur, the adversary cannot distinguish between the coins $g(i,\sigma_1,\sigma_2)$ and random coins, and between a real and random key. Indeed, it means the extractors $\mathsf{Ext}_g,\mathsf{Ext}_h$ cannot recover the values $g(i,\sigma_1^*,\sigma_2^*)$ and σ_1^*,σ_2^*, respectively. By the definition of ERF and IUQ this means that $g(i,\sigma_1^*,\sigma_2^*)$ is uniformly distributed and a random key is indistinguishable from $h(\sigma_1^*,\sigma_2^*)$. Then, if $\mathsf{chal}_g\cup\mathsf{chal}_h$ happens, the adversary can recover the challenge seeds and break the one-wayness properties of both ciphertexts by inspecting the values output by both extractors. We give the OW-CPA adversary $\mathscr{B}_1$ breaking PKE_1 in Figure 4.18, which wins whenever $\mathsf{chal}_g\cup\mathsf{chal}_h$ happens and it picked the correct extracted value. The adversary $\mathscr{B}_1$ picks the second seed σ_2^* at random and runs the adversary $\mathscr{A}$ with both challenge ciphertexts and a random key K, and it can simulate the decapsulation oracle perfectly as the latter does not use the secret key. Then, if $\mathsf{chal}_g\cup\mathsf{chal}_h$ happens, clearly $(i,\sigma_1^*,\sigma_2^*)$ or σ_1^* will be in the output of the extractors until the end of the game. Thus, $\mathscr{B}_1$ can recover σ_1

1. by looking for a tuple of the form (i,σ_1,σ_2^*) for some i,σ_1 in the output of Ext_g. There are at most q_E^1 such tuples, where we recall that q_E^1 is the maximum number of tuples of the form (i,σ_1,σ_2) for a fixed σ_1 or σ_2 output by the extractor.
2. by outputting a random value σ_1 in the output of $\mathsf{Ext}_h(1,\sigma_2^*,\mathscr{L}_H)$. There are at most q_{E_h}

of these values.

Hence, the probability that $\mathscr{B}_1$ wins is at least $\frac{1}{q_E^1+q_{E_h}}\Pr[\mathsf{chal}_g \cup \mathsf{chal}_h]$, as it needs to pick the correct tuple/value. On the other hand, as long as $\mathsf{chal}_g \cup \mathsf{chal}_h$ does not happen, both games are indistinguishable. Hence,

$$\left|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]\right| \leq (q_E^1 + q_{E_h}) \cdot \min\left\{\mathsf{Adv}_{\mathsf{PKE}_1}^{\mathrm{ow-cpa}}(\mathscr{B}_1), \mathsf{Adv}_{\mathsf{PKE}_2}^{\mathrm{ow-cpa}}(\mathscr{B}_2)\right\} \quad .$$

Now, since both K_0 and K_1 are uniformly distributed in Γ^5, $\Pr[G^5 \Rightarrow 1] = \frac{1}{2}$.
Collecting the probabilities and folding similar adversaries into one concludes the proof. Hence, when h is IUQ and g is ERF, Theorem 4.3.2 still holds, but with a bound that might be less tight. $\square$

$\Gamma^i(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(\sigma_1^*, \sigma_2^*) \leftarrow\$ \mathscr{M}_1 \times \mathscr{M}_2$
$\mathsf{coins}_1 \leftarrow g(1, \sigma_1^*, \sigma_2^*)$ $/\!\!/\ \Gamma^0\text{-}\Gamma^4$
$\mathsf{coins}_2 \leftarrow g(2, \sigma_1^*, \sigma_2^*)$ $/\!\!/\ \Gamma^0\text{-}\Gamma^4$
$\mathsf{coins}_1, \mathsf{coins}_2 \leftarrow\$ \mathscr{R}^2$ $/\!\!/\ \Gamma^5$
$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1^*; \mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2, \sigma_2^*; \mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{ct}_i^*) \neq \sigma_i^*$: **abort** $/\!\!/\ \Gamma^1\text{-}\Gamma^5$
$b \leftarrow\$ \{0,1\}$
$K_0 \leftarrow h(\sigma_1^*, \sigma_2^*)$ $/\!\!/\ \Gamma^0\text{-}\Gamma^4$
$K_0 \leftarrow\$ \mathscr{K}$ $/\!\!/\ \Gamma^5$
$K_1 \leftarrow\$ \mathscr{K}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, G, H}(\mathsf{pk}, (\mathsf{ct}_1^*, \mathsf{ct}_2^*), K_b)$
return $1_{b'=b}$

$H(m)$

if $\exists h$ s.t. $(m, h) \in \mathscr{L}_H$:
  return h
if $m = (\sigma_1^* \oplus \sigma_2^*)$: $/\!\!/\ \Gamma^5$
  $\mathsf{chal}^2 = $ **true** $/\!\!/\ \Gamma^5$
  abort $/\!\!/\ \Gamma^5$
$h \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{(m, h)\}$
if $\sigma_1^* \in \mathsf{Ext}_h(1, \sigma_2^*, \mathscr{L}_H)$: $/\!\!/\ \Gamma^5$
  $\mathsf{chal}_h^1 \leftarrow$ **true** $/\!\!/\ \Gamma^5$
if $\sigma_2^* \in \mathsf{Ext}_h(2, \sigma_1^*, \mathscr{L}_H)$: $/\!\!/\ \Gamma^5$
  $\mathsf{chal}_h^2 \leftarrow$ **true** $/\!\!/\ \Gamma^5$
if chal_h^1 **and** chal_h^2 : $/\!\!/\ \Gamma^5$
  abort $/\!\!/\ \Gamma^5$
return h

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2))$

$\mathsf{flag} \leftarrow$ **false**
if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if $\exists((1, \sigma_1, \sigma_2), g_1) \in \mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$
  s.t. $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1; g_1) = \mathsf{ct}_1$
  and $\exists((2, \sigma_1, \sigma_2), g_2) \in \mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$
  s.t. $\mathsf{Enc}_2(\mathsf{pk}_2, \sigma_2; g_2) = \mathsf{ct}_2$: $/\!\!/\ \Gamma^3\text{-}\Gamma^5$
  return $h(\sigma_1, \sigma_2)$ $/\!\!/\ \Gamma^3\text{-}\Gamma^5$
return $\perp$ $/\!\!/\ \Gamma^4\text{-}\Gamma^5$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$ $/\!\!/\ \Gamma^0\text{-}\Gamma^3$
return K' $/\!\!/\ \Gamma^0\text{-}\Gamma^3$

$G(m)$

if $\exists g'$ s.t. $(m, g') \in \mathscr{L}_G$: $g \leftarrow g'$
else : $g \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_G \leftarrow \mathscr{L}_G \cup \{(m, g)\}$
for $((i, \sigma_1, \sigma_2), g) \in \mathsf{Ext}_g(\mathscr{L}_G)$: $/\!\!/\ \Gamma^1\text{-}\Gamma^5$
  if $\mathsf{Dec}_i(\mathsf{sk}_i, \mathsf{Enc}_i(\mathsf{pk}_i, \sigma_i; g)) \neq \sigma_i$: $/\!\!/\ \Gamma^1\text{-}\Gamma^5$
    abort $/\!\!/\ \Gamma^1\text{-}\Gamma^5$
  if $\sigma_1 = \sigma_1^*$ **and** $\sigma_2 \neq \sigma_2^*$: $/\!\!/\ \Gamma^2\text{-}\Gamma^5$
    if $\mathsf{Enc}_1(\mathsf{pk}_1, \sigma_1; g) = \mathsf{ct}_1^*$: $/\!\!/\ \Gamma^2\text{-}\Gamma^5$
      abort $/\!\!/\ \Gamma^2\text{-}\Gamma^5$
  if $\sigma_2 = \sigma_2^*$ **and** $\sigma_1 \neq \sigma_1^*$: $/\!\!/\ \Gamma^2\text{-}\Gamma^5$
    if $\mathsf{Enc}_2(\mathsf{pk}_2, \sigma_2; g) = \mathsf{ct}_2^*$: $\Gamma^2\text{-}\Gamma^5$
      abort $/\!\!/\ \Gamma^2\text{-}\Gamma^5$
if m queried by $\mathscr{A}$
  $\mathscr{L}_{\mathscr{A}} \leftarrow \mathscr{L}_{\mathscr{A}} \cup \{(m, g)\}$
if $((1, \sigma_1^*, \sigma_2^*), g) \in \mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$
  or $((2, \sigma_1^*, \sigma_2^*), g) \in \mathsf{Ext}_g(\mathscr{L}_{\mathscr{A}})$: $/\!\!/\ \Gamma^5$
  abort $/\!\!/\ \Gamma^5$
return g

Figure 4.16: Sequence of games for the proof of Theorem 4.4.1.

$\mathscr{B}_k(\mathscr{A})$

$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$(\sigma_1^*,\sigma_2^*) \leftarrow\$ \mathscr{M}_1 \times \mathscr{M}_2$
$\mathsf{coins}_1 \leftarrow g(1,\sigma_1^*,\sigma_2^*)$
$\mathsf{coins}_2 \leftarrow g(2,\sigma_1^*,\sigma_2^*)$
$\mathsf{ct}_1^* \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1^*;\mathsf{coins}_1)$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2^*;\mathsf{coins}_2)$
if $\exists i \in [2]$ s.t. $\mathsf{Dec}_i(\mathsf{sk}_i,\mathsf{ct}_i^*) \neq \sigma_i^*$: **abort**
$b \leftarrow\$ \{0,1\}$
$K_0 \leftarrow h(\sigma_1^*,\sigma_2^*)$
$K_1 \leftarrow\$ \mathscr{K}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}},G,H}(\mathsf{pk},(\mathsf{ct}_1^*,\mathsf{ct}_2^*),K_b)$
return $1_{b'=b}$

$G(m)$

if $\exists g$ s.t. $(m,g) \in \mathscr{L}_G$:
  return g
$g \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_G \leftarrow \mathscr{L}_G \cup \{(m,g)\}$
for $((i,\sigma_1,\sigma_2),g) \in \mathsf{Ext}_g(\mathscr{L}_G)$:
  if $\mathsf{Dec}_i(\mathsf{sk}_i,\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g)) \neq \sigma_i$:
    abort
  if $\sigma_1 = \sigma_1^*$ **and** $\sigma_2 \neq \sigma_2^*$:
    if $\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1;g) = \mathsf{ct}_1^*$:
      abort
  if $\sigma_2 = \sigma_2^*$ **and** $\sigma_1 \neq \sigma_1^*$:
    if $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2;g) = \mathsf{ct}_2^*$:
      abort

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct} = (\mathsf{ct}_1,\mathsf{ct}_2))$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if $\exists((1,\sigma_1,\sigma_2),g_1) \in \mathsf{Ext}_g(\mathscr{L}_G)$ s.t. $\mathsf{Enc}_1(\mathsf{pk}_1,\sigma_1;g_1) = \mathsf{ct}_1$
    and $\exists((2,\sigma_1,\sigma_2),g_2) \in \mathsf{Ext}_g(\mathscr{L}_G)$
      s.t. $\mathsf{Enc}_2(\mathsf{pk}_2,\sigma_2;g_2) = \mathsf{ct}_2$:
  if k-th query : **abort**
  return $h(\sigma_1,\sigma_2)$
if k-th query :
  $(\sigma_1',\sigma_2') \leftarrow (\mathsf{Dec}_1(\mathsf{sk}_1,\mathsf{ct}_1),\mathsf{Dec}_2(\mathsf{sk}_2,\mathsf{ct}_2))$
  for i s.t. $\nexists((i,\sigma_1,\sigma_2),g_i) \in \mathsf{Ext}_g(\mathscr{L}_G)$
      s.t. $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_i;g_i) = \mathsf{ct}_i$:
    $g_i' \leftarrow g(i,\sigma_1',\sigma_2')$
    if $\mathsf{Enc}_i(\mathsf{pk}_i,\sigma_1';g_i') = \mathsf{ct}_i$: **return** 1
  abort
return $\perp$

Figure 4.17: Adversary $\mathscr{B}_k$ for the proof of Lemma 4.4.1.

4.4.4 Hash combiners.

As some of the proposed functions g use more than one hash functions, these functions are themselves hash combiners. Thus, it is of interest to study the robustness of such constructions. That is, if one of the underlying hash functions is broken (i.e. shown not to behave as a RO), is the g function (thus the whole FO-like combiner) still secure? As one of the main security concerns of the use of FO-like transforms is that the proofs are in the ROM, using robust hash combiners may improve the trust in such constructions.

The last function g in Table 4.1 (p. 75) is actually a robust combiner with respect to the RO property. That is, $G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$ is indistinguishable from a RO, even if G_1 (or G_2) is any function. Hence, if we take both $g(i,\sigma_1,\sigma_2) = G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$ and $h(\sigma_1,\sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$ in the FO-like combiner, we will obtain a secure KEM as long as G_i **and** H_i **and** PKE_i are secure for some $i \in [2]$.

$\mathcal{B}_1^{\mathcal{A},G}(\mathsf{pk}_1,\mathsf{ct}_1^*)$

$(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow\$ \mathsf{Gen}_2(1^\lambda)$
$\sigma_2^* \leftarrow\$ \mathcal{M}_2$
$\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{pk},\sigma_2^*)$
$K \leftarrow\$ \mathcal{K}$
run $\mathcal{A}^{\mathcal{O}_2^{\mathsf{Dec}},G,H}((\mathsf{pk}_1,\mathsf{pk}_2),(\mathsf{ct}_1^*,\mathsf{ct}_2^*),K)$
$\mathcal{L}_G^* \leftarrow \{\sigma_1 : ((1,\sigma_1,\sigma_2^*),g) \in \mathsf{Ext}_g(\mathcal{L}_G)\}$
$\sigma_1 \leftarrow\$ \mathcal{L}_G^* \cup \mathsf{Ext}_h(1,\sigma_2^*,\mathcal{L}_H)$
return σ_1

Figure 4.18: OW-CPA adversary for the proof of Theorem 4.4.1.

Proposition 4.4.3 (Informal). *Let* $g(i,\sigma_1,\sigma_2) = G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$ *and* $h(\sigma_1,\sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$. *We call a tuple* $(G_i, H_i, \mathsf{PKE}_i)$ *secure if* G_i, H_i *are ROs and* PKE_i *is OW-CPA. Let* KEM *be the hybrid KEM resulting from applying* $\mathsf{UT}_\parallel$ *on* PKE_1 *and* PKE_2 *with g and h to derive the deterministic coins and key, respectively. Then,* KEM *is IND-CCA if* $(G_1, H_1, \mathsf{PKE}_1)$ *or* $(G_2, H_2, \mathsf{PKE}_2)$ *(or both) is secure.*

Proof sketch. We assume w.l.o.g. that the tuple $(G_1, H_1, \mathsf{PKE}_1)$ is secure and G_2, H_2 can be any functions and PKE_2 might not be OW-CPA. In addition, we assume G_1, H_1, G_2, H_2 are mutually independent functions (e.g. this can be implemented by RO separation). The result follows simply from the fact that in the IND-CCA game against KEM, as long as G_1 is a RO, the coins $G_1(i,\sigma_1) \oplus G_2(i,\sigma_2)$ are indistinguishable from uniform unless (i,σ_1) is queried, irrespectively of the value $G_2(i,\sigma_2)$. But in turn such a query would break the OW-CPA assumption on PKE_1 (or happens with negligible probability). The same argument for $h(\sigma_1,\sigma_2) = H_1(\sigma_1) \oplus H_2(\sigma_2)$ implies that the key will always be indistinguishable from uniform if H_1 is a RO and PKE_1 is OW-CPA. □

4.5 Implementation

As a proof of concept, we implemented a fully PQ hybrid KEM using two IND-CPA proposals that passed to the Round 2 of the standardisation process and our combiner. As the main goal of our combiner is to increase the security while still offering good performances, we chose HQC and LAC since

1. LAC is one of the most efficient schemes in terms of speed and public key/ciphertext size but it has been attacked recently in [GJY19]. More generally, it seems LAC is more vulnerable to failure attacks than other schemes and that led this scheme to be dropped for Round 3. Thus, using it along another cryptosystem does not imply a large overhead while preventing a failure attack alone against LAC to break the whole scheme.

2. HQC is a code-based scheme that offers good performance, although the hardness

assumption it is based on has not been extensively studied as of yet. Thus, combining it with another efficient scheme might provide more confidence in this scheme at the expense of a small overhead.

3. HQC is code-based while LAC is lattice-based. Therefore, one can hope that any improvement in breaking the assumption of one does not lead to a better cryptanalysis of the other.

4.5.1 Design choices

For both schemes, we used the reference IND-CPA implementations provided by the authors in the second round. Then, we applied our $\mathsf{UT}_{\|}$ combiner. In practice we implemented $G(1,\cdot,\cdot)$ as $\mathsf{SHA256}(\cdot)$, $G(2,\cdot,\cdot)$ as the AES-based expansion function provided by the NIST, and $H(\cdot)$ as $\mathsf{SHA512}(\cdot)$[2]. These choices made the implementation easier as we could stick to most of the author's choices. For example, HQC encryption function in the original FO transform is using a seed output by the AES-based expander and our choice of $G(2,\sigma_1,\sigma_2)$ makes it possible to reuse most of the code.

We implemented two versions of the hybrid cryptosystem, a standard version that we are calling `hqc_lac128` and a parallel version denoted by `hqc_lac128_par`, both using the Level 1 (i.e. aiming at 128 bits of classical security) reference implementations of LAC and HQC. The parallel implementation uses the `pthread` library and is implemented without any other optimisation. In particular, only the encryption of the seeds is parallelised in the encapsulation function (i.e. the encryption functions of LAC and HQC are called in different threads) and only the decryption and re-encryption is parallelised in the decapsulation procedure.

4.5.2 Results and efficiency

We tested both our hybrid schemes on a laptop running Ubuntu 14.04 with an Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz. The results for our hybrid schemes, the original schemes and reference implementations of two other popular lattice-based schemes (Frodo and Kyber) are reported in Table 4.2. The sizes are in bytes and the times are given in microseconds (10^{-6}s) and are averaged over 10 000 runs. Obviously, the size of the public/secret key and ciphertext are the addition of the corresponding ones in LAC and HQC, except for the ciphertext, which is a bit smaller. This follows from the fact that the ciphertext in HQC contains a confirmation hash that we omit in our FO-like combiner. One can see that compared to a proposal with large keys and ciphertexts (i.e. Frodo), our hybrid compares well. In addition, as LAC produces small outputs, the increase compared to HQC is small. That is, the size of the secret key, public key and ciphertext is increased by roughly 33%, 17% and 10%, respectively.

Considering the speed, the non-optimised hybrid `hqc_lac128` performs slightly better than both LAC and HQC run one after the other. However, all procedures are still much faster

[2] In practice a KDF should be used, but for the sake of benchmarking $\mathsf{SHA512}$ is sufficient.

Scheme	SK (B)	PK (B)	CT (B)	KeyGen (μs)	Encaps (μs)	Decaps (μs)
`frodo640`	19888	9616	9720	847.553	4650.037	4602.284
`hqc128`	3165	3125	6234	144.166	298.120	528.624
`kyber512`	1632	800	736	154.077	210.857	263.194
`lac128`	1056	544	712	115.308	199.776	311.709
`hqc_lac128`	4221	3669	6882	260.032	484.969	813.452
`hqc_lac128_par`	4221	3669	6882	162.502	315.137	549.516

Table 4.2: Performance of `hqc_lac128` and `hqc_lac128_par` compared to other schemes. The size of the public/secret key and ciphertext are in bytes. The time for key generation, encapsulation, decapsulation is in microseconds.

than the ones of a slower scheme, like Frodo. On the other hand, the parallelised hybrid `hqc_lac128_par` offers very good performance as one could expect from such a parallelisable design. In particular, we observe only a 13%, 6%, and 4% increase of latency compared to HQC for key generation, encapsulation, and decapsulation, respectively. Therefore, `hqc_lac128_par` can perform nearly as good as HQC on systems that offers efficient parallelisation, such as laptops or any machine with regularly idle processors.

We give on Figure 4.19 a visualisation of the performance of `hqc_lac128` compared to other round 2 candidates with security Level 1. Most of the data comes from the SUPERCOP [Be20] benchmarking system (we picked the results of a test performed on a 2018 Intel Core i7-8809G). All round 2 proposals are represented, except for BIKE, Round 5, and LEDACrypt, which did not have an IND-CCA version benchmarked at the time of the test. We still included the keys and ciphertext sizes of BIKE as they are similar to the ones of HQC.

For the hybrid scheme `hqc_lac128`, we computed the cycles needed for key generation, encapsulation and decapsulation as the sum of the corresponding cycles needed by LAC and HQC. Note that this is a pessimistic approximation as the hybrid system requires less instructions than the sum of both underlying schemes (e.g. we apply some hash functions only once), this is confirmed in practice by the results shown in Table 4.2. We do not plot the parallelised version `hqc_lac128_par` as the sizes are the same as the ones of `hqc_lac128` and the time is upper bounded by the latter as well.

Analysis. From all three graphs in Figure 4.19, we can deduce that our hybrid does not perform particularly well compared to other schemes in these metrics. However, one can see that the bottleneck is the use of HQC here. In particular, `hqc_lac128` performs nearly as well as HQC in the metrics considered. This confirms that boosting security by combining a very efficient scheme with one that is less so does not worsen much the performance of the latter one. In other words, if one is willing to use HQC, one can as well use the hybrid `hqc_lac128` for a very small overhead but arguably much better security.

Finally, one can wonder what is the speedup of our combiners compared to existing ones.

We take as an example the XtM combiner from Bindel et al. [Bin+19a], which applies a special kind of MAC to the ciphertexts and keys. It is proposed to implement this primitive as the concatenation (or the XOR) of two standard MACs. This computation is the main overhead compared to our construction and we simulated it as two calls to SHA256 on both ciphertexts and keys. This takes approximately $40\mu s$ on our setup, hence the speedup when considering `hqc_lac128_par` is slightly over 10% for encapsulation. This obviously depends on many factors like hardware, hash functions, parallelisation, and the underlying schemes. For example, for small ciphertexts the speedup will be negligible while for large ones it will be more important. Finally, we note that PQ schemes are not optimised thus the gain might be more noticeable in the future.

4.5.3 Other hybrid KEMs.

While `hqc_lac128` is an interesting example of the advantages offered by a PQ hybrid KEM, one might wonder what is the optimal combination of schemes according to some metrics. Using the same data [Be20], we computed the theoretical performance of all possible hybrids made of two PKEs based on different assumptions (e.g. code and lattice). We considered the fastest ones in encapsulation/decapsulation and the ones with the smallest public key/-ciphertext size. We present some of the most efficient ones according to these metrics in Table 4.3. **We leave the ones based on SIKE for the sake of completeness but stress that SIKE is broken [CD23]**. We also include a lattice/rank-based hybrid scheme for completeness (i.e. `NTRUhps_rqcI`) and a LAC-RSA hybrid KEM as an interesting comparison. Overall, non-lattice-based schemes are considerably slower than lattice-based ones (although some data on BIKE is missing), thus it seems that combining schemes of these two types will not give small public key *and* fast encapsulation/decapsulation.

We give a visualisation of the performance of these hybrid schemes compared to the NIST proposals (and RSA 2048) in Figure 4.20. On the first figure, one can easily identify the hybrid schemes based on McEliece and NTS on the right. Both the hybrid schemes based on BIKE and `NTRUhps_rqcI` have public key and ciphertext sizes that lie between those of the rank-based proposals and some code-based ones.

On the second figure, one can see that hybrid schemes based on SIKE are slow due to the underlying scheme. On the second figure, one can see that in terms of speed the hybrid systems based on McEliece and NTS offer competitive performance. However, `NTRUhps_rqcI` is the only full PQ hybrid considered that has slightly worse than average performance in all metrics considered (i.e. bandwidth and speed). Interestingly, we see that the decapsulation latency of RSA is one of the worst among the schemes considered, and thus the hybrid `lac_rsa` suffers from slow decapsulation as well.

In general, several lattice-based schemes offer good performance in both the chosen metrics. Hence, the hybrid constructions mostly inherits the advantages and disadvantages of the second PKE scheme used in the construction (i.e. isogeny, code or rank-based). Furthermore,

Scheme	**PK** (B)	**CT** (B)	**Encaps** (cycles)	**Decaps** (cycles)
`kyber512_sike`*	1 178	1 138	17 652 847	18 817 320
`lac128_sike`*	922	1 114	17 677 983	18 871 919
`NTRUhps_sike`*	1 077	1 101	17 643 917	18 826 865
`NTRUhps_bike2`	2 171	2 171	-	-
`lightsaber2_bike2`	2 144	2 208	-	-
`lac_bike2`	2 016	2 184	-	-
`NTRUhps_McEliece`	261 819	827	74 361	168 478
`kyber512_McEliece`	261 920	928	83 291	158 933
`lightsaber2_McEliece`	261 792	864	102 172	186 370
`NTRUhps_NTSkem`	320 187	827	140 165	371 082
`kyber512_NTSkem`	320 288	928	123 001	334 107
`lightsaber2_NTSkem`	320 160	864	141 882	361 544
`NTRUhps_rqcI`	1 552	2 389	374 470	1 265 545

Table 4.3: Selection of efficient hybrid schemes. *SIKE has been broken since the publication of this research [CD23].

one can see from Figure 4.20 that composing a hybrid KEM from an “extreme” scheme (i.e. a scheme that performs very well in one metric but very badly in another) might not be the best option.

It seems that a better approach would be to combine two schemes based on the same type of assumptions. However, that would probably lower the practical security of the hybrid scheme, as a breakthrough in breaking one of the assumptions could automatically imply breaking the other one. A more complete study is out of the scope of this thesis and we leave it as future research.

4.6 Discussion

In this last short section of the chapter, we wish to discuss in more details the security implications of using hybrid KEMs in applications. In particular, we argue that the security boost offered by robust combiners might be greater in practice than what is suggested by the mathematical bounds. Indeed, one can see from security proofs of robust combiners (e.g. proofs of Theorem 4.3.2 & 4.4.1) that at some point between two games Γ^i and Γ^j we define some event E s.t.

$$\left|\Pr[\Gamma^i(\mathscr{A}) \Rightarrow 1] - \Pr[\Gamma^j(\mathscr{A}) \Rightarrow 1]\right| \leq \Pr[\mathsf{E}] \ .$$

Typically, the event E occurring implies that two adversaries can break both underlying schemes with good probability. For instance, in the proof of Theorem 4.3.2 we have (informally) $\Pr[E] \leq q_G \cdot \Pr[\mathscr{B}^1$ breaks $\mathsf{PKE}_1, \mathscr{B}^2$ breaks $\mathsf{PKE}_2]$ for some adversaries $\mathscr{B}^1$ and $\mathscr{B}^2$.

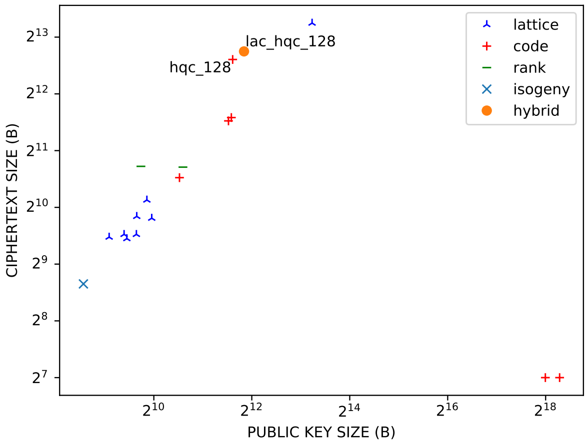
Unfortunately, the term

$$\Pr[\mathscr{B}^1 \text{ breaks } \mathsf{PKE}_1, \mathscr{B}^2 \text{ breaks } \mathsf{PKE}_2]$$

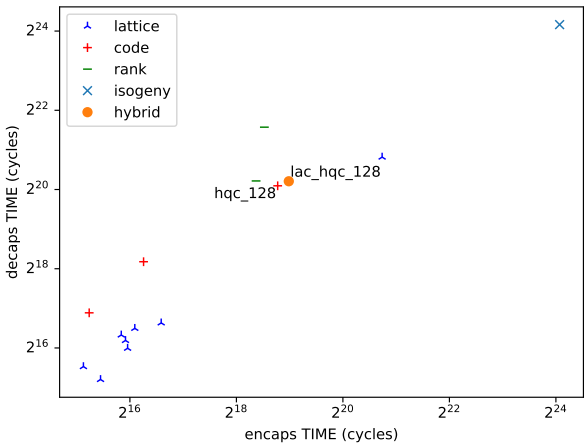
is not very informative in terms of security. Indeed, for example if we set $\mathsf{PKE}_1 = \mathsf{PKE}_2$, then most likely we will have $\Pr[B^1, B^2] \approx \Pr[B^1]$ (where we set $B^i := \mathscr{B}^1$ breaks PKE_i) and the combiner gives no security improvement. More generally, it seems very difficult to formally define the correlation between B^1 and B^2, and thus to give a concrete approximation of $\Pr[B^1, B^2]$. Hence, the bound appears to give no (easily computable) information on the security advantage of using a hybrid combiner.

Another gap between theory and practice in the case of hybrid schemes relates to the notion of security bits. In short, in the NIST PQ standardisation process, a scheme is deemed having λ-bits of security if the complexity to break it is at least the complexity to break AES-λ. In general, we see that if the best known attack (which succeeds with probability 1) against a scheme PKE_i has complexity $\approx 2^{\lambda_i}$, then the complexity of an attack against a hybrid scheme based on PKE_1 and PKE_2 is $\approx 2^{\lambda_1} + 2^{\lambda_2}$. Thus, the increase in the number of security bits is at most one, even if it is required to break two supposedly hard problems to break the hybrid scheme. Overall, it seems to us that such metrics are not the best to quantify the security of hybrid schemes.

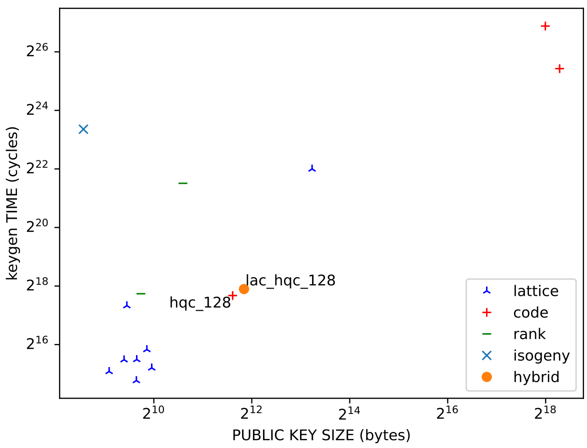
Indeed, for instance one could try to compare the security of an hybrid KEM based on two (seemingly "independent") 128-bits schemes with the one of a 256-bits KEM. In terms of security bits, the hybrid scheme would have less than 129 bits of security while the KEM would benefit of 256 bits of security. However, one might reasonably argue that the probability of a major breakthrough in two different problems believed to be hard by the community is much lower than the probability of one (but even more devastating) breakthrough. The cryptanalysis of SIKE [CD23] and Rainbow [Beu22] are such examples: a 128-bits secure hybrid based on Kyber and e.g. SIKE would still be deemed secure today, while SIKE with the parameters for 256-bits security would not. Moreover, two major breaks would likely occur in a long time frame, giving the time to mitigate the effects of a complete cryptanalysis. Note that such an argument holds only if the correlation between both events is low, that is solving a hard problem (e.g. rank syndrome decoding) does not offer an immediate advantage in solving the other (e.g. LWE). In summary, the practical security of a scheme (hybrid or not) obviously depends on many parameters and knowledge yet to be discovered, but we think that hybrid KEMs offer a greater security boost than what can be deduced from the theoretical bounds only.



(a) Public Key size vs Cipertext size.

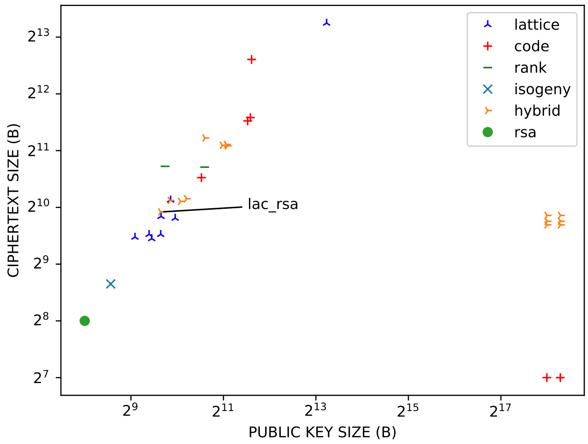


(b) Encapsulation time vs Decapsulation time.

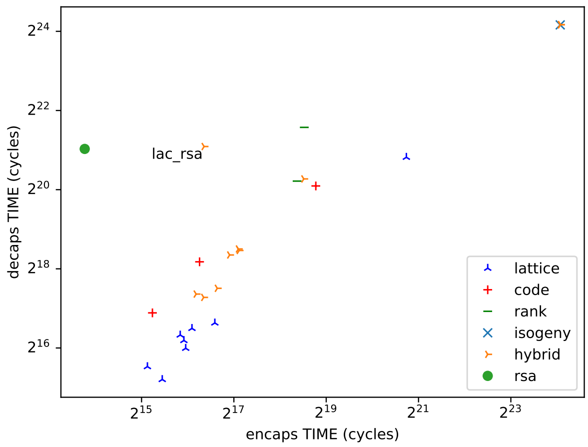


(c) Public Key size vs Key Generation time.

Figure 4.19: Visualisation of the performance of `hqc_lac128` compared to several Level 1 implementation of NIST round 2 proposals.



(a) Public Key size vs Cipertext size.



(b) Encapsulation time vs Decapsulation time.

Figure 4.20: Visualisation of the performance of different hybrid schemes (see Table 4.3) compared to several Level 1 implementation of NIST round 2 proposals.

5 Impossibility of Post-Quantum Shielding Black-Box Constructions of CCA from CPA

We saw in previous chapters how FO-like transforms could be used to construct generically strongly secure KEMs and PKEs out of CPA-secure PKEs. However, we also highlighted the fact that the re-encryption step implied by such constructions is both expensive and vulnerable to misuse attacks. Therefore, a natural problem would be to study whether this re-encryption step can be removed, and we tackle this question in this chapter.

It turns out that classically, this was proven to be impossible by Gertner et al. [GMM07]. In particular, they showed that no *shielding* black-box reduction from IND-CCA to IND-CPA exists. A shielding reduction means that the decryption algorithm of the IND-CCA PKE cannot call the encryption function of the underlying IND-CPA PKE. While their result was shown in the standard model, it readily extends to the ROM, implying that the re-encryption checks cannot be removed from FO-like transforms. We generalise this result to the post-quantum setting.

The results presented in this chapter are joint work with Serge Vaudenay and will be published in the Communications in Cryptology journal [HV24].

5.1 Contributions

Our main contribution is to prove that no *post-quantum shielding* reduction from IND-CCA PKE to IND-CPA PKE exists. Here, unlike in Gertner et al.'s, IND-CCA and IND-CPA are defined relative to quantum adversaries. Moreover, the reduction algorithm is assumed to be quantum as well. However, we still consider classical schemes, i.e. both the IND-CCA and IND-CPA PKEs are assumed to be computable classically. This is why we call this type of reduction *post-quantum*.

From a high-level, the proof uses similar techniques as the classical one. That is, we use the well-known *two oracles* technique by Hsiao et al. [HR04], which is itself a variant of the relativising method introduced by Impagliazzo and Rudich [IR89]. In short, we propose an oracle $\mathcal{O} = (O, R)$ relative to which IND-CPA PKEs exist but IND-CCA schemes Π^O (i.e. Π can query O but not R) do not. One of the main technical difficulties in the proof arises from the fact that the IND-CPA adversaries are quantum, and therefore have quantum access to the oracle. Therefore, we need to show that an adversary that can make quantum queries to $\mathcal{O}$ cannot break the IND-CPA scheme. Our proof relies on reductions from several hard (quantum) problems and thus minimal quantum knowledge is sufficient to verify it.

An obvious limitation, as in the original proof, is that we rule out only *shielding* reductions. However, if non-shielding constructions existed, they would imply a re-encryption step during decryption, as in the Fujisaki-Okamoto (FO) transform. Thus, our result rules out more efficient transforms than the FO one.

5.2 Related Work

Since the seminal paper by Impagliazzo and Rudich [IR89], the topic of black-box separation has been extensively studied (e.g. [AS16; HR04; Sim98]). In particular, as mentioned several times, the present work is a generalisation of a result by Gertner et al. [GMM07]. More recently, Hosoyamada et al. [HY20] defined the notion of quantum black-box reduction. In addition, they showed that there is no quantum black-box reduction from collision-resistant hash functions to one-way permutations [HY20]. Following this work, Cao et al. [CX21] proved that one-way permutations cannot be obtained from different flavours of one-way functions in a quantum black-box way.

Different notions of black-box reductions were first formalised by Reingold et al. [RTV04]. These were then extended by Baecher et al. [BBF13].

5.3 Technical Overview

We use the two-oracle technique by Hsiao et al. [HR04] to rule out post-quantum reductions from (post-quantum) IND-CCA PKE to IND-CPA PKE. That is, we provide an oracle O that helps implement a IND-CPA PKE, and an oracle R that helps break any construction of IND-CCA PKE. More precisely, the oracle O will contain 3 sub-oracles $(\mathbf{g}, \mathbf{e}, \mathbf{d})$, where $\mathbf{g}$ is an ideal key-generation function, $\mathbf{e}$ an ideal public key encryption function, and $\mathbf{d}$ is the corresponding decryption function. Then, $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ will correspond to the IND-CPA PKE scheme. Note that without an additional breaking oracle R, the PKE would be IND-CCA secure against classical or quantum adversaries. Now, R is composed of additional sub-oracles, which are approximately defined as follows.

- $\mathbf{w}$, which takes as input a public key pk and encrypts each bit of the corresponding sk using $\mathbf{e}$. That is, $\mathbf{w}(\mathsf{pk}) \to (\mathbf{e}(\mathsf{pk}, \mathsf{sk}_i))_{i \in [n]}$, where sk is s.t. $\mathbf{g}(\mathsf{sk}) = \mathsf{pk}$.
- $\mathbf{u}$, which takes as input a public key pk and a ciphertext c, and outputs 1 iff both the public key and the ciphertext are valid (i.e. the public key has a corresponding secret

key and the ciphertext has a corresponding pre-image under the given public key).

We note that the set of oracles $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ is the same as the one used in Gertner et al.'s proof [GMM07].

Then, in order to prove the separation, we need to show two results:

1. $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is an IND-CPA PKE even if the adversary has access to $\mathbf{w}$ and $\mathbf{u}$. In the classical setting, this is quite straightforward to prove, as was done by Gertner et al. [GMM07]. In the quantum setting, this is much more tricky as the adversary can now query $\mathbf{w}$ in superposition and the demonstration of this result turns out to be the technical contribution of this chapter. Our proof involves two reductions to quantum problems. We first introduce the IMG problem, where (informally) a quantum adversary must distinguish between two sets of oracles (e_1, e_2, w_1) and (e_1, e_2, w_2), where e_1, e_2 are random injective functions and w_1 (resp. w_2) is a random function that has the same image as e_1 (resp. e_2). We then show that the IND-CPA security of $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ reduces to the IMG problem. Intuitively, in the reduction, the encryption of a 1 (resp. 0) will be simulated by a call to e_1 (resp. e_2) and w_b will simulate the encryption of a bit of sk.

   Finally, we prove that the IMG problem is hard for any quantum adversary by reducing another provably hard problem (namely the set equality problem SETEQ [Zha13]) to it. In SETEQ, the adversary is given two random injective functions f and g s.t. either f, g have the same image or have completely distinct images, and must distinguish between both cases.

   We believe this proof might be of independent interest as it shows security of (ideal) encryption even in the presence of ciphertexts that are highly correlated with the secret-key.

2. Any shielding construction of a PKE from O is insecure against an IND-CCA adversary having access to R. For this, we can simply reuse the proof from Gertner et al. [GMM07] as the classical adversary they build can obviously be implemented quantumly.

We conclude the proof by combining these results and applying usual separation arguments.

5.4 Quantum Algorithms

We formally define in this section quantum oracle-aided algorithms and post-quantum reductions.

First, in order to understand the scope of our result, we need to formally define what kind of quantum algorithms we consider. In this chapter, we will use the following definition, adapted from Hosoyamada et al. [HY20] such that it works with uniform circuits.

Definition 5.4.1 (Uniform quantum). *A quantum algorithm $\mathscr{A}$ is a family of uniform quantum circuits $\{\mathscr{A}_n\}_{n\in\mathbb{N}}$. A family of quantum circuits is uniform if it can be generated by a (classical) deterministic Turing machine.*

We refer the reader to Nishimura et al. [NO02] for more details on uniform quantum circuits.

Definition 5.4.2 (Oracle-aided quantum algorithms). *A quantum oracle is a family of quantum gates $\mathscr{O} = \{\mathscr{O}_n\}_{n\in\mathbb{N}}$. Let $\mathscr{O}_1,\ldots,\mathscr{O}_t$ be a set of t quantum oracles. Then, an oracle-aided quantum algorithm $\mathscr{A}$ is a family of uniform quantum circuits $\{\mathscr{A}_n\}_{n\in\mathbb{N}}$ s.t. on a (classical) input $x \in \{0,1\}^n$, $\mathscr{A}$ runs $\mathscr{A}_n^{\mathscr{O}_{1,n},\ldots,\mathscr{O}_{t,n}}$ on the quantum state $|x,0,0\rangle$, measures the final state and returns the result of the output register. In other words, $\mathscr{A}_n^{\mathscr{O}_{1,n},\ldots,\mathscr{O}_{t,n}}$ can be defined as the unitary operator*

$$\mathscr{A}_n^{\mathscr{O}_{1,n},\ldots,\mathscr{O}_{t,n}} = \left(\prod_{i=1}^{q}(U_{i,t,n}\mathscr{O}_{t,n}\ldots U_{i,1,n}\mathscr{O}_{1,n})\right)U_{0,n}\ ,$$

where $U_{i,j,n}, U_{0,n}$ are some unitary operators and q is the number of queries made by $\mathscr{A}_n$ to the oracles. If an oracle $\mathscr{O}$ is randomised, it is sampled from a given distribution before $\mathscr{A}$ runs $\mathscr{A}_n^{\mathscr{O}_n}$.

Remark. The oracles (classical or quantum) considered in this chapter are stateless. In the quantum setting, that means the oracle does not keep a secret register that evolves with queries. Therefore, we assume that having quantum access to an oracle means having an oracle access to the corresponding unitary. The same assumption stays valid when an algorithm has oracle access to another quantum algorithm.

Now we can define the notion of query magnitude. Informally, this is the quantum equivalent to the probability that an adversary queries a certain value to an oracle.

Definition 5.4.3 (Query magnitude [HY20]). *Let $\Gamma = (\mathscr{O}_1,\ldots,\mathscr{O}_t)$ be a set of fixed (i.e. not randomised) quantum oracles. In addition, let $|\phi_j^i\rangle$ be the state of $\mathscr{A}^\Gamma$ (running on some fixed input x) before the j-th query to an oracle $\mathscr{O}_i$. We can assume w.l.o.g. that the oracle $\mathscr{O}_i$ acts on the first $inp_i + out_i$ qubits of $|\phi_j^i\rangle$ (i.e. inp_i qubits of input and out_i qubits of output). Then there exist $\alpha_z \in \mathbb{C}$ and a state $|\psi_z\rangle$ s.t.*

$$|\phi_j^i\rangle = \sum_{z\in\{0,1\}^{inp_i}} \alpha_z |z,\psi_z\rangle\ .$$

The query magnitude *of z before the j-th query of $\mathscr{A}^\Gamma(x)$ to $\mathscr{O}_i$, for an input $x \in \{0,1\}^n$ is*

$$\mu_{z,j}^{\mathscr{A},\mathscr{O}_i}(x) := |\alpha_z|^2\ .$$

Note that if one measures the first inp_i qubits of $|\phi_j^i\rangle$, z will be the result with probability $\mu_{z,j}^{\mathscr{A},\mathscr{O}_i}(x) = |\alpha_z|^2$.

The total query magnitude *of z is simply the sum of the query magnitude over all queries*

$\mathcal{B}^{\mathcal{A},\Gamma}(x)$

$j \leftarrow\!\$\ [q]$
run $\mathcal{A}^{\Gamma}(x)$ until the j-th query to $\mathcal{O}_i$
$(z, z') \leftarrow\!\$$ measure first register of $|\phi_j^i\rangle$
return z

Figure 5.1: Algorithm $\mathcal{B}$ for Lemma 5.4.1.

$\Psi'_{\Psi,\mathcal{D}_{x,z,\Gamma}}$

$(x, z, \Gamma) \leftarrow\!\$\ \Psi$
parse $(\mathcal{O}_1, \ldots, \mathcal{O}_t) \leftarrow \Gamma$
for $i \in \{1, \ldots, t\}$:
  $\mathcal{O}'_i \leftarrow \mathcal{O}_i$
  for $z' \in \{0,1\}^{inp_i - k}$:
    $y \leftarrow\!\$\ \mathcal{D}_{x,z,\Gamma}$
    // $\mathcal{O}_i = \mathcal{O}'_i$ except on values of the form $(z, \cdot)$
    $\mathcal{O}'_i(z, z') \leftarrow y$
set $\Gamma' \leftarrow (\mathcal{O}'_1, \ldots, \mathcal{O}'_t)$
return (x, z, Γ, Γ')

Figure 5.2: Distribution Ψ' induced by Ψ and $\mathcal{D}_{x,z,\Gamma}$ for Lemma 5.4.1.

$1 \le j \le q$ *made by the adversary to* $\mathcal{O}_i$*:*

$$\mu_z^{\mathcal{A},\mathcal{O}_i}(x) := \sum_{j=1}^{q} \mu_{z,j}^{\mathcal{A},\mathcal{O}_i}(x) \; .$$

Definition 5.4.4 (Quantum-accessible oracles)**.** *Let $\mathcal{O}$ be any classical oracle. The quantum-accessible oracle O induced by $\mathcal{O}$ is a quantum oracle defined as the unitary operator $O : |x, y\rangle \mapsto |x, y + \mathcal{O}(x)\rangle$ for any classical inputs x and y. For the sake of simplicity, in this chapter we denote by $\mathcal{O}$ both a classical oracle and its quantum-accessible oracle counterpart.*

Now we can state the following lemma, which will be useful in our proof. Informally, this lemma says that if a quantum algorithm can distinguish an oracle $\mathcal{O}$ from the same oracle where all values $\mathcal{O}(z, \cdot)$ for z have been changed, then one can extract z with good probability.

Lemma 5.4.1. *Let $n, t \in \mathbb{Z}$ be some integers and Ψ be some distribution that outputs a tuple (x, z, Γ), where $\Gamma = (\mathcal{O}_1, \ldots, \mathcal{O}_t)$ is a sequence of t sub-oracles $\mathcal{O}_i : \{0,1\}^{inp_i} \mapsto \{0,1\}^{out_i}$, $x \in \{0,1\}^n$, and $z \in \{0,1\}^k$ for some $k < inp_i$. In addition, let $\mathcal{D}_{x_d, z_d, \Gamma_d}$ be a distribution parametrised by a tuple (x_d, z_d, Γ_d) that is in the same domain as the output of Ψ defined above.*

Then, we consider the distribution Ψ' induced by Ψ and $\mathcal{D}_{x_d, z_d, \Gamma_d}$ defined by the sampling algorithm given in Figure 5.2. In addition, let $\mathcal{B}$ be the algorithm presented in Figure 5.1. Then, *for any oracle-aided quantum algorithm $\mathscr{A}$ limited to q quantum queries to Γ (or Γ') and any output y*

$$\left|\Pr[\mathscr{A}^{\Gamma}(x) \Rightarrow y] - \Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right| \leq 2q\sqrt{\Pr[\mathscr{B}^{\mathscr{A},\Gamma}(x) \Rightarrow z]}\ ,$$

where $(x, z, \Gamma, \Gamma') \leftarrow\$ \Psi'$ and the probabilities are taken over the internal randomness of the adversaries, the randomness of measurements, and the sampling from Ψ'.

Proof. We first recall a generalised version of the *Swapping Lemma* [Vaz98], proven by Hosoyamada et al. [HY20]:

Lemma 5.4.2 (Generalised Swapping Lemma [HY20])**.** *Let $\Gamma = (\mathscr{O}_1, \ldots, \mathscr{O}_t)$ and $\Gamma' = (\mathscr{O}'_1, \ldots, \mathscr{O}'_t)$ be sequences of fixed (i.e. not randomised) quantum-accessible oracles. In addition, for any pair of quantum-accessible oracles $\mathscr{O}, \mathscr{O}'$, we define $\Delta(\mathscr{O}, \mathscr{O}') := \{x : \mathscr{O}(x) \neq \mathscr{O}'(x)\}$. Then, for any oracle-aided quantum algorithm $\mathscr{A}$ and any input $x \in \{0,1\}^n$*

$$\left|\Pr[\mathscr{A}^{\Gamma}(x) \Rightarrow y] - \Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right| \leq 2\sum_{i=1}^{t}\sqrt{q\sum_{z\in\Delta(\mathscr{O}_i,\mathscr{O}'_i)}\mu_z^{\mathscr{A},\mathscr{O}_i}(x)}$$

for any output y.

We first note that sampling Γ' is the same as sampling (Γ, z, x) and then the set of differing outputs $D \leftarrow\$ \mathscr{D}_{x,z,\Gamma}$. Hence, the left-hand side of the equation can be written as

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\Pr\left[\mathscr{A}^{\Gamma}(x) \Rightarrow y\right]\right] - \mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right]\right| \\
&= \left|\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\Pr[\mathscr{A}^{\Gamma}(x) \Rightarrow y] - \Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right]\right| \\
&\leq \mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\left|\Pr[\mathscr{A}^{\Gamma}(x) \Rightarrow y] - \Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right|\right]
\end{aligned}$$

where we used the linearity of expectation and the inequality $|\mathbb{E}[X]| \leq \mathbb{E}[|X|]$. Now, Γ, Γ', x are fixed in the probabilities above (i.e. we conditioned on Γ, D, z and x). Thus, we can apply Lemma 5.4.2 to get

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\left|\Pr[\mathscr{A}^{\Gamma}(x) \Rightarrow y] - \Pr[\mathscr{A}^{\Gamma'}(x) \Rightarrow y]\right|\right] \\
&\leq 2\sqrt{q}\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\sqrt{\sum_{(z,z')\in\Delta(\mathscr{O}_i,\mathscr{O}'_i)}\mu_{(z,z')}^{\mathscr{A},\mathscr{O}_i}(x)}\right] \\
&\leq 2\sqrt{q\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\sum_{(z,z')\in\Delta(\mathscr{O}_i,\mathscr{O}'_i)}\mu_{(z,z')}^{\mathscr{A},\mathscr{O}_i}(x)\right]} \\
&= 2\sqrt{q\mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\sum_{j=1}^{q}\mu_{(z,\cdot),j}^{\mathscr{A},\mathscr{O}_i}(x)\right]}
\end{aligned}$$

where we used the inequality $\mathbb{E}[\sqrt{X}] \leq \sqrt{\mathbb{E}[X]}$ and we set $\mu^{\mathcal{A},\mathcal{O}_i}_{(z,\cdot),j}(x) = \sum_{(z,z')\in\Delta(\mathcal{O}_i,\mathcal{O}'_i)} \mu^{\mathcal{A},\mathcal{O}_i}_{(z,z'),j}(x)$ for some query j. Now, let Q be the query number sampled uniformly at random by $\mathcal{B}$ and let's assume $Q = j$. Then, the probability that $\mathcal{B}$ outputs z is the probability that the result of measuring the j-th query made by $\mathcal{A}$ is of the form (z,z') for some z'. By the definition of query magnitude, it is at least $\mu^{\mathcal{A},\mathcal{O}_i}_{(z,\cdot),j}(x)$, thus $\Pr[\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z | Q = j] \geq \mu^{\mathcal{A},\mathcal{O}_i}_{(z,\cdot),j}(x)$. Hence,

$$\Pr[\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z] = \frac{1}{q}\sum_{j=1}^{q} \Pr[\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z | Q = j] \geq \frac{1}{q}\sum_{j=1}^{q} \mu^{\mathcal{A},\mathcal{O}_i}_{(z,\cdot),j}(x) \,.$$

Finally, we get

$$\begin{aligned} 2\sqrt{q \mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\sum_{j=1}^{q} \mu^{\mathcal{A},\mathcal{O}_i}_{(z,\cdot),j}(x)\right]} &\leq 2\sqrt{q^2 \mathop{\mathbb{E}}_{\Gamma,x,z,D}\left[\Pr[\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z]\right]} \\ &= 2q\sqrt{\Pr[\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z]} \end{aligned}$$

where the last probability is taken over the internal randomness of $\mathcal{B}$, and the randomness of the measurement, Γ, x and z. Note that we can remove the dependence over D as the event $\{\mathcal{B}^{\mathcal{A},\Gamma}(x) \Rightarrow z\}$ is fully determined by Γ, x, z and the randomness of $\mathcal{B}$. Collecting the inequalities concludes the proof. $\square$

One can observe that the above lemma is a generalised version of OW2H lemma [Unr15] (Lemma 2.3.1).

5.4.1 Post-Quantum reductions

We first define a classical primitive as Baecher et al. [BBF13].

Definition 5.4.5 (Algorithm computing a random variable)**.** *We say an algorithm $\mathcal{A}$ computes a random variable A if $\mathcal{A}$ produces an output with the same distribution as A. In the following, we often write $\mathcal{A}$ to denote both a random variable and the algorithm that computes it.*

Definition 5.4.6 (Classical primitive)**.** *A (classical) primitive $\mathcal{P}$ is a tuple $(\mathcal{F}_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}})$, where $\mathcal{F}_{\mathcal{P}}$ is a set of random variables and $\mathcal{R}_{\mathcal{P}}$ is a relation between two random variables.*

A classical algorithm (i.e. Turing machine) implements *$\mathcal{P}$, or is an* implementation of *$\mathcal{P}$, if it computes f for some $f \in \mathcal{F}_{\mathcal{P}}$.*

*A classical/quantum adversary "*breaks *f" if it computes $\mathcal{A}$ s.t. $(f,\mathcal{A}) \in \mathcal{R}_{\mathcal{P}}$.*

Finally, let $f \in \mathcal{F}_{\mathcal{P}}$ be efficiently computable by a classical algorithm, then if there is no efficient classical (resp. quantum) algorithm $\mathcal{A}$ s.t. $(f,\mathcal{A}) \in \mathcal{R}_{\mathcal{P}}$, we say f is secure *(resp.* post-quantum secure*).*

Remark. In this chapter, we are interested in classically computable primitives (PKEs) that might resist quantum adversaries. Therefore, we do not consider quantum implementations but only quantum adversaries. That is, any implementation can be computed by a classical algorithm but the set of adversaries is the set of efficient quantum algorithms.

Finally, we define the notion of post-quantum black-box reduction.

Definition 5.4.7 (Post-Quantum black-box reduction)**.** *Let $\mathscr{P}$ and $\mathscr{Q}$ be classical primitives. There exists a post-quantum black-box reduction from $\mathscr{Q}$ to $\mathscr{P}$ if there exist an efficient classical algorithm G and an efficient quantum algorithm $\mathscr{S}$ s.t.*

1. *For every (classically computable) $f \in \mathscr{F}_{\mathscr{P}}$, then $G^f \in \mathscr{F}_{\mathscr{Q}}$.*
2. *For every quantum adversary $\mathscr{A}$ and (implementation of) $f \in \mathscr{F}_{\mathscr{P}}$, if $(G^f, \mathscr{A}^f) \in \mathscr{R}_{\mathscr{Q}}$ then $(f, \mathscr{S}^{\mathscr{A},f}) \in \mathscr{R}_{\mathscr{P}}$.*

The second condition can be rewritten as

$$\begin{aligned}&\exists \mathsf{EFF_c}\, G \quad \exists \mathsf{EFF_q}\, \mathscr{S} \quad \forall \mathscr{A} \quad \forall f \in \mathscr{F}_{\mathscr{P}}\\ &(G^f, \mathscr{A}^f) \in \mathscr{R}_{\mathscr{Q}} \Rightarrow (f, \mathscr{S}^{\mathscr{A},f}) \in \mathscr{R}_{\mathscr{P}}\end{aligned}$$

where $\mathsf{EFF_c}$ *and* $\mathsf{EFF_q}$ *stand for efficient classical and efficient quantum, respectively.*

In the post-quantum black-box reduction defined above, we start with a classical primitive $\mathscr{P}$ meant to be post-quantum secure. Then, for a black-box reduction to exist, there must be a classical algorithm that builds a primitive $\mathscr{Q}$ using $\mathscr{P}$. In addition, there must be an efficient quantum reduction algorithm $\mathscr{S}$, which, given quantum black-box access to any (even non efficient) adversary that breaks $\mathscr{Q}$, builds an adversary that breaks $\mathscr{P}$.

Ruling out post-quantum reductions. We show in the following lemma that a two oracles argument as described by Hsiao et al. [HR04] is sufficient to rule out post-quantum reductions. The proof is basically the same as in the classical setting.

Lemma 5.4.3. *Let $\mathscr{P}$ and $\mathscr{Q}$ be classical primitives. Then, there is no post-quantum reduction from $\mathscr{Q}$ to $\mathscr{P}$ if there exist oracles (O, R) s.t.*

1. *There exist efficient classical algorithms f s.t. $f^O \in \mathscr{F}_{\mathscr{P}}$.*
2. *For all efficient classical algorithms G:*
   - *there is an efficient quantum adversary $\mathscr{A}$ s.t. $(G^{f^O}, \mathscr{A}^{O,R}) \in \mathscr{R}_{\mathscr{Q}}$*
   - *for all efficient quantum algorithms $\mathscr{S}$ then $(f^O, \mathscr{S}^{f,O,R}) \notin \mathscr{R}_{\mathscr{P}}$*

Proof. For the sake of contradiction, we assume a pair of oracles (O,R) fulfilling the conditions in Lemma 5.4.3 exists and a post-quantum reduction from $\mathcal{Q}$ to $\mathcal{P}$ exists as well. Let f be the algorithm s.t. $f \in \mathcal{F}_{\mathcal{P}}$ as specified in condition (1). By condition (2), we have that for all G there is an efficient quantum adversary $\mathcal{A}$ s.t. $(G^{f^O}, \mathcal{A}^{O,R}) \in \mathcal{R}_{\mathcal{Q}}$. By the existence of the post-quantum reduction, it means that there exists an efficient quantum reduction $\mathcal{S}$ s.t. $(f^O, \mathcal{S}^{\mathcal{A}',f}) \in \mathcal{R}_{\mathcal{P}}$ with $\mathcal{A}' := \mathcal{A}^{O,R}$. Now, as $f, \mathcal{A}$ are efficient classical and quantum algorithms, one can embed these in $\mathcal{S}$. Hence, there exists an efficient $\mathcal{S}$ s.t. $(f^O, \mathcal{S}^{O,R}) \in \mathcal{R}_{\mathcal{P}}$. This contradicts the second part of condition (2), which completes the proof. □

Informally, the two oracles technique works as follows. One builds an oracle O that trivially implements the primitive $\mathcal{P}$ (i.e. the primitive exists relative to O). Then, we build another oracle R and we show that the primitive is secure against even unbounded quantum adversaries (with bounded number of quantum queries to (O,R)). In particular, this implies that all the security of the primitive must come from O. In a second step, we show that there exists an inefficient adversary $\mathcal{A}$ (with bounded number of queries to (O,R)) that breaks any implementation of $\mathcal{Q}$ relative to O. Then, in a final step, it is argued that $\mathcal{A}$ can be made efficient. In the classical setting, this is done by assuming P = NP or by embedding a PSPACE oracle in R. Looking ahead, this will be sufficient in our case as $\mathcal{A}$ will be classical in our proof. Lemma 5.4.3 then states that this technique is sufficient to rule out post-quantum black-box reductions.

5.5 The Oracle $\mathcal{O}$

We recall that we want to rule out reductions from IND-CCA to IND-CPA using Lemma 5.4.3. That is, we wish to find an oracle $\mathcal{O} = (O,R)$ s.t. an IND-CPA PKE exists relative to this oracle, but IND-CCA PKEs do not. We consider here PKEs that encrypt 1 bit, as they are known to imply PKEs for longer messages. We use the same oracle as the one defined by Gertner et al. [GMM07].

Definition 5.5.1 (Oracle $\mathcal{O}$)**.** *The oracle $\mathcal{O}$ is made of several sub-oracles, more precisely $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$. Each sub-oracle will play a part in the proof: $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ will correspond to the IND-CPA PKE, $(\mathbf{w}, \mathbf{u})$ will help the IND-CCA adversary break the underlying IND-CPA PKE in order to win its own game. More precisely, if we follow the notation of Lemma 5.4.3, $O = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $R = (\mathbf{u}, \mathbf{w})$.*

We now formalise how an oracle

$$\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow\$ \Psi$$

is sampled. For each $n \in \mathbb{N}$, each sub-oracle is generated as follows.

- $\mathbf{g}$*: $\{0,1\}^n \mapsto \{0,1\}^{3n}$ is a random length-tripling one-to-one function. This function will be used as a key-generation function that outputs a public key given a secret key.*

- $\mathbf{e}$*:* $\{0,1\}^{3n} \times \{0,1\} \times \{0,1\}^n \mapsto \{0,1\}^{3n}$ *is s.t.* $\mathbf{e}(\mathsf{pk},\cdot,\cdot)$ *is a random one-to-one function for all fixed* pk*. The oracle* $\mathbf{e}$ *will be used as a bit-encryption function.*

- $\mathbf{d}$*:* $\{0,1\}^n \times \{0,1\}^{3n} \mapsto \{0,1,\perp\}$ *is deterministically defined as follows. The oracle* $\mathbf{d}(\mathsf{sk},\mathsf{ct})$ *outputs* b *s.t.* $\mathbf{e}(\mathbf{g}(\mathsf{sk}), b, r) = \mathsf{ct}$ *if such* r *exists. If not,* $\mathbf{e}$ *outputs* $\perp$*. This oracle will be used as a decryption function.*

- $\mathbf{w}$*:* $\{0,1\}^{3n} \times \{0,1\}^n \mapsto \{0,1\}^{3n\times n} \cup \{\perp\}$ *is defined as follows. The function takes a public key* pk *and an index* i *as inputs, and outputs* $\perp$ *if there is no unique* sk' *s.t.* $\mathbf{g}(\mathsf{sk}') = \mathsf{pk}$*. Otherwise,* $\mathbf{w}(\mathsf{pk}, i)$ *returns a vector of* n *encryptions of the bits of* sk'*:*

$$(\mathbf{e}(\mathsf{pk},\mathsf{sk}'_1, r_{1,i,\mathsf{pk}}),\ldots,\mathbf{e}(\mathsf{pk},\mathsf{sk}'_n, r_{n,i,\mathsf{pk}}))\ ,$$

  where the $r_{k,i,\mathsf{pk}}$ *are sampled at random when* (pk, i) *is queried for the first time. This function returns the bit-by-bit encryption of the secret key corresponding to the input public key, with different random coins indexed by* i*.*

- $\mathbf{u}$*:* $\{0,1\}^{3n} \times \{0,1\}^{3n} \mapsto \{\perp,\top\}$ *takes a public key* pk *and a ciphertext* ct *as inputs and returns* $\top$ *if* $\exists b, r$ *s.t.* $\mathbf{e}(\mathsf{pk}, b, r) = \mathsf{ct}$*. Otherwise it returns* $\perp$*. This function returns whether a ciphertext is valid or not.*

5.6 Hard Problems

We introduce in this section several quantum hard problems that will be used to prove our main technical result.

First, we recall the definition of the (average) set quality (SETEQ) problem.

Definition 5.6.1 (SETEQ)**.** *Let* $\mathsf{Inj}_{n,m}$ *be the set of one-to-one functions from* $\{0,1\}^n$ *to* $\{0,1\}^m$*. We define* $\mathscr{F}_n^b$ *as the following distribution.*

- *If* $b = 0$*: Sample* $f, g \leftarrow\$ \mathsf{Inj}_{n,n+1}$ *s.t.* $\mathsf{Im}(f) = \mathsf{Im}(g)$*.*

- *If* $b = 1$*: Sample* $f, g \leftarrow\$ \mathsf{Inj}_{n,n+1}$ *s.t.* $\mathsf{Im}(f) \cap \mathsf{Im}(g) = \emptyset$*.*

The SETEQ *problem is hard if for any (possibly unbounded) quantum adversary* $\mathscr{A}$ *that makes* $\mathsf{poly}(n)$ *quantum queries to* f, g

$$\left|\Pr[\mathscr{A}^{f,g} \Rightarrow 1 : f, g \leftarrow\$ \mathscr{F}_n^1] - \Pr[\mathscr{A}^{f,g} \Rightarrow 1 : f, g \leftarrow\$ \mathscr{F}_n^0]\right| = \mathsf{negl}(n)\ ,$$

where the probabilities are taken over the quantum randomness and the sampling of f, g*.*

It turns out the SETEQ problem is hard, according to the following theorem by Zhandry [Zha13].

Theorem 5.6.1 (Hardness of SETEQ [Zha13])**.** *Let $\mathscr{F}_n^b$ be as defined above. Then, for any quantum adversary we have*

$$\left|\Pr[\mathscr{A}^{f,g} \Rightarrow 1 : f,g \leftarrow\$ \mathscr{F}_n^1] - \Pr[\mathscr{A}^{f,g} \Rightarrow 1 : f,g \leftarrow\$ \mathscr{F}_n^0]\right| = O(q^3/2^n) ,$$

where q is the number of queries $\mathscr{A}$ makes to f and g.

We now introduce an intermediary problem that we call the IMG problem.

Definition 5.6.2 (IMG problem)**.** *Let $e_0 : \{0,1\}^n \mapsto \{0,1\}^{3n}$ and $e_1 : \{0,1\}^n \mapsto \{0,1\}^{3n}$ be random one-to-one functions s.t. $\mathsf{Im}(e_0) \cap \mathsf{Im}(e_1) = \emptyset$. I.e. e_0 and e_1 are random injective functions s.t. their images are different. Let $f : \{0,1\}^n \mapsto \{0,1\}^n$ be a random function. We define $w_b(\cdot) := e_b(f(\cdot))$. In addition, we define an helper oracle $u(c)$ that returns $\top$ if $c \in \mathsf{Im}(e_0) \cup \mathsf{Im}(e_1)$ and $\perp$ otherwise. The IMG problem is considered hard if for every (possibly unbounded) quantum adversary $\mathscr{A}$ that makes $\mathsf{poly}(n)$ quantum queries to e_0, e_1, w_b, u, we have*

$$\left|\Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] - \Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 1]\right| = \mathsf{negl}(n) ,$$

where the probabilities are taken over the quantum randomness and the sampling of e_0, e_1, f. Concretely, this problem is hard if with a polynomial number of quantum queries one cannot say whether w_b has the same image as e_0 or e_1. Note that we could also define w_b as a random function with domain $\{0,1\}^n$ and codomain $\mathsf{Im}(e_b)$.

Jumping ahead, we will use the above problem with e_b defined as $\mathbf{e}(\mathsf{pk}^*, b, \cdot)$, u as $\mathbf{u}$ and w_b as one part of the $\mathbf{w}$ oracle.

Using this result, we prove that the IMG problem is hard by showing that SETEQ reduces to it.

Lemma 5.6.1 (SETEQ reduces to IMG)**.** *Let $\mathscr{F}_n^b$ be as defined in the SETEQ problem and e_0, e_1, w_b, u as defined in the IMG problem. Then, for any IMG quantum adversary one can build a SETEQ adversary such that*

$$\begin{aligned}&\left|\Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] - \Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 1]\right| \leq \\ &\left|\Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f,g \leftarrow\$ \mathscr{F}_n^1] - \Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f,g \leftarrow\$ \mathscr{F}_n^0]\right| ,\end{aligned}$$

where the number of queries made by $\mathscr{B}$ is roughly twice the number made by $\mathscr{A}$.

Proof. We first state the idea of the proof. In the SETEQ problem, when $b = 1$ (thus $\mathsf{Im}(f) \cap \mathsf{Im}(g) = \emptyset$) one can set $e_0 = f$ and $e_1 = g$ and $w_{b'} = e_{b'} \circ r$ with b' picked at random and r a random function. Minus some technical details, this perfectly simulates an instance of the IMG problem and the probability that the IMG adversary $\mathscr{A}$ outputs b' is the advantage of $\mathscr{A}$ (plus or minus $\frac{1}{2}$) in the IMG problem. Then, if $b = 0$, images of e_0 and e_1 will be the same and it is impossible to distinguish w_0 from w_1. Thus, in this case $\mathscr{A}$ outputs 0 or 1 with probability

$\frac{1}{2}$. Hence, if $\mathscr{A}$ makes the correct guess with probability p in a correct instance of the IMG problem, the SETEQ reduction $\mathscr{B}$ has an advantage of $p-\frac{1}{2}$, which is equal to $\mathscr{A}$'s advantage.

More formally, the reduction $\mathscr{B}^{f,g}$ sets $\mathscr{A}$'s oracles as follows. First, $\mathscr{B}$ samples a random one-to-one function $h \leftarrow\$ \mathsf{Inj}_{n+1,3n}$, a random function $r:\{0,1\}^n \mapsto \{0,1\}^n$, and a random bit b'. Then, each oracle is set as

- $e_0 := h \circ g$.
- $e_1 := h \circ f$.
- $w_{b'} := e_{b'} \circ r$.
- $u(c)$: return $\top$ if $c \in \mathsf{Im}(h)$, otherwise return $\perp$. Note that the check $c \in \mathsf{Im}(h)$ can be done because $\mathscr{B}$ is an unbounded adversary which sampled h.

Each oracle can be implemented in a quantum circuit that makes 2 calls to the quantum oracles f or g. For instance, the unitary $U_{e_0} : |x,y,z\rangle \mapsto |x, y+e_0(x), z\rangle$ can be implemented as

$$U_{e_0} : |x,y,0,z\rangle \xrightarrow{g} |x,y,g(x),z\rangle \xrightarrow{h} |x,y+h(g(x)),g(x),z\rangle \xrightarrow{g} |x,y+h(g(x)),0,z\rangle \ .$$

The adversary $\mathscr{B}^{f,g}$ runs $b'' \leftarrow \mathscr{A}^{e_0,e_1,w_{b'},u}$ and returns $1_{b'=b''}$. We distinguish two cases:

- $b = 1$ ($\mathsf{Im}(f) \cap \mathsf{Im}(g) = \emptyset$): By definition g and f are one-to-one functions from $\{0,1\}^n$ to $\{0,1\}^{n+1}$ and h is a random one-to-one function from $\{0,1\}^{n+1}$ to $\{0,1\}^{3n}$. Moreover, as the images of g and f are distinct, e_0 and e_1 are random one-to-one functions from $\{0,1\}^n$ to $\{0,1\}^{3n}$ s.t. $\mathsf{Im}(e_0) \cap \mathsf{Im}(e_1) = \emptyset$. In addition, $w_{b'}$ is defined as $e_{b'} \circ r$ and $u(c)$ returns whether $c \in \mathsf{Im}(e_0) \cup \mathsf{Im}(e_1)$. Therefore,

$$\begin{aligned}\Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f,g \leftarrow\$ \mathscr{F}_n^1] &= \Pr[\mathscr{A}^{e_0,e_1,w_{b'},u} \Rightarrow b' : b' \leftarrow\$ \{0,1\}] \\ &= \frac{1}{2}\Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] + \frac{1}{2}\Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 0] \ ,\end{aligned}$$

  where $(e_0, e_1, w_{b'}, u)$ follow the same distribution as in the IMG problem.

- $b = 0$ ($\mathsf{Im}(f) = \mathsf{Im}(g)$): In this case, $\mathsf{Im}(e_0) = \mathsf{Im}(e_1) = \mathsf{Im}(w_0) = \mathsf{Im}(w_1)$. As r is a random function and cannot be accessed by the adversary, w_0 and w_1 are perfectly indistinguishable. More precisely, given all values of $e_0, e_1, w_{b'}$ (we omit u as it is independent of b'), the optimal distinguisher would output the b that maximises $\Pr[e_b(r(0)) = w_{b'}(0), \ldots, e_b(r(2^n-1)) = w_{b'}(2^n-1) | w_{b'}, e_0, e_1]$. The only randomness here is the one from r, as all values of $e_0, e_1, w_{b'}$ are known. Now,

$$\Pr_r[e_b(r(0)) = w_{b'}(0), \dots, e_b(r(2^n-1)) = w_{b'}(2^n-1) | w_{b'}, e_0, e_1] =$$
$$\Pr_r[r(0) = e_b^{-1}(w_{b'}(0)), \dots, r(2^n-1) = e_b^{-1}(w_{b'}(2^n-1)) | w_{b'}, e_0, e_1] =$$
$$\frac{1}{2^{n2^n}}$$

for both $b' = 0$ or $b' = 1$, as r is a random function. Hence, even with an unbounded number of queries to $e_0, e_1, w_{b'}$, $\Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] = \Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 1]$. Therefore,

$$\Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f, g \leftarrow\$ \mathscr{F}_n^0] = \Pr[\mathscr{A}^{e_0,e_1,w_{b'},u} \Rightarrow b' : b' \leftarrow\$ \{0,1\}] = \frac{1}{2} .$$

Finally, we get that for any IMG adversary $\mathscr{A}$ that makes q quantum queries, there exists an (unbounded) SETEQ adversary $\mathscr{B}$ s.t.

$$2 \cdot \left| \Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f, g \leftarrow\$ \mathscr{F}_n^1] - \Pr[\mathscr{B}^{f,g} \Rightarrow 1 : f, g \leftarrow\$ \mathscr{F}_n^0] \right| =$$
$$2 \cdot \left| \Pr[\mathscr{A}^{e_0,e_1,w_{b'},u} \Rightarrow b' : b' \leftarrow\$ \{0,1\}] - \frac{1}{2} \right| =$$
$$\left| \Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] - \Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 1] \right| ,$$

where $\mathscr{B}$ makes at most $2q$ queries, which concludes the proof. □

Corollary 5.6.1 (Hardness of IMG)**.** *The* IMG *is hard for quantum algorithms. More precisely, for any* IMG *quantum adversary, we have*

$$\left| \Pr[\mathscr{A}^{e_0,e_1,w_1,u} \Rightarrow 1] - \Pr[\mathscr{A}^{e_0,e_1,w_0,u} \Rightarrow 1] \right| = O(q^3/2^n) ,$$

where q is the number of quantum queries made by $\mathscr{A}$.

Finally, we define partial inverse functions and recall a lemma by Cao et al. [CX21].

Definition 5.6.3 (Partial inverse function)**.** *Let $f : \{0,1\}^n \mapsto \{0,1\}^{n+m}$ be some injective function and $x^* \in \{0,1\}^n$. Then, we define the partial inverse function $f^{-1}_{\neq x^*}$ as*

$$f^{-1}_{\neq x^*}(y) = \begin{cases} x, & \text{if } \exists x \neq x^* \text{ s.t. } f(x) = y \\ \perp, & \text{if } \nexists x \text{ s.t. } f(x) = y \\ \perp, & \text{if } y = f(x^*) \end{cases} .$$

In other words, $f^{-1}_{\neq x^*}$ inverts f except on $y = f(x^*)$.

Lemma 5.6.2 (Lemma 5 [CX21])**.** *Let $f \leftarrow\$ \mathsf{Inj}_{n,n+m}$ be a random injective function, $x^* \leftarrow\$ \{0,1\}^n$, and $f^{-1}_{\neq x^*}$ be the partial inverse function. Then, for any (possible unbounded) quantum adversary*

$\mathscr{A}$ making $\mathsf{poly}(n)$ *quantum queries to* $f, f^{-1}_{\neq x^*}$*, we have*

$$\Pr[\mathscr{A}^{f, f^{-1}_{\neq x^*}}(f(x^*)) \Rightarrow x^* : x^* \leftarrow\!\!\$ \{0,1\}^n, f \leftarrow\!\!\$ \mathsf{Inj}_{n,n+m}] = \mathsf{negl}(n) ,$$

where the probability is taken over the randomness of $\mathscr{A}$, f and x^*. I.e. inverting $f(x^*)$ given f and the partial inverse function is hard.

5.7 Existence of IND-CPA PKE

We first define what a (1-bit) *PKE relative to an oracle* is.

Definition 5.7.1 (PKE relative to O)**.** *Let $O = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ be an oracle. A* valid *PKE construction relative to O is of the form* $\mathsf{PKE}^O = (\mathsf{Gen}^O, \mathsf{Enc}^O, \mathsf{Dec}^O)$*, where for all $n \in \mathbb{N}$ and some constants $\rho_0, \rho_1, \rho_2, \rho_3$,* $(\mathsf{Gen}^O, \mathsf{Enc}^O, \mathsf{Dec}^O)$ *is as follows.*

- $\mathsf{Gen}^O : \{0,1\}^n \mapsto \{0,1\}^{n^{\rho_0}} \times \{0,1\}^{n^{\rho_1}}$. *We consider* $\mathsf{Gen}^O(S) = (SK, PK)$ *as a key generation function that takes a seed S and outputs a pair of secret/public keys (SK, PK).*
- $\mathsf{Enc}^O : \{0,1\}^{n^{\rho_1}} \times \{0,1\} \times \{0,1\}^{n^{\rho_2}} \mapsto \{0,1\}^{n^{\rho_3}}$. *We consider* $\mathsf{Enc}^O(PK, PT, R) = CT$ *as an encryption function that takes as inputs a public key PK, a bit PT, and random coins R, and outputs a ciphertext CT.*
- $\mathsf{Dec}^O : \{0,1\}^{n^{\rho_0}} \times \{0,1\}^{n^{\rho_3}} \mapsto \{0,1\} \cup \{\bot\}$. *We consider* $\mathsf{Dec}^O(SK, CT) = PT'$ *as a decryption function that takes as inputs a secret key SK and a ciphertext CT, and outputs a plaintext bit PT' or the error symbol $\bot$.*

We also require perfect correctness, that is for any $PT \in \{0,1\}$, $R \in \{0,1\}^{n^{\rho_2}}$ and $S \in \{0,1\}^n$,

$$\mathsf{Dec}^O(SK, \mathsf{Enc}^O(PK, PT, R)) = PT$$

for $(SK, PK) = \mathsf{Gen}^O(S)$. In addition, w.l.o.g. we assume there are constants s and q s.t. for any security parameter n, $(\mathsf{Gen}^O, \mathsf{Enc}^O, \mathsf{Dec}^O)$ *make at most n^q queries to O and each query is at most of size n^s. In addition, the running time of* $(\mathsf{Gen}^O, \mathsf{Enc}^O, \mathsf{Dec}^O)$ *must be polynomial in n as well.*

We now prove the main theorem, that is $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is IND-CPA relative to the oracle $\mathscr{O}$.

Theorem 5.7.1. *Let* $\mathsf{PKE_q}^{\mathscr{O}} = (g^{\mathbf{g}}, \mathbf{e}, \mathbf{d})$ *be a PKE relative to $\mathscr{O}$, where $g^{\mathbf{g}}(s)$ sets* $\mathsf{sk} \leftarrow s$ *and returns* $(\mathsf{sk}, \mathbf{g}(\mathsf{sk}))$*. Then, for any (possibly unbounded) quantum adversary $\mathscr{A}$ we have*

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathscr{A}^{\mathscr{O}}, \mathsf{PKE_q}^{\mathscr{O}}} = \mathsf{negl}(n) ,$$

where the number of quantum queries made by $\mathscr{A}$ to $\mathscr{O}$ is polynomial in n.

$\Gamma^{0\text{-}2}$

$b \leftarrow\$ \{0,1\}; \mathsf{sk}^* \leftarrow\$ \{0,1\}^n$
$\mathsf{pk}^* \leftarrow \mathbf{g}(\mathsf{sk}^*)$
$r^* \leftarrow\$ \{0,1\}^n$
$\mathsf{ct}^* \leftarrow \mathbf{e}(\mathsf{sk}^*, b, r^*)$
$\forall \mathsf{pk} \in \{0,1\}^{3n}, i \in \{0,1\}^n :$
$\quad r_{i,\mathsf{pk}} \leftarrow\$ \{f : \{0,1\}^n \mapsto \{0,1\}^n\}$
$b' \leftarrow\$ \mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \quad /\!/ \ \Gamma^0$
$b' \leftarrow\$ \mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \quad /\!/ \ \Gamma^1$
$b' \leftarrow\$ \mathscr{B}^{\mathbf{g},\mathbf{e}',\mathbf{d}',\mathbf{w}',\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \quad /\!/ \ \Gamma^2$
return $b' = b$

$\mathbf{d}'(\mathsf{sk}, \mathsf{ct})$

if $\mathsf{sk} = \mathsf{sk}^*$:
$\quad$ **return** $\perp$
return $\mathbf{d}(\mathsf{sk}, \mathsf{ct})$

$\mathbf{w}'(\mathsf{pk}, i)$

if $\exists \mathsf{sk}$ s.t. $g(\mathsf{sk}) = \mathsf{pk}$:
$\quad r \leftarrow (\mathbf{e}'(\mathsf{pk}, \mathsf{sk}_1, r_{1,\mathsf{pk}}(i)), \ldots,$
$\qquad \mathbf{e}'(\mathsf{pk}, \mathsf{sk}_n, r_{n,\mathsf{pk}}(i)))$
$\quad$ **return** r
return $\perp$

$\mathbf{e}'(\mathsf{pk}, b, r)$

if $\mathsf{pk} = \mathsf{pk}^*$ **and** $r = r^*$:
$\quad$ **return** $\perp$
return $\mathbf{e}(\mathsf{pk}, b, r)$

Figure 5.3: Games Γ^0-Γ^2 for the proof of Thm 5.7.1.

Proof. We proceed with a sequence of hybrid games Γ^0-Γ^2 shown in Figure 5.3.

Game Γ^0: It is the original IND-CPA game. We recall that a quantum (sub-)oracle $\mathbf{o}$ is a family of quantum circuits: $\mathbf{o} = \{\mathbf{o}_i\}_{i \in \mathbb{N}}$, where $\mathbf{o} \in (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$. In the IND-CPA game with security parameter n, we assume the adversary *only* queries oracle circuits $\mathbf{o}_n$. As the adversary's input is independent of any suboracle $\mathbf{o}_i$, $i \neq n$ it does not change the distribution of the output. For the sake of simplicity, we write $\mathbf{o}$ for $\mathbf{o}_n$.

Game Γ^1: We modify the $\mathbf{d}$ oracle into an identical oracle $\mathbf{d}'$ except that $\mathbf{d}'(\mathsf{sk}^*, \cdot) = \perp$, where $\cdot$ denotes any value in $\{0,1\}^{3n}$ and sk^* is the challenge secret key (i.e. $\mathbf{g}(\mathsf{sk}^*) = \mathsf{pk}^*$). That is, the $\mathbf{d}'$ oracle does not reply to decryption queries that could help the adversary decrypt the challenge ciphertext ct^*. By Lemma 5.4.1, we have

$$\left|\Pr[\mathscr{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b] - \Pr[\mathscr{A}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b]\right|$$
$$\leq 2q\sqrt{\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^*]}\ ,$$

where $\mathscr{B}$ runs $\mathscr{A}$ until some random quantum query q_i, measures the input register, and outputs the first n bits of the result. Now we prove the following lemma.

Lemma 5.7.1. $\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^*] = \mathsf{negl}(n)$.

Proof. We proceed by building a sequence of hybrid games where the oracle $\mathbf{w}$ is modified.

We first recall that

$$\mathbf{w}(\mathsf{pk}, i) := (\mathbf{e}(\mathbf{g}(\mathsf{sk}), \mathsf{sk}_1, r_{1,i,\mathsf{pk}}), \ldots, \mathbf{e}(\mathbf{g}(\mathsf{sk}), \mathsf{sk}_n, r_{n,i,\mathsf{pk}})) ,$$

where the values $r_{k,i,\mathsf{pk}}$ are sampled at random and pk is s.t. $\mathbf{g}(\mathsf{sk}) = \mathsf{pk}$. Equivalently, we can write

$$\mathbf{w}(\mathsf{pk}, i) := \big(\mathbf{e}(\mathbf{g}(\mathsf{sk}), \mathsf{sk}_1, r_{1,\mathsf{pk}}(i)), \ldots, \mathbf{e}(\mathbf{g}(\mathsf{sk}), \mathsf{sk}_n, r_{n,\mathsf{pk}}(i))\big) ,$$

where $r_{k,\mathsf{pk}} : \{0,1\}^n \mapsto \{0,1\}^n$ are random functions.

$\underline{\mathbf{w}^1}$: Let $e_b(\cdot) := \mathbf{e}(\mathsf{pk}^*, b, \cdot)$. We modify $\mathbf{w}$ into an oracle $\mathbf{w}^1$ s.t.

$$\mathbf{w}^1(\mathsf{pk}, i) = \begin{cases} \mathbf{w}(\mathsf{pk}, i), & \text{if } \mathsf{pk} \neq \mathsf{pk}^* \\ (e_0(r_{1,\mathsf{pk}}(i)), e_{\mathsf{sk}_2^*}(r_{2,\mathsf{pk}}(i)), \ldots, e_{\mathsf{sk}_n^*}(r_{n,\mathsf{pk}}(i))), & \text{if } \mathsf{pk} = \mathsf{pk}^* \end{cases} .$$

In other words, when pk^* is queried, the encryption of the first bit of sk^* is replaced by the encryption of a zero. All other values returned are the same as in the original $\mathbf{w}$ oracle. We now wish to upper bound

$$\begin{aligned} &\left|\Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^*\right] - \Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^1,\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^*\right]\right| = \\ &\frac{1}{2}\left|\Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^* \middle| \mathsf{sk}_1^* = 1\right] - \Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^1,\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow \mathsf{sk}^* \middle| \mathsf{sk}_1^* = 1\right]\right| \end{aligned} \tag{5.1}$$

where the equality follows from the fact that $\mathbf{w}$ is identically distributed to $\mathbf{w}^1$ if $\mathsf{sk}_1^* = 0$. We show that for any adversary $\mathscr{B}$, one can construct a IMG adversary $\mathscr{C}$ s.t. Eq. (5.1) is upper bounded by the advantage of $\mathscr{C}$. We show the reduction in Figure 5.4.

First, we see $\mathscr{C}$ simulates perfectly the oracles for queries independent of $(\mathsf{sk}^*, \mathsf{pk}^*)$. Indeed, $\mathscr{C}$ samples a valid function g and random injective functions $e(g(\mathsf{sk}), \cdot, \cdot)$ for each $\mathsf{sk} \neq \mathsf{sk}^*$. Then, $\mathscr{C}$ can use its knowledge of these functions to reply to any query $e'(\mathsf{pk}, \cdot, \cdot)$, $d'(\mathsf{sk}, \cdot)$, $w'(\mathsf{pk}, \cdot)$ or $u'(\mathsf{pk}, \cdot)$ with $\mathsf{pk} \neq \mathsf{pk}^*, \mathsf{sk} \neq \mathsf{sk}^*$ as in the original game played by $\mathscr{B}$.

Then, $\mathscr{C}$ sets the encryption oracle for pk^* as

$$e'(\mathsf{pk}^*, b, r) = \begin{cases} e_0(r), & \text{if } b = 0 \\ e_1(r) & \text{if } b = 1 \end{cases} ,$$

where e_0, e_1 are $\mathscr{C}$'s own oracles. As e_0, e_1 are random one-to-one functions s.t. their image do not intersect, $e'(\mathsf{pk}^*, \cdot, \cdot)$ is also a random one-to-one function $\{0,1\}^{n+1} \mapsto \{0,1\}^{3n}$. Therefore, e' simulates perfectly $\mathbf{e}$. Then, d' simulates perfectly $\mathbf{d}'$ as $\perp$ is returned if it is queried on $(\mathsf{sk}^*, \cdot)$. Similarly, u' perfectly simulates $\mathbf{u}$ by using $\mathscr{C}$'s own u oracle to reply to queries of the form $u'(\mathsf{pk}^*, \cdot)$. Finally, $w'(\mathsf{pk}^*, \cdot)$ perfectly simulates $\mathbf{w}$ when $w_b := e_1(r(\cdot))$ and perfectly simulates $\mathbf{w}^1$ when $w_b := e_0(r(\cdot))$, where r is a random function. Indeed, when $\mathscr{C}$ plays the IMG game with $b = 1$, on a query $w'(\mathsf{pk}^*, \cdot)$ made by $\mathscr{B}$, $\mathscr{C}$ outputs a ciphertext with the first component set to $e_1(r(\cdot)) = e_{\mathsf{sk}_0^*}(r(\cdot))$ (i.e. the "encryption" of the first bit of sk^*, which is equal to 1). Similarly, when $\mathscr{C}$ plays the IMG game with $b = 0$, the returned ciphertext has a first component set to $e_0(r(\cdot))$, as in the $\mathbf{w}^1$ oracle. Hence, $\mathscr{C}$ playing the IMG game with bit $b = 1$ (resp. $b = 0$) perfectly simulates $\mathscr{B}$'s view with oracle $\mathbf{w}$ (resp. $\mathbf{w}^1$) and we have

$$\begin{aligned}
&\left|\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^*] - \Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^1,\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^*]\right| \\
&= \frac{1}{2}\left|\Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^* \middle| \mathsf{sk}_1^* = 1\right] - \Pr\left[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^1,\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^* \middle| \mathsf{sk}_1^* = 1\right]\right| \\
&= \left|\Pr[\mathscr{C}^{e_0,e_1,w_1,u} \Rightarrow 1] - \Pr[\mathscr{C}^{e_0,e_1,w_0,u} \Rightarrow 1]\right| = \mathsf{negl}(n)\ ,
\end{aligned}$$

where the last equality follows from Corollary 5.6.1.

<u>$\mathbf{w}^j$</u>: We successively modify the oracle $\mathbf{w}^1$ into an oracle $\mathbf{w}^j$, $j \in [n]$ s.t. on a query $(\mathsf{pk}^*,\cdot)$, the i-th first components of the resulting ciphertexts are encryptions of a 0 instead of the i-th bit of the challenge secret key. Formally, we have

$$\mathbf{w}^j(\mathsf{pk}, i) = \begin{cases} \mathbf{w}(\mathsf{pk}, i), & \text{if } \mathsf{pk} \neq \mathsf{pk}^* \\ (\ldots, e_0(r_{j,\mathsf{pk}}(i)), e_{\mathsf{sk}_{j+1}^*}(r_{j+1,\mathsf{pk}}(i))), \ldots), & \text{if } \mathsf{pk} = \mathsf{pk}^* \end{cases} .$$

By a similar reduction to the IMG problem as before, we have for all $j \in \{1, \ldots, n-1\}$

$$\left|\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^j,\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^*] - \Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^{j+1},\mathbf{u}}(\mathsf{pk}^*,\mathsf{ct}^*) \Rightarrow \mathsf{sk}^*]\right| = \mathsf{negl}(n)\ .$$

<u>$\mathbf{w}^n$</u>: Now, $\mathbf{w}^n(\mathsf{pk}^*,\cdot)$ returns a vector of ciphertexts encrypting 0 which means we do not use sk^* in $\mathbf{w}^n$ anymore. In order to conclude the proof of the lemma, we wish to show that

$$\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^n,\mathbf{u}}(g(\mathsf{sk}^*),\mathsf{ct}^*) \Rightarrow \mathsf{sk}^*] = \mathsf{negl}(n)\ .$$

One can see that the oracles $(\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^n,\mathbf{u})$ never invert pk^* or use the secret key sk^* anymore. The only exception is the decryption oracle that returns $\perp$ whenever sk^* is equal to the queried sk. However, this condition can be checked by verifying whether $g(\mathsf{sk}) = \mathsf{pk}^*$, as g is one-to-one. Hence, we are going to show that if $\mathscr{B}$ outputs sk^*, one can build an adversary $\mathscr{D}$ that inverts g on a random image, having access to a partial inverse oracle. We show the adversary in Figure 5.5. As in Lemma 5.6.2, $\mathscr{D}^{g,g^{-1}_{\neq \mathsf{sk}^*}}$ receives $g(\mathsf{sk}^*)$, where g is a random injective function, sk^* is sampled at random, and the goal is to recover sk^*. Note that sk^* and $\mathsf{pk}^* = g(\mathsf{sk}^*)$ are distributed as in $\mathscr{B}$'s game. Then, $\mathscr{D}$ generates a challenge ciphertext ct^* using pk^* and runs $\mathscr{B}$ while simulating the oracles $(\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w}^n,\mathbf{u})$ as follows.

- $g(\mathsf{sk})$: $\mathscr{D}$ uses its own g oracle to reply to $\mathscr{B}$'s queries to $\mathbf{g}$. As they are similarly distributed and $\mathsf{pk}^* = g(\mathsf{sk})^*$, the simulation is perfect.
- $e'(\mathsf{pk}, b, r)$: It simply returns the evaluation of $e(\mathsf{pk}, b, r)$, where $e(\mathsf{pk},\cdot,\cdot)$ is a random one-to-one function sampled by $\mathscr{D}$. This simulates perfectly $\mathbf{e}$.

$\mathscr{C}^{\mathscr{B}, e_0, e_1, w_b, u}$

$b' \leftarrow\$ \{0,1\}; \mathsf{sk}^* \leftarrow\$ \{0,1\}^n$
$\mathsf{sk}_0^* \leftarrow 1$
sample $g \leftarrow\$ \mathsf{Inj}_{n,3n}$
$\mathsf{pk}^* \leftarrow g(\mathsf{sk}^*)$
$\forall \mathsf{pk} \in \{0,1\}^{3n}$ s.t. $\mathsf{pk} \neq \mathsf{pk}^*$:
  sample $e(\mathsf{pk}, \cdot, \cdot) \leftarrow\$ \mathsf{Inj}_{n+1,3n}$
$\forall \mathsf{pk} \in \{0,1\}^{3n}, i \in [n]$:
  $r_{i,\mathsf{pk}} \leftarrow\$ \{f : \{0,1\}^n \mapsto \{0,1\}^n\}$
$r^* \leftarrow\$ \{0,1\}^n; \mathsf{ct}^* \leftarrow e_{b'}(r^*)$
run $\mathsf{sk}' \leftarrow\$ \mathscr{B}^{g, e', d', w', u'}(\mathsf{pk}^*, \mathsf{ct}^*)$
return $1_{\mathsf{sk}' = \mathsf{sk}^*}$

$e'(\mathsf{pk}, b, r)$

if $\mathsf{pk} = \mathsf{pk}^*$:
  return $e_b(r)$
return $e(\mathsf{pk}, b, r)$

$d'(\mathsf{sk}, \mathsf{ct})$

if $\mathsf{sk} = \mathsf{sk}^*$:
  return $\perp$
if $\exists (b, r)$ s.t. $e(g(\mathsf{sk}), b, r) = \mathsf{ct}$:
  return b
return $\perp$

$w'(\mathsf{pk}, i)$

if $\mathsf{pk} = \mathsf{pk}^*$:
  $r \leftarrow (w_b(i), e_{\mathsf{sk}_2^*}(r_{2,\mathsf{pk}}(i)), \ldots, e_{\mathsf{sk}_n^*}(r_{n,\mathsf{pk}}(i)))$
  return r
if $\exists \mathsf{sk}$ s.t. $g(\mathsf{sk}) = \mathsf{pk}$:
  $r \leftarrow (e(\mathsf{pk}, \mathsf{sk}_1, r_{1,\mathsf{pk}}(i)), \ldots, e(\mathsf{pk}, \mathsf{sk}_n, r_{n,\mathsf{pk}}(i)))$
  return r
return $\perp$

$u'(\mathsf{pk}, \mathsf{ct})$

if $\mathsf{pk} = \mathsf{pk}^*$:
  return $u(\mathsf{ct})$
if $\exists (b, r)$ s.t. $e(\mathsf{pk}, b, r) = \mathsf{ct}$:
  return $\top$
return $\perp$

Figure 5.4: $\mathscr{C}$ adversary.

- $d'(\mathsf{sk}, \mathsf{ct})$: It returns the decryption of ct or $\perp$ if $\mathsf{sk} = \mathsf{sk}^*$, as in the oracle $\mathbf{d}'$. Note that $\mathscr{D}$ uses its own oracle g to check whether $g(\mathsf{sk}) = \mathsf{pk}^*$.
- $w'(\mathsf{pk}, i)$: It simulates $\mathbf{w}^n$ perfectly. Indeed, if $\mathsf{pk} = \mathsf{pk}^*$ it returns a vector of ciphertexts encrypting 0. Otherwise, $\mathscr{D}$ uses its own $g^{-1}_{\neq \mathsf{sk}^*}$ to invert pk and encrypts the bits of the corresponding secret key.
- $u(\mathsf{pk}, \mathsf{ct})$: It simulates perfectly $\mathbf{u}$ as $\mathscr{D}$ uses its knowledge of $e(\mathsf{pk}, \cdot, \cdot)$ to check whether ct is a valid image.

Note that while the simulated oracles are described in a classical way, $\mathscr{D}$ implements them as quantum accessible oracles. This can be done with a polynomial number of quantum queries to its own oracles, as described before. Finally, we get

$$\Pr[\mathscr{B}^{\mathbf{g}, \mathbf{e}, \mathbf{d}', \mathbf{w}^n, \mathbf{u}}(g(\mathsf{sk}^*), \mathsf{ct}^*) \Rightarrow \mathsf{sk}^*] = \Pr[\mathscr{D}^{g, g^{-1}_{\neq \mathsf{sk}^*}}(g(\mathsf{sk}^*)) \Rightarrow \mathsf{sk}^*] = \mathsf{negl}(n) \,,$$

where the last equality follows from Lemma 5.6.2. Collecting the probabilities as in a standard hybrid argument concludes the proof of Lemma 5.7.1. □

Game Γ^2: We recall that Γ^1 is the IND-CPA game except the oracle $\mathbf{d}$ has been modified into

$\mathcal{D}^{\mathcal{B},g,g^{-1}_{\neq \mathsf{sk}^*}}(\mathsf{pk}^* = g(\mathsf{sk}^*))$

$b' \leftarrow\$ \{0,1\}$
$\forall \mathsf{pk} \in \{0,1\}^{3n}$:
  sample $e(\mathsf{pk},\cdot,\cdot) \leftarrow\$ \mathsf{Inj}_{n+1,3n}$
  $\forall i \in \{0,1\}^n$:
    $r_{i,\mathsf{pk}} \leftarrow\$ \{f : \{0,1\}^n \mapsto \{0,1\}^n\}$
$r^* \leftarrow\$ \{0,1\}^n; \mathsf{ct}^* \leftarrow e(\mathsf{pk}^*, b', r^*)$
$\mathsf{sk}' \leftarrow\$ \mathcal{B}^{g,e',d',w',u'}(\mathsf{pk}^*, \mathsf{ct}^*)$
return sk'

$e'(\mathsf{pk}, b, r)$

return $e(\mathsf{pk}, b, r)$

$d'(\mathsf{sk}, \mathsf{ct})$

if $g(\mathsf{sk}) = \mathsf{pk}^*$:
  return $\perp$
if $\exists (b, r)$ s.t. $e(g(\mathsf{sk}), b, r) = \mathsf{ct}$:
  return b
return $\perp$

$w'(\mathsf{pk}, i)$

if $\mathsf{pk} = \mathsf{pk}^*$:
  $r \leftarrow (e(0, r_{1,\mathsf{pk}}(i)), \ldots, e(0, r_{n,\mathsf{pk}}(i)))$
  return r
$\mathsf{sk} \leftarrow g^{-1}_{\neq \mathsf{sk}^*}(\mathsf{pk})$
if $\mathsf{sk} = \perp$: **return** $\perp$
$r \leftarrow (e(\mathsf{pk}, \mathsf{sk}_1, r_{1,\mathsf{pk}}(i)), \ldots, e(\mathsf{pk}, \mathsf{sk}_n, r_{n,\mathsf{pk}}(i)))$
return r

$u'(\mathsf{pk}, \mathsf{ct})$

if $\exists (b, r)$ s.t. $e(\mathsf{pk}, b, r) = \mathsf{ct}$:
  return $\top$
return $\perp$

Figure 5.5: $\mathcal{D}$ adversary.

an oracle $\mathbf{d}'$ that returns $\perp$ on a query $(\mathsf{sk}^*, \cdot)$. Now, we modify Γ^1 into a game Γ^2 by building another "encryption" oracle $\mathbf{e}'$ that returns $\perp$ whenever queried on (pk^*, b, r^*) for any bit b, where r^* is the randomness used to compute the challenge ciphertext (i.e. $\mathsf{ct}^* = \mathbf{e}(\mathsf{pk}^*, b, r^*)$). Formally,

$$\mathbf{e}'(\mathsf{pk}, b, r) = \begin{cases} \perp, & \text{if } \mathsf{pk} = \mathsf{pk}^* \wedge r = r^* \\ \mathbf{e}(\mathsf{pk}, b, r), & \text{otherwise} \end{cases} .$$

In addition, we modify $\mathbf{w}$ into a $\mathbf{w}'$ oracle s.t. it queries $\mathbf{e}'$ instead of $\mathbf{e}$. Note that as $\mathbf{w}$ (resp. $\mathbf{w}'$) encrypts n bits in parallel, one quantum query to $\mathbf{w}$ (resp. $\mathbf{w}'$) can be computed with n quantum queries to $\mathbf{e}$ (resp. $\mathbf{e}'$). Thus, in total, there are at most $q + qn$ queries made to $\mathbf{e}$ or $\mathbf{e}'$, where q is the number of queries made by $\mathcal{A}$. Then, Γ^2 is the same as Γ^1 except $\mathcal{A}$ has quantum oracle access to $\mathbf{e}'$ and $\mathbf{w}'$ instead of $\mathbf{e}$ and $\mathbf{w}$. As in the previous transition $\Gamma^0 \to \Gamma^1$, one can apply Lemma 5.4.1 to get

$$\begin{aligned} &\left| \Pr[\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b] - \Pr[\mathcal{A}^{\mathbf{g},\mathbf{e}',\mathbf{d}',\mathbf{w}',\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b] \right| \\ &\leq 2(q + qn)\sqrt{\Pr[\mathcal{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow r^*]} , \end{aligned}$$

where $(\mathsf{pk}^*, \mathsf{ct}^* = \mathbf{e}(\mathsf{pk}^*, b, r^*))$ is as in the IND-CPA game, and $\mathcal{B}$ runs $\mathcal{A}$ until some random quantum query q_i to $\mathbf{e}$ (made by $\mathcal{A}$ or $\mathbf{w}$), measures the input register and outputs the last n bits of the result. As before, we are going to upper bound the right-hand side of the equation by the probability a quantum adversary $\mathscr{F}$ inverts a random one-to-one length-tripling function with the help of a partial inverse oracle. This time, $\mathscr{F}$ will simulate queries of the form $\mathbf{e}(\mathsf{pk}^*, b, \cdot)$ using the function $e \in \mathsf{Inj}_{n,3n}$ it wants to invert. We show $\mathscr{F}$ in Figure 5.6. As in previous reductions, $\mathscr{F}$ samples its own functions $g, r_{\mathsf{pk},i}$ and $e(\mathsf{pk}, \cdot, \cdot)$ for $\mathsf{pk} \neq \mathsf{pk}^*$. Using these, it can reply consistently to $\mathscr{B}$'s queries that do not involve sk^* or pk^*. In addition, $\mathscr{F}$ samples a random challenge bit b' that plays the role of the challenge bit of the IND-CPA game. Then, it sets

$$e'(\mathsf{pk}^*, b, r) = \begin{cases} e(r), & \text{if } b = b' \\ e_{1-b'}(r), & \text{if } b = 1 - b' \end{cases},$$

where e is the function $\mathscr{F}$ wants to invert and $e_{1-b'} \in \mathsf{Inj}_{n,3n}$ is sampled by $\mathscr{F}$. Now, as both $e, e_{1-b'}$ are injective functions in $\mathsf{Inj}_{n,3n}$, the probability that $e'(\mathsf{pk}^*, \cdot, \cdot)$ is not a random function from $\mathsf{Inj}_{n+1,3n}$ is $\Pr[\mathsf{coll}] = \Pr[\mathsf{Im}(e) \cap \mathsf{Im}(e_{1-b'}) \neq \emptyset] = O(\frac{1}{2^n})$. Thus, assuming coll does not occur, $e'(\mathsf{pk}^*, \cdot, \cdot)$ follows the same distribution as $\mathbf{e}$. In addition, $\mathscr{F}$ can simulate perfectly $\mathbf{d}$ and $\mathbf{w}$ using its knowledge of sk^* and its own oracles/functions. In particular, each quantum query $\mathbf{w}(\mathsf{pk}^*, \cdot)$ can be simulated with at most n quantum queries to its oracle e. Finally, queries of the form $u'(\mathsf{pk}^*, \mathsf{ct})$ for some $\mathsf{ct} \in \{0,1\}^{3n}$ can be simulated perfectly, as:

- if $\mathsf{ct} = \mathsf{ct}^*$: $\mathscr{F}$ can return $\top$ as ct^* is a valid ciphertext.
- if $\mathsf{ct} \neq \mathsf{ct}^*$: $\mathscr{F}$ can query its oracle $e^{-1}_{\neq r^*}$ to check whether ct is a valid ciphertext of the form $\mathsf{ct} = e'(\mathsf{pk}^*, b', r)$, for some r. If that is not the case, $\mathscr{F}$ further checks whether $\mathsf{ct} = e'(\mathsf{pk}^*, 1 - b', r)$ for some r using its knowledge of $e_{1-b'}$.
- if the two previous conditions are not fulfilled, then ct is not a valid ciphertext.

Hence, if coll does not occur, $\mathscr{F}$ simulates perfectly $\mathscr{B}$'s view and we get

$$\Pr[\mathscr{B}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow r^*] \leq O\left(\frac{1}{2^n}\right) + \Pr[\mathscr{F}^{e, e^{-1}_{\neq r^*}}(e(r^*)) \Rightarrow r^*] = \mathsf{negl}(n)\,,$$

where the last equality follows from Lemma 5.6.2. Thus,

$$\left|\Pr[\mathscr{A}^{\mathbf{g},\mathbf{e},\mathbf{d}',\mathbf{w},\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b] - \Pr[\mathscr{A}^{\mathbf{g},\mathbf{e}',\mathbf{d}',\mathbf{w}',\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b]\right| = \mathsf{negl}(n).$$

Finally, we argue that

$$\Pr[\mathscr{A}^{\mathbf{g},\mathbf{e}',\mathbf{d}',\mathbf{w}',\mathbf{u}}(\mathsf{pk}^*, \mathsf{ct}^*) \Rightarrow b] = \frac{1}{2}\,.$$

Indeed, we recall that the challenge ciphertext is $\mathsf{ct}^* = \mathbf{e}(\mathsf{pk}^*, b, r^*)$, where $\mathbf{e}(\mathsf{pk}^*, \cdot, \cdot)$ is a random injective function and b is a random bit. Then, the decryption oracle $\mathbf{d}'$ is useless as $\mathbf{d}'(\mathsf{sk}^*, \cdot)$ returns $\bot$, thus $\mathscr{A}$ cannot invert ct^*. In addition, no oracle (i.e. $\mathbf{e}'$ or $\mathbf{w}'$) ever returns $\mathbf{e}(\mathsf{pk}^*, b, r^*)$ for any bit b (i.e. $\bot$ is returned in both cases). Finally, $\mathbf{u}(\mathsf{pk}^*, \mathbf{e}(\mathsf{pk}^*, b, r^*))$ returns $\top$ for both $b = 0$ and $b = 1$. Hence, given $\mathscr{A}$'s view, $\Pr[\mathsf{ct}^* = \mathbf{e}(\mathsf{pk}^*, 0, r^*)] = \Pr[\mathsf{ct}^* = \mathbf{e}(\mathsf{pk}^*, 1, r^*)]$ and $\mathscr{A}$ cannot distinguish. Therefore, $\Pr[\Gamma^2 \Rightarrow 1] = \frac{1}{2}$ and collecting the probabilities concludes the proof. $\square$

$\mathscr{F}^{\mathscr{B},e,e^{-1}_{\neq r^*}}(\mathsf{ct}^* = e(r^*))$

$b' \leftarrow\!\!\$\, \{0,1\}; \mathsf{sk}^* \leftarrow\!\!\$\, \{0,1\}^n$
sample $g \leftarrow\!\!\$\, \mathsf{Inj}_{n,3n}$
$\mathsf{pk}^* \leftarrow g(\mathsf{sk}^*)$
$\forall \mathsf{pk} \neq \mathsf{pk}^* \in \{0,1\}^{3n}$:
  sample $e(\mathsf{pk},\cdot,\cdot) \leftarrow\!\!\$\, \mathsf{Inj}_{n+1,3n}$
$\forall \mathsf{pk} \in \{0,1\}^{3n}, \forall i \in \{0,1\}^n$:
    $r_{i,\mathsf{pk}} \leftarrow\!\!\$\, \{f : \{0,1\}^n \mapsto \{0,1\}^n\}$
$e_{1-b'} \leftarrow\!\!\$\, \mathsf{Inj}_{n,3n}$
$\mathsf{sk}' \leftarrow\!\!\$\, \mathscr{B}^{g,e',d',w',u'}(\mathsf{pk}^*,\mathsf{ct}^*)$
return sk'

$e'(\mathsf{pk}, b, r)$

if $\mathsf{pk} = \mathsf{pk}^*$:
  if $b = b'$: **return** $e(r)$
  else : **return** $e_{1-b}(r)$
return $e(\mathsf{pk}, b, r)$

$d'(\mathsf{sk}, \mathsf{ct})$

if $\mathsf{sk} = \mathsf{sk}^*$:
  return $\perp$
if $\exists (b, r)$ s.t. $e(g(\mathsf{sk}), b, r) = \mathsf{ct}$:
  return b
return $\perp$

$w'(\mathsf{pk}, i)$

if $\exists \mathsf{sk}$ s.t. $g(\mathsf{sk}) = \mathsf{pk}$:
  $r \leftarrow (e'(\mathsf{pk}, \mathsf{sk}_1, r_{1,\mathsf{pk}}(i)), \ldots, e'(\mathsf{pk}, \mathsf{sk}_n, r_{n,\mathsf{pk}}(i)))$
  return r
return $\perp$

$u'(\mathsf{pk}, \mathsf{ct})$

if $\mathsf{pk} = \mathsf{pk}^*$:
  if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\top$
  if $e^{-1}_{\neq r^*}(\mathsf{ct}) \neq \perp$: **return** $\top$
  if $\exists r$ s.t. $e_{1-b'}(r) = \mathsf{ct}$: **return** $\top$
  return $\perp$
if $\exists (b, r)$ s.t. $e(\mathsf{pk}, b, r) = \mathsf{ct}$:
  return $\top$
return $\perp$

Figure 5.6: $\mathscr{F}$ adversary.

Corollary 5.7.1. *Let* $\mathsf{PKE_q}^{\mathscr{O}} = (g^{\mathbf{g}}, \mathbf{e}, \mathbf{d})$ *be a PKE relative to* $\mathscr{O}$*, where* $g^{\mathbf{g}}(s)$ *sets* $\mathsf{sk} \leftarrow s$ *and returns* $(\mathsf{sk}, \mathbf{g}(\mathsf{sk}))$*. Then, we have*

$$\Pr_{\mathscr{O} \leftarrow\$ \Psi}\left[\forall \mathsf{EFF_q}\ \mathscr{A} : \mathsf{Adv}^{\text{ind-cpa}}_{\mathscr{A}^{\mathscr{O}}, \mathsf{PKE_q}^{\mathscr{O}}} = \mathsf{negl}(n)\right] = 1$$

where $\mathsf{EFF_q}$ *stands for "efficient quantum". In other words, for measure* 1 *of oracles,* $\mathsf{PKE_q}^{\mathscr{O}}$ *is* IND-CPA *secure.*

Proof. This follows from a now standard trick in impossibility results based on the Borel-Cantelli lemma, Markov inequalities, and a counting argument (e.g. see Lemma 2 and 5 by Buldas et al. [BN13]). Note, however, that for the proof to work, the set of efficient quantum adversaries must be countable. This is the case here, as we consider *uniform* quantum circuits, which are countable (as they can be generated by deterministic Turing Machines). □

5.8 Non-existence of IND-CCA PKE

We first recall which type of constructions we will rule out, namely *shielding constructions.*

Definition 5.8.1 (Shielding construction)**.** *A valid PKE construction relative to* $O = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ $\mathsf{PKE}^O = (\mathsf{Gen}^O, \mathsf{Enc}^O, \mathsf{Dec}^O)$ *is* shielding *iff the decryption function* Dec *never queries the oracles* $\mathbf{e}$*. In other words, we can write* $\mathsf{PKE}^O = (\mathsf{Gen}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Enc}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Dec}^{\mathbf{g},\mathbf{d}})$.

Informally, the decryption function of a PKE resulting from a shielding transform never queries the encryption function of the underlying PKEs.

Now, in order to complete the proof of the impossibility result, we need to show that any shielding black-box construction

$$\mathsf{PKE}^O = (\mathsf{Gen}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Enc}^{\mathbf{g},\mathbf{e},\mathbf{d}}, \mathsf{Dec}^{\mathbf{g},\mathbf{d}})$$

is not IND-CCA secure. We can simply reuse Gertner et al.'s result [GMM07], as they showed there exists a classical IND-CCA adversary that breaks any shielding PKE construction. This implies that there is such a quantum adversary as well.

This is stated in the following theorem.

Theorem 5.8.1 (Theorem 2 [GMM07])**.** *Let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be any shielding construction. Then, there exists a (non-efficient) adversary* $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ *making a polynomial number of queries to* (O, R) *s.t.*

$$\mathsf{Adv}^{\text{ind-cca}}_{\mathscr{A}^{O,R},\mathsf{PKE}^O} \geq 1 - \frac{1}{n},$$

where the probability of the advantage is taken over the randomness of the game and of the adversary, and the sampling of $(O, R) \leftarrow\$ \Psi$*, where* Ψ *is defined as in Definition 5.5.1.*

Proof sketch. We recall the idea of the proof here.

1. In the first step, the public keys $\mathbf{g}(\mathsf{sk})$ for some sk's embedded into the public key PK (which is output by Gen^O) are collected. In order to do this, the adversary executes $\mathsf{Enc}^O(PK, M, R)$ for many different M and R, collecting all pk in queries $\mathbf{e}(\mathsf{pk}, \cdot, \cdot)$ made by Enc. Obviously not all pk's possibly embedded in PK are recovered as some could never be used, but the useful ones (most likely) are. Indeed, the secret keys sk's that are going to be used in decryption should correspond to the public keys used in encryption. Thus, the main goal of the next steps will be to invert the public keys pk's that have been collected in this part.

2. In this step, the public keys corresponding to the IND-CPA scheme are inverted. This is the only part where the decryption oracle provided to the classical adversary in the IND-CCA game is used. The approximate idea is the following. Many ciphertexts $C = \mathsf{Enc}^O(PK, M, R)$ for a random bit M and coins R are generated. Then, the process is repeated but in each encryption, some query $\mathbf{e}(\mathsf{pk}, b, r)$ (for some b and r) made by Enc is replaced by some value $\mathbf{e}(\mathsf{pk}, \mathsf{sk}_i, r')$ obtained through the $\mathbf{w}$ oracle, where

$\mathsf{pk} = \mathbf{g}(\mathsf{sk})$. Let C' be such a modified ciphertext and $C = \mathsf{Enc}^{O}(PK, M, R)$ the original one. Then, C' is queried to the decryption oracle to get $M' = \mathsf{Dec}^{O}(SK, C')$. We first observe that if $\mathsf{sk}_i = b$, then M' should be equal to M. Indeed, we replaced $\mathsf{ct} := \mathbf{e}(\mathsf{pk}, b, r)$ by $\mathsf{ct}' := \mathbf{e}(\mathsf{pk}, b, r')$, but since Dec cannot query $\mathbf{e}$, it cannot distinguish ct from ct′. Now we can distinguish two cases:

- $M \neq M'$: By the previous observation, it means that (most likely) $b \neq \mathsf{sk}_i$ and thus $\mathsf{sk}_i = 1 - b$ can be recovered, as b is known.
- $M = M'$: Either $\mathsf{sk}_i = b$ or the ciphertext corresponding to the modified query (or the decryption of the ciphertext) does not impact the decryption result. However, by repeating many times the experiment with different (M, R), it is possible to distinguish both cases with high probability and one can recover the corresponding bit of the secret key sk.

Note that if no ciphertext of the form $\mathbf{e}(\mathsf{pk}, \cdot, \cdot)$ ever impacts the decryption, the secret key sk s.t. $\mathbf{g}(\mathsf{sk}) = \mathsf{pk}$ will not be recovered using this technique. However, it also means that recovering such a secret key is not important as it is not used in decryption. Hence, after this step, all useful sk's should be recovered with high probability.

3. In the last step, using the knowledge of the secret keys recovered and of the queries made throughout the different experiments, the adversary builds a key SK' and simulates the decryption algorithm Dec^{O} using its own version $\widehat{\mathsf{Dec}}^{\hat{O}}$. Then, with high probability we will have $\widehat{\mathsf{Dec}}^{\hat{O}}(SK', C^*) = M^*$, where C^* is the challenge ciphertext and M^* the challenge bit of the IND-CCA game (remember we consider 1-bit PKEs). This step is the only non-efficient one, as the adversary needs to sample an oracle $\hat{O}$ consistent with the values observed in the previous step.

□

Corollary 5.8.1. *If* P = NP*, for measure one of oracles* (O, R)*, there exists an* efficient *adversary* $\mathscr{A}$ *that breaks the IND-CCA security of every shielding construction* $\mathsf{PKE}^{O} = (\mathsf{Gen}^{O}, \mathsf{Enc}^{O}, \mathsf{Dec}^{O})$.

Proof. This follows from Theorem 5.8.1 and the fact that the adversary defined in the proof is efficient if P = NP. Indeed, the adversary is efficient except in the last step, where it samples an oracle that must be consistent with the queries seen. Sampling such an oracle is equivalent to sampling an NP witness, which can be done efficiently if P = NP. More details can be found in the original proof [GMM07]. □

It follows that that disproving the previous result would imply proving P ≠ NP. However, we note that the assumption P=NP is not necessary. One can also embed a PSPACE oracle in the breaking oracle R, then the proof holds as $\mathrm{P}^{\mathrm{PSPACE}} = \mathrm{NP}^{\mathrm{PSPACE}}$.

The main result of this chapter then follows.

Theorem 5.8.2. *There is no post-quantum shielding black-box construction of IND-CCA PKE from IND-CPA PKEs.*

Proof. From Corollaries 5.7.1 and 5.8.1 we know that for measure one of oracles (O,R), IND-CPA PKEs exist but IND-CCA PKEs do not. Thus, there exists a tuple of oracle (O,R) s.t. IND-CPA PKEs exist but IND-CCA PKEs do not. Hence, the conditions for Lemma 5.4.3 to hold are fulfilled and that concludes the proof. $\square$

6 On IND-qCCA Security in the ROM and Its Applications: CPA Security Is Sufficient for TLS 1.3

We showed in the previous chapter that CCA-secure PKE or, equivalently, CCA-secure KEMs can only be obtained from CPA-secure PKE in a black-box way at the price of a re-encryption check. In addition to be quite costly in terms of computation time, we saw that this step can be tricky to implement properly, which can lead to security issues. Since it is impossible to remove it to get IND-CCA in a general way, a natural question is whether IND-CCA security is really necessary in the post-quantum protocols that are meant to replace existing classical ones.

For example, if we consider the flagship example of cryptographic protocols, namely TLS in its current version 1.3, we know that it is secure classically assuming the underlying primitives are themselves secure. However, as TLS relies on Diffie-Hellman, it would obviously be vulnerable to quantum adversaries. Therefore, in order to remedy this, several post-quantum variants of TLS 1.3 have been proposed. The simplest one is what we will call PQ TLS 1.3, which has been implemented as part of the OQS-OpenSSL project [SM23]. In this version, the changes compared to the standard version of the TLS 1.3 handshake are minimal. That is, the client's (resp. the server's) Diffie-Hellman share is replaced by a public key (resp. a ciphertext encapsulated under the public key), and the shared secret is the key encapsulated in the ciphertext. Several works have analysed the performance and implementation challenges of OQS-OpenSSL (e.g. [CPS19; PST20]).

More recently, based on the observation that (post-quantum) KEM public keys/ciphertexts are usually more compact than (post-quantum) public keys/signatures, Schwabe et al. [SSW20a] proposed KEMTLS as a variant of the TLS 1.3 handshake. The main difference between PQ TLS 1.3 and KEMTLS is that the latter uses a KEM for implicit server authentication instead of a signature. This reduces the overall bandwidth of the handshake and the computation time on the server-side. Thus, two KEMs are used in KEMTLS: one for establishing an ephemeral shared secret and the other one to authenticate the server. While the latter KEM needs to be IND-CCA secure as it uses long-term keys, the authors showed that IND-1CCA security of the former is sufficient for the whole handshake to be secure. That is, the KEM needs to be secure against an adversary that can make a *unique* decapsulation query. Similarly, in the security proof of TLS 1.3 handshake by Dowling et al. [Dow+20], DH key-exchange can be replaced by an IND-1CCA KEM and the proof would still go through.

Therefore, it seems that sometimes IND-1CCA KEMs are sufficient for security, and thus we focus on this primitive in this chapter. More specifically, we study whether IND-1CCA KEMs can be obtained from CPA-secure PKEs through a more efficient transform than FO-like ones (in the ROM and QROM). We reply in the affirmative by showing that IND-1CCA KEMs with much faster decapsulation than FO-derived IND-CCA KEMs can be obtained from any CPA-secure PKE. Using similar tools, we also study the security of the PQ TLS 1.3 handshake when the KEM used for key exchange is only CPA-secure.

The content of this chapter is a joint work with Serge Vaudenay and was published at EUROCRYPT 2022 [HV22].

6.1 Contributions

We show how to build an efficient IND-qCCA KEM (i.e. the adversary can only make q decapsulation queries with q constant) from any OW-CPA PKE in the (Q)ROM. The bound has a loss factor of 2^q, making it insecure or impractical for large q. However, such construction is sufficient to build an efficient IND-1CCA KEM from any OW-CPA public-key encryption scheme. The transform simply sends a confirmation hash along the ciphertext encrypting the seed. In addition, we prove the security of this construction in the QROM as well.

Such a transform might be useful in several applications such as the KEMTLS protocol [SSW20a] mentioned above, PQ variants of TLS 1.3, or ratcheting, as discussed in Section 6.6.

Similarly, we show that deriving the key as $K := H(m, \mathsf{ct})$, where m is the seed encrypted in the ciphertext ct, holds an IND-qCCA KEM in the ROM. The security bound is less tight compared to the first transform, having a $\approx q_H^{2q}$ factor, where q_H is the number of queries an adversary can make to the random oracle H. The intuition is that any decapsulation query that returns $H(m, \mathsf{ct})$ with $\mathsf{ct} \neq \mathsf{ct}^*$ does not help much the adversary to recover the real key $H(m^*, \mathsf{ct}^*)$ due to the independence of RO values. However, each query to the decapsulation oracle still leaks a little information (such as equality between decrypted values), leading to the $\approx q_H^{2q}$ factor.

Compared to the FO transform and its variants, our CPA-to-qCCA transforms offer several advantages. The main one is a significant speedup of the decapsulation, as there is no need for re-encryption. Depending on the cost of encryption of the underlying scheme, the difference can be large. For instance, as shown in Table 6.1 (we leave the benchmarks for SIKE for the sake of completeness), removing the re-encryption check in Kyber or Lightsaber cuts by more than 50% the decapsulation time on our setup. The speedup is even larger for Frodo ($> 6\times$), which has a slow underlying encryption procedure compared to the decryption. We also note that our transform does not perform de-randomisation of the underlying encryption (i.e. computing the random coins for encapsulation as the hash of the message/seed), removing

Scheme	Decaps with re-enc. (μs)	Decaps without re-enc. (μs)	Speedup
SIKE*	2316	1020	2.27
Kyber	6.1	2.8	2.17
Lightsaber	11.1	4.0	2.78
Frodo-AES	295.0	48.3	6.11

Table 6.1: Benchmark of Decaps with/without re-encryption with liboqs (avx2 enabled, NIST security level I) on Ubuntu 21.04, Intel Core i7-1165G7@2.8Ghz.*SIKE has been broken since the publication of this research [CD23].

the need for an additional random oracle.

Another interesting feature of the second transform (i.e. the one where the key is derived as $H(m, \mathrm{ct})$) is that the symmetric structure of the underlying KEM, if it exists, is preserved. That is, if the underlying KEM is a non-interactive key-exchange (NIKE), the scheme output by our transform will still be a NIKE. For instance, the IND-qCCA KEM derived from Diffie-Hellman or CSIDH [Cas+18] with our second transform will be a NIKE.

We then consider the PQ TLS 1.3 handshake as implemented in OQS-OpenSSL [SM23]. Based on the observation that the key-schedule computes the keys as key-derivation functions (KDFs) applied on the shared secret and (the hash of) the transcript so far (including the ciphertext), we prove that if the KEM is OW-CPA secure, then the handshake is secure in the MultiStage model of Dowling et al. [Dow+20]. The proof is inspired by the proof of security of our second transform. Note that this result holds in the ROM (the KDFs/hash function are assumed to be ROs) and the security bound is very much "non-tight". Still, this shows that CPA-secure KEMs are sufficient for the TLS 1.3 handshake to be secure, solving an open problem raised by several authors (e.g. [Dow+20; PST20]). Then, since one can consider DH as a KEM, this implies that TLS 1.3 is secure as long as the *computational Diffie-Hellman* (CDH) problem is hard, showing that the PRF-ODH assumption used in the original proof [Dow+20] is not necessary (in the ROM). We note that this last result can also be derived from the fact that DH as used in TLS 1.3 is a IND-1CCA KEM in the ROM, assuming that CDH is hard. We prove this in Appendix A.

Finally, in Section 6.6, we discuss possible use cases of IND-qCCA in the context of communication protocols and ratcheting primitives. In particular, we note that IND-1CCA security is sufficient in many recent applications as the trend is to move to forward secure schemes, which discard key pairs after one use.

Remark on IND-CPA vs IND-1CCA

We note that plain IND-CPA PQ schemes are often not IND-1CCA. In particular, it is stated in Section 4.3 of the KEMTLS paper [SSW20b]:

"We leave as an open question to what extent non-FO-protected post-quantum KEMs may be secure against a single decapsulation query, but at this point IND-CCA is the safe choice."

The answer to this question obviously depends on how the "non-FO protected" IND-CPA PKE is used as a KEM. However, if it is used in the trivial way (i.e. $m \leftarrow\$ \mathcal{M}$, $K := H(m), \mathsf{ct} := \mathsf{Enc}(\mathsf{pk}, m)$), the resulting KEM can usually be broken with 1 query for most of the PQ schemes. The adversary receives $K^*, \mathsf{ct}^* := \mathsf{Enc}(\mathsf{pk}, m^*)$, queries $\mathsf{ct}^* + \delta$ and gets back $H(m^*)$ with high probability, if δ is "small". Then, it can just compare whether $H(m^*) = K^*$ or not and break IND-1CCA security. The reaction attacks (like the ones presented in Chapter 3) requiring thousands of queries mentioned in the same paper [SSW20b] are key-recovery attacks, not distinguishing attacks. The simple distinguishing adversary given above actually gives a good intuition of why adding a confirmation hash $H'(m, \mathsf{ct})$ along the ciphertext as in our first transform yields an IND-qCCA KEM. In order to submit a valid decapsulation query, the adversary must compute $H'(m, \mathsf{ct})$ with $\mathsf{ct} \neq \mathsf{ct}^*$. Hence, the adversary itself needs to query $H'(m, \mathsf{ct})$ beforehand, thus it knows m and the decapsulation query is (nearly) useless.

6.2 Related Work

The notion of bounded IND-CCA (i.e. IND-qCCA) has been studied in several works. Bellare et al. [BS99] defined the notion of indistinguishability under parallel attack (IND-PA), which can be seen as a generalisation of IND-1CCA, where the adversary can submit *once* a vector of ciphertexts to a decryption oracle. Cramer et al. [Cra+07] defined IND-qCCA and showed that one can build an IND-qCCA PKE from any CPA-secure PKE in a black-box manner in the standard model, using one-time signatures. While this construction is valid in the standard model and ours in the ROM only, their reduction is inefficient compared to FO transforms, which we aim to improve. Following their work, Peirera et al. [Per+10] built a more efficient IND-qCCA PKE based on the CDH assumption and Yamakawa et al. [Yam+15] proposed other constructions based on the factoring and bilinear CDH assumptions. As far as we know, we are the first to note that a IND-qCCA KEM can be obtained from any CPA-secure PKE through a very simple and efficient transform in the ROM.

Starting from the original Fujisaki-Okamoto transform [FO99; FO13], many works have been dedicated to building variants of FO with tighter security bounds in the QROM (e.g. [HHK17; Bin+19b; Kuc+20; SXY18]). While these are CPA-to-CCA transforms, ours guarantee qCCA security only but at a lesser computational cost.

Dowling et al. [Dow+20] proved the security of the standard TLS 1.3 handshake in their MultiStage security model. We extend their result by showing that TLS 1.3 security still holds if the DH key-exchange is replaced by a CPA-secure KEM (in the ROM). In turn, this also implies that the CDH assumption is sufficient for proving the security of the original TLS 1.3, which was based on the PRF-ODH assumption so far. In two more recent papers, Diemert and Jager [DJ21] and Davis and Günther [DG21] aimed at proving a tighter security bound for

$\text{IND-qCCA}_{\mathsf{KEM}}(\mathscr{A})$	$\text{IND-qCCA'}^{b}_{\mathsf{KEM}}(\mathscr{A})$	**Oracle** $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct})$
$i \leftarrow 0$	$i \leftarrow 0$	**if** $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$	$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$	**if** $i = q$: **return** $\perp$
$b \leftarrow\!\!\$\ \{0,1\}$	$\mathsf{ct}^*, K_0 \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$	$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{ct}^*, K_0 \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk})$	$K_1 \leftarrow\!\!\$\ \mathscr{K}$	$i \leftarrow i + 1$
$K_1 \leftarrow\!\!\$\ \mathscr{K}$	$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}}(\mathsf{pk}, \mathsf{ct}^*, K_b)$	**return** K'
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}}(\mathsf{pk}, \mathsf{ct}^*, K_b)$	**return** b'	
return $1_{b'=b}$		

Figure 6.1: Equivalent IND-qCCA games and the decapsulation oracle.

TLS 1.3. Their proofs are valid in the ROM and are based on the Strong Diffie-Hellman (SDH) assumption. Our result on TLS 1.3 is complementary to theirs in the sense that we prove that TLS security holds under a weaker assumption but with a looser security bound.

Brendel et al. [Bre+17] studied the PRF-ODH assumption. In particular, they showed that PRF-ODH is hard if the SDH assumption holds in the ROM. The PRF-ODH notion considered in their work is generic as the adversary can query two types of “decapsulation” oracles multiple times. On the other hand, if we restrict ourselves to the notion where the adversary can make a unique query (which is sufficient for TLS 1.3 security), we show in Appendix A that CDH hardness is sufficient.

Finally, following the KEMTLS paper [SSW20a], several variants of the protocol (e.g. [SSW21; Gün+22]) as well as a post-quantum replacement of X3DH [Bre+22] have been using IND-1CCA KEM as a building block, showing the growing importance of such a notion.

6.3 IND-qCCA KEM

We first need to formally define the notion of IND-qCCA, which is defined as IND-CCA (see Definition 2.2.10) but the adversary is limited to q queries in the game.

Definition 6.3.1 (KEM IND-qCCA)**.** *We consider the games induced by the pseudocode on the left in Figure 6.1. A KEM scheme* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is IND-qCCA if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{ind-qcca}}_{\mathscr{A},\mathsf{KEM}} := \left| \Pr\left[\text{IND-qCCA}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

Equivalently, we can consider the game given in the middle in Figure 6.1. Then, a KEM scheme $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* IND-qCCA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{ind-qcca}'}_{\mathscr{A},\mathsf{KEM}} := \left| \Pr\left[\text{IND-qCCA'}^{1}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] - \Pr\left[\text{IND-qCCA'}^{0}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] \right| = \mathsf{negl}(\lambda).$$

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\, \mathsf{gen^P}(1^\lambda)$	$\sigma \leftarrow\!\$\, \mathscr{M}$	$(\mathsf{ct}_0', \mathsf{tag}') \leftarrow \mathsf{ct}$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct}_0 \leftarrow\!\$\, \mathsf{enc^P}(\mathsf{pk}, \sigma)$	$\sigma' \leftarrow \mathsf{dec^P}(\mathsf{sk}, \mathsf{ct}_0')$
	$\mathsf{tag} \leftarrow H'(\sigma, \mathsf{ct}_0)$	**if** $H'(\sigma', \mathsf{ct}_0') \neq \mathsf{tag}'$:
	$K \leftarrow H(\sigma)$	**return** $\perp$
	return $K, (\mathsf{ct}_0, \mathsf{tag})$	**return** $H(\sigma')$

Figure 6.2: $\mathsf{T_{CH}}$ transform.

6.4 OW-CPA to IND-qCCA Transforms

We first prove the following simple lemma, which states that a OW-CPA PKE is OW-PCA up to a factor 2^q, where q is the number of queries one can make to the plaintext-checking oracle.

Lemma 6.4.1. *Let* PKE *be a PKE. Then, for any efficient OW-PCA adversary* $\mathscr{A}$ *making at most* q *queries to the PCO oracle, there exists a OW-CPA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{A}) \leq 2^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) .$$

Proof. We can simply see that the PCO oracle returns 1 bit of information, thus PKE loses at most q bits of security when a PCO oracle is available. More formally, given $\mathscr{A}$, one can build $\mathscr{B}$ as follows. It passes its input to $\mathscr{A}$ and simulates the PCO oracle by sampling a response at random in $\{0,1\}$. Then, it returns the response of $\mathscr{A}$. Its probability of success is $\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) \geq \frac{1}{2^q} \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{A})$, as the probability the q responses are correct is $\frac{1}{2^q}$. □

We consider the transform $\mathsf{T_{CH}}$ given in Figure 6.2. This construction takes a PKE $\mathsf{PKE} = (\mathsf{gen^P}, \mathsf{enc^P}, \mathsf{dec^P})$ and outputs a KEM $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$. Note that $\mathsf{T_{CH}}$ is basically the REACT transform [OP01] without the asymmetric part (to get a KEM instead of a PKE).

We now show that the resulting KEM is IND-qCCA assuming the underlying PKE is OW-PCA.

Theorem 6.4.1. *We consider two random oracles* $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. *Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a* δ*-correct* PKE. *Then, for any IND-qCCA adversary* $\mathscr{A}$ *that makes at most* q_H *(resp.* $q_{H'}$*) queries to* H *(resp.* H'*), there exists a OW-PCA adversary* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathscr{A}) \leq \frac{(q + q_{H'} + 1)^2}{2^n} + \delta + \frac{q}{2^n} + (q_H + q_{H'}) \cdot \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{B}) ,$$

where $\mathscr{B}$ *makes at most* q *queries to its plaintext-checking oracle. In addition, if* PKE *is a deterministic encryption scheme, the bound becomes*

$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathscr{A}) \leq \frac{(q + q_{H'} + 1)^2}{2^n} + \delta + \frac{q}{2^n} + \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{B}) .$$

$\Gamma^{0-3}(\mathscr{A})$

$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\, \mathsf{Gen}(1^\lambda)$
$b \leftarrow\!\$\, \{0,1\}$
$\sigma^* \leftarrow\!\$\, \{0,1\}^n$
$\mathsf{ct}_0^* \leftarrow\!\$\, \mathsf{enc}^{\mathsf{P}}(\mathsf{pk},\sigma^*)$
$K_0 \leftarrow H(\sigma^*); h^* \leftarrow H'(\sigma^*,\mathsf{ct}_0^*)$
$K_1 \leftarrow\!\$\, \mathscr{K}$
$\mathsf{ct}^* \leftarrow (\mathsf{ct}_0^*, h^*)$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}},H,H'}(\mathsf{pk},\mathsf{ct}^*,K_b)$ // $\Gamma^{0\text{-}1}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec2}},H,H'}(\mathsf{pk},\mathsf{ct}^*,K_b)$ // $\Gamma^{2\text{-}3}$
if query : **abort** // Γ^3
return $1_{b'=b}$

$H(\sigma)$

if $\exists h$ s.t. $(\sigma,h) \in \mathscr{L}_H$:
  return h
if $\sigma = \sigma^*$: query $\leftarrow$ **true** // Γ^3
$h \leftarrow\!\$\, \{0,1\}^n$
$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{(\sigma,h)\}$
return h

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries :
  return $\perp$
$(\mathsf{ct}_0, h) \leftarrow \mathsf{ct}$
if $\mathsf{ct}_0 = \mathsf{ct}_0^*$ **or** $h = h^*$: // $\Gamma^{1\text{-}3}$
  return $\perp$ // $\Gamma^{1\text{-}3}$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{P}}(\mathsf{sk},\mathsf{ct}_0')$
if $H'(\sigma',\mathsf{ct}_0) \neq h$:
  return $\perp$
return $H(\sigma')$

$H'(\sigma,\mathsf{ct})$

if $\exists h$ s.t. $((\sigma,\mathsf{ct}),h) \in \mathscr{L}_{H'}$:
  return h
if $\sigma = \sigma^*$: // Γ^3
  query $\leftarrow$ **true** // Γ^3
$h \leftarrow\!\$\, \{0,1\}^n$
$\mathscr{L}_{H'} \leftarrow \mathscr{L}_{H'} \cup \{((\sigma,\mathsf{ct}),h)\}$
if $\exists x, x', h$ s.t. $x \neq x'$
    $\wedge\, (x,h) \in \mathscr{L}_{H'}$
    $\wedge\, (x',h) \in \mathscr{L}_{H'}$:
  abort
return h

Oracle $\mathscr{O}^{\mathsf{Dec2}}(\mathsf{ct})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries :
  return $\perp$
$(\mathsf{ct}_0, h) \leftarrow \mathsf{ct}$
if $\mathsf{ct}_0 = \mathsf{ct}_0^*$ **or** $h = h^*$:
  return $\perp$
if $\exists \sigma$ s.t. $((\sigma,\mathsf{ct}_0),h) \in \mathscr{L}_{H'}$:
  if $\mathscr{O}^{\mathsf{PCO}}(\sigma,\mathsf{ct}_0)$:
    return $H(\sigma)$
return $\perp$

Figure 6.3: Sequence of games for the proof of Theorem 6.4.1. $\mathscr{O}^{\mathsf{PCO}}$ is defined as in the OW-PCA game (see Figure 2.4).

Proof. We proceed by game hopping, the sequence of games is presented in Figure 6.3. Let $\mathscr{L}_H$ (resp. $\mathscr{L}_{H'}$) be the list of queries (x,h) made to the RO H (resp. H') s.t. $H(x) = h$ (resp. $H'(x) = h$). In addition, let the challenge ciphertext be $\mathsf{ct}^* = (\mathsf{ct}_0^*, h^*)$, and σ^* be s.t. $\mathsf{enc}^{\mathsf{P}}(\mathsf{pk},\sigma^*) = \mathsf{ct}_0^*$. We start with game Γ^0 which is the IND-qCCA game, except we abort if the adversary finds a collision on H' (i.e. $H'(x) = H'(x')$ for $x \neq x'$ and $(x,h),(x',h) \in \mathscr{L}_{H'}$). This happens with probability at most $\frac{(q+q_{H'}+1)^2}{2^n}$ and we have

$$\left|\Pr\left[\mathrm{IND-qCCA}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] - \Pr[\Gamma^0(\mathscr{A}) \Rightarrow 1]\right| \leq \frac{(q+q_{H'}+1)^2}{2^n} .$$

<u>Game Γ^1 :</u> The decapsulation oracle is modified s.t. it returns $\perp$ whenever ct_0^* or h^* is queried (note that both cannot be submitted at the same time). This game is the same as Γ^0 except if the oracle in Γ^0 does not return $\perp$ on such queries. Let bad be this event. We split this into two cases:

- $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct}_0^*, h \neq h^*) \neq \perp$. This happens only if

  $$H'(\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_0^*), \mathsf{ct}_0^*) = h \neq h^* = H'(\sigma^*, \mathsf{ct}_0^*) \ .$$

  In turn, this implies that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_0^*) \neq \sigma^*$ and thus it is a correctness error. Such an error happens with probability at most δ.

- $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct}_0 \neq \mathsf{ct}_0^*, h^*) \neq \perp$. It means that $h^* = H'(\sigma^*, \mathsf{ct}_0^*) = H'(\sigma', \mathsf{ct}_0)$, with $\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_0)$, which is not possible since $\mathsf{ct}_0 \neq \mathsf{ct}_0^*$ and we assume no collision occurs.

Therefore, overall $\Pr[\mathsf{bad}] \leq \delta$ and

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \Pr[\mathsf{bad}] \leq \delta \ .$$

<u>Game Γ^2 :</u> We modify the decapsulation oracle into another oracle $\mathcal{O}^{\mathsf{Dec}^2}$ as follows. On a decapsulation query (ct_0, h) (with $\sigma' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_0)$):

1. If there is no $((*, \mathsf{ct}_0), h)$ in $\mathcal{L}_{H'}$: return $\perp$. This differs from the previous game only if $h = H'(\sigma', \mathsf{ct}_0)$ but (σ', ct_0) was never queried to H'. As the RO values are uniformly distributed, this happens with probability at most $\frac{1}{2^n}$.
2. If $((\sigma, \mathsf{ct}_0), h) \in \mathcal{L}_{H'}$ for some σ: If $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0) := 1_{\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_0) = \sigma} = 1$, return $H(\sigma)$. Otherwise, return $\perp$. This perfectly simulates the previous oracle as $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0) = 1$ iff $\sigma = \sigma'$ and we know $h = H(\sigma = \sigma', \mathsf{ct}_0)$.
   Note that there is at most one σ s.t. $((\sigma, \mathsf{ct}_0), h) \in \mathcal{L}_{H'}$ as we assume no collision occurs. In particular, it means that $\mathcal{O}^{\mathsf{PCO}}$ is called at most once every decapsulation query.

Therefore, by a union bound we get

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \frac{q}{2^n} \ .$$

<u>Game Γ^3 :</u> Finally, we abort whenever $\mathcal{A}$ queries σ^* to H or $(\sigma^*, \cdot)$ to H'. Let this event be query. Note that $\mathcal{A}$ could also learn the value of $H(\sigma^*)$ through a query to $\mathcal{O}^{\mathsf{Dec2}}$. However, the latter oracle would return $H(\sigma^*)$ only if $\mathcal{A}$ queried $H'(\sigma^*, \cdot)$ before (thus triggering query).

Then, we can build a OW-PCA adversary $\mathcal{B}$ (shown in Figure 6.4) that perfectly simulates $\mathcal{A}$'s view as long as query does not happen. More precisely, $\mathcal{B}$ can simulate the decapsulation oracle using its PCO oracle. Then, on input $(\mathsf{pk}, \mathsf{ct}_0^*)$, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk}, (\mathsf{ct}_0^*, h^*), K^*)$, where h^* and K^* are picked at random. Unless query occurs, $\mathcal{A}$ cannot distinguish between these random

$\mathscr{B}^{\mathscr{O}^{\mathsf{PCO}}}(\mathsf{pk}, \mathsf{ct}^*)$

init $\mathscr{L}_H, \mathscr{L}_{H'} \leftarrow \emptyset$
$h^* \leftarrow\!\$\ \{0,1\}^n$
$K^* \leftarrow\!\$\ \{0,1\}^n$
simulate H, H' for $\mathscr{A}$ with lazy sampling:
run $\mathscr{A}^{H,H',\mathscr{O}^{\mathsf{Dec2}}}(\mathsf{pk}, (\mathsf{ct}^*, h^*), K^*)$
$\sigma' \leftarrow\!\$\ \{\sigma : \sigma \in \mathscr{L}_H^{\mathscr{A}} \vee (\sigma, *) \in \mathscr{L}_{H'}\}$
return σ'

Oracle $\mathscr{O}^{\mathsf{Dec2}}(\mathsf{ct})$

if more than q queries :
  return $\perp$
$(\mathsf{ct}_0, h) \leftarrow \mathsf{ct}$
if $\mathsf{ct}_0 = \mathsf{ct}_0^*$ **or** $h = h^*$:
  return $\perp$
if $\exists \sigma$ s.t. $((\sigma, \mathsf{ct}), h) \in \mathscr{L}_{H'}$:
  if $\mathscr{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct}_0)$: **return** $H(\sigma)$
return $\perp$

Figure 6.4: $\mathscr{B}$ adversary for the proof of Theorem 6.4.1.

h^*, K^* and the real ones. Finally, if query occurs, $\mathscr{B}$ can recover σ^* with probability $\frac{1}{q_H + q_{H'}}$ by sampling a random σ from $S = \{\sigma : (\sigma, *) \in \mathscr{L}_H^{\mathscr{A}} \vee ((\sigma, *), *) \in \mathscr{L}_{H'}\}$, where $\mathscr{L}_H^{\mathscr{A}}$ is the set of queries to H made by $\mathscr{A}$. Thus,

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq (q_H + q_{H'}) \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{B}) \,,$$

where $\mathscr{B}$ makes q query to the PCO oracle. Note that if PKE is deterministic, $\mathscr{B}$ can check whether $\mathsf{Enc}(\mathsf{pk}, \sigma) = \mathsf{ct}_0^*$ for all $\sigma \in S$ to find σ^*. This fails only if there exists $\sigma' \neq \sigma^*$ s.t. $\mathsf{Enc}(\mathsf{pk}, \sigma') = \mathsf{ct}_0^*$. In turn this implies that there exists $\sigma \in S \cup \{\sigma^*\}$ that would break correctness, but such an event is already covered by the previous δ factor. In this case, we obtain

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{B}) \,.$$

Finally, since $\mathscr{A}$ cannot query σ^* to H anymore, it cannot distinguish between a random key and $H(\sigma^*)$. Hence, $\Pr[\Gamma^3 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities concludes the proof. □

Corollary 6.4.1. *We consider two random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let KEM be the KEM resulting from applying the $\mathsf{T_{CH}}$ transform to a δ-correct PKE. Then, for any IND-qCCA adversary $\mathscr{A}$ that makes at most q_H (resp. $q_{H'}$) queries to H (resp. H'), there exists a OW-CPA adversary $\mathscr{B}$ s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathscr{A}) \leq \frac{(q + q_{H'})^2}{2^n} + \delta + \frac{q}{2^n} + (q_H + q_{H'} + q)2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathscr{B}) \,.$$

If PKE is deterministic, we get

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathscr{A}) \leq \frac{(q + q_{H'})^2}{2^n} + \delta + \frac{q}{2^n} + 2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathscr{B}) \,.$$

In particular, in the case of IND-1CCA (i.e. $q = 1$), if the underlying PKE is OW-CPA, then the KEM obtained from the $\mathsf{T_{CH}}$ transform is IND-1CCA with a security loss of ≈ 1 bit compared to the OW-CPA advantage (if we omit the other negligible terms). Finally, we note that as q is a constant that does not depend on the security parameter of the PKE, if the OW-CPA advantage of the PKE is negligible, so is the KEM IND-qCCA one. However, in practice, we would need to set the security parameter to a very large value to guarantee security for more than a few queries.

6.4.1 Security in the QROM.

We also show that the T_{CH} transform is secure in the Quantum Random Oracle Model (QROM) by proving that Theorem 6.4.1 holds in the QROM.

Theorem 6.4.2. *We consider two quantum random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let KEM be the KEM resulting from applying the T_{CH} transform to a PKE. Then, for any IND-qCCA adversary $\mathscr{A}$ that makes at most q_H (resp. $q_{H'}$) quantum queries to H (resp. H'), there exists a OW-PCA adversary $\mathscr{B}$ that makes at most q queries to its plaintext-checking oracle s.t.*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathscr{A}) \leq \delta + 2(q_{H'} + q_H + q)\sqrt{\mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-pca}}(\mathscr{B}) + \epsilon_1 + q((q_{H'} + 2q)\epsilon_2 + 2\epsilon_3)}\ ,$$

where $\epsilon_1 := \frac{40e^2(q_{H'}+q+1)^3+2}{2^n}$, $\epsilon_2 := 8\sqrt{2/2^n}$ and $\epsilon_3 := 2/2^n$.

Proof. Thanks to the use of the extractable RO-simulator (see Definition 2.3.1), the proof technique is very similar to the classical one and most of the QROM subtleties are abstracted away. The sequence of games is shown in Figure 6.5.

<u>Game Γ^0 :</u> This is the original IND-CCA game.

<u>Game Γ^1 :</u> The decapsulation oracle is modified s.t. it returns $\perp$ whenever $(\mathsf{ct}^*, \cdot)$ is queried to $\mathscr{O}^{\mathsf{Dec}}$ (note that $(\mathsf{ct}^*, \mathsf{tag}^*)$ cannot be queried). This game is the same as Γ^0 except when the oracle in Γ^0 does not return $\perp$ on such queries. Now, let's assume $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct}^*, \mathsf{tag} \neq \mathsf{tag}^*) \neq \perp$. This happens only if

$$H'(\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}^*), \mathsf{ct}^*) = \mathsf{tag} \neq \mathsf{tag}^* = H'(\sigma^*, \mathsf{ct}^*)\ .$$

In turn, this implies that $\mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}^*) \neq \sigma^*$ and thus the challenge ciphertext in the IND-CCA game would trigger a correctness error. Such an error happens with probability at most δ. Therefore, overall

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \leq \delta\ .$$

<u>Game Γ^2 :</u> We replace the challenge key K^* and tag tag^* by random values. As the key is now always random we have

$$\Pr[\Gamma^2 \Rightarrow 1] = \frac{1}{2}\ .$$

$\Gamma^{0-2}, \Upsilon^{0\text{-}7}(\mathscr{A})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$
$b \leftarrow_\$ \{0,1\}$
$\sigma^* \leftarrow_\$ \{0,1\}^n$
$K_0 \leftarrow H(\sigma^*); \mathsf{ct}^* \leftarrow_\$ \mathsf{enc}^{\mathsf{P}}(\mathsf{pk}, \sigma^*)$
$\mathsf{tag}^* \leftarrow H'(\sigma^*, \mathsf{ct}^*)$
$K_0 \leftarrow_\$ \{0,1\}^n; \mathsf{tag}^* \leftarrow_\$ \{0,1\}^n$ // $\Gamma^2, \Upsilon^{0\text{-}7}$
$K_1 \leftarrow_\$ \{0,1\}^n$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, H, H'}(\mathsf{pk}, (\mathsf{ct}^*, \mathsf{tag}^*), K_b)$ // $\Gamma^0\text{-}\Gamma^1$
return $1_{b'=b}$ // $\Gamma^0\text{-}\Gamma^2$
$\sigma' \leftarrow_\$ \mathsf{Ext}^{\mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, H, H'}}(\mathsf{pk}, (\mathsf{ct}^*, \mathsf{tag}^*), K_b)$ // $\Upsilon^{0\text{-}5}$
$\sigma' \leftarrow_\$ \mathsf{Ext}^{\mathscr{A}^{\mathscr{O}^{\mathsf{Dec2}}, H, H'}}(\mathsf{pk}, (\mathsf{ct}^*, \mathsf{tag}^*), K_b)$ // $\Upsilon^{6\text{-}7}$
for $i \in [q]$: $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) \leftarrow \mathsf{S.Ext}(\mathsf{tag}_i)$ // $\Upsilon^{1\text{-}2}$
for $i \in [q]$: $\sigma' \leftarrow \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct}_i); H'(\sigma', \mathsf{ct}_i)$ // Υ^7
return $1_{\sigma'=\sigma^*}$ // $\Upsilon^{0\text{-}7}$

$H(\sigma), H'(\sigma, \mathsf{ct})$

use standard QROs to reply // $\Gamma^0\text{-}\Gamma^1$
use two QROs $(H, H'_{\mathsf{ct}_0^*})$ and $H'_{\neq \mathsf{ct}_0^*}$: // Γ^2, Υ^0
use compressed oracle instead of $H'_{\neq \mathsf{ct}_0^*}$: // $\Upsilon^{1\text{-}7}$

Oracle $\mathscr{O}^{\mathsf{Dec}}(\mathsf{ct}, \mathsf{tag})$

$i \leftarrow$ query number
if $(\mathsf{ct}, \mathsf{tag}) = (\mathsf{ct}^*, \mathsf{tag}^*)$: **return** $\perp$
if $i > q$: **return** $\perp$
if $\mathsf{ct} = \mathsf{ct}^*$: // $\Gamma^1\text{-}\Gamma^2, \Upsilon^{0\text{-}5}$
  return $\perp$ // $\Gamma^1\text{-}\Gamma^2, \Upsilon^{0\text{-}5}$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct})$
$(\widehat{\sigma}, \widehat{\mathsf{ct}}) \leftarrow \mathsf{S.Ext}(\mathsf{tag})$ // Υ^5
$\mathsf{tag}' \leftarrow H'(\sigma', \mathsf{ct})$
$(\widehat{\sigma}, \widehat{\mathsf{ct}}) \leftarrow \mathsf{S.Ext}(\mathsf{tag})$ // $\Upsilon^{3\text{-}4}$
if $\mathsf{tag}' \neq \mathsf{tag}$: **return** $\perp$
if $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) = \perp$: // $\Upsilon^{4\text{-}5}$
  $\mathsf{bad}'_i \leftarrow$ **true**; **abort** // $\Upsilon^{4\text{-}5}$
if $(\sigma'_i, \mathsf{ct}_i) \neq (\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) \neq \perp$: // $\Upsilon^{2\text{-}5}$
  $\mathsf{bad} \leftarrow$ **true**; **abort** // $\Upsilon^{2\text{-}5}$
return $H(\sigma')$

Oracle $\mathscr{O}^{\mathsf{Dec2}}(\mathsf{ct}, \mathsf{tag})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries : **return** $\perp$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct})$ // Υ^6
$(\widehat{\sigma}, \widehat{\mathsf{ct}}) \leftarrow \mathsf{S.Ext}(\mathsf{tag})$
$\mathsf{tag}' \leftarrow H'(\sigma', \mathsf{ct})$ // Υ^6
if $(\widehat{\sigma}, \widehat{\mathsf{ct}}) \neq \perp$ **and** $\widehat{\mathsf{ct}} = \mathsf{ct}$ **and** $\mathscr{O}^{\mathsf{PCO}}(\widehat{\sigma}, \widehat{\mathsf{ct}})$:
  return $H(\widehat{\sigma})$
return $\perp$

Figure 6.5: Sequence of games for Theorem 6.4.2.

We now consider H' as two random oracles H'_{ct^*} and $H'_{\neq\mathsf{ct}^*}$, where the former is called on queries of the form $H'(\mathsf{ct}^*, \cdot)$, and the latter on queries of the form $H'(\mathsf{ct} \neq \mathsf{ct}^*, \cdot)$.

Then, by the OW2H lemma (Lemma 2.3.1) applied on (H, H'_{ct^*}) with F as in Figure 6.6, we have

$$\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1] \leq 2(q_{H'} + q_H + q) \cdot \sqrt{\Pr[\Upsilon \Rightarrow 1]}\,,$$

where Υ^0 is the same game as Γ^2, except that we measure the input register of a random quantum query made to H, H'_{ct^*} (by the adversary or the decapsulation oracle) and outputs 1 iff this is equal to the challenge seed σ^*. Note that the number of queries made to this oracle (i.e. to H and H'_{ct^*}) throughout the game is at most $q_{H'} + q_H + q$ as the adversary can make $q_{H'} + q_H$ queries to the oracles and the decapsulation oracle makes 1 query to H (and none to H'_{ct^*} due to the change in the previous game).

$\mathsf{F}(\sigma^*, (K^*, \mathsf{tag}^*))$

```
1 : (pk, sk) ←$ Gen(1^λ)
2 : ct* ←$ enc^P(pk, σ*)
3 : return (pk, (ct*, tag*), K*)
```

Figure 6.6: F function for applying the AOW2H lemma in the proof of Theorem 6.4.2.

Game Υ^1 : From now on, let $(\mathsf{ct}_1, \mathsf{tag}_1), \ldots, (\mathsf{ct}_q, \mathsf{tag}_q)$ be the queries to the decapsulation oracle, and $\sigma'_i := \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct}_i)$ be the decrypted seed from the i-th ciphertext. We modify Υ^0 s.t. the compressed oracle of the extractable RO-simulator S is used instead of the standard RO for $H'_{\neq \mathsf{ct}^*}$. For the sake of simplicity, we will refer to the compressed RO as $H'_{\neq \mathsf{ct}^*}$ (instead of $\mathsf{S}.H'_{\neq \mathsf{ct}^*}$). In addition, at the end of the game, we call the extractor on all tags $\mathsf{tag}_1, \ldots, \mathsf{tag}_q$, to get extracted values $(\widehat{\sigma}_1, \widehat{\mathsf{ct}}_1), \ldots, (\widehat{\sigma}_q, \widehat{\mathsf{ct}}_q)$. As the standard and compressed oracles are indistinguishable (Property 1 of Def. 2.3.1), and the extractor calls are made at the end of the game, this does not change anything to the outcome and we have

$$\Pr[\Upsilon^0 \Rightarrow 1] = \Pr[\Upsilon^1 \Rightarrow 1] \; .$$

Game Υ^2 : Let bad be the event that on *any* query $(\mathsf{ct}_i, \mathsf{tag}_i)$, the decapsulation oracle outputs no error (i.e. $H'(\sigma'_i, \mathsf{ct}_i) = \mathsf{tag}_i$) but the corresponding extracted values at the end are such that $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) \neq \perp$ and $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) \neq (\sigma'_i, \mathsf{ct}_i)$. Then, by Property 8 of Def. 2.3.1, we have that $\Pr[\mathsf{bad}] \le \epsilon_1$, where $\epsilon_1 := \frac{40e^2(q_{H'}+q+1)^3+2}{2^n}$. Now, let Υ^2 be the same as Υ^1, except we abort if bad happens. We get

$$\Pr[\Upsilon^1 \Rightarrow 1] - \Pr[\Upsilon^2 \Rightarrow 1] \le \epsilon_1 \; .$$

We note that in the game description in Figure 6.5, we check whether bad happens in the decapsulation oracle for the sake of presentation, even though it is not technically correct (i.e. the values $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i)$ are not defined at this time). This issue will disappear in the next game.

Game Υ^3 : We now move all extractions to the corresponding decapsulation query, just after the tag is verified. We have that moving each extraction to the decapsulation oracle implies at most $(q_{H'} + q)$ swaps with RO queries to H'. Thus, by Property 4 of Def. 2.3.1 we get

$$\Pr[\Upsilon^1 \Rightarrow 2] - \Pr[\Upsilon^1 \Rightarrow 1] \le q(q_{H'} + q)\epsilon_2 \; ,$$

where $\epsilon_2 := 8\sqrt{2/2^n}$.

Game Υ^4 : Let bad'_i be the event that on a query $(\mathsf{ct}_i, \mathsf{tag}_i)$, the decapsulation oracle outputs no error (i.e. $H'(\sigma'_i, \mathsf{ct}_i) = \mathsf{tag}_i$) but the corresponding extracted values at the end are such that $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) = \perp$. By Property 7 of Def. 2.3.1, this happens with probability at most $\epsilon_3 = 2 \cdot 2^{-n}$. Let

Υ^4 be the same as Υ^3 except we abort if bad'_i happens for any $i \in \{1, \ldots, q\}$. Then, we have

$$\Pr[\Upsilon^3 \Rightarrow 1] - \Pr[\Upsilon^4 \Rightarrow 1] \leq q\epsilon_3 .$$

<u>Game Υ^5 :</u> In the decapsulation oracle, we move the classical RO query made for tag verification after the extraction. By Property 4 of Def. 2.3.1, we have that

$$\Pr[\Upsilon^4 \Rightarrow 1] - \Pr[\Upsilon^5 \Rightarrow 1] \leq q\epsilon_2 .$$

<u>Game Υ^6 :</u> We modify the previous game such that after the extraction in the decapsulation oracle, we call a PCO oracle on the extracted values that returns a bit r that is 1 iff $\mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \widehat{\mathsf{ct}}_i) = \widehat{\sigma}_i$. Then, we modify the decapsulation oracle s.t. we return $H(\widehat{\sigma}_i)$ iff $r = 1$ *and* $\widehat{\mathsf{ct}}_i = \mathsf{ct}_i$. Otherwise, $\perp$ is returned. If the extracted values are null, $\perp$ is returned as well. Now we argue that the decapsulation oracle in Υ^5 returns identical outputs as the ones in the previous game. We split the analysis in two cases:

1. Assume $H(\sigma') \neq \perp$ is output by $\mathsf{O}^{\mathsf{Dec}}(\mathsf{ct}_i)$ in Υ^5. Since we assume bad and bad'_i do not occur, it means that $(\widehat{\sigma}_i, \widehat{\mathsf{ct}}_i) = (\sigma'_i, \mathsf{ct}_i)$, and thus the decapsulation oracle in Υ^6 returns $H(\sigma'_i) = H(\widehat{\sigma}_i)$.
2. Assume $H(\sigma') = \perp$ is output by $\mathsf{O}^{\mathsf{Dec}}(\mathsf{ct}_i)$ in Υ^5 (i.e. the tag verification failed). In addition, let's assume toward contradiction that $\mathsf{O}^{\mathsf{Dec}}(\mathsf{ct}_i)$ outputs $H(\widehat{\sigma}_i) \neq \perp$ in Υ^6. Since the checks passed, we know that $\widehat{\mathsf{ct}_i} = \mathsf{ct}_i$ and $\mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \widehat{\mathsf{ct}}_i) = \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct}_i) = \widehat{\sigma}_i = \sigma'_i$. By Property 6 of Def. 2.3.1, we know that $H'(\widehat{\sigma}_i, \widehat{\mathsf{ct}_i}) = H'(\sigma'_i, \mathsf{ct}_i) = \mathsf{tag}_i$ except with probability at most $\frac{2}{2^n} = \epsilon_3$. Hence, this contradicts the fact that the tag verification would have failed in $\mathsf{O}^{\mathsf{Dec}}(\mathsf{ct}_i)$ in Υ^5.

We also remove the original tag verification and logic, as these instructions do not influence on the probability of success. In the end, we get

$$\Pr[\Upsilon^5 \Rightarrow 1] - \Pr[\Upsilon^6 \Rightarrow 1] \leq q\epsilon_3 .$$

<u>Game Υ^7 :</u> In this last game, we move hash queries to H' made during tag verification in the decapsulation oracle to the end of the game. Note that the outputs of the decapsulation oracle are independent of the tag verification now, so we can apply Property 8 of Def. 2.3.1 again. We also remove the call to the decryption procedure as it is not useful anymore. Finally, we change the game s.t. the random query that is measured is taken uniformly at random from the queries made by the adversary to H or H', and the queries made by the decapsulation oracle to H. In other words, we forget about the queries to H' that have been moved at the end of the game. As these queries are classical and are never equal to $H'(\mathsf{ct}^*, \sigma^*)$, this does not lower the probability of Υ^7 to output 1 compared to the previous game.

$\mathsf{Gen}(1^\lambda)$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\$ \mathsf{gen}^\mathsf{P}(1^\lambda)$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^\mathsf{P}(\mathsf{sk},\mathsf{ct})$
return $(\mathsf{pk},\mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^\mathsf{P}(\mathsf{pk},\sigma)$	**if** $\sigma' = \perp$: **return** $\perp$
	$K \leftarrow H(\sigma,\mathsf{ct})$	**return** $H(\sigma',\mathsf{ct})$
	return K,ct	

Figure 6.7: $\mathsf{T_H}$ transform.

Thus, we have

$$\Pr[\Upsilon^6 \Rightarrow 1] - \Pr[\Upsilon^7 \Rightarrow 1] \le (q-1)q\epsilon_2$$

as we move at most q queries to H' and each is going to be swapped with at most $(q-1)$ calls to the extractor.

Now, one can see that a OW-PCA adversary $\mathscr{B}$ can perfectly simulate Υ^7 as both the challenge key K_b and tag tag^* are random, and the decapsulation oracle can be perfectly simulated with a plaintext-checking oracle (which is called at most q times). In addition, whenever Υ^7 outputs 1, $\mathscr{B}$ can recover σ^* from the query measurement. Hence, we have

$$\Pr[\Upsilon^7 \Rightarrow 1] \le \mathsf{Adv}^{\mathrm{ow-pca}}_{\mathsf{PKE}}(\mathscr{B}) \ .$$

Collecting the probabilities concludes the proof. □

Corollary 6.4.2. *We consider two quantum random oracles $H, H' : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_{CH}}$ *transform to a δ-correct* PKE*. Then, for any IND-qCCA adversary $\mathscr{A}$ that makes at most q_H (resp. $q_{H'}$) queries to H (resp. H'), there exists a OW-CPA adversary $\mathscr{B}$ s.t.*

$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathscr{A}) \le \delta + 2(q_{H'} + q_H + q)\sqrt{2^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathscr{B}) + \epsilon_1 + q((q_{H'} + 2q)\epsilon_2 + 2\epsilon_3)} \ ,$$

where $\epsilon_1 := \frac{40e^2(q_{H'}+q+1)^3+2}{2^n}$, $\epsilon_2 := 8\sqrt{2/2^n}$ *and* $\epsilon_3 := 2/2^n$.

6.4.2 Hashing the plaintext and ciphertext

One can also wonder what is the leakage of the decapsulation oracle in the ROM, when the key is simply the hash of the seed and the plaintext. That is, we consider the simple PKE to KEM transform given in Figure 6.7, which we call $\mathsf{T_H}$. Note that this is the same transform as the $\mathsf{U}^\perp$ transform from Hofheinz et al. [HHK17] presented in Figure 2.17. We now show that if q is small (logarithmic in the security parameter), then $\mathsf{T_H}$ outputs a secure IND-qCCA scheme in the ROM, given that the underlying PKE is OW-CPA.

Theorem 6.4.3. *We consider a random oracle $H : \{0,1\}^* \mapsto \{0,1\}^n$. Let* KEM *be the KEM resulting from applying the* $\mathsf{T_H}$ *transform to a δ-correct PKE* PKE *(which never queries H). Then, for any*

$\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$

sort $\mathcal{L}_H$ according to query order :
$\mathcal{L}_H = ((\sigma_i, \mathsf{ct}_i), K_i)_{i \in \{1,\ldots,|\mathcal{L}_H|\}}$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct})$
if $\sigma' = \perp$: **return** $\perp_d$
for $i \in \{1, \ldots, |\mathcal{L}_H|\}$:
  if $\mathsf{ct}_i = \mathsf{ct}$ **and** $\sigma' = \sigma_i$:
    return i
return $\perp$

Figure 6.8: $\mathcal{O}^{\mathsf{i}}$ oracle for the proof of Theorem 6.4.3.

IND-qCCA adversary $\mathcal{A}$ that makes at most q_H queries to H, there exists a OW-CPA adversary $\mathcal{B}$ s.t.

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq q_H \cdot ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) .$$

If PKE *is deterministic, we get*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) .$$

Proof. We start by defining an oracle $\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$ (see Figure 6.8). This oracle returns the index i s.t. $((\sigma_i, \mathsf{ct}_i), K_i) \in \mathcal{L}_H$ (we first sort $\mathcal{L}_H$ according to some fixed order) and $\mathsf{ct}_i = \mathsf{ct}$ and $\mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_i) = \sigma_i$. If such a i does not exist and $\mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct}_i) = \perp$ it returns $\perp_d$, otherwise it returns $\perp$.

Now we show how to simulate the IND-qCCA decapsulation oracle in the ROM, using $\mathcal{O}^i$ and $\mathcal{O}^{\mathsf{PCO}}$ only. The original (resp. modified) oracles $\mathcal{O}^{\mathsf{Dec}}$ and H (resp. $\mathcal{O}^{\mathsf{Dec}'}$ and H') are on the left (resp. right) in Figure 6.9. We now prove that any IND-qCCA adversary cannot distinguish between the real and modified oracles.

First, we show that the outputs of the ROs H and H' on any query (σ, ct) have the same distribution, given the adversary's view. We break this into four subcases:

- (σ, ct) was queried before to H (resp. H'): In this case, both H and H' return the value h returned on the previous similar query. Thus, we assume from now on that every RO query made by the adversary is unique.
- ct was never queried to the decapsulation oracle before: In this case, both H and H' return a random value h and store the query/response in $\mathcal{L}_H$.
- ct was queried to the decapsulation oracle before: In both cases (original and modified oracles) one can see that if the decryption of ct either fails or $\sigma' = \mathsf{dec}^{\mathsf{p}}(\mathsf{sk}, \mathsf{ct})$ is different from σ, then the output of the decapsulation oracle is independent of $H(\sigma, \mathsf{ct})$ (and $H'(\sigma, \mathsf{ct})$). In both cases, the ROs sample a fresh value (H' will do so because $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$ will output 0 in this case, as $\sigma \neq \sigma'$ or the ciphertext is not valid). Now, if ct decrypts

Oracle $\mathcal{O}^{\mathsf{Dec}}(\mathsf{ct})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries :
  return $\perp$
$\sigma' \leftarrow \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct})$
if $\sigma' = \perp$: **return** $\perp$
return $H(\sigma', \mathsf{ct})$

$H(\sigma, \mathsf{ct})$

if $\exists h$ s.t. $((\sigma, \mathsf{ct}), h) \in \mathcal{L}_H$:
  return h
$h \leftarrow\$ \{0,1\}^n$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{((\sigma, \mathsf{ct}), h)\}$
return h

Oracle $\mathcal{O}^{\mathsf{Dec}'}(\mathsf{ct})$

if $\mathsf{ct} = \mathsf{ct}^*$: **return** $\perp$
if more than q queries : **return** $\perp$
if $\exists K$ s.t. $(\mathsf{ct}, K) \in \mathcal{L}_K$:
  return K
$i \leftarrow \mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$
if $i = \perp_d$: **return** $\perp$
if $i \neq \perp$:
  $((\sigma_i, \mathsf{ct}_i), K_i) \leftarrow \mathcal{L}_H[i]$
  return K_i // return i-th valued returned by H'
$K \leftarrow\$ \{0,1\}$
$\mathcal{L}_K \leftarrow \mathcal{L}_K \cup \{(\mathsf{ct}, K)\}$
return K

$H'(\sigma, \mathsf{ct})$

if $\exists h$ s.t. $((\sigma, \mathsf{ct}), h) \in \mathcal{L}_H$:
  return h
if $\exists K$ s.t. $(\mathsf{ct}, K) \in \mathcal{L}_K$:
  if $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$:
    $\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{((\sigma, \mathsf{ct}), K)\}$
    return K
$h \leftarrow\$ \{0,1\}^n$
$\mathcal{L}_H \leftarrow \mathcal{L}_H \cup \{((\sigma, \mathsf{ct}), h)\}$
return h

Figure 6.9: Original and modified oracles for the proof of Theorem 6.4.3.

to σ, the original decapsulation oracle outputs $H(\sigma, \mathsf{ct})$. In the modified game, the decapsulation oracle outputs a random K. Indeed, as we assume (σ, ct) was never queried to H, $\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$ outputs $\perp$. Then, the modified RO will output the same K, as $\mathcal{O}^{\mathsf{PCO}}(\sigma, \mathsf{ct})$ will verify. In both cases, the ROs output the same value as the decapsulation oracle.

We now show that the decapsulation oracles $\mathcal{O}^{\mathsf{Dec}}$ and $\mathcal{O}^{\mathsf{Dec}'}$ are indistinguishable. Let ct be the queried ciphertext and $\sigma = \mathsf{dec}^{\mathsf{P}}(\mathsf{sk}, \mathsf{ct})$.

- $\mathsf{ct} = \mathsf{ct}^*$: both oracles return $\perp$.
- $\sigma = \perp$: Both oracles return $\perp$, as $\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$ returns $\perp_d$.
- $H(\sigma, \mathsf{ct})$ (resp. $H'(\sigma, \mathsf{ct})$) was never queried. Both oracles return a random value if ct was never queried, or a consistent value if it was. It is straightforward to see this is the case in the original oracle. In the modified oracle, as $H'(\sigma, \mathsf{ct})$ was never queried, we have $\mathcal{O}^{\mathsf{i}}(\mathcal{L}_H, \mathsf{ct})$ that returns $\perp$. Thus, the decapsulation oracle returns a random K if ct was not queried or a consistent K if it was.

$\mathscr{B}(\mathsf{pk}, \mathsf{ct}^*)$

```
init 𝓛_H, 𝓛_K ← ∅
init 𝓛_q ← []
K* ←$ 𝒦
run 𝒜^{H'', 𝒪^{Dec''}}(pk, ct*, K*)
sample random query (σ', ct') made to H''
return σ'
```

$H''(\sigma, \mathsf{ct})$

```
i_q ← query number
if ∃h s.t. ((σ, ct), h) ∈ 𝓛_H :
  return h
if ∃K s.t. (ct, K) ∈ 𝓛_K :
  if 𝓛_q[ct] = i_q :
    𝓛_H ← 𝓛_H ∪ {((σ, ct), K)}
    return K
h ←$ {0,1}^n
𝓛_H ← 𝓛_H ∪ {((σ, ct), h)}
return h
```

Oracle $\mathscr{O}^{\mathsf{Dec}''}(\mathsf{ct})$

```
if ct = ct* : return ⊥
if more than q queries : return ⊥
if ∃K s.t. (ct, K) ∈ 𝓛_K :
  return K
i ←$ {1, ..., q_H, ⊥, ⊥_d}
if i = ⊥_d: return ⊥
if i ≠ ⊥:
  (ct_i, K_i) ← 𝓛_H[i]
  return K_i        // return i-th valued returned by H''
K ←$ {0,1}
𝓛_K ← 𝓛_K ∪ {(ct, K)}
𝓛_q[ct] ←$ {0, ..., q_H}
return K
```

Figure 6.10: $\mathscr{B}$ adversary for the proof of Theorem 6.4.3.

- $H(\sigma, \mathsf{ct})$ (resp. $H'(\sigma, \mathsf{ct})$) was queried and it output K. Both oracles return K. In the modified decapsulation oracle, $\mathscr{O}^{\mathsf{i}}(\mathscr{L}_H, \mathsf{ct})$ will output a valid i s.t. $H'(\sigma_i, \mathsf{ct}) = h_i$ and h_i is returned. Thus, the answer is consistent with the RO.

Now we can prove the theorem by game hopping as before. We define Γ^0 as the original IND-qCCA game.

Game Γ^1 : We modify the original IND-qCCA game into another game Γ^1 where the random/decapsulation oracles are the modified ones (i.e. H' and $\mathscr{O}^{\mathsf{Dec}'}$) described above. As shown, both games are indistinguishable and thus

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| = 0 .$$

Game Γ^2 : We replace the challenge key by a random one, as in the previous proof. Then, similarly, the real key is indistinguishable from a random one unless $H(\sigma^*, \mathsf{ct}^*)$ is queried. We define this event as query and

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] .$$

We can upper bound this probability by the advantage of a OW-CPA adversary $\mathscr{B}$ against PKE. That is, given a IND-qCCA adversary playing game Γ^2, we build an adversary $\mathscr{B}$ as shown in Figure 6.10. One can see that if $\mathscr{B}$ was simulating $\mathscr{A}$ with the H' and $\mathscr{O}^{\mathsf{Dec}'}$ oracles (instead of its own oracles H'' and $\mathcal{O}^{\mathsf{Dec}''}$), the simulation would be perfect as long as query did not occur. Then, whenever query would happen, $\mathcal{B}$ would recover σ^* with probability $\frac{1}{q_H}$. Now $\mathcal{B}$ does not simulate the modified oracles perfectly but instead makes some guessing in its own oracles H'' and $\mathcal{O}^{\mathsf{Dec}''}$:

- $\mathcal{O}^{\mathsf{Dec}''}$: In line 5, i is picked at random instead of being the returned value of the $\mathcal{O}^{\mathsf{i}}$ oracle. On each query the simulation is perfect with probability $1/(q_H+2)$ and overall with probability $\frac{1}{(q_H+2)^q}$, as there are at most q queries to this oracle. In line 12, we associate a random index to each ct s.t. $(\mathsf{ct},*) \in \mathcal{L}_K$.
- H'': In line 5, when $(\mathsf{ct},*) \in \mathcal{L}_K$, instead of querying the plaintext-checking oracle we check whether the corresponding sampled index $\mathcal{L}_q[\mathsf{ct}]$ is equal to the query number. If it is, we reply with K s.t. $(\mathsf{ct},K) \in \mathcal{L}_K$ otherwise we proceed as before (i.e. as in H'). Let's assume w.l.o.g that each query to H'' is unique. For each ct s.t. $(\mathsf{ct},*) \in \mathcal{L}_K$, there can be at most one query (σ,ct) s.t. $\mathcal{O}^{\mathsf{PCO}}(\sigma,\mathsf{ct})$ returns 1 (it is when σ is the decryption of ct). Here, $\mathcal{B}$ guesses beforehand which query it is (or if no such query will be made) and gets the correct answer with probability $\frac{1}{q_H+1}$. Note that $\mathcal{B}$ needs to make one guess per query to $\mathcal{O}^{\mathsf{Dec}''}$ (not per query to H''). Overall, the probability H'' simulates correctly H' is $\frac{1}{(q_H+1)^q}$.

From this we can deduce that $\mathcal{B}$ correctly simulates Γ^2 with probability $\frac{1}{((q_H+1)(q_H+2))^q}$ and wins the OW-CPA game with probability at least $\frac{1}{q_H} \cdot \Pr[\mathsf{query}]$. Hence,

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq q_H \cdot ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) .$$

Note that when PKE is deterministic, in order to recover σ^*, $\mathcal{B}$ can check which σ' queried is s.t. $\mathsf{Enc}(\mathsf{pk}^*,\sigma') = \mathsf{ct}^*$ instead of guessing. This works as long as the challenge seed σ^* and queried seeds are correct. If that is not the case, one can build an adversary that wins the correctness game defined in Figure 2.2. Note that this adversary knows which will be the correct seed as it is given sk and the PKE is deterministic. As the correctness advantage is upper bounded by δ, we obtain that for deterministic PKEs the last inequality becomes

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) .$$

Finally, in game Γ^2, the challenge key is always random and thus $\Pr[\Gamma^2 \Rightarrow 1] = \frac{1}{2}$. Collecting the probabilities concludes the proof.

□

6.5 CPA-security Is Sufficient for TLS 1.3 in the ROM

We show in this section that a CPA-secure KEM is sufficient for the handshake in TLS 1.3 to be secure in the ROM. The security bound is very loose, but this still solves an interest-

$\text{IND-1CCA-MAC}_{\mathsf{KEM}}(\mathscr{A})$

$b \leftarrow\!\$\ \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$
$\mathsf{ct}^*, K^* \leftarrow\!\$\ \mathsf{Encaps}(\mathsf{pk})$
$n^* \leftarrow\!\$\ \{0,1\}^n$
$\mathsf{HS}^* \leftarrow G(K^*)$
$\mathsf{CHTS}_0 \leftarrow H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{SHTS}_0 \leftarrow H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{dHS}_0 \leftarrow H_3(\mathsf{HS}^*)$
$(\mathsf{CHTS}_1, \mathsf{SHTS}_1, \mathsf{dHS}_1) \leftarrow\!\$\ \{0,1\}^{3n}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, \mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}, G, \{H_i\}_{i\in[4]}}\big(\mathsf{pk}, \mathsf{ct}^*, n^*,$
$\qquad (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)\big)$
$\textbf{return } 1_{b'=b}$

Oracle $\mathscr{O}^{\mathsf{Dec}}((\mathsf{ct}, n))$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
if $K' = \perp$: **return** $\perp$
$\mathsf{HS}' \leftarrow G(K')$
$\mathsf{CHTS} \leftarrow H_1(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{tk_c} \leftarrow H_D(\mathsf{CHTS})$
$\mathsf{tk_s} \leftarrow H_D(\mathsf{SHTS})$
return $(\mathsf{tk_c}, \mathsf{tk_s})$

Oracle $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{HS}' \leftarrow G(K')$
$\mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{fk}_S \leftarrow H_4(\mathsf{SHTS})$
if $\mathsf{MAC.Vrf}(\mathsf{fk}_S, \mathsf{txt}, \mathsf{tag}) = \textbf{true}$:
  return HS'
return $\perp$

Figure 6.11: IND-1CCA-MAC game.

ing open problem. TLS 1.3 only supports DH key-exchange but it can be trivially modified to support KEMs as done in several PQ variants of TLS (e.g.[SM23; Cel+21]). That is, the client runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}$ and sends pk as its share (instead of g^x). Then, the server runs $K, \mathsf{ct} \leftarrow\!\$\ \mathsf{Encaps}(\mathsf{pk})$ and sends ct as its secret share (instead of g^y). Finally, the client runs $K \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$ and the shared secret is set to K. By abuse of language, we refer to this modified protocol as TLS 1.3 in what follows. An overview of this modified handshake is given in Figure 6.15.

6.5.1 IND-1CCA-MAC

In order to show that a CPA-secure KEM is sufficient for TLS 1.3 to be secure, we first introduce an intermediary notion of security for KEMs, called IND-1CCA-MAC. This security definition has no application and will serve only as a useful intermediary building block for the proof.

Definition 6.5.1 (IND-1CCA-MAC)**.** *We consider the games defined in Figure 6.11. Let $\mathscr{K}$ be the key space, G, H_1, H_2, H_3, H_4, and H_D be key-derivation functions with images in $\{0,1\}^n$, H_T be a hash function with images in $\{0,1\}^n$, and* MAC *a MAC scheme. A KEM scheme* $\mathsf{KEM} =$

$(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* IND-1CCA-MAC *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}_{KEM}^{\text{ind-1cca-mac}}(\mathscr{A}) := \left|\Pr\left[\text{IND-1CCA-MAC}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)\ ,$$

where $\Pr\left[\text{IND-1CCA-MAC}^{b}_{\mathsf{KEM}}(\mathscr{A}) \Rightarrow 1\right]$ *is the probability that* $\mathscr{A}$ *wins the* IND-1CCA-MAC *game defined in Figure 6.11.*

In this game, the adversary receives a challenge ciphertext encapsulating a key K, a nonce n^*, and either three secrets $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ derived from K through a key schedule, or three random secrets. Jumping ahead, these three values are computed (nearly) in the same way as their identically named counterparts in the modified TLS 1.3 protocol. The adversary has also access to two oracles that it can query *at most once.* The first is simply a decapsulation oracle that applies a key schedule (similar to TLS's) on the decapsulated key and returns two secrets $\mathsf{tk_c}$ and $\mathsf{tk_s}$. The second oracle takes a ciphertext (which must be different than the challenge ciphertext), a tag, and some data. Then, the ciphertext is decrypted to recover a secret HS' that is passed through a key schedule to get a MAC key fk_S. Finally, the oracle checks whether tag is a valid MAC on the data with the key fk_S. If this is the case it returns HS', otherwise it returns an error $\perp$. Informally, this last oracle outputs the root secret HS if the adversary can forge a valid tag corresponding to the tuple (ct, n). In the TLS proof, this will be used to argue that if a participant can send a valid tag, it should know the root secret HS.

6.5.2 OW-CPA implies IND-1CCA-MAC

First, we briefly define the notion of MAC unforgeability we will need.

Definition 6.5.2 (MAC EUF-0T)**.** *Let* $\mathsf{MAC} = (\mathsf{MAC.Vrf}, \mathsf{MAC.Tag})$ *be a message authentication code scheme (MAC). We say* MAC *is EUF-0T if for any efficient adversary* $\mathscr{A}$*,*

$$\mathsf{Adv}_{\mathsf{MAC}}^{\text{euf-0t}}(\mathscr{A}) := \Pr[\mathsf{MAC.Vrf}(K, M, T) = 1 : (M, T) \leftarrow\!\!\$\ \mathscr{A}; K \leftarrow\!\!\$\ \mathscr{K}]$$

is negligible in the security parameter, where the probability is taken over the sampling of the key and the randomness of the adversary.

We now prove that any OW-CPA KEM is also IND-1CCA-MAC secure in the ROM if the MAC used is EUF-0T secure. More precisely, the KDFs G, H_1, H_2, H_3, H_4, and H_D, and the hash function H_T in the IND-1CCA-MAC games are assumed to be ROs.

Theorem 6.5.1. *Let* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ *be a KEM. Let the KDFs and the hash function in the IND-1CCA-MAC game be modelled as random oracles. Then, for any efficient adversary* $\mathscr{A}$ *making at most* $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4} q_{H_D}, q_{H_T}$ *queries to* $G, H_1, H_2, H_3, H_4, H_D, H_T$ *respectively,*

$\Gamma^{0\text{-}6}_{\mathsf{KEM}}(\mathscr{A})$

$b \leftarrow\!\$ \{0,1\}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow\!\$ \mathsf{Gen}(1^\lambda)$
$\mathsf{ct}^*, K^* \leftarrow\!\$ \mathsf{Encaps}(\mathsf{pk})$
$n^* \leftarrow\!\$ \{0,1\}^n$
$\mathsf{HS}^* \leftarrow G(K^*)$
$\mathsf{CHTS}_0 \leftarrow H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{SHTS}_0 \leftarrow H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$
$\mathsf{dHS}_0 \leftarrow H_3(\mathsf{HS}^*)$
$(\mathsf{CHTS}_1, \mathsf{SHTS}_1, \mathsf{dHS}_1) \leftarrow\!\$ \{0,1\}^{3n}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, \mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}, H_1, H_2}(\mathsf{pk}, \mathsf{ct}^*, n^*,$
$\quad (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$ // Γ^0-Γ^3
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}'}, \mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}, H_1', H_2'}(\mathsf{pk}, \mathsf{ct}^*, n^*,$
$\quad (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$ // Γ^4-
if collision on H_T : **abort** // Γ^1-
if $\mathscr{A}$ queries $H_i(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $i \in [2]$ or $H_3(\mathsf{HS}^*)$:
$\quad$ **abort** // Γ^6
$\quad$ **if** $\mathscr{A}$ did not query $G(K^*)$: **abort** // Γ^5
return $1_{b'=b}$

Oracle $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$\mathsf{HS}' \leftarrow G(K')$; $\mathsf{SHTS} \leftarrow H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$
$\mathsf{fk}_S \leftarrow H_4(\mathsf{SHTS})$
if $\mathsf{SHTS} = \mathsf{SHTS}_b$: // Γ^2-
$\quad$ **abort** // Γ^2-
if $\mathsf{MAC.Vrf}(\mathsf{fk}_S, \mathsf{txt}, \mathsf{tag}) = \mathbf{true}$:
$\quad$ **if** $\mathscr{A}$ did not query $H_4(\mathsf{SHTS})$: // Γ^2-
$\quad\quad$ **abort** // Γ^2-
$\quad$ **if** $\mathscr{A}$ did not query $H_2(\mathsf{HS}', H_T(\mathsf{ct}, n))$: // Γ^3-
$\quad\quad$ **abort** // Γ^3-
$\quad$ **return** HS'
return $\perp$

$H_j(\mathsf{HS}, y)$, $j \in \{1,2\}$

if $\nexists(\mathsf{ct}, n)$ s.t. $((\mathsf{ct}, n), y) \in \mathscr{L}_{H_T}$: // Γ^1-
$\quad h \leftarrow\!\$ \{0,1\}^n$; **return** h // Γ^1-
usual lazy sampling

Figure 6.12: Games for the proof of Theorem 6.5.1. The adversary has access to all the other ROs G, H_3, H_4 and H_D, even if it is not explicited in the games. H_1', H_2' and $\mathscr{O}^{\mathsf{Dec}'}$ are defined in Figure 6.13.

there exists a OW-CPA adversary $\mathscr{B}$ s.t.

$$\mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathscr{A}) \leq \mathsf{Adv}^{\text{euf-0t}}_{\mathsf{MAC}}(\mathscr{B}) + \frac{3q_{H_1} + 4q_{H_2} + q_{H_3} + q_{H_4} + q_{H_D} + 1}{2^n}$$
$$+ \frac{(q_{H_T} + 4)^2}{2^n} + q_G (q_{H_1} + 2)^2 (q_{H_2} + 2)^3 \cdot \mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{KEM}}(\mathscr{C}),$$

where $\mathscr{B}$ has approximately the same running time as $\mathscr{A}$.

Proof. The first step of the proof is very similar to the proof of Theorem 6.4.3. Indeed, one can see that the decapsulation oracle outputs secrets that are computed as (a function of) $H_i(\mathsf{HS}, H_T(\mathsf{ct}, n))$, where H_i and H_T are ROs. Note that the only difference is that H_T is applied on (ct, n). However, as H_T is a RO, this difference will not matter much in the proof. Hence, as in Theorem 6.4.3, one can program the ROs s.t. the decapsulation oracle $\mathscr{O}^{\mathsf{Dec}}$ can be simulated without the secret key. In a second step, we show that the adversary can also simulate the $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}$ oracle with good probability. More precisely, let HS be the secret corresponding to the submitted ciphertext ct. Then, either $H_2(\mathsf{HS}, H_T(\mathsf{ct}, n))$ has been queried by the adversary, or it is very unlikely that $\mathscr{A}$ knows the MAC key fk_S. In the first case we can recover HS from the list of queries, and in the second we can return $\perp$ as most likely the MAC verification will fail.

We proceed with a sequence of games, which are given in detail in Figure 6.12.

Game Γ^0 : This is the original IND-1CCA-MAC game. From now on, we assume w.l.o.g. that each query to ROs are unique (i.e. they never repeat).

Game Γ^1 : We modify the previous game as follows. First, we abort if a collision on H_T occurs in the game. As there are at most $q_{H_T}+4$ queries to H_T in the game, a collision occurs with probability less than $\frac{(q_{H_T}+4)^2}{2^n}$. Then, on adversary's queries $H_j(\mathsf{HS}, y)$, $j \in \{1,2\}$, if $H_T(\mathsf{ct}, n) = y$ was never queried by $\mathscr{A}$ for some (ct, n), we mark y as *unpaired* and return a random value. The only way it differs from the previous game, is if a query $H_j(\mathsf{HS}, y)$ for an *unpaired* y is performed by the game (i.e. not by the adversary), either before or after y was marked as *unpaired.* Now, $\mathscr{A}$ does not get any information about values $H_T(\mathsf{ct}, n)$ from the game (or oracles), except a few values $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ (or values that depends on these), for some HS. Note that these values completely "hide" the result of the H_T query, as H_j is a RO. Hence, the best strategy for $\mathscr{A}$ to query $H_j(\mathsf{HS}, y)$ s.t. y is *unpaired* but is queried by the game at some point, is to try random values for y. As the game makes at most 2 queries to H_1 (one in the challenge part and one in the decapsulation oracle) and 3 queries to H_2 (one in the challenge part and one in each oracle), the probability that a random unpaired y is s.t. y was the result of a H_T query by the game at some point is at most $\frac{2}{2^n}$ for a H_1 call, and $\frac{3}{2^n}$ for a H_2 call. Overall, we have

$$|\Pr[\Gamma^0 \Rightarrow 1] - \Pr[\Gamma^1 \Rightarrow 1]| \le \frac{2q_{H_1} + 2q_{H_2}}{2^n} .$$

We note that this step ensures that on a query $H_j(\mathsf{HS}, y)$ one can recover a unique tuple (ct, n) s.t. $H_T(\mathsf{ct}, n) = y$, or a random value is returned.

Game Γ^2 : We modify the original game s.t. we abort whenever the MAC verification succeeds on the query $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ but $\mathsf{fk}_S := H_4(\mathsf{SHTS})$ was never queried, where $\mathsf{SHTS} := H_2(G(K), H_T(\mathsf{ct}, n))$ and $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$. If that is the case, it means the MAC key $\mathsf{fk}_S := H_4(\mathsf{SHTS})$ is indistinguishable from a random value for $\mathscr{A}$, but it managed to forge a valid tag. Thus, one can build an adversary $\mathscr{B}$ that breaks MAC unforgeability. More formally, $\mathscr{B}$ samples a pair of keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{Gen}$, generates a valid input for $\mathscr{A}$ and simulates the decryption oracle with the secret key. Then, when $\mathscr{A}$ submits $(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ to $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}$, $\mathscr{B}$ outputs $(\mathsf{txt}, \mathsf{tag})$ as a forgery. We also abort if the value SHTS computed in the oracle is s.t. $\mathsf{SHTS} = \mathsf{SHTS}_b$. As there are no collision on H_T and $(\mathsf{ct}, n) \neq (\mathsf{ct}^*, n^*)$, this happens with probability at most $\frac{1}{2^n}$. Then, we have

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \le \mathsf{Adv}^{\mathsf{euf-0t}}_{\mathsf{MAC}}(\mathscr{B}) + \frac{1}{2^n} .$$

Game Γ^3 : We abort whenever the MAC verification succeeds on the query $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$ but $H_2(G(K), H_T(\mathsf{ct}, n))$ was never queried, where $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$. By the previous game, it means that the adversary queried $\mathsf{SHTS} := H_2(G(K), H_T(\mathsf{ct}, n))$ to H_4 without having queried

Oracle $\mathcal{O}^{\mathsf{Dec}'}(\mathsf{ct}, n)$

if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
if more than 1 query : **return** $\perp$
$q_1 \leftarrow_\$ \{0, \ldots, q_{H_1}\}$
$q_2 \leftarrow_\$ \{0, \ldots, q_{H_2}\}$
$i \leftarrow \mathcal{O}^{\mathsf{i}}(\mathcal{L}_{H_1}, \mathsf{ct}, n)$
if $i = \perp_d$: **return** $\perp$
if $i \neq \perp$:
  // get i-th valued returned by H_1
  $((\mathsf{HS}_i, \mathsf{ct}_i, n_i), h_i) \leftarrow \mathcal{L}_{H_1}[i]$
  $\mathsf{CHTS} \leftarrow h_i$
else :
  $\mathsf{CHTS} \leftarrow_\$ \{0,1\}$
$\mathcal{L}_K^1 \leftarrow (\mathsf{ct}, n, \mathsf{CHTS})$
$i \leftarrow \mathcal{O}^{\mathsf{i}}(\mathcal{L}_{H_2}, \mathsf{ct}, n)$
if $i \neq \perp$:
  // get i-th valued returned by H_2
  $((\mathsf{HS}_i, \mathsf{ct}_i, n_i), h_i) \leftarrow \mathcal{L}_{H_2}[i]$
  $\mathsf{SHTS} \leftarrow h_i$
else :
  $\mathsf{SHTS} \leftarrow_\$ \{0,1\}$
$\mathcal{L}_K^2 \leftarrow (\mathsf{ct}, n, \mathsf{SHTS})$
return $(H_D(\mathsf{CHTS}), H_D(\mathsf{SHTS}))$

$H'_j(\mathsf{HS}, y),\ j \in \{1,2\}$

if $\nexists (\mathsf{ct}, n)$ s.t. $((\mathsf{ct}, n), y) \in \mathcal{L}_{H_T}$:
  $h \leftarrow_\$ \{0,1\}^n$; **return** h
set (ct, n) s.t. $((\mathsf{ct}, n), y) \in \mathcal{L}_{H_T}$
if $\mathcal{L}_K^j = (\mathsf{ct}, n, h)$ for some h :
  if $\mathsf{HS} = G(\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}))$:
    $\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{((\mathsf{HS}, \mathsf{ct}, n), h)\}$
    return h
$h \leftarrow_\$ \{0,1\}^n$
$\mathcal{L}_{H_j} \leftarrow \mathcal{L}_{H_j} \cup \{((\mathsf{HS}, \mathsf{ct}, n), h)\}$
return h

$\mathcal{O}^{\mathsf{i}}_G(\mathcal{L}, n, \mathsf{ct})$

sort $\mathcal{L}$ according to query order :
  $\mathcal{L} = ((\mathsf{HS}_i, \mathsf{ct}_i, n_i), h_i)_{i \in \{1, \ldots, |\mathcal{L}_H|\}}$
$K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
if $K' = \perp$: **return** $\perp_d$
$\mathsf{HS}' \leftarrow G(K')$
for $i \in \{1, \ldots, |\mathcal{L}|\}$:
  if $(\mathsf{ct}_i, n_i) = (\mathsf{ct}, n)$ **and** $\mathsf{HS}' = \mathsf{HS}_i$:
    return i
return $\perp$

Figure 6.13: Simulation of decapsulation and random oracles with sub-oracle $\mathcal{O}^{\mathsf{i}}_G$ for the proof of Theorem 6.5.1. Note that as we assume that each query to H_j is unique, H'_j does not check whether a query was previously made.

$H_2(G(K), H_T(\mathsf{ct}, n))$ beforehand. If we analyse what information $\mathcal{A}$ has about $\mathsf{SHTS} \neq \mathsf{SHTS}_b$ if it did not query $H_2(G(K), H_T(\mathsf{ct}, n))$, we see that the only potential "leakage" is from a decapsulation query that returns $\mathsf{tk_s} := H_D(\mathsf{SHTS})$, where H_D is a RO perfectly hiding SHTS.

Thus, the best strategy for $\mathcal{A}$ to find SHTS without querying H_2 is to query random values $x \in \{0,1\}^n$ to H_D or H_4 until it finds x s.t. $H_D(x) = \mathsf{tk_s}$ or $H_4(x) = \mathsf{fk}_S$. This happens with probability at most $\frac{q_{H_D} + q_{H_4}}{2^n}$. Hence, we have

$$|\Pr[\Gamma^2 \Rightarrow 1] - \Pr[\Gamma^3 \Rightarrow 1]| \leq \frac{q_{H_D} + q_{H_4}}{2^n} .$$

<u>Game Γ^4 :</u> We program both ROs H_1 and H_2 s.t. we can perfectly simulate the decapsulation oracle with an oracle $\mathcal{O}^{\mathsf{i}}_G$. This follows exactly the idea of the proof of Theorem 6.4.3. First, we introduce an oracle $\mathcal{O}^{\mathsf{i}}_G$ in Figure 6.13 that takes a list of RO queries, a nonce n, and a ciphertext ct, then checks whether $(G(K), H_T(\mathsf{ct}, n))$ (where K is the key encapsulated in ct) was ever

queried and if that is the case, the index of the corresponding query. This is exactly the same as the oracle $\mathcal{O}^{\mathsf{i}}$ in the proof of Theorem 6.4.3, except we query the decapsulated K to the RO G and there is the additional nonce. Thus, we can program the ROs H_j, $j \in \{1,2\}$ and simulate the (1-time) decapsulation oracle as shown in Figure 6.13.

The simulation works nearly as in the proof of Theorem 6.4.3. Let ct be the unique decapsulation query, $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$ and $\mathsf{HS} := G(K)$. For $j \in [2]$, the simulated decapsulation oracle checks whether $(G(K), H_T(\mathsf{ct}, n))$ was already queried to H_j using $\mathcal{O}^{\mathsf{i}}_G$, if that is the case it recovers the corresponding value, otherwise it means $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ was never queried by the adversary *nor* the challenger, as $(\mathsf{ct}, n) \neq (\mathsf{ct}^*, n^*)$. Thus it samples the hash value at random, queries it to H_D and returns it to the adversary.

The simulation of H_j is such that it is consistent with the values returned by the simulated decapsulation oracle. First, if $H_j(\mathsf{HS}, y)$ is queried s.t. y is *unpaired*, we can simply return a random value, this is consistent with the game. Then, if y is not *unpaired*, one can recover the unique (as there are no collision) tuple (ct, n) s.t. $y = H_T(\mathsf{ct}, n)$. We consider from now on only queries with y s.t. $H_T(\mathsf{ct}, n) = y$ for some (ct, n). On a query $H_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$, if (ct, n) was already queried to the decapsulation oracle, then $h := H_j(\mathsf{HS}, \mathsf{ct}, n)$ was set by $\mathcal{O}^{\mathsf{Dec}'}$ iff $\mathsf{HS} = G(K)$, where $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$. Hence, we return the same K if $G(\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})) = \mathsf{HS}$. Otherwise we sample a random value and return it. Note that this is the only place where the secret key sk is used anymore (except implicitly in the $\mathcal{O}^{\mathsf{i}}_G$ oracle). The simulation is perfect and therefore we have

$$|\Pr[\Gamma^3 \Rightarrow 1] - \Pr[\Gamma^4 \Rightarrow 1]| = 0 .$$

<u>Game Γ^5 :</u> In game Γ^5, we abort whenever the adversary did not query $G(K^*)$ (which is equal to HS^*) but it queried $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$. Note that the (modified) decryption oracle never queries $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$. In addition, the challenge values given to $\mathcal{A}$ are either perfectly random or completely hide HS^*. Thus, the probability that $\mathcal{A}$ queries HS^* to H_1, H_2 or H_3 is upper bounded by $\frac{q_{H_1}+q_{H_2}+q_{H_3}}{2^n}$ and hence we have

$$|\Pr[\Gamma^4 \Rightarrow 1] - \Pr[\Gamma^5 \Rightarrow 1]| \leq \frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n} .$$

<u>Game Γ^6 :</u> Finally, in game Γ^6 we abort whenever $H_1(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$, $H_2(\mathsf{HS}^*, H_T(\mathsf{ct}^*, n^*))$ or $H_3(\mathsf{HS}^*)$ is queried by the adversary. Let query be this event. By the previous game, it means that K^* was queried to G before query happens. Finally, as in the previous proofs, we can upper bound $\Pr[\mathsf{query}]$ by the advantage of a OW-CPA adversary times a constant. The challenge keys $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ are sampled at random in the reduction, as long query does not happen both the real and random cases are perfectly indistinguishable. We present such a OW-CPA adversary $\mathcal{C}$ in Figure 6.14. The only challenge for $\mathcal{C}$ is to simulate the oracles without having access to the secret key.

- $\mathcal{O}^{\mathsf{Dec}''}_{\mathsf{MAC}}$: This oracle returns something else than $\perp$ iff $(\mathsf{HS}, H_T(\mathsf{ct}, n))$ was queried to H_2, where $\mathsf{HS} := G(K)$ and $K := \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$. Hence, one can simply pick a random value $r \leftarrow\$ \{0, \ldots, q_{H_2}\}$ and guess whether $\mathcal{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct})$ fails (if $r = 0$) or succeeds and HS is in the r-th query made to H_2. In the latter case, one can recover HS in the r-th query and return it. Overall the simulation works with probability $\frac{1}{q_{H_2}+1}$.

- $\mathcal{O}^{\mathsf{Dec}''}$: In this oracle, the secret key is used only in the $\mathcal{O}^{\mathsf{i}}_G$ sub-oracle. A reply of $\mathcal{O}^{\mathsf{i}}_G$ is in the set $\{\perp, \perp_d, 1, \ldots, q_{H_j}\}$ for $j \in [2]$. Thus, one can guess the correct reply by sampling a random value in that set, which gives a success probability of $\frac{1}{(q_{H_j}+2)}$. Overall, there are at most 2 calls to $\mathcal{O}^{\mathsf{i}}_G$ (one for $j = 1$ and $j = 2$) and therefore the probability that the simulation is successful is $\frac{1}{(q_{H_1}+2)(q_{H_2}+2)}$.

- H''_j, $j \in [2]$: The only time the secret key is used is when there is a query $(\mathsf{HS}, H_T(\mathsf{ct}, n))$ s.t. (ct, n) was already queried to $\mathcal{O}^{\mathsf{Dec}'}$ (i.e. $\mathcal{L}^j_K = (\mathsf{ct}, n, h)$ for some h). In this case h is returned iff $G(\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})) = \mathsf{HS}$ (let's call this Condition (1)). Recalling that queries to H_j never repeat by assumption, there will be at most one query $H''_j(\mathsf{HS}, H_T(\mathsf{ct}, n))$ s.t. (ct, n) was queried to the decapsulation oracle and Condition (1) is fulfilled. Hence, one can simulate H_j by sampling an index $q_j \in \{0, \ldots, q_{H_j}\}$ and returning h (if it exists) in the q_j-th query or never in case $q_j = 0$. This successfully simulates H_j with probability $\frac{1}{(q_{H_j}+1)}$. Overall, the probability that both H_1 and H_2 are simulated correctly is $\frac{1}{(q_{H_1}+1)(q_{H_2}+1)}$.

The other ROs can be simulated perfectly by $\mathcal{C}$ using lazy sampling. Overall, $\mathcal{C}$ simulates perfectly $\mathcal{A}$'s view in game Γ^6 (as long as query does not occur) with probability

$$p = \frac{1}{(q_{H_2}+1)^2(q_{H_1}+2)(q_{H_2}+2)(q_{H_1}+1)} .$$

Then if query happens, K^* will be in the list of queries made by $\mathcal{A}$ to G. The adversary can guess which one it is and succeeds with probability $\frac{1}{q_G}$. Hence, we have

$$|\Pr[\Gamma^5 \Rightarrow 1] - \Pr[\Gamma^6 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \leq q_G(q_{H_1}+2)^2(q_{H_2}+2)^3 \cdot \mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{KEM}}(\mathcal{C}) .$$

Finally, in game Γ^6, as $H_1(\mathsf{HS}^*, \mathsf{ct}^*, n^*)$, $H_2(\mathsf{HS}^*, \mathsf{ct}^*, n^*)$ or $H_3(\mathsf{HS}^*)$ cannot be queried anymore, the challenge keys are perfectly indistinguishable from random for the adversary. Hence,

$$\Pr[\Gamma^6 \Rightarrow 1] = \frac{1}{2} .$$

Collecting the probabilities concludes the proof.

□

$\mathscr{C}(\mathsf{pk}, \mathsf{ct}^*)$

init $\mathscr{L}_H, \mathscr{L}_K \leftarrow \emptyset$
$q_1 \leftarrow\$ \{0,1,\ldots,q_{H_1}\}$; $q_2 \leftarrow\$ \{0,1,\ldots,q_{H_2}\}$
$n^* \leftarrow\$ \{0,1\}^n$
$(\mathsf{CHTS}^*, \mathsf{SHTS}^*, \mathsf{dHS}^*) \leftarrow\$ \{0,1\}^{3n}$
run $\mathscr{A}^{\mathscr{O}^{\mathsf{Dec}''}, \mathscr{O}^{\mathsf{Dec}''}_{\mathsf{MAC}}, H_1'', H_2'', G, H_3, H_4, H_D}$
$\quad(\mathsf{pk}, \mathsf{ct}^*, n^*, (\mathsf{CHTS}^*, \mathsf{SHTS}^*, \mathsf{dHS}^*))$
sample random query K made to G
return K

$H_j''(\mathsf{HS}, y),\ j \in [2]$

if $\nexists(\mathsf{ct}, n)$ s.t. $((\mathsf{ct}, n), y) \in \mathscr{L}_{H_T}$:
$\quad h \leftarrow\$ \{0,1\}^n$; **return** h
set (ct, n) s.t. $((\mathsf{ct}, n), y) \in \mathscr{L}_{H_T}$
$i_q \leftarrow$ query number
if $\exists h$ s.t. $((\mathsf{HS}, \mathsf{ct}, n), h) \in \mathscr{L}_{H_j}$:
$\quad$ **return** h
if $\mathscr{L}_K^j = (\mathsf{ct}, n, h)$ for some h :
$\quad$ **if** $i_q = q_j$:
$\quad\quad \mathscr{L}_{H_j} \leftarrow \mathscr{L}_{H_j} \cup \{((\mathsf{HS}, \mathsf{ct}, n), h)\}$
$\quad\quad$ **return** h
$h \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_{H_j} \leftarrow \mathscr{L}_{H_j} \cup \{((\mathsf{HS}, \mathsf{ct}, n), h)\}$
return h

Oracle $\mathscr{O}^{\mathsf{Dec}''}_{\mathsf{MAC}}(\mathsf{ct}, n, \mathsf{tag}, \mathsf{txt})$

if more than 1 query : **return** $\perp$
if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
$r \leftarrow\$ \{0,\ldots,q_{H_2}\}$
if $r = 0$: **return** $\perp$
if less than r queries have been made to H_2 :
$\quad$ **abort**
get r-th query $(\mathsf{HS}, \mathsf{ct}, n)$ made to H_2
return HS

Oracle $\mathscr{O}^{\mathsf{Dec}''}(\mathsf{ct}, n)$

if $(\mathsf{ct}, n) = (\mathsf{ct}^*, n^*)$: **return** $\perp$
if more than 1 query : **return** $\perp$
$q_1 \leftarrow\$ \{0,\ldots,q_{H_1}\}$
$q_2 \leftarrow\$ \{0,\ldots,q_{H_2}\}$
$i \leftarrow\$ \{1,\ldots,q_{H_1}, \perp, \perp_d\}$
if $i = \perp_d$: **return** $\perp$
if $i \neq \perp$:
$\quad ((\mathsf{HS}_i, \mathsf{ct}_i, n_i), h_i) \leftarrow \mathscr{L}_{H_1}[i]$
$\quad \mathsf{CHTS} \leftarrow h_i$
else :
$\quad \mathsf{CHTS} \leftarrow\$ \{0,1\}$
$\mathscr{L}_K^1 \leftarrow (\mathsf{ct}, n, \mathsf{CHTS})$
$i \leftarrow\$ \{1,\ldots,q_{H_2}, \perp, \perp_d\}$
if $i \neq \perp$:
$\quad ((\mathsf{HS}_i, \mathsf{ct}_i, n_i), h_i) \leftarrow \mathscr{L}_{H_2}[i]$
$\quad \mathsf{SHTS} \leftarrow h_i$
else :
$\quad \mathsf{SHTS} \leftarrow\$ \{0,1\}$
$\mathscr{L}_K^2 \leftarrow (\mathsf{ct}, n, \mathsf{SHTS})$
return $(H_D(\mathsf{CHTS}), H_D(\mathsf{SHTS}))$

Figure 6.14: $\mathscr{C}$ adversary for the proof of Theorem 6.5.1.

6.5.3 MultiStage security

We briefly recall in this section the notion of MultiStage security as defined by Dowling et al. [Dow+20]. We refer the reader to their work for further details and discussion.

MultiStage syntax

Each protocol has a set of properties encoded in a tuple $(\mathsf{M}, \mathsf{AUTH}, \mathsf{FS}, \mathsf{USE}, \mathsf{REPLAY})$ which respectively indicates the number of stages in the protocol, the stage at which a key becomes (unilaterally or mutually) authenticated, which keys are forward secret, which keys are meant to be used internally/externally to the protocol and finally which stage is "replayable".

Then, we denote by $\mathcal{U}$ the set of honest participants and each session is defined as $\pi = (U, V, n) \in \mathcal{U} \times \mathcal{U} \times \mathbb{N}$, which denotes the n-th session of participant U with intended partner session V. In addition, each participant can have a long-term secret such as a secret key or pre-shared secret. Then, each session has a list of properties:

- $\mathsf{id} \in \mathcal{U}$: the identity of the session owner.
- $\mathsf{pid} \in \mathcal{U} \cup \{*\}$: the identity of the intended partner.
- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$: the role of the session (e.g. client/server for TLS).
- $\mathsf{auth} \in \mathsf{AUTH}$: the intended authentication type.
- $\mathsf{pssid} \in \{0,1\}^* \cup \{\perp\}$: the identifier of the pre-shared secret, when any.
- $\mathsf{st}_{\mathsf{exec}} \in \{\mathsf{running}, \mathsf{accepted}, \mathsf{rejected}\}^M$: indicates whether the session is running the i-th stage, has accepted or rejected the i-th key.
- $\mathsf{stage} \in \{0, \ldots, M\}$: the current stage.
- $\mathsf{sid} \in (\{0,1\}^* \cup \{\perp\})^M$: indicates the session identifier in each stage.
- $\mathsf{cid} \in (\{0,1\}^* \cup \{\perp\})^M$: indicates the contributive identifier in each stage.
- $\mathsf{key} \in (\{0,1\}^* \cup \{\perp\})^M$: indicates the key established in each stage. The key key_i is set only when the key was accepted in stage i.
- $\mathsf{st}_{\mathsf{key}} \in \{\mathsf{fresh}, \mathsf{revealed}\}^M$: indicates the state of a session key in each stage.
- $\mathsf{tested} \in \{\mathsf{true}, \mathsf{false}\}^M$: tested_i indicates whether key_i has been tested.
- $\mathsf{corrupted} \in \{0, \ldots, M, \infty\}^M$: indicates which stage the session was in when a Corrupt query was issued by the adversary (0 if it was before the session started and ∞ if no party involved is corrupted).

We say two sessions π and π' are *partnered* if $\pi.\mathsf{sid} = \pi'.\mathsf{sid} \neq \perp$ and $\pi.\mathsf{role} \neq \pi'.\mathsf{role}$. Similarly, two sessions are *contributive partners* if $\pi.\mathsf{cid} = \pi'.\mathsf{cid} \neq \perp$ and $\pi.\mathsf{role} \neq \pi'.\mathsf{role}$.

MultiStage adversarial model

In the MultiStage security model, the adversary is able to create sessions and make the send/receive messages. In addition, it can also reveal sessions keys and corrupt long-term secrets. Finally, it can issue test queries, which return a real or random session key and the adversary must distinguish between both cases. More precisely, the oracles are defined as follows.

- $\mathsf{NewSession}(U, V, \mathsf{role})$: returns a new session π with owner V, role role and intended partner session V. If U is corrupted, $\pi.\mathsf{corrupted} \leftarrow 0$ is set.

- $\mathsf{Send}(\pi, m)$: sends a message m on behalf of session π. If a key is accepted during the processing of this query, the process is stopped and accepted is returned to the adversary, who can then test the key before it is used. In order to continue the process, the adversary can query $\mathsf{Send}(\pi, \mathsf{continue})$. On key acceptance at stage i, if there exists a partnered session π' s.t. $\pi'.\mathsf{tested}_i = \mathsf{true}$, then $\pi.\mathsf{tested}_i \leftarrow \mathsf{true}$ is set. If key_i is an internal key, we furthermore set $\pi.\mathsf{key}_i \leftarrow \pi'.\mathsf{key}_i$.

- $\mathsf{Reveal}(\pi, i)$: returns $\pi.\mathsf{key}_i$ if it exists and $\perp$ otherwise. Then, $\pi.\mathsf{st}_{\mathsf{key}_i} \leftarrow \mathsf{revealed}$ is set.

- $\mathsf{Corrupt}(U)$ or $\mathsf{Corrupt}(U, V, \mathsf{pssid})$: reveals the long-term or pre-shared secret, respectively. It also marks U (resp. (U, V, pssid)) as corrupted and sets the corresponding labels in each session π with $\pi.\mathsf{id} = U$ as corrupted. We refer the interested reader to the original work [Dow+20] for more details on each case and the handling of flags depending on the forward-security level required.

- $\mathsf{Test}(\pi, i)$: tests the session key at stage i. This oracle depends on a random bit b (the goal for $\mathscr{A}$ is to guess b). If $\pi_{\mathsf{st.exec},i} \neq \mathsf{accepted}$ or $\pi.\mathsf{tested}_i = \mathsf{true}$, it returns $\perp$. If stage i is internal and there exists a partnered session π' s.t. $\pi'_{\mathsf{st}_{\mathsf{exec}},i} \neq \mathsf{accepted}$, we set a lost flag to true. Other flags are set depending on the level of authentication (see Dowling et al. [Dow+20] for more details). Then, $\pi.\mathsf{tested}_i$ is set to **true**. If $b = 0$, a key K is sampled at random and if $b = 1$ K is set to the real key $\pi.\mathsf{key}_i$. If the session key is internal, $\pi.\mathsf{key}_i$ is replaced by K (thus K will be used for any future use of $\pi.\mathsf{key}_i$ in the protocol). If the key is external, the oracle simply returns K. Finally, if there exists a partnered session π' s.t. π' has accepted the key at stage i, we set $\pi'.\mathsf{tested}_i$ to **true** and if the key is internal we set $\pi'.\mathsf{key}_i \leftarrow \pi.\mathsf{key}_i$.

MultiStage game

We can now describe the game that defines MultiStage security.

Definition 6.5.3. *Let* KE *be a key-exchange with properties* $(\mathsf{M}, \mathsf{AUTH}, \mathsf{FS}, \mathsf{USE}, \mathsf{REPLAY})$. *For any efficient adversary* $\mathscr{A}$ *playing the following game* $\mathsf{MultiStage}_{\mathsf{KE}}(\mathscr{A})$*:*

Setup: *The random bit* $b \leftarrow\$ \{0,1\}$ *is sampled, the* lost *flag is set to* **false** *and in a public key variant, long-term* $(\mathsf{pk}_U, \mathsf{sk}_U)$ *are generated for all* $U \in \mathcal{U}$.

Query: *The adversary* $\mathcal{A}$ *receives the public keys and can call every oracle defined above.*

Guess: *The adversary outputs a guess* b'.

Finalise: *The* lost *flag is set to* **true** *if there exist* π, π' *s.t.* $\pi.\mathsf{sid}_i = \pi'.\mathsf{sid}_i$, $\pi.\mathsf{st}_{\mathsf{key}\,i} = \mathsf{revealed}$ *and* $\pi'.\mathsf{tested}_i = \textbf{true}$. *If* $\mathsf{lost} = \textbf{true}$ *the game outputs a random bit, otherwise it outputs* $1_{b=b'}$.

We define the MultiStage advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}_{\mathsf{KE}}^{\text{multi-stage}}(\mathcal{A}) = \Pr[\mathsf{MultiStage}_{\mathsf{KE}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2}\,.$$

Then, we say KE *is MultiStage secure if for any efficient* $\mathcal{A}$ *the advantage* $\mathsf{Adv}_{\mathsf{KE}}^{\text{multi-stage}}(\mathcal{A})$ *is negligible in the security parameter.*

6.5.4 TLS 1.3 in the MultiStage model

We describe the parameters of the TLS 1.3 full 1-RTT handshake relevant to our proof in the MultiStage model. The number of stages is $M = 6$, forward-secrecy is required (i.e. $\mathsf{FS} = 1$), the handshake traffic keys are used internally while other keys are external (i.e. $\mathsf{USE} = (\mathsf{internal}: \{1,2\}, \mathsf{external}: \{3,4,5,6\})$). The first stages of our modified TLS 1.3 1-RTT handshake are shown in Figure 6.15, for a detailed description of all the keys and stages, we refer the reader to Figure 1 in Downing et al. [Dow+20].

The session identifiers are set when a key is accepted in a given stage, they include a label and the transcript up to this point:

$$\begin{aligned}
\mathsf{sid}_1 &= (\text{“}CHTS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS})\\
\mathsf{sid}_2 &= (\text{“}SHTS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS})\\
\mathsf{sid}_3 &= (\text{“}CATS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS}, \mathsf{EE}, \mathsf{CR}^*, \mathsf{SCRT}, \mathsf{SCV}, \mathsf{SF})\\
\mathsf{sid}_4 &= (\text{“}SATS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS}, \mathsf{EE}, \mathsf{CR}^*, \mathsf{SCRT}, \mathsf{SCV}, \mathsf{SF})\\
\mathsf{sid}_5 &= (\text{“}EMS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS}, \mathsf{EE}, \mathsf{CR}^*, \mathsf{SCRT}, \mathsf{SCV}, \mathsf{SF})\\
\mathsf{sid}_6 &= (\text{“}RMS\text{”}, \mathsf{CH}, \mathsf{CKS}, \mathsf{SH}, \mathsf{SKS}, \mathsf{EE}, \mathsf{CR}^*, \mathsf{SCRT}, \mathsf{SCV}, \mathsf{SF}, \mathsf{CCRT}^*, \mathsf{CCV}^*, \mathsf{CF})
\end{aligned}$$

where $*$ marks elements used only in the *mutual authentication mode*. The contributive identifiers are the same as the sid except in stage 1 and 2. That is, $\mathsf{cid}_i = \mathsf{sid}_i,\ i \in \{3,4,5,6\}$. In stages 1 and 2, a client (resp. server) session sets $\mathsf{cid}_1 = (\text{“}CHTS\text{”}, \mathsf{CH}, \mathsf{CKS})$, $\mathsf{cid}_2 = (\text{“}SHTS\text{”}, \mathsf{CH}, \mathsf{CKS})$ upon sending (resp. receiving) the CH (+ CKS) messages, then they set $\mathsf{cid}_1 = \mathsf{sid}_1$ and $\mathsf{cid}_2 = \mathsf{sid}_2$ upon receiving (resp. sending) the SH and SKS messages.

Now, as a client session only accepts the first stage key after receiving the SH message, a contributive partner of a tested client session will have the same $\mathsf{cid}_1 = \mathsf{sid}_1$. Hence it means the client and server sent and received the same messages in the first stage. On the other hand, a server session accepts the first stage key (and thus can be tested) after receiving the $\mathsf{CH}, \mathsf{CKS}$ messages only. Hence, in this case it guarantees that the client and server sessions got the same client messages but not necessarily that the server messages are the same.

6.5.5 Security of TLS 1.3 with IND-1CCA-MAC KEM

We can now use the (slightly modified) notion IND-1CCA-MAC KEM to prove the security of the TLS 1.3 handshake in the multi-stage security model.

The security of (the original) TLS 1.3 handshake was proven by Dowling et al. [Dow+20] and we refer the reader to their work for a complete analysis of the handshake. We will simply show that IND-1CCA-MAC KEMs, thus OW-CPA KEMs (if the MAC is secure), can be used in place of the original snPRF-ODH assumption for DH key-exchange.

First, we show the relevant part of the (full 1-RTT) handshake of TLS 1.3 in Figure 6.15. One can see that the key schedule is nearly identical to the ones used in the IND-1CCA-MAC game. Note that several simplifications have been made and several steps irrelevant to our proofs are missing. In particular, we do not see the derivation of the finals keys, which all depend on the secret dHS. As we will show, the intermediary secrets $(\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS})$ are secure (i.e. indistinguishable from random for a Multi-Stage adversary), thus all subsequent keys will be secure as well, assuming the KDFs are secure. Finally, we write $\mathsf{HKDF.Exp}_i(\mathsf{HS}, T_2)$ for $\mathsf{HKDF.Exp}(\mathsf{HS}, \mathsf{label}_i, T_2)$, where label_i is some string. As we assume the KDFs $\mathsf{HKDF.Ext}$ and $\mathsf{HKDF.Exp}$ to be ROs, this denotes the fact that the label implements oracle separation.

The security of the modified 1-RTT TLS 1.3 handshake is stated in the following theorem.

Theorem 6.5.2. *Let* $\mathsf{HKDF.Ext}$, $\mathsf{HKDF.TK}$ *and* $\mathsf{HKDF.Exp}_j$, $j \in \{0,4,5,6\}$ *(the KDFs in TLS 1.3) be random oracles. Let* Hash *(the hash function used to compute the hashed transcripts* T_i*) be a RO, and* Sig *the signature scheme used for server authentication (not shown in Figure 6.15). For any Multi-Stage efficient adversary* $\mathcal{A}$ *there exist efficient adversaries* $\{\mathcal{B}_i\}_{i\in[6]}$ *s.t.*

$$\begin{aligned}\mathsf{Adv}^{\text{multi-stage}}_{\mathsf{TLS1.3-1RTT}}(\mathcal{A}) \le 6t_s\Big(&\mathsf{Adv}^{\text{coll}}_{H}(\mathcal{B}_1) + t_u \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig}}(\mathcal{B}_2)\\ &+ t_s\Big(\mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathcal{B}_3) + 2\cdot \mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Exp}}(\mathcal{B}_4)\\ &+ \mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Ext}}(\mathcal{B}_5) + \mathsf{Adv}^{\text{prf}}_{\mathsf{HKDF.Exp}}(\mathcal{B}_6)\Big)\Big),\end{aligned}$$

where t_s *(resp.* t_u*) is the maximal number of sessions (resp. users). Note that for the sake of the comparison with the original bound, we keep several PRF advantages and the collision advantage in the bound, even though they could be replaced by negligible terms, as the KDFs and* Hash *are ROs.*

TLS 1.3 with KEM Handhsake

Client		**Server**
$(\mathsf{sk},\mathsf{pk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$		
$\mathsf{CH} : n_c \leftarrow\$ \{0,1\}^{256}$		
$+\,\mathsf{pk}$		
	$\mathsf{dES} \leftarrow \mathsf{HKDF}(\text{constant})$	
	$\xrightarrow{\quad\mathsf{CH}\quad}$	
		$K, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk})$
		$\mathsf{SH} : n_s \leftarrow\$ \{0,1\}^{256}$
		$+\,\mathsf{ct}$
	$\xleftarrow{\quad\mathsf{SH}\quad}$	
$K \leftarrow \mathsf{Decaps}(\mathsf{sk},\mathsf{ct})$		
	$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\mathsf{dES}, K)$	
	$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, T_2)$	
	$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, T_2)$	
	$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, T_0)$	
	.... (Stage 1) accept $\mathsf{tk_c} \leftarrow \mathsf{HKDF.TK}(\mathsf{CHTS})$	
	.... (Stage 2) accept $\mathsf{tk_s} \leftarrow \mathsf{HKDF.TK}(\mathsf{SHTS})$	
	$\dots$	
	$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$	
		$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$
	$\xleftarrow{\quad\{\mathsf{SF}\}\quad}$	
if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**		
	$\dots$	

Figure 6.15: TLS 1.3 handshake with KEM. $\{\dots\}$ indicates an encrypted message with $\mathsf{tk_S}$, T_i is the hash of the transcript up to message i. For simplicity, the CH (resp. SH) message captures both the $ClientHello$ and $ClientKeyShare$ (resp. $ServerHello$ and $ServerKeyShare$). Only the relevant steps for the proof are shown. Keys in the remaining stages (3-6, not shown) are all derived from dHS.

Proof. As hinted above, the idea of the proof is simply to replace the snPRF-ODH step of the original proof by using our IND-1CCA-MAC. Note that while the snPRF-ODH assumption is used to replace the root secret HS by a random one, we will be able to replace the values $(\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS})$ by random ones in one step, due to the structure of the IND-1CCA-MAC definition. From a high-level point of view, the proof goes through because CHTS and SHTS are computed similarly as in the $\mathsf{T_H}$ transform (i.e. the secrets are the hashed seed and ciphertext) and thus resist to 1 adversarial decapsulation query. Then, dHS is used only once a MAC has been verified, which implies that an adversary relaying a correct tag should already know the root key HS.

The proof proceeds by a sequence of games. As the first transitions are the same as in the original proof by Dowling et al. (proof of Theorem 5.2 [Dow+20]) we do not explain them in detail.

Game Γ^0 : The first game is the original Multi-Stage game.

Game Γ^1 : We modify the game s.t. $\mathscr{A}$ can only make one Test query. This brings the $6t_s$ factor in the security bound.

Game Γ^2 : The game aborts if a collision on the hash function Hash occurs. We recall that Hash is used to compute the hash of the transcripts (the values T_i in Figure 6.15).

We can then split the proof into two different cases: (A) $\mathscr{A}$ tests a session that does not have a contributive partner or (B) $\mathscr{A}$ tests a session with a contributive partner. In case (A), one can show that the probability of $\mathscr{A}$ winning the game is upper bounded by $t_u \mathsf{Adv}^{\mathrm{euf-cma}}_{\mathsf{Sig}}(\mathscr{B}_2)$ for an adversary $\mathscr{B}_2$. Thus, we focus on case (B).

Game $\Gamma^{B.0}$: This is the same as Γ^2 conditioned on the fact that $\mathscr{A}$ tests a session with a contributive partner.

Game $\Gamma^{B.1}$: The adversary guesses which session will be the contributive partner at the beginning of the game. As there are at most t_s sessions, this incurs a loss factor of t_s in the rest of the proof.

Game $\Gamma^{B.2}$: This is the only game transition that will differ from the original proof. Let π_c be the client session that is either tested or the contributive partner of the tested session. Similarly, let π_s be the server session that is either tested or the contributive partner (note that a session and its contributive partner always have opposite role). Let (ct, n_s) (resp. (pk, n_c)) be the SH (resp. CH) message sent by π_s (resp. π_c), where ct is the ciphertext, n_s (resp. n_c) the nonce of the server (resp. client) session. Note that by an abuse of notation, we assume SH (resp. CH) includes the server's (resp. client's) share. Then, in this game, we make the following changes:

1. We replace the derived secrets $(\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS})$ in π_s by random ones.
2. If π_c receives (ct, n_s) in the SH message, we replace $(\mathsf{CHTS}, \mathsf{SHTS}, \mathsf{dHS})$ with the *same* random secrets as in the previous point.

Now, we can argue that distinguishing $\Gamma^{B.2}$ from $\Gamma^{B.1}$ implies breaking the IND-1CCA-MAC security of KEM. First, we notice that $T_2 := \mathsf{Hash}(\mathsf{CH}, \mathsf{SH}) = \mathsf{Hash}(\mathsf{pk}, n_c, \mathsf{ct}, n_s)$. Hence, the

KDF $\mathsf{HKDF.Exp}_j(\cdot, T_2)$, $j \in \{4,5\}$ can be written as $H_j(\cdot, H_T(\mathsf{ct}, n_s)), i \in \{1,2\}$ where H_j and H_T are ROs, if we omit the public key and the client nonce, which are not important in the proof. Similarly, as T_0 and dES are constant, one can write $\mathsf{HKDF.Ext}(\mathsf{dES}, \cdot)$ as $G(\cdot)$, $\mathsf{HKDF.Exp}_0(\cdot, T_0)$ as $H_3(\cdot)$, and $\mathsf{HKDF.Exp}_6(\cdot)$ as $H_4(\cdot)$, with G, H_3 and H_4 some ROs. Finally, one can rename $\mathsf{HKDF.TDK}$ as H_D, where H_D is a RO. Hence, one can see that the key-schedule becomes exactly the one of the IND-1CCA-MAC game. Now let's explain how the reduction will work. We split $\Gamma^{B.2}$ into 2 cases:

- Case 1: The tested session is the client session π_c. As π_c can only be tested after receiving the SH message and π_s is a contributive partner, it means that the SH message sent by π_s is the same as the one received by π_c. In particular it means that we make both changes mentioned above. Then the reduction is straightforward. The IND-1CCA-MAC adversary $\mathscr{B}_3$ receives a tuple $(\mathsf{pk}^*, \mathsf{ct}^*, n^*, (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$. It simulates the tested session π_c with these values. In particular, it sends pk^* in the CH message, it uses n^* as the nonce of the contributive session π_s, $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ as the secrets of π_s and ct^* as the ciphertext sent in the SH generated by π_s. Finally, to simulate π_c after receiving SH, we use the same challenge secrets $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$. In case $b = 0$, this perfectly simulates $\Gamma^{B.1}$ (the secrets correspond to ct^*) and in case $b = 1$ this perfectly simulates $\Gamma^{B.2}$. Therefore, in Case 1 we have

$$\mathsf{Adv}^{\Gamma^{b.1}}_{\mathsf{TLS1.3-1RTT}}(\mathscr{A}) \le \mathsf{Adv}^{\Gamma^{b.2}}_{\mathsf{TLS1.3-1RTT}}(\mathscr{A}) + \mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathscr{B}_3)\,.$$

  Note that we did not even need the oracles provided to $\mathscr{B}_3$ in this case.

- Case 2: The tested session is the server session π_s. Again, either the SH sent by π_s is the same as the one received by π_c and the reduction $\mathscr{B}_3$ is the same as in Case 1, or (ct, n_s) is not the same as the SH message received by π_c. In the latter case, we build the reduction $\mathscr{B}_3$ as follows. Again, $\mathscr{B}_3$ receives a tuple $(\mathsf{pk}^*, \mathsf{ct}^*, n^*, (\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b))$, uses pk^* in the CH, (ct^*, n^*) as the SH sent by π_s and $(\mathsf{CHTS}_b, \mathsf{SHTS}_b, \mathsf{dHS}_b)$ as the secrets derived by π_s after receiving CH. Then, on the modified $\mathsf{SH} = (\mathsf{ct}', n_s') \neq (\mathsf{ct}^*, n_s^*)$ sent by $\mathscr{A}$ to π_c, $\mathscr{B}_3$ queries its decryption oracle $\mathscr{O}^{\mathsf{Dec}}$ to obtain the correct $(\mathsf{tk_c}, \mathsf{tk_s})$. Therefore, $\mathscr{B}_3$ can correctly simulate π_c and any Reveal queries until the SF message, as no other secrets are needed. Then, when π_c receives the SF message, which is a tag on T_7, it queries $\mathscr{O}^{\mathsf{Dec}}_{\mathsf{MAC}}(\mathsf{ct}', n', \mathsf{SF}, T_7)$. If the tag in SF is correct (i.e. correspond to a MAC on T_7 with a key derived from the secret encapsulated in ct'), $\mathscr{B}_3$ gets $\mathsf{HS} := G\big(\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}')\big)$ and can derive all secrets to simulate π_c correctly. Otherwise, the oracle returns $\perp$, which means the MAC is not valid and $\mathscr{B}_3$ aborts the client session π_c. Again, this perfectly simulates π_c behaviour. Hence, the adversary can simulate perfectly $\mathscr{A}$'s view in game Γ^{B_1} in case $b = 0$ or game $\Gamma^{B.2}$ in case $b = 1$, and we obtain

$$\mathsf{Adv}^{\Gamma^{b.1}}_{\mathsf{TLS1.3-1RTT}}(\mathscr{A}) \le \mathsf{Adv}^{\Gamma^{b.2}}_{\mathsf{TLS1.3-1RTT}}(\mathscr{A}) + \mathsf{Adv}^{\text{ind-1cca-mac}}_{\mathsf{KEM}}(\mathscr{B}_3)\,.$$

Game $\Gamma^{B.3}$: Note that from the previous game, all "main" secrets in the tested session are random and independent of any session (except the partnered session in case π_c is the tested session and received the correct SH from $\mathscr{A}$). Hence, one can replace the relevant transport keys $(\mathsf{tk}_c, \mathsf{tk}_s)$ by random values (i.e. the ones in the π_s session, and, in case π_c received the correct SH, the ones in π_c as well). Overall this step transition is correct if $\mathsf{HKDF.TK}$ is a PRF.
Game $\Gamma^{B.4}$: We replace the relevant master secret MS by a random value. Again the transition is correct if the KDF is a PRF.
Game $\Gamma^{B.5}$: Finally, all the remaining keys are replaced by random values in the tested session (and in the partnered session if π_c the received the correct SH). Again, this is correct if the KDF used is a PRF. Then, all keys in the tested session are random and independent of values in any other session (except the partnered session as mentioned before). Hence, $\mathscr{A}$ cannot win as the tested keys are always random. This concludes the proof. □

Similarly, one can prove the security of the modified TLS 1.3 PSK-(EC)DHE 0-RTT handshake. Note that in our case the key-exchange will be done with KEMs, but we keep the "-(EC)DHE" in the name for consistency with the original protocol. We state this in the following informal theorem.

Theorem 6.5.3. *The modified TLS 1.3 handshake in the pre-shared key (optional) 0-RTT mode with key-exchange (i.e. TLS 1.3 PSK-(EC)-DHE 0-RTT) is secure in the MultiStage model if the underlying KEM is OW-CPA (and signature, MAC, etc. are secure), in the sense of Dowling et al. [Dow+20].*

Proof. The only step in the original proof involving the KEMs can be dealt with a similar reduction from IND-1CCA-MAC as in the proof of Theorem 6.5.2. □

Corollary 6.5.1. *The original TLS 1.3 handshake is MultiStage secure in the ROM if the CDH problem is hard (and the signature, MAC, etc. are secure). Stronger assumptions used in previous proofs (e.g. PRF-ODH [Dow+20]) are not necessary.*

Proof. This simply follows from the fact that DH can be described as a KEM $(\mathsf{sk}, \mathsf{pk}) := (x, g^x)$, $(K, \mathsf{ct}) := (\mathsf{pk}^y, g^y)$ and $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) := \mathsf{ct}^x$. Integrating this KEM in our modified TLS 1.3 handshake results in the standard TLS 1.3 handshake. Finally, this KEM is OW-CPA as long as the CDH problem is hard, thus by Theorems 6.5.1 and 6.5.2, the handshake is secure. One can also directly show that DH as used in TLS 1.3 is a IND-1CCA KEM. We provide such a proof in Appendix A. □

Remarks. Note that due to non-tightness of the bound in Theorem 6.5.1, the overall bound for TLS security is very much non-tight. This is clearly not sufficient to guarantee security in practice, and we leave as an interesting open question the improvement of the bounds. In addition, we leave security in the QROM as future work.

6.6 Impact

The transforms introduced in Section 6.4 produce IND-qCCA KEMs without any de-randomisation and re-encryption steps. Thus, using IND-1CCA ephemeral KEMs obtained through these transforms could speed up the decapsulation process in several protocols.

KEMTLS. As discussed in the introduction of this chapter, improving the KEMTLS protocol [SSW20a] was the main motivation behind this research. In particular, a more efficient decapsulation in the ephemeral KEM would decrease overall latency and computation on the client-side. In particular, this could be of interest for less powerful clients like IoT devices, which would not need to perform re-encryption. Overall, the efficiency gain in practice would obviously depend on the ephemeral KEM used, as encryption is expensive in some schemes while it is not in others.

The same remarks apply to the very recent variants of KEMTLS with pre-distributed keys proposed by Günther et al. [Gün+22] and Schwabe et al. [SSW21].

Note also that following a similar proof as the one in Section 6.5, we conjecture that one should be able to prove that CPA-security of the ephemeral KEM should suffice for KEMTLS to be secure in the ROM (but at the expense of a non-tight security bound, as in the TLS case).

TLS 1.3. TLS 1.3 only supports ephemeral DH as a key-exchange. In turn, in the original security proof [Dow+20], the snPRF-ODH assumption is used for the key-exchange security. The snPRF-ODH assumption can be seen as a variant of the hashed Diffie-Hellman assumption with a 1-time "decapsulation" oracle. More precisely, an adversary is given (g, g^u, g^v) and either $y_0 := \mathsf{PRF}(g^{uv}, \mathsf{ad}^*)$ or a random y_1, where ad^* is some auxiliary data chosen by the adversary. Then, the adversary must distinguish between y_0 and y_1 with the help of *one* query to an oracle $\mathcal{O}((x, \mathsf{ad}) \neq (g^u, \mathsf{ad}^*)) := \mathsf{PRF}(x^v, \mathsf{ad})$.

One can notice that snPRF-ODH security is very close to IND-1CCA security transposed to DH key-exchange. Actually, one can show that IND-1CCA KEM is sufficient for the PQ TLS 1.3 handshake to hold. Indeed, instead of using our IND-1CCA-MAC assumption in the proof, one can use the decapsulation oracle of the IND-1CCA adversary to recover the key if needed. One can check the transition between games $B.1$ and $B.2$ in the proof of KEMTLS security [SSW20a] for more details.

Therefore, using IND-1CCA KEMs in the PQ TLS 1.3 handshake seems a sound idea, as in this case the security bound will offer better guarantees than with a OW-CPA KEM. In addition, the handshake would be faster using IND-1CCA KEMs generated by our transforms instead of the slower IND-CCA KEMs derived with FO.

Finally, by Corollary 6.5.1, we now know that the snPRF-ODH assumption is not necessary in the ROM for TLS 1.3 to be secure (even though the security bound is very much non-tight), but CDH is sufficient. Alternatively, as shown in Appendix A, DH as used in TLS 1.3 is actually an IND-1CCA KEM (≈ snPRF-ODH) in the ROM if CDH holds. This gives a tighter security bound compared to Corollary 6.5.1.

Ratcheting. IND-1CCA security is also a property used (often implicitly) in several works on ratcheting. For instance, Jost et al. [JMM19] build a *healable and key-updating public-key encryption* scheme based on a *one time* IND-CCA2 PKE (with authenticated data). The latter primitive can easily be made out of an IND-1CCA KEM using KEM/DEM techniques. Another paper by Poettering et al. [PR18] introduces a construction of *unidirectional ratcheted key exchange* (URKE) that is based (implicitly) on IND-1CCA KEMs, as noticed by Balli et al. [BRV20].

In another recent paper, Brendel et al. [Bre+22] propose a post-quantum alternative to the Signal handshake based on KEMs and designated verifier signature schemes. They first define a core protocol that uses two KEMs in the same vein as KEMTLS: one with long-term keys for implicit authentication of one of the parties and another one with ephemeral keys for guaranteeing forward security. Again, the latter one requires only IND-1CCA security for the handshake to be secure. Similarly, in the full Signal-like handshake built upon the core protocol (called SPQR), three KEMs are used and one requires only IND-1CCA security. Looking ahead, we will use a generalised variant of IND-1CCA security (called IND-1BatchCCA) in the next chapter to build our own PQ variant of the Signal handshake.

Concerns over key-reuse. The main security risk of using an IND-1CCA KEM instead of its IND-CCA counterpart is the vulnerability to key-reuse/misuse attacks. Indeed, if a system/protocol is mis-implemented s.t. the IND-1CCA KEM is used with a "static" public key instead of an ephemeral one, an adversary might be able to recover the secret key after several decryption queries as shown in Chapter 3. In KEMTLS, this risk is mitigated by the use of an IND-CCA KEM in addition to the ephemeral one (which can be IND-1CCA). In particular, the final shared key is derived from shares of both KEMs. Thus, even if the public key meant to be ephemeral is reused, the final shared key should remain "secure" (but forward security would be lost).

In other systems (e.g. TLS 1.3), the risk of key recovery after a few reuses could be mitigated by using hybrid cryptography. For instance, a very efficient IND-CCA KEM could be combined with an IND-1CCA one. That would improve the overall security and resistance against key-reuse attacks at a small cost (see Chapter 4 for a complete discussion on hybrid schemes).

7 K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures

We saw in the previous chapter that TLS 1.3 can easily be adapted to the post-quantum setting. It is not the case of all protocols, as for instance X3DH, the key-agreement protocol used in Signal, cannot simply be instantiated with KEMs while keeping all its security properties, in particular deniability. In this chapter, we introduce K-Waay, a post-quantum key-exchange protocol that could replace X3DH and that provides similar security guarantees.

This research is a joint work with Daniel Collins, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. The corresponding paper was accepted at USENIX Security 2024. A chapter based on the same material will appear in Daniel Collins' PhD thesis. Therefore, in order to avoid any confusion, we now detail the contributions of the author of the present dissertation:

- Main idea behind this research, i.e. the K-Waay protocol with the signed ephemeral split-KEM keys, and with a modified Frodo key-exchange as the split-KEM scheme.
- Identification of the shortcomings of the original split-KEM security definitions.
- Definition of appropriate split-KEM security notions (with input from Daniel Collins). I.e. the definitions of deniability, decaps-OW-CPA, UNF-1KCA, and IND-1BatchCCA (including idea of the key-reuse "trick").
- Help with the deniability proof of K-Waay and with the game hops involving the split-KEM in the security proof of K-Waay.
- Security proofs (deniability, OW-CPA security, decaps-OW-CPA) of our Frodo-inspired split-KEM (joint work with Ngoc Khan Nguyen).
- Idea and ROM/QROM security proofs of the T_{CH} transform.
- Supervision of the implementation of the benchmarking system and of the corresponding experiments (conducted by Nicolas Rolin).
- Analysis and discussion (i.e. Section 7.9) is joint work with Daniel Collins.

In terms of sections, the contributions of the author of this thesis can be roughly summarised as follows (ignoring the introductory sections):

- Section 7.5 with input from Daniel Collins.
- Design of the protocol presented in Section 7.7 and help for the parts relevant to split-KEM in the security proofs (Section 7.7.1).
- Sections 7.8.3 and 7.8.4 with Ngoc Khan Nguyen.
- Section 7.8.5.
- Section 7.9 with Daniel Collins, excluding implementation of the benchmarking system.

Although it is hard to define where one's contribution starts and ends, results presented outside of the sections listed above can be regarded as existing work extracted from the original paper [Col+ar] and included for the sake of completeness.

7.1 Background

In order to properly understand our contribution, we first need to introduce X3DH in more details, as well as the efforts that have been made to make it quantum resistant.

In the classical X3DH protocol, parties first upload their keying material to a central server or public key infrastructure in a so-called *prekey bundle*. A party can then derive a session key by downloading their partner's bundle and performing three (or four) Diffie- Hellman key exchanges with a mixture of ephemeral and long-term (resp. plus semi-static) keys, ensuring at least confidentiality even if the ephemeral or long-term key of each party is corrupted. In particular, no signatures are used after signed prekeys are uploaded: at that point, the DH exchanges provide implicit authentication guarantees. Consequently, the protocol provides a level of *deniability*: informally, a participant can deny having performed key exchange with its counterpart. This is a privacy guarantee that prevents (at least on a cryptographic level) a conversation transcript from incriminating an unsuspecting party, which is especially pertinent in situations like whistleblowing and protesting.

Conscious of the quantum threat, Signal announced and rolled out in 2023 their initial *hybrid* post-quantum key exchange solution, namely the "PQXDH protocol".[1] Like in X3DH, several Diffie-Hellman key exchanges are performed at once, but in PQXDH, parties upload prekey bundles that also contain a Kyber-1024 public key that the initiator additionally encapsulates to the responder with. Moreover, prekey bundles are still signed with the same *classical* signature scheme as regular X3DH. Although PQXDH appears to provide post-quantum confidentiality [Bha+23], it does not provide post-quantum *authentication* as a quantum attacker can trivially forge prekey bundles by breaking the classical signature scheme.

[1] https://signal.org/docs/specifications/pqxdh.

A natural direction for building a post-quantum equivalent of X3DH is to emulate X3DH's symmetric structure with a post-quantum construction. As a result, Brendel et al. [Bre+21] introduced the syntax of such a primitive, which they called *split-KEM*. In a split-KEM, a party B encapsulates to their partner A by using their own secret $\mathsf{sk_B}$ and their partner's public key $\mathsf{pk_A}$ to produce a ciphertext, then A decapsulates it using $\mathsf{sk_A}$ and $\mathsf{pk_B}$. The authors define indistinguishability-based security notions and notice that the Frodo [Bos+16] lattice-based key-exchange fulfils the split-KEM syntax and the weakest notion of indistinguishability they define. Although they present an X3DH-like protocol, they do not define a security model, and, looking ahead, their split-KEM security notions do not suffice to construct an X3DH-like key exchange with both authenticity and deniability.

In two other recent works, Hashimoto et al. [Has+21; Has+22] and Brendel et al. [Bre+22] concurrently proposed instead to construct X3DH-like key exchange using KEMs directly. In order to ensure deniability, two seemingly different approaches were proposed: Hashimoto et al. [Has+21; Has+22] apply ring signatures while Brendel et al. [Bre+22] use a flavour of designated verifier signatures; these primitives were later shown to be equivalent [Has+22].

As described in the aforementioned papers, the currently most efficient post-quantum ring signatures [BKP20; ESZ22; LAZ19; LN22; Yue+21] are proven to be secure in the ROM and can enjoy signatures that are a handful of kilobytes large, making them practical. Often, however, the constructions do not come with a security proof in the quantum random oracle model (QROM). In this vein, parameters are generally optimistically chosen as the security loss incurred by proofs in the ROM is not taken into account when setting these, without even mentioning QROM loss, which is usually much larger. Further, security notions can differ between papers, making it less clear exactly when they are appropriate for use. More generally, it is of interest to determine the cost (or overhead) that deniability incurs in (X3DH-like) key exchange. Towards this goal, Hashimoto et al. [Has+22] provide benchmarks for their baseline, non-deniable X3DH-like protocol based on signatures and KEMs, and Brendel et al. [Bre+22] consider parameter sizes for (but do not benchmark) existing ring and designated verifier signature schemes.

While the use of ring signatures to build PQ and deniable X3DH is at least theoretically understood, this far from exhausts the protocol design space. Motivated by this and the above discussion, we therefore address the following problem in this chapter:

Can we design a provably-secure, efficient, and deniable post-quantum X3DH alternative that does not *require ring signatures?*

7.2 Contributions

In this chapter, we propose an efficient, deniable, and post-quantum X3DH-like protocol without ring signatures that we call K-Waay. Our contributions can be summarised as follows:

- Towards building our protocol, we revisit the split-KEM formalism proposed by Brendel et al. [Bre+21] and deduce that several additional properties, namely notions of authenticity and deniability, are needed to construct a secure X3DH-like deniable authenticated key exchange protocol (DAKE).

- We propose K-Waay, a X3DH-like DAKE that uses a deniable and unforgeable split-KEM at its core. Our protocol uses a signature scheme to sign prekeys, and then uses an ephemeral KEM, a long-term KEM, and the split-KEM for the final key exchange step.

- The main drawback of a naive version of our protocol is that parties can run out of ephemeral keys, thus making the protocol synchronous if this happens (e.g. Bob needs to wait for Alice's fresh ephemeral key before sending a message). While such a problem would rarely occur in practice, given enough keys are uploaded on the server, we propose a simple trick that makes the reuse of ephemeral keys possible on the receiver's side for messages they received while offline. We think this trick could be of independent interest as it – perhaps surprisingly – allows for a specific kind of key reuse for a split-KEM that is *not* IND-CCA secure. This technique is inspired by the IND-qCCA transforms defined in Chapter 6.

- We prove key indistinguishability in our model that captures ephemeral key reuse and session state exposure, and prove a variant of deniability that strengthens the notion from Brendel et al [Bre+22] by additionally leaking the victim's session state to the adversary in the security game.

- We instantiate a post-quantum split-KEM secure under our new security notions derived from the Frodo key exchange protocol (FrodoKEX) [Bos+16] based on the plain LWE assumption. The parameters we choose provide strong security guarantees, providing more than 192 bits of classical and quantum security for our core split-KEM security notions OW-CPA, decaps-OW-CPA, and deniability. We then use a transform in the (Q)ROM to prove it UNF-1KCA and IND-1BatchCCA (i.e. our new unforgeability and indistinguishability definitions for split-KEM). This construction incurs a security loss as usual in the (Q)ROM, but our final split-KEM still provides around 128 (resp. 64) bits of security in the ROM (resp. QROM) assuming the adversary is limited to 2^{64} (resp. quantum) random oracle queries. In other words, our parameters take into account the loss due to the (Q)ROM proof.

- We benchmark our protocol K-Waay using our modified version of FrodoKEX (which we call FrodoKEX+) as the split-KEM, along with standard X3DH and the two previous proposals for PQ X3DH-like AKE [Has+22; Bre+22]. We find that while K-Waay has larger prekeys, it is approximately 6× faster compared to these. In addition, the only non-standard primitive we use in K-Waay (i.e. FrodoKEX+) is based on both an assumption (i.e. LWE) and a scheme (FrodoKEM) that have been thoroughly scrutinised by the cryptographic community. Overall, we believe our protocol more mature for short to medium-term integration compared to previous work based on ring signatures.

7.3 Additional Related Work

The security of X3DH has been modelled in detail by Cohn-Gordon et al. [Coh+20]. Vatandas et al. [Vat+20] investigated the deniability of X3DH and similar key exchange protocols under the deniability notion of Di Raimondo et al. [RGK06], requiring strong knowledge-of-exponent-type assumptions to prove X3DH secure. In 2022, Dobson and Galbraith [DG22] proposed a X3DH-like protocol based on SIDH, which is thus now broken. More recently, Kiltz et al. [Kil+23] proved X3DH tightly-secure in the generic group model under a new multi-user assumption although they do not allow the adversary to expose parties' session states.

Unger and Goldberg built a number of different DAKEs [UG15; UG18]. However, the protocols do not provide post-quantum guarantees: even though it is suggested in the latter work [UG18] to add a PQ KEM for post-quantum *confidentiality*. Nevertheless, the protocols provide relatively strong online deniability (i.e. where a judge and a party can communicate while trying to frame another party) at the expense of stronger primitives like dual-receiver encryption and non-committing encryption.

In 2021, Alwen et al. [Alw+21] defined the notion of authenticated key encapsulation mechanism (AKEM) and some security definitions. AKEM captures the same primitive as a split-KEM, but we opted for the syntax and language of the latter as it was meant to be used in a X3DH-like protocol.

Cremers and Feltz [CF11] introduced peer deniability, which captures the kind of participation deniability property we are after in this chapter, namely that a party cannot deny using a system but can deny communicating with a particular party. However, their security notion does not require the simulator to output the session key and the adversary to distinguish between the real and simulated key, and so composability issues may arise from using it.

7.4 Technical Overview

As the length of this chapter is consequent, we review in this section the techniques developed and used in the design of K-Waay.

7.4.1 X3DH-like key exchange

A quantum-secure X3DH-like protocol should satisfy certain properties. Apart from satisfying standard authenticated key exchange (AKE) properties like secrecy and authentication, it should also be *asynchronous*. That is, parties should be able to upload keying material to a central server, after which an initiating party can derive a session key immediately with their counterpart who may be offline. This also entails *receiver-obliviousness* in the language of Hashimoto et al. [Has+22] as the initial key upload should not depend on the keys of any other party. Another is *deniability*, allowing parties to claim that they plausibly did not participate in the key exchange. Note that we cannot possibly ensure that parties can claim that they never uploaded prekeys as they are signed (and using e.g. ring signatures would violate receiver-obliviousness). Finally, a DAKE should, like X3DH, provides security guarantees even if the session state of a party is leaked.

7.4.2 Revisiting split-KEM

In an attempt to model the primitive central to X3DH-like AKE, Brendel et al. [Bre+22] introduced *split-KEM*, which is similar to a standard KEM except the encapsulator can contribute to the derived key with their own secret key. However, we discovered that the accompanying security definitions were not sufficient to use such a primitive as the main component of an AKE. The reason being that their notions ensure that an encapsulated ciphertext will not leak information on its encapsulated key, but not that only the sender can send a "legitimate" ciphertext to the receiver (or that only the sender and receiver can derive a common key). In other words, there is no guarantee of implicit authentication. Therefore, we introduce the notion of unforgeability against one-known-ciphertext attacks for split-KEM (UNF-1KCA), which roughly states that if Alice receives a message allegedly sent by Bob, either Bob really sent it or the decapsulation will fail. Jumping ahead, this will be used in the security proof of the protocol to argue that either the adversary relayed a legit split-KEM ciphertext to the receiver, or the sender aborts as the ciphertext is forged.

We also introduce an intermediary notion of decaps-OW-CPA, which says that an adversary should not be able to recover a key *decapsulated* by some party without knowing the sender's or the receiver's secret key. We then prove that our lattice-based split-KEM satisfies such a definition and we apply some transform in the (Q)ROM to obtain a UNF-1KCA split-KEM.

Finally, we also define the notion of deniability for split-KEM, which states that no party J can be convinced that a party B sent a given ciphertext to A, even knowing A's secret key but assuming both parties did not deviate from the protocol. This models a setting where A communicates with B and later tries to frame the latter by giving the transcript and their own secret key to J.

7.4.3 Construction

As any X3DH-like protocol, our construction works in 4 phases: long-term key generation, prekey generation, send, and receive. The first observation we make is that in X3DH implementation, prekey bundles are signed with a long-term signing key before being uploaded to the server. This fact is often abstracted away in formal analysis as it hurts the claims one can make about the deniability of X3DH: as a signature is undeniable by definition, users cannot deny they *participated* in the protocol. Based on this, our goal was to achieve some level of peer-deniability [CF11], where parties can deny they communicated with someone in particular, and to leverage the fact that we use signatures to authenticate the prekeys. Our protocol works then as follows (see Figure 7.1 for a high-level overview). The long-term key

$$
\begin{array}{lcl}
\underline{\text{Bob } (\mathsf{lpk_A}, \mathsf{vk_A})} & & \underline{\text{Alice } (\mathsf{lpk_B}, \mathsf{vk_B})} \\
(\mathsf{esk}_\mathsf{B}^\mathsf{skem}, \mathsf{epk}_\mathsf{B}^\mathsf{skem}) \leftarrow\!\!\$\; \mathsf{KeyGenB_{sKEM}} & & (\mathsf{esk}_\mathsf{A}^\mathsf{skem}, \mathsf{epk}_\mathsf{A}^\mathsf{skem}) \leftarrow\!\!\$\; \mathsf{KeyGenA_{sKEM}} \\
 & & (\mathsf{esk}_\mathsf{A}^\mathsf{kem}, \mathsf{epk}_\mathsf{A}^\mathsf{kem}) \leftarrow\!\!\$\; \mathsf{KeyGen_{KEM}} \\
 & \xleftarrow{\{\mathsf{epk}_\mathsf{A}^\mathsf{skem}, \mathsf{epk}_\mathsf{A}^\mathsf{kem}\}_{\mathsf{sk_A}}} \quad \xrightarrow{\{\mathsf{epk}_\mathsf{B}^\mathsf{skem}\}_{\mathsf{sk_B}}} & \\
\hline
K_\ell, \mathsf{ct}_\ell \leftarrow\!\!\$\; \mathsf{Encaps_{KEM}}(\mathsf{lpk_A}) & & \\
K_e^\mathsf{skem}, \mathsf{ct}_e^\mathsf{skem} \leftarrow\!\!\$\; \mathsf{Encaps_{sKEM}}(\mathsf{epk}_\mathsf{A}^\mathsf{skem}, \mathsf{esk}_\mathsf{B}^\mathsf{skem}) & & \\
K_e^\mathsf{kem}, \mathsf{ct}_e^\mathsf{kem} \leftarrow\!\!\$\; \mathsf{Encaps_{KEM}}(\mathsf{epk}_\mathsf{A}^\mathsf{kem}) & & \\
K \leftarrow \mathsf{KDF}(K_\ell, K_e^\mathsf{skem}, K_e^\mathsf{kem}) & \xrightarrow{\mathsf{ct}_\ell, \mathsf{ct}_e^\mathsf{skem}, \mathsf{ct}_e^\mathsf{kem}} & \\
 & & K'_\ell \leftarrow \mathsf{Decaps_{KEM}}(\mathsf{lsk_A}, \mathsf{ct}_\ell) \\
 & & K_e'^{\,\mathsf{skem}} \leftarrow \mathsf{Decaps_{sKEM}}(\mathsf{epk}_\mathsf{B}^\mathsf{skem}, \mathsf{esk}_\mathsf{A}^\mathsf{skem}, \mathsf{ct}_e^\mathsf{skem}) \\
 & & K_e'^{\,\mathsf{kem}} \leftarrow \mathsf{Decaps_{KEM}}(\mathsf{esk}_\mathsf{A}^\mathsf{kem}, \mathsf{ct}_e^\mathsf{kem}) \\
 & & K'_\ell \stackrel{?}{=} \perp \vee K_e'^{\,\mathsf{skem}} \stackrel{?}{=} \perp \vee K_e'^{\,\mathsf{kem}} \stackrel{?}{=} \perp \\
 & & K' \leftarrow \mathsf{KDF}(K'_\ell, K_e'^{\,\mathsf{skem}}, K_e'^{\,\mathsf{kem}})
\end{array}
$$

Figure 7.1: High-level overview of the K-Waay protocol. Values in brackets $\{\cdot\}_{\mathsf{sk}}$ are signed with sk and the signature is verified upon reception. For clarity, we omit the calculation and addition of session identifier sid to KDF.

pair consists of a KEM and signature key pair, the latter being used to sign the prekey, which comprises an ephemeral KEM key pair and ephemeral split-KEM key pair. The former is used for forward secrecy while the second is used for the implicit authentication of the sender. Although usually ephemeral keys cannot be used for authentication as they are dynamic, in our case we can since they are authenticated (i.e. signed) by their owner. Then, the sender encapsulates against both KEM public keys of the receiver, and uses their own split-KEM secret key and the receiver's public key to derive a split-KEM ciphertext. Upon decapsulation, the receiver recovers the three keys and combines them using a PRF to derive the shared key.

Ephemeral split-KEM key reuse. The way our protocol is described above works perfectly well if the split-KEM satisfies the UNF-1KCA unforgeability notion introduced above. However, in practice, it could happen that some party, say Alice, is offline for too long and all their ephemeral split-KEM keys have been used. If that occurs, another sender would have to wait for Alice to come online and upload new keys before they can send her a message.

We fix this issue by modifying the protocol as follows: when Alice's ephemeral public keys have run out on the server, a sender can simply reuse one of them. Then, when Alice is back online, she groups the ciphertexts corresponding to the same public key and decrypts all ciphertexts in a group *at once*. If one or more of the decapsulations of the split-KEM ciphertexts in a group fails, Alice outputs $\perp$ for *all* ciphertexts and e.g., restarts the protocol. Otherwise, Alice proceeds as before (and *never* decapsulates again using the same split-KEM key). We formally model this key reuse with an algorithm BatchReceive that takes as input a given session state and one or more messages to be received.

Security. We show this version of the protocol is secure assuming the split-KEM satisfies a stronger notion than IND-CPA that we call IND-1BatchCCA. This definition is the same as traditional IND-CPA (adapted to the split-KEM syntax), except the adversary can query a decapsulation oracle *once* with multiple public keys and ciphertexts, and the oracle returns $\perp$ if *one or more* of the decapsulations failed, and the resulting keys otherwise. We show that one can easily build an IND-1BatchCCA split-KEM out of a CPA secure one in the (Q)ROM, conveniently using the same transform mentioned above that builds a UNF-1KCA scheme out of a decaps-OW-CPA one.

As in previous protocols ([Bre+22; Has+22]), the long-term KEM provides implicit authentication of the receiver as only they can decrypt. The ephemeral KEM provides forward secrecy, and the UNF-1KCA/IND-1BatchCCA split-KEM provides implicit authentication of the sender, as it guarantees that only the sender could have sent a ciphertext that correctly decapsulates (unforgeability), and no adversary knows what is inside that ciphertext (indistinguishability), even after seeing the decapsulation of one batch of ciphertexts encapsulated against the same public key (if all decapsulated correctly). We note that the sender-to-receiver authentication depends both on a long-term key (i.e. the signing key), and an ephemeral one (the split-KEM key). Consequently, our model (that allows session state exposure) is more restrictive than that of Hashimoto et al. [Has+22], since in particular it suffices for the adversary to learn a receiver's ephemeral state during key exchange to forge a message that the receiver accepts. Intuitively, this is because a split-KEM is effectively a symmetric primitive.

Deniable split-KEM from lattices. We provide the first lattice-based split-KEM which satisfies both deniability and UNF-1KCA security. Our starting point is the Frodo key-exchange (FrodoKEX) [Bos+16], which was identified (among other schemes) as a split-KEM by Brendel et al. [Bre+21], the security of which relies on the well-known Learning with Errors (LWE) problem [Reg05]. We highlight that the vanilla construction of FrodoKEX does not enjoy the aforementioned properties. Indeed, when looking closely at the security games of deniability, partial information about the secret keys are revealed, which makes a reduction to LWE completely non-trivial[2]. We circumvent this problem in two ways.

First, we reduce deniability of our scheme to a so-called Extended-LWE problem [AP12], where in addition to a standard LWE instance, the adversary is given a short random combination of the secret coefficients. We show that deniability of our scheme reduces straightforwardly to Extended-LWE and then, following the methodology of Alperin-Sheriff and Peikert [AP12], we

[2]Nevertheless, we found no deniability/UNF-1KCA attack on FrodoKEX [Bos+16].

reduce it further to plain LWE. In order to make the reduction tighter, we use an odd modulo, unlike in the original FrodoKEX scheme.

Then, towards UNF-1KCA security, we slightly modify the Frodo split-KEM by introducing masking terms that make the security proof go through. In Section 7.9.2 we discuss the necessity of these (seemingly artificial) changes.

7.5 Split-KEM

As mentioned above, the primitive at the core of our protocol is a split-KEM, which we present in this section. It was first defined by Brendel et al. [Bre+21].

Definition 7.5.1 (Split-KEM)**.** *An (asymmetric) split-KEM* sKEM *is a tuple of four efficient algorithms* $(\mathsf{KeyGenA}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *defined as follows:*

- $(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$ *(resp.* $(\mathsf{pk_B}, \mathsf{sk_B}) \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$*): The key generation function of the first/second party takes the security parameter* λ *as input, and outputs a pair of public/secret keys* $(\mathsf{pk_A}, \mathsf{sk_A})$ *(resp.* $(\mathsf{pk_B}, \mathsf{sk_B})$*).*
- $K, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$*: The encapsulation function takes the public key* $\mathsf{pk_A}$ *of a party* A *and the other party's secret key* $\mathsf{sk_B}$ *as inputs, and outputs a ciphertext* ct *and a key* K*.*
- $K/\bot \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct})$*: The decapsulation function takes the secret key* $\mathsf{sk_A}$ *of a party* A*, the other party's public key* $\mathsf{pk_B}$ *and a ciphertext* ct *as inputs, and outputs a key* K *or the error symbol* $\bot$*.*

We say a split-KEM is δ*-correct if*

$$\Pr\left[K \neq K' : \begin{array}{c} (\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow\$ \mathsf{KeyGenA}(1^\lambda); \\ (\mathsf{pk_B}, \mathsf{sk_B}) \leftarrow\$ \mathsf{KeyGenB}(1^\lambda); \\ K, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B}); \\ K' \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct})] \end{array}\right] \leq \delta \, .$$

Intuitively, a split-KEM is similar to a normal KEM except material from both participants is used for encapsulation (i.e. the final key will depend on both parties' secret/public keys). In a X3DH-like protocol, it can be used to implicitly authenticate the party encapsulating. In the language of Brendel et al. [Bre+21], our notion of split-KEM is "asymmetric", as it is assumed that B always encapsulates and A always decapsulates. This is sufficient for our purpose, but we note that all the results presented in this chapter can be adapted to a symmetric split-KEM where $\mathsf{KeyGenA} = \mathsf{KeyGenB}$.

$\underline{\text{IND-1BatchCCA}_{\mathsf{sKEM}}(\mathscr{A})}$

$b \leftarrow\$ \{0,1\}$
$q \leftarrow 0$
$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$
$K_0, \mathsf{ct}^* \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$
$K_1 \leftarrow\$ \mathscr{K}$
$b' \leftarrow\$ \mathscr{A}^{\text{BatchDec}}(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}^*, K_b)$
$\textbf{return } 1_{b'=b}$

$\underline{\text{BatchDec}(\{(\mathsf{pk}_i, \mathsf{ct}_i)\}_{i=1}^d)}$

$\textbf{if } q = 1 : \textbf{return } \bot$
$\textbf{else } : q \leftarrow q + 1$
$\textbf{for } i \in \{1, \ldots, d\} :$
$\quad \textbf{if } (\mathsf{pk}_i, \mathsf{ct}_i) = (\mathsf{pk_B}, \mathsf{ct}^*) : \textbf{return } \bot$
$\quad K_i' \leftarrow \mathsf{Decaps}(\mathsf{pk}_i, \mathsf{sk_A}, \mathsf{ct}_i)$
$\textbf{if } K_1 = \bot \vee \ldots \vee K_d = \bot : \textbf{return } \bot$
$\textbf{return } (K_1, \ldots, K_d)$

$\underline{\text{OW-CPA}_{\mathsf{sKEM}}(\mathscr{A})}$

$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$
$K^*, \mathsf{ct}^* \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$
$K' \leftarrow\$ \mathscr{A}(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}^*)$
$\textbf{return } 1_{K'=K^*}$

Figure 7.2: IND-1BatchCCA and OW-CPA games.

7.5.1 Security

We will need several security properties from the split-KEM to prove our whole protocol secure. We first define one-wayness (OW-CPA[3]) for sKEM, which is very similar to the usual one for KEM, and another new notion called IND-1BatchCCA. Looking ahead, we will show that any OW-CPA split-KEM can easily be transformed into a IND-1BatchCCA one in the (Q)ROM.

Definition 7.5.2 (split-KEM OW-CPA)**.** *We consider the* OW-CPA *game defined in Figure 7.2. A split-KEM scheme* $\mathsf{sKEM} = (\mathsf{KeyGen_A}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* OW-CPA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{sKEM}}(\mathscr{A}) := \Pr[\text{OW-CPA}_{\mathsf{sKEM}}(\mathscr{A}) \Rightarrow 1] = \mathsf{negl}\,.$$

Definition 7.5.3 (split-KEM IND-1BatchCCA)**.** *We consider the* IND-1BatchCCA *game defined in Figure 7.2. Let* $\mathscr{K}$ *be a finite key space. A split-KEM scheme over* $\mathscr{K}$ $\mathsf{sKEM} = (\mathsf{KeyGen_A}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* IND-1BatchCCA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{ind-1batchcca}}_{\mathsf{sKEM}}(\mathscr{A}) := \left|\Pr[\text{IND-1BatchCCA}_{\mathsf{sKEM}}(\mathscr{A}) \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{negl}\,.$$

We also recall the different notions of indistinguishability for (asymmetric) split-KEM defined by Brendel et al. [Bre+21]:

[3] Strictly speaking, the notion does not correspond to the "chosen-plaintext attack" setting, but we call it OW-CPA nonetheless as it resembles the OW-CPA notion for KEMs (Definition 2.2.11).

$\text{xy-IND-CCA}_{\mathsf{sKEM}}(\mathscr{A})$

$b \leftarrow\!\$\, \{0,1\}$
$n_x \leftarrow 0; n_y \leftarrow 0;$
$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\!\$\, \mathsf{KeyGenA}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\!\$\, \mathsf{KeyGenB}(1^\lambda)$
$K_0, \mathsf{ct}^* \leftarrow\!\$\, \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$
$K_1 \leftarrow\!\$\, \mathscr{K}$
$b' \leftarrow\!\$\, \mathscr{A}^{\text{DEC,ENC}}(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}^*, K_b)$
$\textbf{return } 1_{b'=b}$

$\mathsf{ENC}(\mathsf{pk})$

$\textbf{if } n_y \geq \mathrm{y} : \textbf{return } \perp$
$n_y \leftarrow n_y + 1$
$K, \mathsf{ct} \leftarrow\!\$\, \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$\textbf{if } (\mathsf{pk}, \mathsf{ct}) = (\mathsf{pk_A}, \mathsf{ct}^*) : \textbf{return } \perp$
$\textbf{return } K, \mathsf{ct}$

$\mathsf{DEC}(\mathsf{pk}, \mathsf{ct})$

$\textbf{if } n_x \geq \mathrm{x} : \textbf{return } \perp$
$n_x \leftarrow n_x + 1$
$\textbf{if } (\mathsf{pk}, \mathsf{ct}) = (\mathsf{pk_B}, \mathsf{ct}^*) : \textbf{return } \perp$
$\textbf{return } \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct})$

Figure 7.3: xy-IND-CCA games for an "asymmetric" split-KEM from Brendel et al. [Bre+21], where x,y ∈ {n, s, m}. When doing the comparison on the first line of both oracles, we assume n = 0, s = 1 and m = ∞.

Definition 7.5.4 (split-KEM xy-IND-CCA)**.** *We consider the* xy-IND-CCA *game defined in Figure 7.3. A split-KEM scheme* $\mathsf{sKEM} = (\mathsf{KeyGen_A}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* xy-IND-CCA*, with* x, y ∈ {n, s, m} *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{xy-ind-cca}}_{\mathsf{sKEM}}(\mathscr{A}) := \left| \Pr\left[\text{xy-IND-CCA}_{\mathsf{sKEM}}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2} \right| = \mathsf{negl}\,.$$

These indistinguishability notions range from nn-IND-CCA, which is similar to some kind of IND-CPA security as the adversary has no access to encapsulation or decapsulation oracles, to mm-IND-CCA, which captures strong IND-CCA security for split-KEMs. More generally, all notions are of the form xy-IND-CCA, x, y ∈ {n, s, m}, where x (resp. y) specifies the number of queries an adversary can make to the decapsulation (resp. encapsulation) oracle (i.e. none, single, or many).

On the original split-KEM security. We recall that the advantage of split-KEMs over normal KEMs is that they capture the fact that the party encapsulating can contribute (*static*) keying material towards the shared key, whereas it is not the case with KEMs, as the encapsulation function only takes the receiving party's public key as input (and random coins). In particular, this means that KEMs cannot be used for implicit authentication of the encapsulator, unlike split-KEMs. However, we argue that the original xy-IND-CCA definitions for split-KEMs [Bre+21] do not capture implicit authentication either and thus are not suited for their purpose (i.e. building an asynchronous DAKE). In fact, any IND-CPA (resp. IND-CCA) KEM can easily be converted to an (asymmetric) split-KEM satisfying nn-IND-CCA (resp. mm-IND-CCA).

More formally, imagine a setting where Alice and Bob know each other's public key, and Bob wants to implicitly authenticate to Alice using a split-KEM. In addition, we assume that a mm-IND-CCA split-KEM sKEM_0 exists (note mm-IND-CCA security is the strongest so this holds for all weaker notions). We first modify sKEM_0 such that on a special ciphertext $\mathsf{ct}^\star$ not in the original ciphertext space, Decaps returns a constant key $K^\star$. Let's call this modified scheme sKEM. We observe that sKEM is still mm-IND-CCA secure as no adversary can break an honestly-generated challenge ciphertext. Now, implicit authentication means that if Alice decapsulates a ciphertext and obtains a key K, then only Bob knows K. However, in our case, any adversary can send $\mathsf{ct}^\star$ to Alice and set their own key to $K^\star$. Both the adversary and Alice will share the same key and implicit authentication does not hold. In a way, xy-IND-CCA security does not prevent forgeries.

UNF-1KCA. This leads us to define our notion of UNF-1KCA security for split-KEMs below which, along with OW-CPA (which can be turned into IND-1BatchCCA), guarantees that only Bob (and obviously Alice) can know the result of Alice's decapsulation on some ciphertext. More precisely, UNF-1KCA ensures that no adversary can forge a valid split-KEM ciphertext for B even knowing a ciphertext that was computed with respect to a public key chosen by the adversary[4], under the condition that the public key used for encapsulation and the known ciphertext are different from the pair made of A's public key and the ciphertext output by the adversary. We also define a security notion called decaps-OW-CPA that will serve as a building block to build UNF-1KCA. The decaps-OW-CPA notion ensures that it is hard for an adversary knowing a ciphertext ct (under an adversarially-chosen public key) to come up with a ciphertext ct' (possibly equal to ct) and a key K' such that the decapsulation of ct' returns K'.

Definition 7.5.5 (split-KEM UNF-1KCA)**.** *We consider the* UNF-1KCA *game defined in Figure 7.4. A split-KEM scheme* $\mathsf{sKEM} = (\mathsf{KeyGen}_\mathsf{A}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* UNF-1KCA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{unf-1kca}}_{\mathsf{sKEM}}(\mathscr{A}) := \Pr[\text{UNF-1KCA}_{\mathsf{sKEM}}(\mathscr{A}) \Rightarrow 1] = \mathsf{negl}\,.$$

Definition 7.5.6 (split-KEM decaps-OW-CPA)**.** *We consider the* decaps-OW-CPA *game defined in Figure 7.4. A split-KEM scheme* $\mathsf{sKEM} = (\mathsf{KeyGen}_\mathsf{A}, \mathsf{KeyGenB}, \mathsf{Encaps}, \mathsf{Decaps})$ *is* decaps-OW-CPA *if for any efficient adversary* $\mathscr{A}$ *we have*

$$\mathsf{Adv}^{\text{decaps-ow-cpa}}_{\mathsf{sKEM}}(\mathscr{A}) := \left|\Pr\left[\text{decaps-OW-CPA}_{\mathsf{sKEM}}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2}\right| = \mathsf{negl}\,.$$

7.5.2 Deniability

We finally state the notion of split-KEM deniability we would like to achieve.

[4]Looking ahead, the fact that the public key is adversarially-chosen will be useful for proving security under key-compromise impersonation attacks for our full protocol.

$\text{UNF-1KCA}_{\mathsf{sKEM}}(\mathscr{A})$	$\text{decaps-OW-CPA}_{\mathsf{sKEM}}(\mathscr{A})$
$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$	$b \leftarrow\$ \{0,1\}$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$	$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}(\mathsf{pk_A}, \mathsf{pk_B})$	$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$
$K_\mathsf{B}, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$	$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}(\mathsf{pk_A}, \mathsf{pk_B})$
$\mathsf{ct}' \leftarrow\$ \mathscr{A}(\mathsf{st}, \mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$	$K_\mathsf{B}, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$\textbf{if } (\mathsf{ct}, \mathsf{pk}) = (\mathsf{ct}', \mathsf{pk_A}) : \textbf{return } 0$	$K'_\mathsf{A}, \mathsf{ct}' \leftarrow\$ \mathscr{A}(\mathsf{st}, \mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct})$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$	$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$\textbf{if } K_\mathsf{A} = \perp : \textbf{return } 0$	$\textbf{if } K_\mathsf{A} = \perp : \textbf{return } 0$
$\textbf{return } 1$	$\textbf{return } 1_{K_\mathsf{A} = K'_\mathsf{A}}$

Figure 7.4: Games UNF-1KCA and decaps-OW-CPA.

$\text{DENY}^{\mathsf{REAL}}_{\mathsf{sKEM,Sim}}(\mathscr{A})$	$\text{DENY}^{\mathsf{SIM}}_{\mathsf{sKEM,Sim}}(\mathscr{A})$
$(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$	$(\mathsf{pk_A}, \mathsf{sk_A}) \leftarrow\$ \mathsf{KeyGenA}(1^\lambda)$
$(\mathsf{pk_B}, \mathsf{sk_B}) \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$	$(\mathsf{pk_B}, \mathsf{sk_B}) \leftarrow\$ \mathsf{KeyGenB}(1^\lambda)$
$K, \mathsf{ct} \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$	$K, \mathsf{ct} \leftarrow\$ \mathsf{Sim}(\mathsf{pk_B}, \mathsf{sk_A})$
$b \leftarrow\$ \mathscr{A}(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{sk_A}, K, \mathsf{ct})$	$b \leftarrow\$ \mathscr{A}(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{sk_A}, K, \mathsf{ct})$
$\textbf{return } b$	$\textbf{return } b$

Figure 7.5: Deniability game.

Definition 7.5.7 (Deniability)**.** *We consider the game shown in Figure 7.5. We say a split-KEM* sKEM *is* DENY *if there exists a simulator* Sim *s.t. for all efficient adversaries* $\mathscr{A}$*, we have*

$$\mathsf{Adv}^{\text{deny}}_{\mathsf{sKEM,Sim}}(\mathscr{A}) := \left|\Pr[\text{DENY}^{\mathsf{REAL}}_{\mathsf{sKEM,Sim}}(\mathscr{A}) \Rightarrow 1] - \Pr[\text{DENY}^{\mathsf{SIM}}_{\mathsf{sKEM,Sim}}(\mathscr{A}) \Rightarrow 1]\right| = \mathsf{negl}\,.$$

Informally, the setting considered is the following. Alice and Bob use the split-KEM to establish a shared key (we assume the public keys are only used for this one exchange), and Alice (while following the protocol) wants to frame Bob and prove that he did communicate with her. Therefore, after receiving Bob's ciphertext and deriving the key, Alice gives both public keys, the derived key, the ciphertext and her own secret key to a judge (i.e. the adversary) that must decide whether Bob actually sent the ciphertext that was used to derive the key or not. The scheme is deniable if there is a simulator that, given Alice's view, outputs a ciphertext and a key indistinguishable from the ones output by Bob.

7.6 Model for DAKE

In this section, we describe our model for deniable authenticated key exchange (DAKE) that we tailor to the semantics and flow of X3DH.

7.6.1 Syntax

A DAKE is a tuple of four efficient algorithms $(\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Send}, \mathsf{BatchReceive})$ defined as follows:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$. This function takes as input the security parameter λ and outputs the long-term public/secret key pair of the caller.

- $(\mathsf{st}_i, \mathsf{prek}_i) \leftarrow\$ \mathsf{Init}(\mathsf{sk}_i, \mathsf{role})$. This function takes as inputs a long-term secret key sk_i and a role $\mathsf{role} \in \{\mathsf{sender}, \mathsf{receiver}\}$ and outputs a session state st_i and a prekey bundle prek_i. Init models the creation of key material that will be uploaded to the public key infrastructure by both parties (e.g., a prekey bundle in X3DH). The output values depend only on the public key of party i executing the function.

- $(\mathsf{k}, m) \leftarrow\$ \mathsf{Send}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{st}_i, \mathsf{prek}_j)$. This function takes as inputs the secret key of the executing party i, the public key of the intended recipient pk_j, party i's session state st_i and the (claimed) prekey bundle of the intended recipient prek_j, and outputs a key k and a message m.

- $\{k_s\}_s \leftarrow \mathsf{BatchReceive}(\mathsf{sk}_i, \mathsf{st}_i, \{\mathsf{pk}_j, \mathsf{prek}_j, m_j\}_j)$. This function takes as inputs the secret key of the executing party i, an ephemeral state of party i st_i and a vector of size $d \geq 1$ of the form $(\mathsf{pk}_j, \mathsf{prek}_j, m_j)$ for party i's session with the public key of the (claimed) sender pk_j, the (claimed) prekey bundle of party j prek_j and a message m_j, and outputs a vector of d keys $(\mathsf{k}_1, \ldots, \mathsf{k}_d)$, some or all of which may be $\perp$.

Init explicitly captures parties uploading ephemeral keys to a central server in the first protocol step. This contrasts with the formal modelling in some previous works on X3DH-like key exchange [Bre+22; Has+22] that model a three-move key exchange with a single initiator.

The most novel part of our primitive is $\mathsf{BatchReceive}$ which captures ephemeral key reuse when uploaded ephemeral keys are exhausted. In the case of key exhaustion, when a party comes back online, they execute $\mathsf{BatchReceive}$ several times (once per ephemeral state st_i), where the number of inputs of the form $(\mathsf{pk}_j, \mathsf{prek}_j, m_j)$ in a given $\mathsf{BatchReceive}$ call corresponds to how many times st_i is re-used. Otherwise, $\mathsf{BatchReceive}$ can be used as in standard AKE with a single value $(\mathsf{pk}_j, \mathsf{prek}_j, m_j)$ as input.

7.6.2 Security model

We now describe the security model we consider for our DAKE, which extends existing models in some ways to support $\mathsf{BatchReceive}$.

Parties and sessions

We assume that there are n parties $P_1, \ldots P_n$ (or $1, \ldots, n$), where party P_i (resp. or i) is associated with long-term key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ output by KeyGen. Each party runs one or more *sessions* (sometimes called *oracles* [Bre+21]), where the s-th session of P_i is denoted by π_i^s. Each session π_i^s is associated with the following local fields:

- sid, the session identifier or session id.
- pid, the partner identifier.
- $\mathsf{role} \in \{\perp, \mathsf{sender}, \mathsf{receiver}\}$, the role of P_i.
- $\mathsf{status} \in \{\perp, \mathsf{accept}, \mathsf{reject}\}$, the status of π_i^s.
- k, the session key.
- st, the session state.
- rand, the session randomness.

All fields are initialised to $\perp$ except rand which is initialised to uniform randomness. A session either has role sender or receiver, and its counterpart, its *partner* pid, has the other role; note a receiver may have several counterparts (capturing ephemeral key reuse).

Fields pid, role, status and rand in session π_i^s are set directly by the challenger, and the rest are (sometimes implicitly) set by the underlying DAKE algorithms called by the challenger. Moreover, in light of the definition of BatchReceive, sid, pid and k are *vectors* for a receiver ($\mathsf{role} = \mathsf{receiver}$); we sometimes write $\vec{\mathsf{sid}}$, $\vec{\mathsf{pid}}$ and $\vec{\mathsf{k}}$ for clarity to indicate this.

Suppose P_i is acting as a receiver. Initially, P_i calls Init, and then eventually calls BatchReceive. Before this point, one or more senders P_j (i.e., parties with $\mathsf{role} = \mathsf{sender}$) may call Init and then Send with respect to the output prek from P_i's Init call (assuming honest message delivery), which output messages of the form m_j. Finally, P_i invokes BatchReceive with one or more m_j values as input. A party has $\mathsf{status} = \mathsf{accept}$ if and only if $\mathsf{k} \neq \perp$[5], and stores any session state after calling Init and before setting $\mathsf{status} \neq \perp$ due to a Send or BatchReceive call in st.

Partnering

We define partnering between two sessions to capture security using session identifiers:

Definition 7.6.1 (Partnering). *For any (i, P_j, s, t), we say that sessions π_i^s and π_j^t are partners if*

1. $\pi_i^s.\mathsf{role} \neq \pi_j^t.\mathsf{role}$.

[5] In particular, BatchReceive may output several keys; as long as at least one of them is not $\perp$, the calling party accepts.

2. *If* $\pi_i^s.\mathsf{role} = \mathsf{sender}$*, then* $\pi_i^s.\mathsf{pid} = j$ *and* $i \in \pi_j^t.\vec{\mathsf{pid}}$*. If* $\pi_i^s.\mathsf{role} = \mathsf{receiver}$*, then* $j \in \pi_i^s.\vec{\mathsf{pid}}$ *and* $i = \pi_j^t.\mathsf{pid}$.

3. *If* $\pi_i^s.\mathsf{role} = \mathsf{sender}$*, then* $\pi_i^s.\mathsf{sid} \in \pi_j^t.\vec{\mathsf{sid}} \neq \perp$*. If* $\pi_i^s.\mathsf{role} = \mathsf{receiver}$*, then* $\pi_j^t.\mathsf{sid} \in \pi_i^s.\vec{\mathsf{sid}} \neq \perp$.

Looking ahead, this definition ensures that two sessions can only be partners if they both have set $\mathsf{status} = \mathsf{accept}$. Our definition mainly differs from previous work in that there can be many senders (and thus partnered sessions) for a given receiver. Ignoring this aspect, our definition is only slightly different from that of Hashimoto et al. [Has+22] in that we restrict sid to be not equal to $\perp$; this is an artifact of the fact we model "four-move" key exchange (including prekey uploading).

KIND Security Game

We first define key indistinguishability (KIND) and then define deniability separately. Following previous work, we define a KIND experiment played between a challenger C and adversary $\mathscr{A}$ in text below. The experiment $\mathrm{KIND}^n_{\mathsf{DAKE}}$ is parameterised by the DAKE DAKE and integer n, the number of parties (honest or otherwise) in the lifetime of the game's execution. The game is divided into distinct phases defined as follows.

Setup. C first uniformly samples challenge bit $b \in \{0,1\}$. Then, for each party P_i, C calls $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$ and provides $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ and 1^λ as input to $\mathscr{A}$.

Phase 1. $\mathscr{A}$ adaptively makes any number of the following queries in any order:

- $\mathrm{EXEC}(i, s, \mathsf{prek}, m)$: $\mathscr{A}$ starts or runs the next step of execution in session π_i^s. In each call, C uses randomness tape $\pi_i^s.\mathsf{rand}$ as needed.
  - To start the execution in session π_i^s not previously started, $\mathscr{A}$ calls $\mathrm{EXEC}(i, s, \mathsf{prek}, m)$ with special input $m = (\mathtt{start}, \mathsf{sender}, j)$ (resp. $(\mathtt{start}, \mathsf{receiver}, \vec{j})$) (where $\mathtt{start}$ is defined only in the context of this game) that, if not previously called, sets $\pi_i^s.\mathsf{pid} = j$ (resp. $\pi_i^s.\mathsf{pid} = \vec{j}$) and $\pi_i^s.\mathsf{role} = \mathsf{sender}$ (resp. $\pi_i^s.\mathsf{role} = \mathsf{receiver}$); observe input prek is ignored by C. Then, C invokes $(\mathsf{st}_i, \mathsf{prek}_i) \leftarrow\!\!\$\ \mathsf{Init}(\mathsf{sk}_i, \mathsf{role})$ and outputs prek_i to $\mathscr{A}$.
  - Given that P_i has started in π_i^s, $\pi_i^s.\mathsf{status} = \perp$ and $\pi_i^s.\mathsf{role} = \mathsf{sender}$, when $\mathscr{A}$ calls $\mathrm{EXEC}(i, s, \mathsf{prek}, \perp)$, C invokes $(\mathsf{k}, m) \leftarrow\!\!\$\ \mathsf{Send}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{st}_i, \mathsf{prek})$ (where $j = \pi_i^s.\mathsf{pid}$), returns output m to $\mathscr{A}$ and sets $\pi_i^s.\mathsf{status}$ to reject (resp. accept) if $\mathsf{k} = \perp$ (resp. $\mathsf{k} \neq \perp$).
  - If $\pi_i^s.\mathsf{role} = \mathsf{receiver}$ and $\pi_i^s.\mathsf{status} = \perp$, when $\mathscr{A}$ calls $\mathrm{EXEC}(i, s, \{s_j, \mathsf{prek}_j, m_j\}_{j \in \vec{j}'})$, C aborts if $\vec{j}' \neq \pi_i^s.\mathsf{pid}$ and otherwise invokes

$\mathsf{k} \leftarrow \mathsf{BatchReceive}(\mathsf{sk}_i, \mathsf{st}_i, \{\mathsf{pk}_j, \mathsf{prek}_j, m_j\}_j)$ and outputs to $\mathscr{A}$ $\perp$ if BatchReceive fails (resp. nothing otherwise) and sets $\pi_i^s.\mathsf{status}$ to reject (resp. accept).

- LTK(i) outputs sk_i. P_i is hereafter *corrupted*.
- REGISTER(pk_i, i) registers a new party P_i for $i > n$ not previously registered, sets their long-term public key to pk_i and distributes pk_i to all other oracles; P_i is immediately marked as *corrupted*.
- STATE(i, s) outputs $\pi_i^s.\mathsf{st}$, which is hereafter *revealed*.
- KEY(i, s, j) outputs $\pi_i^s.\mathsf{k}_j$ if $\pi_i^s.\mathsf{role} = \mathsf{receiver}$ and $\pi_i^s.\mathsf{status} \neq \perp$ and otherwise outputs $\pi_i^s.\mathsf{k}$.

Test. When $\mathscr{A}$ decides to move to the next phase, it issues the following query TEST which (if successful) returns either a real or random key:

- TEST(i, s, j): If $\pi_i^s.\mathsf{status} \neq \mathsf{accept}$, C returns $\perp$. Otherwise:
  - If $\pi_i^s.\mathsf{role} = \mathsf{sender}$, C aborts if $j \neq \pi_i^s.\mathsf{pid}$, and otherwise returns either $\pi_i^s.\mathsf{k}$ if $b = 0$ or a uniformly sampled key k if $b = 1$;
  - If $\pi_i^s.\mathsf{role} = \mathsf{receiver}$, C aborts if $j \not\in \pi_i^s.\vec{\mathsf{pid}}$, and otherwise returns either $\pi_i^s.\mathsf{k}_j$ if $b = 0$ or a uniformly sampled key k if $b = 1$.

  At this point, π_i^s (which we say is with respect to key j if $\pi_i^s.\mathsf{role} = \mathsf{receiver}$) is said to be the *test session*.

Phase 2. $\mathscr{A}$ adaptively issues queries as in Phase 1.

Guess, freshness and correctness. After Phase 2, $\mathscr{A}$ outputs bit b'. Suppose that $\mathscr{A}$ made query TEST(i, s, j), i.e., π_i^s is the test session with respect to key j and $j \in \pi_i^s.\mathsf{pid}$ (with equality at least when $\pi_i^s.\mathsf{role} = \mathsf{sender}$). The following *freshness* conditions are checked by C; if any condition *is not satisfied*, C sets b' to a uniform bit (i.e., $\mathscr{A}$ gains no advantage):

1. KEY(i, s, j') has not been queried, where j' is arbitrary if $\pi_i^s.\mathsf{role} = \mathsf{sender}$ and $j' = j$ if $\pi_i^s.\mathsf{role} = \mathsf{receiver}$.
2. If π_i^s and π_j^t are partners, then KEY(j, t, i') has not been queried, where i' is arbitrary if $\pi_i^s.\mathsf{role} = \mathsf{sender}$ and $i' = i$ if $\pi_i^s.\mathsf{role} = \mathsf{receiver}$.
3. P_i is not corrupted <u>or</u> $\pi_i^s.\mathsf{st}$ has not been revealed.
4. If π_i^s and π_j^t are partners, then P_j is not corrupted <u>or</u> $\pi_j^t.\mathsf{st}$ has not been revealed.

5. If π_i^s has no partner session, then P_j is not corrupted when $\pi_i^s.\mathsf{status} = \perp$.

6. If π_i^s has no partner session, then if $\pi_i^s.\mathsf{role} = \mathsf{sender}$, for any session π_j^t such that prek_j was both output by $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$ and input to Send in π_i^s by C, P_j is not corrupted <u>or</u> $\pi_j^t.\mathsf{st}$ is not revealed.

7. If π_i^s has no partner session, then if $\pi_i^s.\mathsf{role} = \mathsf{receiver}$, for any session π_j^t such that prek_j was both output by $\mathsf{Init}(\mathsf{sk}_j, \mathsf{sender})$ and input to $\mathsf{BatchReceive}$ in π_i^s by C, $\pi_j^t.\mathsf{st}$ is not revealed <u>and</u> $\pi_i^s.\mathsf{st}$ is not revealed.

Then, the following *correctness* conditions are checked by C which, iterating over all relevant parties i, j, k, only consider the subset of sessions corresponding to *honest* protocol runs where $\mathscr{A}$ faithfully follows the protocol specification. If any condition *is satisfied*, C sets $b = b'$ (i.e., $\mathscr{A}$ wins):

1. There exist distinct sessions π_i^s and π_j^t such that $\pi_i^s.\mathsf{role} = \pi_j^t.\mathsf{role}$ and either 1) $\pi_i^s = \mathsf{receiver}$ and $\pi_i^s.\mathsf{sid}_j = \pi_j^t.\mathsf{sid}_i$ or 2) $\pi_i^s.\mathsf{sid} = \pi_j^t.\mathsf{sid}$.

2. Assuming $\pi_i^s.\mathsf{role} = \mathsf{receiver}$, there exist sessions π_i^s with respect to key j and π_j^t that are partners such that $\pi_i^s.\mathsf{k}_j \neq \pi_j^t.\mathsf{k}$ (analogously when $\pi_i^s.\mathsf{role} = \mathsf{sender}$).

3. There exist distinct sessions π_i^s, π_j^t and π_k^u such that $\pi_i^s.\mathsf{status} = \pi_j^t.\mathsf{status} = \pi_k^u.\mathsf{status} = \mathsf{accept}$ and $\pi_i^s.\mathsf{sid}_k = \pi_j^t.\mathsf{sid}_k = \pi_k^u.\mathsf{sid}$ (assuming i, j are receivers here but analogously in other cases).

Finally, the game outputs 1 if and only if $b = b'$.

Security is formally captured in Definition 7.6.2 below.

Definition 7.6.2 (DAKE key indistinguishability)**.** *We consider the KIND game described above. We say a DAKE* DAKE *is* KIND *if for all efficient adversaries* $\mathscr{A}$ *and polynomially-bounded n (the total number of parties), we have*

$$\mathsf{Adv}^{\mathrm{kind}}_{\mathsf{DAKE},n}(\mathscr{A}) := \left|\Pr[\mathrm{KIND}^{n}_{\mathsf{DAKE}}(\mathscr{A}) \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{negl}\,.$$

Discussion. Following previous work, we define freshness conditions to prevent the adversary from mounting trivial attacks. Conditions 1 to 5 correspond exactly to the forward-secure variant of security by Hashimoto et al. [Has+22]. Due to the design of our DAKE K-Waay, we additionally restrict the adversary via conditions 6 and 7. The clauses in these conditions essentially due to the fact that in K-Waay the only secret keying material required to call Send is an ephemeral split-KEM secret. For example, suppose that the tested π_i^s is the receiver. Due to the "symmetric" nature of split-KEM, without these restrictions, revealing $\pi_i^s.\mathsf{st}$ allows the adversary to inject to P_i by simulating Send (akin to a key-compromise impersonation (KCI) attack using P_i's ephemeral state) and trivially distinguish. Consequently, we restrict session state exposure in this case.

Apart from the fact that we make several extensions to typical AKE modelling to capture $\mathsf{BatchReceive}$, the game is closest to that of Hashimoto et al.'s [Has+22] except that we additionally enforce correctness checks as in Brendel et al.'s model [Bre+22]. To capture partnering, we consider partner and key identifiers that may be *vectors* for a receiver, such that several sender sessions may be partnered with a receiver session if, for a given sender session, it partners with a part/component of the receiver session. We do not capture semi-static keys explicitly as in [Bre+22], although in principle they could be captured in Init. Like [Has+22], our game supports message injection, session state exposure (revealing) (unlike [Bre+22]), session key exposure, long-term key exposure (corruption) and adversarial long-term key registration (also considered corruption). During execution, a single challenge test query is made by the adversary that reveals a real or random key output in some session. For $\mathsf{BatchReceive}$ which can output several keys, just one of the output keys are tested.

Trivial attacks. We restrict the adversary's behaviour to prevent trivial attacks by defining *freshness* predicates. Due to our protocol's design, our notion restricts more than the full forward security notion under session state exposure defined by Hashimoto et al. [Has+22]. Our freshness predicates imply weak forward secrecy and implicit authentication given session state exposure is not allowed (enforced in some recent works like [Bad+15; Coh+19]). Brendel et al.'s model provides these guarantees but additionally protects against randomness exposure [Bre+22], whereas we allow exposures on session states under some conditions unlike them.

7.6.3 Deniability

We next introduce our security notion for a deniable DAKE. To this end, we introduce security game $\mathrm{DENY}^{exp}_{\mathsf{DAKE},\mathsf{Sim}}$ in Figure 7.6.

Definition 7.6.3 (DAKE deniability)**.** *We consider the game shown in Figure 7.6. We say a DAKE* DAKE *is* DENY^{exp} *for* $exp \in \{\mathsf{true}, \mathsf{false}\}$ *if there exists an efficient simulator* Sim *s.t. for all efficient adversaries* $\mathscr{A}$ *and polynomially-bounded* n*, we have*

$$\mathsf{Adv}^{\mathsf{deny}}_{\mathsf{DAKE},\mathsf{Sim},\mathsf{exp}}(\mathscr{A}) := \left|\Pr[\mathrm{DENY}^{exp}_{\mathsf{DAKE},n,\mathsf{Sim}}(\mathscr{A}) \Rightarrow 1] - \frac{1}{2}\right| = \mathsf{negl}\,.$$

Our definition captures the following deniability property. Initially, the judge $\mathscr{A}$ is given the long-term keys of all parties. $\mathscr{A}$ then observes honest protocol runs between pairs of parties (via CHAL). Depending on the challenge bit b, either Send or a simulator Sim that takes as input the secret keying material of the receiver trying to frame the sender is executed in each run. Moreover, $\mathscr{A}$ is given the prekey messages independent of b and, if the parameter exp is

$\mathrm{DENY}^{exp}_{\mathsf{DAKE},n,\mathsf{Sim}}(\mathscr{A})$

$b \leftarrow\$ \{0,1\}$
$L \leftarrow \emptyset$
for $i \in [n]$:
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$ \mathsf{KeyGen}(1^\lambda)$
  $L \leftarrow L \cup \{(\mathsf{pk}_i, \mathsf{sk}_i)\}$
$b' \leftarrow\$ \mathscr{A}^{\mathrm{CHAL}}(L)$
return $1_{b'=b}$

$\mathrm{CHAL}(i, j)$

require $i \in [n] \land j \in [n]$
$(k, m) \leftarrow (\perp, \perp)$
$(\mathsf{st}_i, \mathsf{prek}_i) \leftarrow\$ \mathsf{Init}(\mathsf{sk}_i, \mathsf{sender})$
$(\mathsf{st}_j, \mathsf{prek}_j) \leftarrow\$ \mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$
if $b = 0$: $(\mathsf{k}, m) \leftarrow\$ \mathsf{Send}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{st}_i, \mathsf{prek}_j)$
else : $(k, m) \leftarrow\$ \mathsf{Sim}(\mathsf{sk}_j, \mathsf{pk}_i, \mathsf{st}_j, \mathsf{prek}_i, \mathsf{prek}_j)$
$T \leftarrow (\mathsf{prek}_i, \mathsf{prek}_j, m)$
if $exp = \mathsf{true}$: **return** $(\mathsf{k}, T, \mathsf{st}_j)$
else : **return** (k, T)

Figure 7.6: Deniability game.

set to true, also the session state of the receiver in each protocol run. The goal of the adversary is to distinguish whether Send or Sim is being called.

Our notion $\mathrm{DENY}^{\mathsf{false}}$ corresponds most closely with that of Brendel et al. [Bre+22] which was also adopted by Cremers et al. [CZ24]. Due to how Brendel et al.'s AKE primitive is defined, they also consider semi-static key pairs which are also given to the adversary. $\mathrm{DENY}^{\mathsf{true}}$ provides stronger deniability, corresponding in practice to a receiver who co-operates with a judge by handing over the entire contents of their device. Although incomparable formally, our DAKE would not be considered deniable under a notion like that of Brendel et al. [Bre+22] since their protocol does not formally model long-term signatures. Note that our definition, like Brendel et al.'s [Bre+22], can be straightforwardly converted to a "simulation-based" notion like Definition 7.5.7.

Finally, observe that our definition, like that of Brendel et al. [Bre+22] does not consider deniability for the receiver but only for the sender. One could define such a notion, in which the goal is for the judge (adversary) to distinguish between the output of BatchReceive and a simulator Sim that has access to the long-term and ephemeral states of all corresponding senders and is given (honest) ciphertexts output by Send as input. Here, one could argue deniability for K-Waay using the security of the ephemeral KEM and then the KDF.

7.7 K-Waay: Post-Quantum X3DH from Split-KEM

We present our DAKE K-Waay (Key-Exchange with Asynchrony, Authentication, and Peer-Deniability) in Figure 7.7.

Each party is associated with a long-term public/secret key pair which in K-Waay comprises of a signature and KEM key pair generated in KeyGen. In Init, ephemeral KEM and split-KEM keys for both parties are generated and the public keys are signed with the long-term signature key.

```
Init(sk_i, role)
  // prekey generation/upload
  if role = sender :
    (espk_i, essk_i) ←$ KeyGenA_sKEM(1^λ)
    ekpk_i ← ⊥
  else :
    (espk_i, essk_i) ←$ KeyGenB_sKEM(1^λ)
    (ekpk_i, eksk_i) ←$ KeyGen_EKEM(1^λ)
  σ_i ←$ Sign_Sig(sk_i.ssk, (espk_i, ekpk_i))
  prek_i ← (espk_i, ekpk_i, σ_i)
  return (st_i = (essk_i, eksk_i, prek_i), prek_i)
```

```
KeyGen(1^λ)
  // long-term key generation
  (kpk, ksk) ←$ KeyGen_LKEM(1^λ)
  (spk, ssk) ←$ KeyGen_Sig(1^λ)
  pk ← (spk, kpk)
  sk ← (ssk, ksk)
  return (pk, sk)
```

```
Send(sk_i, pk_j, st_i, prek_j)
  (essk_i, eksk_i, prek_i) ← st_i
  (espk_j, ekpk_j, σ_j) ← prek_j
  require Vrfy_Sig(pk_j.spk, (espk_j, ekpk_j), σ_j)
  (K_ℓ, ct_ℓ) ←$ Encaps_LKEM(pk_j.kpk)
  (K_k, ct_k) ←$ Encaps_EKEM(ekpk_j)
  (K_s, ct_s) ←$ Encaps_sKEM(espk_j, essk_i)
  m ← (ct_ℓ, ct_k, ct_s)
  sid ← P_i||P_j||pk_i||pk_j||prek_i||prek_j||m
  k ← KDF(K_ℓ, K_k, K_s, sid)
  return (k, m)
```

```
BatchReceive(sk_i, st_i, S = {pk_j, prek_j, m_j}_j)
  (essk_i, eksk_i, prek_i) ← st_i
  fail ← false; k_j ← ⊥
  for j : (pk_j, prek_j, m_j) ∈ S :
    (ct_ℓ, ct_k, ct_s) ← m_j
    (espk_j, ekpk_j, σ_j) ← prek_j
    if ¬Vrfy_Sig(pk_j.spk, (espk_j, ekpk_j), σ_j) :
      k_j ← ⊥
      continue
    K_ℓ ← Decaps_LKEM(sk_i.ksk, ct_ℓ)
    K_k ← Decaps_EKEM(eksk_i, ct_k)
    K_s ← Decaps_sKEM(espk_j, essk_i, ct_s)
    sid ← P_j||P_i||pk_j||pk_i||prek_j||prek_i||m_j
    if K_s = ⊥ : fail ← true
    if (K_ℓ = ⊥) ∨ (K_k = ⊥) ∨ (K_s = ⊥) :
      k_j ← ⊥
    else : k_j ← KDF(K_ℓ, K_k, K_s, sid)
  if fail : return ⊥^|S|
  else : return {k_j}_j
```

Figure 7.7: K-Waay: X3DH-like DAKE from IND-CCA KEMs EKEM and LKEM, SUF-CMA signature scheme Sig and IND-1BatchCCA and UNF-1KCA split-KEM sKEM.

$\text{PRF}_F(\mathscr{A})$	$\mathcal{O}_{\mathsf{prf}}(a,b,c)$
1 : Sample random function G	1 : **if** $b = 0$:
2 : $k \leftarrow\$ \mathcal{K}$	2 : **return** $F_k(a,b,c)$
3 : $b \leftarrow\$ \{0,1\}$	3 : **else** :
4 : $b' \leftarrow\$ \mathscr{A}^{\mathcal{O}_{\mathsf{prf}}}(1^\lambda)$	4 : **return** $G(a,b,c)$
5 : **return** $1_{b'=b}$	

Figure 7.8: PRF game for function F_k taking three arguments as input.

After initialisation, the sender P_i (sometimes called the initiator) invokes Send that takes the prekey prek_j output by the receiver P_j's Init call as input. After verifying the signature in prek_j, P_i encapsulates to 1) the long-term KEM key of P_j; 2) the ephemeral KEM key contained in prek_j; and 3) the ephemeral split-KEM key contained in prek_j. Note that the split-KEM provides implicit authentication (without it, Send could be simulated without secrets). P_i then combines the encapsulated keys using a KDF and outputs the key and its message for P_j consisting of the three encapsulation ciphertexts. Receiving is analogous: receiver P_i verifies P_j's prekey, decapsulates using its three respective secret keys, and derives the session key. If P_i's prekeys have run out, it is possible that multiple P_j's have sent using the same prekey prek_i. In that case, P_i decapsulates for all sessions using the same secret keys but aborts if *any* split-KEM decapsulations failed in *any* of the sessions (a signature check failing does not however lead to the receiver aborting). We assume that for a given $\mathsf{BatchReceive}(\mathsf{sk}_i, \mathsf{st}_i, S)$ call, each element of S corresponds to a different party.

7.7.1 Security

Before proving the security of K-Waay, we introduce the notion of a triple PRF (3PRF), which generalises the by now common notion of a *dual PRF* [Bel06b]. Looking ahead, we will assume in the security proof that the key-derivation function (KDF) used in our protocol fulfils this property. A triple PRF F_{triple} can be trivially constructed in the random oracle model.

Definition 7.7.1 (Triple PRF). *Let $F : \mathcal{K} \times \mathcal{K} \times \mathcal{K} \times D \to R$ be a function. We consider the game shown in Figure 7.8. We say that F is a* 3PRF *if for all efficient adversaries, we have*

$$\mathsf{Adv}_F^{3\mathrm{prf}}(\mathscr{A}) := \max_{i \in \{1,2,3\}} \left| \Pr\left[\mathrm{PRF}_{F_i}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2} \right| = \mathsf{negl}\ ,$$

where F_i denotes F keyed in its i-th argument for $i \in \{1,2,3\}$.

Theorem 7.7.1. *Consider a δ_{EKEM}-correct* IND-CCA KEM EKEM*, a δ_{LKEM}-correct* IND-CCA KEM LKEM*, a δ_{Sig}-correct* SUF-CMA *signature scheme* Sig*, and a δ_{sKEM}-correct* IND-1BatchCCA, UNF-1KCA split-KEM sKEM *and* 3PRF KDF *used to build K-Waay (Figure 7.7). Then, we have that for polynomially-bounded n and every efficient adversary $\mathscr{A}$ that makes at* *most q oracle queries, one can build an adversary $\mathscr{B}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\text{kind}}_{\mathsf{K\text{-}Waay},n}(\mathscr{A}) \leq \frac{q}{3}\cdot\left(\delta_{\mathsf{Sig}}+\delta_{\mathsf{LKEM}}+\delta_{\mathsf{EKEM}}+\delta_{\mathsf{sKEM}}\right)+\\ 2q^2\cdot(\epsilon_{\mathsf{EKEM}}+\epsilon_{\mathsf{LKEM}}+2\epsilon_{\mathsf{KDF}}+2\epsilon_{\mathsf{Sig}})+\\ q^3\cdot\left(\epsilon_{\mathsf{EKEM}}+\epsilon_{\mathsf{LKEM}}+\epsilon_{\mathsf{sKEM}}+3\epsilon_{\mathsf{KDF}}\right),\end{aligned}$$

where $\epsilon_{\mathsf{EKEM}} = \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{EKEM}}(\mathscr{B})$, $\epsilon_{\mathsf{LKEM}} = \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{LKEM}}(\mathscr{B})$, $\epsilon_{\mathsf{Sig}} = \mathsf{Adv}^{\text{suf-cma}}_{\mathsf{Sig}}(\mathscr{B})$, $\epsilon_{\mathsf{sKEM}} = \mathsf{Adv}^{\text{ind-1batchcca}}_{\mathsf{sKEM}}(\mathscr{B}) + \mathsf{Adv}^{\text{unf-1kca}}_{\mathsf{sKEM}}(\mathscr{B})$, *and* $\epsilon_{\mathsf{KDF}} = \mathsf{Adv}^{\text{3prf}}_{\mathsf{KDF}}(\mathscr{B})$.

Proof. Our proof proceeds by constructing sequences of hybrids, which we first summarise. Let Game_1 be exactly the KIND game played with respect to DAKE K-Waay (Figure 7.7). We first transition to Game_2, which differs from Game_1 in that honest protocol runs, all $\mathsf{Vrfy}_{\mathsf{Sig}}$ checks in BatchReceive calls are removed and Decaps calls are replaced by the output of the Encaps calls in the corresponding Send calls whenever they are consistent. To this end, we invoke the correctness of K-Waay's building blocks. Then, we transition to Game_3 in which the challenger immediately outputs the session π_i^s that the adversary makes real-or-random challenge query TEST(i, s, j^*) with respect to. We then partition $\mathscr{A}$'s possible executions of Game_3 into several events.

Suppose π_i^s has a partner session (with respect to key j^* if π_i^s.role = receiver) (event E_p), say π_j^t. Observe that by definition of partnering and construction of the protocol (in particular by definition of sid), it follows that partnered sessions correspond to *honest* protocol runs. Then, considering π_i^s and π_j^t, if the receiver's session state, say π_j^t.st, is revealed (event $E_p \wedge E_{c1}$), we reduce to the IND-CCA security of the long-term KEM LKEM, since the freshness conditions imply P_j must not have been corrupted. Otherwise (event $E_p \wedge \neg E_{c1}$), we reduce to the IND-CCA security of the ephemeral KEM EKEM. After both cases, we transition to an unwinnable game by keying KDF with the now uniformly random key output by the respective KEM call, a transition we perform repeatedly and omit from this description hereafter. Otherwise (event $\neg E_p$), we consider whether party P_i in test session π_i^s has the role sender or receiver:

- π_i^s.role = sender (event $\neg E_p \wedge E_s$): As P_j can only be corrupted after P_i accepts, we first use the SUF-CMA security of Sig to argue that P_i's Send call in the test session must be with honestly-generated input (prek). Then, let E_{c2} be the event that P_j is corrupted. Given $\neg E_p \wedge E_s \wedge E_{c2}$, we reduce to the security of EKEM, since by freshness the state π_j^t.st associated with prek must not have been exposed. Otherwise ($\neg E_p \wedge E_s \wedge \neg E_{c2}$) we reduce to the security of LKEM.
- π_i^s.role = receiver (event $\neg E_p \wedge \neg E_s$): As above, we first argue using SUF-CMA security that input prek_j used in the test session's BatchReceive call must have been honestly generated. Then by freshness, we know that neither π_i^s.st nor π_j^t.st associated with prek_j are revealed, in which case we first reduce to the UNF-1KCA security of sKEM to prevent injections on the split-KEM ciphertext, after which we reduce to the IND-1BatchCCA

security of sKEM.

Let $\mathsf{Adv}^{\mathrm{gi}}_{\mathsf{DAKE},n}(\mathscr{A})$ be the advantage of adversary $\mathscr{A}$ in winning game Game_i for relevant i which we introduce below. Furthermore, let $\mathsf{Adv}^{\mathrm{gi}}_{\mathsf{DAKE},n}(\mathscr{A}, E)$ be the same advantage except restricted to event E, so in particular if $\mathsf{Adv}^{\mathrm{gi}}_{\mathsf{DAKE},n}(\mathscr{A})$ is of the form $\Pr[X] - \frac{1}{2}$, $\mathsf{Adv}^{\mathrm{gi}}_{\mathsf{DAKE},n}(\mathscr{A}, E)$ is of the form $\Pr[X \wedge E] - \frac{1}{2}$.

$\underline{\mathsf{Game}_1}$: This is the original key indistinguishability game.

$\underline{\mathsf{Game}_2}$: This differs from Game_1 in that, in honest protocol runs, all signature verification calls in BatchReceive calls are removed and the output of Decaps calls are replaced with the output of the corresponding Encaps call in Send. It follows at this point that the three correctness checks in the KIND game evaluate to true. Since for a given $\mathsf{BatchReceive}(\cdot,\cdot,S)$ call there must be $|S|$ corresponding Send and Init calls, there are at most $q/3$ iterations of the **for** loop in BatchReceive (counting over all such calls in a given execution of Game_1). It then follows from a standard hybrid argument and the correctness of Sig, LKEM, EKEM and sKEM that

$$\mathsf{Adv}^{\mathrm{g1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\mathrm{g2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \frac{q}{3} \cdot \left(\delta_{\mathsf{Sig}} + \delta_{\mathsf{LKEM}} + \delta_{\mathsf{EKEM}} + \delta_{\mathsf{sKEM}}\right) .$$

$\underline{\mathsf{Game}_3}$: This differs from Game_3 in that the challenger immediately outputs the session π_i^s that the adversary $\mathscr{A}$ calls $\mathrm{TEST}(i, s, j^*)$ with respect to. Noting that there are at most q such possible sessions and applying a standard argument, it follows that

$$\mathsf{Adv}^{\mathrm{g2}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq q \cdot \mathsf{Adv}^{\mathrm{g3}}_{\mathsf{DAKE},n}(\mathscr{A}) .$$

Case 1: Test session π_i^s is partnered (Game_{3a} and Game_{3b}):

$\underline{\mathsf{Game}_{3a.1}}$: Let E_p be the event that test session π_i^s has a partner, say π_j^t. Let E_{c1} be the event that the ephemeral state st of the receiver (in π_i^s and π_j^t) is revealed. Games $\mathsf{Game}_{3a.i}$ are defined given $E_p \wedge E_{c1}$. $\mathsf{Game}_{3a.1}$ differs from Game_3 in that the game initially outputs π_j^t, the partner of π_i^s (observe that $j = j^*$ where j^* is defined in the previous hop), as well as a bit indicting whether π_i^s is the sender or receiver. By the same reasoning as above, we have

$$\mathsf{Adv}^{\mathrm{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, E_p \wedge E_{c1}) \leq 2q \cdot \mathsf{Adv}^{\mathrm{g3a.1}}_{\mathsf{DAKE},n}(\mathscr{A}) .$$

$\underline{\mathsf{Game}_{3a.2}}$: $\mathsf{Game}_{3a.2}$ differs from $\mathsf{Game}_{3a.1}$ in that the output key K in the call to LKEM.Encaps and the corresponding LKEM.Decaps call or calls (which are guaranteed to exist given E_p, and K is identical by definition of Game_2) made in the test and partner sessions with respect to the receiver's public key and secret key, respectively, are replaced with a key k uniformly sampled by the challenger. Observe that since E_{c1} holds, by freshness, P_j cannot be corrupted, and thus we reduce to the security of LKEM.

Let $\mathscr{A}'$ be a IND-CCA adversary who simulates for $\mathsf{Game}_{3a.1}/\mathsf{Game}_{3a.2}$ adversary $\mathscr{A}$ as follows. Let pk be the IND-CCA challenge public key, (ct^*, K^*) be the challenge ciphertext and key respectively.

In the Setup phase, $\mathscr{A}'$ uniformly samples bit b_{sim}, calls $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow\!\$\, \mathsf{KeyGen}(1^\lambda)$ locally for $\ell \neq k$ where k is the sender, sets $\mathsf{pk}_k \leftarrow \mathsf{pk}$, and returns $\{\mathsf{pk}_1, \ldots \mathsf{pk}_n\}$ and 1^λ to $\mathscr{A}$. Observe here (and later for $\mathsf{Game}_{3b.2}$) that, since E_p holds, we have matching sid values for test session π_i^s and partner π_j^t. Note by construction of sid, the presence of substring $\mathsf{prek}_i||\mathsf{prek}_j$ and m in the common value sid implies that Send must have been called honestly in π_i^s and also in $\mathsf{BatchReceive}$ for tuple $(\mathsf{pk}_j, \mathsf{prek}_j, m_j)$ in π_j^t for E_p to hold. Thus, we do not need to consider injections in the test session itself (although we have to in general in the $\mathsf{BatchReceive}$ call).

Before proceeding, we argue that $\mathscr{A}'$ can simulate on behalf of parties with a maliciously-registered long-term key locally, which applies here and in the rest of the proof. Since π_i^s is partnered, as argued above, π_i^s and π_j^t correspond to honest (completed) executions, and so neither P_i and P_j can be malicious. For unpartnered sessions, since Send and $\mathsf{BatchReceive}$ cannot be called by the game, the test session π_i^s cannot be corrupted itself (since testing requires $\pi_i^s.\mathsf{status} \neq \perp$), and otherwise condition 5 restricts the non-tested party P_j from being corrupted, thus precluding its key from being registered maliciously. Finally, computation involving messages or prekey bundles from maliciously-registered parties does not require any secret material not already known to $\mathscr{A}'$.

In Phase 1, when $\mathscr{A}$ calls $\mathrm{EXEC}(k', \cdot, \cdot, \cdot)$ where k' corresponds to the sender in π_i^s and π_j^t and the challenger is supposed to invoke Send, $\mathscr{A}'$ replaces the call to $\mathsf{Encaps}_{\mathsf{LKEM}}$ with the output (ct^*, K^*), and otherwise simulates locally. When $\mathscr{A}$ calls $\mathrm{EXEC}(k, \cdot, \cdot, \cdot)$ corresponding to the receiver in π_i^s and π_j^t and the challenger is supposed to invoke $\mathsf{BatchReceive}$, $\mathscr{A}'$ replaces the output of the relevant $\mathsf{Decaps}_{\mathsf{LKEM}}$ calls corresponding to either i or j, depending on who is the receiver, with K^* and the output of other calls $\mathsf{Decaps}_{\mathsf{LKEM}}$ with the output obtained from $\mathrm{DEC}(\cdot)$; $\mathscr{A}'$ otherwise simulates locally.

In the Test phase, i.e. when $\mathscr{A}$ calls $\mathrm{TEST}(i, s, j)$, $\mathscr{A}'$ simulates with respect to bit b_{sim}. $\mathscr{A}'$ then simulates Phase 2 as above and the rest of the game locally, ultimately outputting the same bit as $\mathscr{A}$; observe that $\mathscr{A}'$ can efficiently evaluate the freshness conditions. Since $\mathscr{A}'$ perfectly simulates $\mathsf{Game}_{3a.1}$ when playing with respect to challenge bit 0 and $\mathsf{Game}_{3a.2}$ when it is 1, it follows that

$$\mathsf{Adv}^{\mathrm{g3a.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\mathrm{g3a.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{LKEM}}(\mathscr{A}') \ .$$

$\underline{\mathsf{Game}_{3a.3}}$: This differs from $\mathsf{Game}_{3a.2}$ in that, for the test and partner sessions, the call to KDF made in Send and the corresponding calls made in $\mathsf{BatchReceive}$ with respect to ciphertext ct_ℓ output by Send are replaced with uniformly sampled keys. Let $\mathscr{A}'$ be a PRF adversary playing with respect to KDF keyed in its first argument simulating for $\mathsf{Game}_{3a.2}/\mathsf{Game}_{3a.3}$ adversary $\mathscr{A}$ as follows. $\mathscr{A}'$ simulates locally all calls except the Send and $\mathsf{BatchReceive}$ calls made in π_i^s and π_j^t, where it replaces the relevant calls $\mathsf{KDF}(K_\ell, K_k, K_s, \mathsf{sid})$ with the call $\mathscr{O}_{\mathsf{prf}}(K_k, K_s, \mathsf{sid})$. Since

K_ℓ is uniform (by definition of $\mathsf{Game}_{3a.2}$) and, by definition of freshness, K_ℓ is not revealed to $\mathscr{A}$, the simulation is perfect and we have

$$\mathsf{Adv}^{\mathrm{g3a.2}}_{\mathsf{DAKE},n}(\mathscr{A}) \le \mathsf{Adv}^{\mathrm{g3a.3}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{3prf}}_{\mathsf{KDF}}(\mathscr{A}') .$$

Finally, we have $\mathsf{Adv}^{\mathrm{g3a.3}}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$ since the output of TEST is identical regardless of the challenge bit and it is not otherwise used by the challenger or leaked to the adversary.

$\underline{\mathsf{Game}_{3b.1}}$: We now consider the case when $E_p \wedge \neg E_{c1}$, i.e. the case where the receiver's session state st in π_i^s and π_j^t is not revealed. $\mathsf{Game}_{3b.1}$ differs from Game_3 in that the game initially outputs π_j^t, the partner of π_i^s, as well as a bit indicating whether π_i^s is the sender or receiver. Since $\mathsf{Game}_{3b.1}$ is exactly $\mathsf{Game}_{3a.1}$, we have

$$\mathsf{Adv}^{\mathrm{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, E_p \wedge \neg E_{c1}) \le q \cdot \mathsf{Adv}^{\mathrm{g3b.1}}_{\mathsf{DAKE},n}(\mathscr{A}) .$$

$\underline{\mathsf{Game}_{3b.2}}$: In $\mathsf{Game}_{3b.2}$, the output of $\mathsf{Encaps}_{\mathsf{EKEM}}$ and the corresponding $\mathsf{Decaps}_{\mathsf{EKEM}}$ call or calls in the test session are replaced with a uniformly random key k. IND-CCA adversary $\mathscr{A}'$ simulates for $\mathsf{Game}_{3b.1}/\mathsf{Game}_{3b.2}$ adversary $\mathscr{A}$ as follows. $\mathscr{A}'$ follows the same broad approach as the adversary defined in the hop between $\mathsf{Game}_{3a.1}$ and $\mathsf{Game}_{3a.2}$. In particular, $\mathscr{A}'$ simulates the receiver in their session's call to Init except it uses the IND-CCA challenge public key pk, replaces the output of EKEM in the test session Encaps and the corresponding Decaps calls with the challenge ciphertext and key, and replaces other Decaps calls with calls to oracle DEC. By the same reasoning as before, it follows that

$$\mathsf{Adv}^{\mathrm{g3b.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \le \mathsf{Adv}^{\mathrm{g3b.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{EKEM}}(\mathscr{A}') .$$

$\underline{\mathsf{Game}_{3b.3}}$: This replaces the relevant outputs of KDF in the test session with a uniformly random key. As in $\mathsf{Game}_{3a.3}$, this game is now unwinnable, i.e. $\mathsf{Adv}^{\mathrm{g3b.3}}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$. As before, we reduce to the security of KDF, except now we key KDF in the PRF game with the second argument K_k. We then arrive at

$$\mathsf{Adv}^{\mathrm{g3b.2}}_{\mathsf{DAKE},n}(\mathscr{A}) \le \mathsf{Adv}^{\mathrm{g3b.3}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{3prf}}_{\mathsf{KDF}}(\mathscr{A}') .$$

Case 2: Test session π_i^s is unpartnered and $\pi_i^s.\mathsf{role} = \mathsf{sender}$ (Game_{3c}):

$\underline{\mathsf{Game}_{3c.1}}$: Let $\pi_i^s.\mathsf{pid} = j$ and E_s be the event that $\pi_i^s.\mathsf{role} = \mathsf{sender}$. $\mathsf{Game}_{3c.1}$ differs from Game_3 in that the challenger immediately outputs j. By a standard argument, we have

$$\mathsf{Adv}^{\mathrm{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, \neg E_p \wedge E_s) \le q \cdot \mathsf{Adv}^{\mathrm{g3c.1}}_{\mathsf{DAKE},n}(\mathscr{A}) .$$

$\underline{\mathsf{Game}_{3c.2}}$: This differs from $\mathsf{Game}_{3c.1}$ in that the challenger aborts if the call $\mathsf{Send}(\mathsf{sk}_i, \mathsf{pk}_j, \mathsf{st}_i, \mathsf{prek}_j)$ in the test session is such that prek_j was not previously output by a call to $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$. Note that by freshness condition 5 that P_j must not be corrupted until after $\pi_i^s.\mathsf{status}$ is changed from $\perp$, which, by definition of E_s, means until after it is set to accept. In order for Send to accept on input $\mathsf{prek}_j = (\mathsf{espk}_j, \mathsf{ekpk}_j, \sigma_j)$ not previously output by $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$ (and thus for the game to abort), $\mathscr{A}$ needs to find a different prek_j such that $\mathsf{VrfySig}(\mathsf{pk}_j.\mathsf{spk}, (\mathsf{espk}_j, \mathsf{ekpk}_j), \sigma_j)$ (by construction of Send). Using this observation, we reduce to the SUF-CMA security of Sig.

Let $\mathscr{A}'$ be a SUF-CMA adversary simulating for $\mathsf{Game}_{3c.1}$ adversary $\mathscr{A}$. Let pk be the SUF-CMA challenge public key. In the Setup phase, $\mathscr{A}'$ sets $\mathsf{pk}_j = \mathsf{pk}$ and otherwise simulates locally. In particular, unlike in previous hops, $\mathscr{A}'$ also samples the random $\mathsf{Game}_{3c.1}$ bit. In each subsequent phase, for each call $\mathrm{EXEC}(j, u, \cdot, m)$ such that $m = (\mathtt{start}, \mathsf{role}, \cdot)$, $\mathscr{A}'$ replaces the $\mathsf{Sign}_{\mathsf{Sig}}(\mathsf{sk}_j.\mathsf{ssk}, (\mathsf{espk}_j, \mathsf{ekpk}_j))$ call in $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$ by a call to $\mathrm{SIGN}((\mathsf{espk}_j, \mathsf{ekpk}_j))$, and otherwise simulates the call locally. When the challenger calls $\mathsf{Send}(\cdot, \mathsf{pk}_j, \cdot, \mathsf{prek}_j)$ where $\mathsf{prek}_j = (\mathsf{espk}_j, \mathsf{ekpk}_j, \sigma_j)$, $\mathscr{A}'$ checks whether 1) $(\mathsf{espk}_j, \mathsf{ekpk}_j)$ was previously queried to SIGN which output σ_j and 2) $\mathsf{Vrfy}_{\mathsf{Sig}}(\mathsf{pk}, (\mathsf{espk}_j, \mathsf{ekpk}_j), \sigma_j) = 1$. Given 1) and 2) both hold, $\mathscr{A}'$ returns $(m, \sigma) = ((\mathsf{espk}_j, \mathsf{ekpk}_j), \sigma_j)$ to its challenger. $\mathscr{A}'$ otherwise simulates locally, aborting if $\mathscr{A}$ outputs a bit. The simulation is perfect and it follows that

$$\mathsf{Adv}^{\mathrm{g3c.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\mathrm{g3c.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{suf-cma}}_{\mathsf{Sig}}(\mathscr{A}') \ .$$

$\underline{\mathsf{Game}_{3c.3}}$: In $\mathsf{Game}_{3c.3}$, the challenger initially outputs π_j^t, where π_j^t is the session that prek is output by $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$ and input to the Send call in test session π_i^s.

By a standard failure event argument, we have

$$\mathsf{Adv}^{\mathrm{g3c.2}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq q \cdot \mathsf{Adv}^{\mathrm{g3c.3}}_{\mathsf{DAKE},n}(\mathscr{A}) \ .$$

$\underline{\mathsf{Game}_{3c.4a.1}}$: Let E_{c2} be the event that P_j is corrupted. We construct hybrid sequence $\mathsf{Game}_{3c.4a}$ (resp. $\mathsf{Game}_{3c.4b}$) to deal with the case that E_{c2} holds (resp. does not hold). $\mathsf{Game}_{3c.4a.1}$ differs from $\mathsf{Game}_{3c.3}$ in that the output of $\mathsf{Encaps}_{\mathsf{EKEM}}$ in the Send call in test session π_i^s and of the (possible) corresponding $\mathsf{Decaps}_{\mathsf{EKEM}}$ calls in π_j^t are replaced with uniformly random output. By freshness, P_j's session state $\pi_j^t.\mathsf{st}$ associated with prek input to the test Send call is not revealed.

IND-CCA adversary $\mathscr{A}'$ simulates for $\mathsf{Game}_{3c.4a.1}$ adversary $\mathscr{A}$ as follows. Let $(\mathsf{pk}, k, \mathsf{ct})$ the challenge public key, key and corresponding ciphertext (respectively) of $\mathscr{A}'$. $\mathscr{A}'$ embeds pk in session π_j^t by replacing the public key output by $\mathsf{KeyGen}_{\mathsf{EKEM}}$ in $\mathsf{Init}(\mathsf{sk}_j, \mathsf{receiver})$ with pk, which outputs prek_j. Upon prek_j being input to Send in the test session, $\mathscr{A}'$ replaces the output of $\mathsf{Encaps}_{\mathsf{EKEM}}$ with (k, ct). When the challenger calls $\mathsf{BatchReceive}(\mathsf{sk}_j, \cdot, \{\cdot, \cdot, m_{j'} = (\cdot, \mathsf{ct}', \cdot)\}_{j'})$ in session π_j^t, if $\mathsf{ct}' = \mathsf{ct}$, $\mathscr{A}'$ replaces the output of $\mathsf{Decaps}_{\mathsf{EKEM}}$ with k; else, $\mathscr{A}'$ replaces the call $\mathsf{Decaps}_{\mathsf{EKEM}}(\cdot, \mathsf{ct}')$ with the call $\mathrm{DEC}(\mathsf{ct}')$. $\mathscr{A}'$ otherwise simulates locally and outputs the same bit as $\mathscr{A}$. By similar reasons to before, we have

$$\mathsf{Adv}^{\mathrm{g3c.3}}_{\mathsf{DAKE},n}(\mathscr{A}, E_{c2}) \leq \mathsf{Adv}^{\mathrm{g3c.4a.1}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\mathrm{ind-cca}}_{\mathsf{EKEM}}(\mathscr{A}') \ .$$

Game$_{3c.4a.2}$: This replaces the output of KDF in the Send and BatchReceive calls as before in the test session and π_j^t with uniformly random keys. By the exact same argument as for Game$_{3b.3}$, we have $\mathsf{Adv}^{\text{g3c.4a.2}}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$ and

$$\mathsf{Adv}^{\text{g3c.4a.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3c.4a.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{3prf}}_{\mathsf{KDF}}(\mathscr{A}') \; .$$

Game$_{3c.4b.1}$: We assume $\neg E_{c2}$, i.e. that P_j is not corrupted. We reduce to the IND-CCA security of LKEM. The reduction follows the same high-level strategy as previous hops (embedding the challenge pk in pk_j and the challenge in the test Send call and possibly the corresponding BatchReceive call), noting that non-challenge $\mathsf{Decaps}_{\mathsf{LKEM}}(\mathsf{sk}_j, \cdot)$ queries are replaced with calls to DEC. We then have

$$\mathsf{Adv}^{\text{g3c.3}}_{\mathsf{DAKE},n}(\mathscr{A}, \neg E_{c2}) \leq \mathsf{Adv}^{\text{g3c.4b.1}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{ind-cca}}_{\mathsf{LKEM}}(\mathscr{A}') \; .$$

Game$_{3c.4b.2}$: As in Game$_{3c.4a.2}$, this replaces the output of KDF in the Send and BatchReceive calls in π_i^s and π_j^t with a uniformly random key. As argued several times above, it follows that $\mathsf{Adv}^{\text{g3c.4b.2}}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$ and

$$\mathsf{Adv}^{\text{g3c.4b.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3c.4b.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{3prf}}_{\mathsf{KDF}}(\mathscr{A}') \; .$$

Case 3: Test session π_i^s is unpartnered and $\pi_i^s.\mathsf{role} = \mathsf{receiver}$ (Game$_{3d}$):

Game$_{3d.1}$: Game$_{3d.1}$ differs from Game$_3$ in that the challenger immediately outputs j, the third argument in $\mathscr{A}$'s TEST(i, s, j) call. As for Game$_{3c.1}$, we have

$$\mathsf{Adv}^{\text{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, \neg E_s) \leq q \cdot \mathsf{Adv}^{\text{g3d.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \; .$$

Game$_{3d.2}$: This differs from Game$_{3d.1}$ in that the challenger aborts if the call $\mathsf{BatchReceive}(\mathsf{sk}_i, \mathsf{st}_i, \{\mathsf{pk}_{j'}, \mathsf{prek}_{j'}, m\}_{j'})$ in the test session is such that prek_j was not previously output by a call to $\mathsf{Init}(\mathsf{sk}_j, \mathsf{sender})$. As in Game$_{3c.2}$, P_j must not be corrupted until after $\pi_i^s.\mathsf{status}$ is set to accept. By reducing to SUF-CMA security essentially as in Game$_{3c.2}$, it follows that

$$\mathsf{Adv}^{\text{g3d.1}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3d.2}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{suf-cma}}_{\mathsf{Sig}}(\mathscr{A}) \; .$$

Game$_{3d.3}$: This differs from Game$_{3d.2}$ in that the challenger initially outputs π_j^t, the session that generated prek_j which formed part of the input to BatchReceive in the test session π_i^s. By a standard argument we have

$$\mathsf{Adv}^{\text{g3d.2}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq q \cdot \mathsf{Adv}^{\text{g3d.3}}_{\mathsf{DAKE},n}(\mathscr{A}) \; .$$

Game$_{3d.4}$: This differs from Game$_{3d.3}$ in that the challenger aborts if the

$\mathsf{Send}(\mathsf{sk}_j, \mathsf{pk}_i, \mathsf{st}_j, \mathsf{prek}_i)$ call in session π_j^t (if it exists) and the relevant component in the $\mathsf{BatchReceive}$ call in the test session π_i^s were not both with respect to honestly generated split-KEM keying material (namely, an honestly generated split-KEM public key from prek_i and prek_j from the previous hop) and the same split-KEM ciphertext. By freshness, neither of the two ephemeral states $\pi_i^s.\mathsf{st}$ and $\pi_j^t.\mathsf{st}$ are revealed. Consequently, we reduce to the UNF-1KCA security of split-KEM sKEM.

UNF-1KCA adversary $\mathscr{A}'$ simulates for $\mathsf{Game}_{3d.3}/\mathsf{Game}_{3d.4}$ adversary $\mathscr{A}$ as follows. Let $(\mathsf{pk_A}, \mathsf{pk_B})$ be the two challenge public keys given to $\mathscr{A}'$. In the $\mathsf{Init}(\mathsf{sk}_i, \mathsf{receiver})$ call in session π_i^s, $\mathscr{A}'$ simulates except replaces the call to $\mathsf{KeyGenA}_{\mathsf{sKEM}}$ by $\mathsf{pk_A}$. Similarly, in the $\mathsf{Init}(\mathsf{sk}_j, \mathsf{sender})$ call in session π_j^t, $\mathscr{A}'$ replaces $\mathsf{KeyGenB}_{\mathsf{sKEM}}$ by $\mathsf{pk_B}$. In the $\mathsf{Send}(..., \mathsf{prek})$ call in session π_j^t where $\mathsf{prek} = (\mathsf{espk}, ...)$, $\mathscr{A}'$ outputs espk to its UNF-1KCA challenger, receives $(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}, K_B)$ from its challenger, and replaces the call to $\mathsf{Encaps}_{\mathsf{sKEM}}$ with tuple (ct, K_B). Finally, when the $\mathsf{BatchReceive}(\mathsf{sk}_i, \cdot, \{\cdot, \mathsf{prek}_{j'}, m = (\cdot, \cdot, \mathsf{ct}_s)\}_{j'})$ call in test session π_i^s is made, $\mathscr{A}'$ outputs ct_s corresponding to $j' = j$ to its challenger. As the simulation is perfect and the probability that $\mathscr{A}'$ wins is exactly the probability that 1) $(\mathsf{ct}, \mathsf{pk_A}) \neq (\mathsf{ct}_s, \mathsf{pk})$ and 2) relevant $\mathsf{Decaps}_{\mathsf{sKEM}}$ call in $\mathsf{BatchReceive}$ outputs $k \neq \perp$, it follows by a standard failure event argument that

$$\mathsf{Adv}^{\text{g3d.3}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3d.4}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{unf-1kca}}_{\mathsf{sKEM}}(\mathscr{A}') .$$

$\underline{\mathsf{Game}_{3d.5}}$: This differs from $\mathsf{Game}_{3d.4}$ in that the output k of the relevant test session split-KEM decapsulation and the corresponding encapsulation (if it exists) are both replaced by a uniformly random key. Note that by definition of $\mathsf{Game}_{3d.4}$, $\mathscr{A}$ can only input an honestly generated split-KEM ciphertext to the $\mathsf{BatchReceive}$ call in the test session from P_j and that the split-KEM public key in P_j's corresponding Send call (if it exists) must be honestly generated. We therefore reduce to the IND-1BatchCCA security of sKEM. We embed the IND-1BatchCCA keys $\mathsf{pk_A}$, $\mathsf{pk_B}$ in the simulation as in the previous hop. When $\mathscr{A}$ queries $\mathrm{EXEC}(i, s, S = \{s_{j'}, \mathsf{prek}_{j'}, m_{j'}\}_{j'})$, $\mathscr{A}'$ replaces all $\mathsf{Decaps}_{\mathsf{sKEM}}$ calls involving $\mathsf{sk_A}$ *except* the call corresponding to the test session by the output of its query to oracle BatchDec, replaces this final $\mathsf{Decaps}_{\mathsf{sKEM}}$ call with the IND-1BatchCCA challenge key and otherwise simulates locally. It follows that

$$\mathsf{Adv}^{\text{g3d.4}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3d.5}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{ind-1batchcca}}_{\mathsf{sKEM}}(\mathscr{A}') .$$

$\underline{\mathsf{Game}_{3d.6}}$: This game replaces the relevant invocation of KDF in the test session's $\mathsf{BatchReceive}$ call by a uniformly random value. Note as usual that $\mathsf{Adv}^{\text{g3d.6}}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$. By keying KDF in its third argument as a PRF and a standard argument it follows that

$$\mathsf{Adv}^{\text{g3d.5}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3d.6}}_{\mathsf{DAKE},n}(\mathscr{A}) + \mathsf{Adv}^{\text{3prf}}_{\mathsf{KDF}}(\mathscr{A}') .$$

Finally note that by the triangle inequality, we have, among other inequalities:

$$\mathsf{Adv}^{\text{g3}}_{\mathsf{DAKE},n}(\mathscr{A}) \leq \mathsf{Adv}^{\text{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, E_p) + \mathsf{Adv}^{\text{g3}}_{\mathsf{DAKE},n}(\mathscr{A}, \neg E_p) .$$

$\mathsf{Sim}(\mathsf{sk}_j, \mathsf{pk}_i, \mathsf{st}_j, \mathsf{prek}_i, \mathsf{prek}_j)$

$(\mathsf{essk}_j, \mathsf{eksk}_j, \mathsf{prek}_j) \leftarrow \mathsf{st}_j$

$(\mathsf{espk}_i, \mathsf{ekpk}_i, \sigma_i) \leftarrow \mathsf{prek}_i$

$(\mathsf{espk}_j, \mathsf{ekpk}_j, \sigma_j) \leftarrow \mathsf{prek}_j$

$\mathsf{spk} \leftarrow \mathsf{GetPK}(\mathsf{sk}_j.\mathsf{ssk})$

$\textbf{require } \mathsf{Vrfy}_{\mathsf{Sig}}(\mathsf{spk}, (\mathsf{espk}_j, \mathsf{ekpk}_j), \sigma_j) = 1$

$(K_\ell, \mathsf{ct}_\ell) \leftarrow\!\!\$\ \mathsf{Encaps}_{\mathsf{LKEM}}(\mathsf{pk}_j.\mathsf{kpk})$

$(K_k, \mathsf{ct}_k) \leftarrow\!\!\$\ \mathsf{Encaps}_{\mathsf{EKEM}}(\mathsf{ekpk}_j)$

$(K_s, \mathsf{ct}_s) \leftarrow\!\!\$\ \mathsf{Sim}_{\mathsf{sKEM}}(\mathsf{espk}_i, \mathsf{essk}_j)$

$m \leftarrow (\mathsf{ct}_\ell, \mathsf{ct}_k, \mathsf{ct}_s)$

$\mathsf{sid} \leftarrow P_i || P_j || \mathsf{pk}_i || \mathsf{pk}_j || \mathsf{prek}_i || \mathsf{prek}_j || m$

$\mathsf{k} \leftarrow \mathsf{KDF}(K_\ell, K_k, K_s, \mathsf{sid})$

$\textbf{return } (\mathsf{k}, m)$

Figure 7.9: Simulator Sim for the deniability game where we assume we have a function GetPK(sk) that takes a signature secret key as input and outputs the corresponding public key.

The result follows using this observation and by combining the sequences of hybrids together in a standard way. □

Theorem 7.7.2. *Consider deniable split-KEM* sKEM *with simulator* $\mathsf{Sim}_{\mathsf{sKEM}}$ *used to build K-Waay (Figure 7.7). Then, we have that for every efficient adversary $\mathscr{A}$ that makes at most q oracle queries, there exists an efficient* Sim *s.t. one can build an adversary $\mathscr{B}$ such that for* $\mathsf{exp} \in \{\mathsf{true}, \mathsf{false}\}$ *we have*

$$\mathsf{Adv}^{\mathrm{deny}}_{\mathsf{K\text{-}Waay},\mathsf{Sim},\mathsf{exp}}(\mathscr{A}) \leq q \cdot \mathsf{Adv}^{\mathrm{deny}}_{\mathsf{sKEM},\mathsf{Sim}_{\mathsf{sKEM}}}(\mathscr{B}) \,.$$

Proof. We construct a sequence of hybrids and reduce to the deniability of sKEM (i.e. $\mathrm{DENY}_{\mathsf{sKEM},\mathsf{Sim}_{\mathsf{sKEM}}}$ security) in each step. Before this, we define the simulator Sim that we use in the proof, which uses the simulator $\mathsf{Sim}_{\mathsf{sKEM}}$ as a subroutine.

Observe in K-Waay that, given an honestly generated prek_j, any party with knowledge only of public keying material can simulate all steps in Send except for the $\mathsf{Encaps}_{\mathsf{sKEM}}$ call which requires sender P_i's secret key. Thus, our simulator Sim (Figure 7.9) simulates these steps and since it takes the receiver's key sk_j as input it can also invoke the deniability simulator $\mathsf{Sim}_{\mathsf{sKEM}}$ to complete the call.

Let Γ_0 be the DAKE DENY game instantiated with K-Waay. For $i \in [q]$, let Γ_i be the same as Γ_{i-1} except that in the i-th CHAL call, the call to Send is replaced with a call to Sim. Note that the steps executed in Send only differ in that it calls $\mathsf{Encaps}_{\mathsf{sKEM}}$ rather than $\mathsf{Sim}_{\mathsf{sKEM}}$.

For $i \in [q]$, let $\mathscr{B}$ be a split-KEM DENY adversary with input $(\mathsf{pk}_{\mathsf{A}}, \mathsf{pk}_{\mathsf{B}}, \mathsf{sk}_{\mathsf{B}}, K, \mathsf{ct})$ from its challenger playing $\mathrm{DENY}^{\mathsf{REAL}}$ given $\mathscr{A}$ is playing Γ_{i-1} and $\mathrm{DENY}^{\mathsf{SIM}}$ if it is playing Γ_i. $\mathscr{B}'$

locally simulates long-term public key generation and the first $i-1$ calls to CHAL. When $\mathscr{A}$ makes their i-th call to CHAL, $\mathscr{B}$ simulates CHAL until it reaches the **if** statement except that it replaces the output of calls $\mathsf{KeyGenA}/\mathsf{KeyGenB}$ calls in Init calls with $\mathsf{pk_A}/\mathsf{pk_B}$. Then, instead of executing the **if**/**else** block in CHAL, $\mathscr{B}$ simulates Sim except that it replaces the output of the call to $\mathsf{Sim_{sKEM}}$ with (K, ct). $\mathscr{B}$ then simulates locally, and returns (k, T, st_r) (where st_r contains $\mathsf{sk_B}$) if $exp = \mathsf{true}$ and returns (k, T) otherwise. $\mathscr{B}$ continues simulating locally and finally outputs the same bit as $\mathscr{A}$. Noting that DAKE deniability game $\mathrm{DENY}_{\mathsf{K\text{-}Waay,Sim}}$ considers only honest executions of $\mathsf{K\text{-}Waay}$, it follows that the simulation is perfect, and so by $\mathrm{DENY}_{\mathsf{sKEM}}$ security we have

$$\left|\mathsf{Adv}^{\Gamma_{i-1}}_{\mathsf{DAKE}}(\mathscr{A}) - \mathsf{Adv}^{\Gamma_i}_{\mathsf{DAKE}}(\mathscr{A})\right| \leq \mathsf{Adv}^{\mathrm{deny}}_{\mathsf{sKEM,Sim_{sKEM}}}(\mathscr{B}) \ .$$

By application of the triangle inequality and telescoping sums:

$$\left|\mathsf{Adv}^{\Gamma_0}_{\mathsf{DAKE}}(\mathscr{A}) - \mathsf{Adv}^{\Gamma_q}_{\mathsf{DAKE}}(\mathscr{A})\right| \leq q \cdot \mathsf{Adv}^{\mathrm{deny}}_{\mathsf{sKEM,Sim_{sKEM}}}(\mathscr{B}) \ .$$

To complete the proof, observe that $\mathsf{Adv}^{\Gamma_q}_{\mathsf{DAKE},n}(\mathscr{A}) = 0$ since CHAL behaves identically independent of challenge bit b. □

7.8 Deniable Split-KEM from Lattices

In this section we build an efficient deniable split-KEM under the hardness of LWE. We start by introducing briefly several concepts of lattice-based cryptography that we use to design the scheme.

7.8.1 Lattice toolbox

L_∞ and L_α norms. We start by recalling what the L_∞ and L_α norms over $\mathbb{Z}_q$ are. For an element w in $\mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $|\langle w\rangle_q|$. Then, we define the L_∞ and L_α norms for $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ over $\mathbb{Z}_q$ as follows:

$$\|\mathbf{w}\|_\infty = \max_{j\in[n]} \|w_j\|_\infty, \ \ \|\mathbf{w}\|_\alpha = \sqrt[\alpha]{\|w_1\|^\alpha_\infty + \ldots + \|w_n\|^\alpha_\infty}.$$

By default, $\|\mathbf{w}\| := \|\mathbf{w}\|_2$.

Probability distributions. We will use the binomial distribution Bin_1 which is defined as $\mathsf{Bin}_1(-1) = \mathsf{Bin}_1(1) = 1/4$ and $\mathsf{Bin}_1(0) = 1/2$.

Rounding functions. Given two parameters q and $B < \log q - 1$, we define the rounding function $\lfloor\cdot\rceil_{q,B}$ and the cross-rounding function $\langle\cdot\rangle_{q,B}$ as follows:

$$\lfloor\cdot\rceil_{q,B} : v \mapsto \left\lfloor \frac{2^B}{q} \cdot v \right\rceil \bmod 2^B, \;\; \langle\cdot\rangle_{q,B} : v \mapsto \left\lfloor \frac{2^{B+1}}{q} \cdot v \right\rfloor \bmod 2\,,$$

for any $v \in \mathbb{Z}_q$.

Reconciliation function. We recall the (generalised) reconciliation mechanism from Bos et al. and Peikert [Bos+16; Pei14], which for every approximate agreement in $\mathbb{Z}_q$ allows extracting shared bits. We refer the reader to the aforementioned works for more details. Let q be a positive integer. Let B be the number of bits we want to extract from one coefficient in $\mathbb{Z}_q$ so that $B < \log q - 1$. Now, for any $v \in \mathbb{Z}_q$, which is represented as an integer in $[0, q)$, we define the following functions.

Definition 7.8.1 (Randomised doubling function (dbl))**.** *For any $v \in \mathbb{Z}_q$, we define $\mathsf{dbl}(\cdot)$ as*

$$\mathsf{dbl}(v) : v \mapsto 2v - e, \quad e \leftarrow\!\!\$\, \mathsf{Bin}_1\,.$$

Then, we have the following property which comes from [Bos+16, Claim 3.1].

Lemma 7.8.1. *Let q be odd. If $v \in \mathbb{Z}_q$ is uniformly random and $\bar{v} \leftarrow\!\!\$\, \mathsf{dbl}(v) \in \mathbb{Z}_{2q}$, then $\lfloor\bar{v}\rceil_{2q,B}$ is uniformly random given $\langle\bar{v}\rangle_{2q,B}$.*

Now, we are ready to define the reconciliation function $\mathsf{Rec} : \mathbb{Z}_{2q} \times \mathbb{Z}_2 \to \mathbb{Z}_{2^B}$.

Definition 7.8.2 (Reconciliation function (Rec))**.** *For any $w \in \mathbb{Z}_{2q}$ and bit $b \in \{0,1\}$, let v be the closest element to $w \in \mathbb{Z}_{2q}$ s.t. $\langle v\rangle_{2q,B} = b$. Then, we define Rec as*

$$\mathsf{Rec}(w, b) := \lfloor v\rceil_{2q,B}\,.$$

The next result gives an important property of the reconciliation function Rec, as described by Peikert [Pei14, Section 3.2].

Lemma 7.8.2. *Let q be odd and $\bar{v} \leftarrow\!\!\$\, \mathsf{dbl}(v)$. If $|v - w| \leq \lfloor\frac{q}{2^{B+2}}\rfloor$ then*

$$\mathsf{Rec}(2w, \langle\bar{v}\rangle_{2q,B}) = \lfloor\bar{v}\rceil_{2q,B}.$$

Finally, we define the $\mathsf{HelpRec} : \mathbb{Z}_q \mapsto \{0,1\}$ function as follows:

Definition 7.8.3 ($\mathsf{HelpRec}$ function)**.** *On any input $v \in \mathbb{Z}_q$,*

$$\mathsf{HelpRec}(v) := \langle\bar{v}\rangle_{2q,B}, \quad \textit{where } \bar{v} \leftarrow \mathsf{dbl}(v)\,.$$

$\mathrm{LWE}_{n,m,\chi,q}(\mathscr{A})$

$b \leftarrow\$ \{0,1\}$
$\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}$
$\mathbf{t} \leftarrow\$ \mathbb{Z}_q^{n}$
$\mathbf{s} \leftarrow\$ \chi^{m}$
$\mathbf{e} \leftarrow\$ \chi^{n}$
if $b = 0$:
  $b' \leftarrow \mathscr{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$
else :
  $b' \leftarrow \mathscr{A}(\mathbf{A}, \mathbf{t})$
return $1_{b=b'}$

$\mathrm{ELWE}_{n,m,\bar{n},\chi,q}(\mathscr{A})$

$b \leftarrow\$ \{0,1\}$
$\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}$
$\mathbf{t} \leftarrow\$ \mathbb{Z}_q^{n}$
$\mathbf{e} \leftarrow\$ \chi^{n}$
$(\mathbf{Z}, \mathbf{W}) \leftarrow\$ \chi^{\bar{n}\times m} \times \chi^{\bar{n}\times n}$
if $b = 0$:
  $b' \leftarrow \mathscr{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q, \mathbf{Z}, \mathbf{W}, \mathbf{Z}\mathbf{s} + \mathbf{W}\mathbf{e} \bmod q)$
else :
  $b' \leftarrow \mathscr{A}(\mathbf{A}, \mathbf{t}, \mathbf{Z}, \mathbf{W}, \mathbf{Z}\mathbf{s} + \mathbf{W}\mathbf{e} \bmod q)$
return $1_{b=b'}$

Figure 7.10: LWE and ELWE games.

All the functions above can be naturally generalised to take as input vectors and matrices over $\mathbb{Z}_q$ by applying the function to each of the coefficients.

Learning-with-Errors. Security of our lattice constructions relies on the Learning-with-Errors (LWE) problem introduced by Regev [Reg05]. In this chapter we will consider the case where both the secret and error coefficients come from a probability distribution over $\mathbb{Z}$.

Definition 7.8.4 ($\mathsf{LWE}_{n,m,\chi,q}$)**.** *Let $n, m \in \mathbb{N}$ and χ be a probability distribution over $\mathbb{Z}$. The* LWE *problem asks the adversary $\mathscr{A}$ to distinguish between the following two distributions:*

1. $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$ *for* $\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}$*, a secret* $\mathbf{s} \leftarrow\$ \chi^{m}$*, and error* $\mathbf{e} \leftarrow\$ \chi^{n}$*.*
2. $(\mathbf{A}, \mathbf{t}) \leftarrow\$ \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n}$*.*

The advantage of an adversary $\mathscr{A}$ is then defined as

$$\mathsf{Adv}^{\mathrm{lwe}}_{n,m,\chi,q}(\mathscr{A}) := \left| \Pr\left[\mathrm{LWE}_{n,m,\chi,q}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2} \right| ,$$

where LWE is the game defined on the left of Figure 7.10.

7.8.2 Extended-LWE

Our proof of deniability for the split-KEM will involve a new security assumption, which we call the Extended-LWE problem (ELWE). Intuitively, it is similar to the plain LWE problem, but the adversary is now also given random linear combinations of the secrets and errors.

Definition 7.8.5 ($\mathsf{ELWE}_{n,m,\bar{n},\chi,q}$)**.** *Let $n, m \in \mathbb{N}$ and χ be a probability distribution over $\mathbb{Z}$. The* ELWE *problem asks the adversary $\mathscr{A}$ to distinguish between the following two cases:*

1. $\big(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q, \mathbf{Z}, \mathbf{W}, \mathbf{Z}\mathbf{s} + \mathbf{W}\mathbf{e} \bmod q\big)$ *for* $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, *secret* $\mathbf{s} \leftarrow\!\!\$\ \chi^m$, *error* $\mathbf{e} \leftarrow\!\!\$\ \chi^n$, *and* $(\mathbf{Z}, \mathbf{W}) \leftarrow\!\!\$\ \chi^{\bar{n}\times m} \times \chi^{\bar{n}\times n}$,

2. $\big(\mathbf{A}, \mathbf{t}, \mathbf{Z}, \mathbf{W}, \mathbf{Z}\mathbf{s} + \mathbf{W}\mathbf{e} \bmod q\big)$ *for* $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, $\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, *secret* $\mathbf{s} \leftarrow\!\!\$\ \chi^m$, *error* $\mathbf{e} \leftarrow\!\!\$\ \chi^n$, *and* $(\mathbf{Z}, \mathbf{W}) \leftarrow\!\!\$\ \chi^{\bar{n}\times m} \times \chi^{\bar{n}\times n}$.

Formally, we define the advantage of an ELWE adversary $\mathscr{A}$ *as*

$$\mathsf{Adv}^{\mathrm{elwe}}_{n,m,\bar{n},\chi,q}(\mathscr{A}) = \left|\Pr\left[\mathrm{ELWE}_{n,m,\bar{n},\chi,q}(\mathscr{A}) \Rightarrow 1\right] - \frac{1}{2}\right| ,$$

where ELWE is the game defined on the right in Figure 7.10.

This problem is a natural generalisation of the Extended-LWE problem by Alperin-Sheriff and Peikert [AP12], where now $(\mathbf{Z}, \mathbf{W})$ are matrices and not just vectors. Here, we also simplify the definition and assume that the coefficients of $\mathbf{Z}$ and $\mathbf{W}$ come from the same distribution χ as the secrets and errors.

We show in the following theorem that the hardness of this newly introduced ELWE problem reduces to the hardness of LWE.

Theorem 7.8.1. *Let* q *be an odd prime and* χ *be symmetric around* 0. *For any efficient* $\mathsf{ELWE}_{n,m,\bar{n},\chi,q}$ *adversary* $\mathscr{A}$ *there exists an efficient* $\mathsf{LWE}_{n+m,m,\chi,q}$ *adversary* $\mathscr{B}$ *such that*

$$\mathsf{Adv}^{\mathrm{elwe}}_{n,m,\bar{n},\chi,q}(\mathscr{A}) \leq 1/\delta_{\mathsf{elwe}} \cdot \mathsf{Adv}^{\mathrm{lwe}}_{n+m,m,\chi,q}(\mathscr{B}) + \mathsf{negl}(n) ,$$

where

$$\delta_{\mathsf{elwe}} := \Pr\left[\mathbf{Z}(\mathbf{e} - \mathbf{d}) = \mathbf{0} \pmod{q} : \mathbf{Z} \leftarrow\!\!\$\ \chi^{\bar{n}\times(n+m)}, \mathbf{e}, \mathbf{d} \leftarrow\!\!\$\ \chi^{n+m}\right]. \tag{7.1}$$

Proof. The proof is given in Appendix B. □

7.8.3 Construction

We can now present our Frodo-inspired [Bos+16] split-KEM. The scheme is given in Figure 7.11. The key generation works as follows. The public key pk_A for party A is a pair $(\mathbf{A}, \mathbf{B}_\mathsf{A})$, where $\mathbf{A}$ is a uniformly random matrix over $\mathbb{Z}_q$ given as a public parameter, and $\mathbf{B}_\mathsf{A} := \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$ where $\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\ \chi^{n\times\bar{n}}$. The secret key becomes a pair $\mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A})$. Similarly, the public key pk_B for party B is a pair $(\mathbf{A}, \mathbf{B}_\mathsf{B})$, where $\mathbf{B}_\mathsf{B} := \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$, while the secret key is $\mathsf{sk}_\mathsf{B} = (\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B})$, where $\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\bar{n}\times n}$.

Then, B samples a matrix $\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\bar{n}\times\bar{n}}$ and computes the matrix $\mathbf{V} := \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B}$. Next, it computes $\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$ and $\mathbf{K} \leftarrow \mathsf{Rec}(\mathbf{V}, \mathsf{ct})$. Then, B outputs ct. Then, party A decapsulates as follows: given $(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathsf{ct})$, it computes $\mathbf{V}' = \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$ and $\mathbf{K}' = \mathsf{Rec}(2\mathbf{V}', \mathsf{ct})$. Finally, A returns the key $\mathbf{K}'$.

$\mathsf{KeyGenA}(1^\lambda)$

- $\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\ \chi(\mathbb{Z}_q^{n\times\overline{n}})$
- $\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
- $\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
- $\mathsf{pk}_\mathsf{A} \leftarrow (\mathbf{A}, \mathbf{B}_\mathsf{A})$
- $\mathsf{sk}_\mathsf{A} \leftarrow (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A})$
- **return** $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A})$

$\mathsf{Encaps}(\mathsf{pk}_\mathsf{A} = (\mathbf{A}, \mathbf{B}_\mathsf{A}), \mathsf{sk}_\mathsf{B} = (\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B}, \mathbf{F}_\mathsf{B}))$

- // We assume B encapsulates
- $\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\bar{n}\times\bar{n}}$
- $\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B}$
- $\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
- $\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
- **return** $(\mathbf{K}, \mathsf{ct})$

$\mathsf{KeyGenB}(1^\lambda)$

- $\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times n}$
- $\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
- $\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
- $\mathsf{pk}_\mathsf{B} \leftarrow (\mathbf{A}, \mathbf{B}_\mathsf{B})$
- $\mathsf{sk}_\mathsf{B} \leftarrow (\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B}, \mathbf{F}_\mathsf{B})$
- **return** $(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{B})$

$\mathsf{Decaps}(\mathsf{pk}_\mathsf{B} = (\mathbf{A}, \mathbf{B}_\mathsf{B}), \mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A}), \mathsf{ct})$

- $\mathbf{V}' \leftarrow \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$
- $\mathbf{K}' \leftarrow \mathsf{Rec}(2\mathbf{V}', \mathsf{ct})$
- **return** $\mathbf{K}'$

Figure 7.11: Our variant of FrodoKEX [Bos+16] expressed as a split-KEM. The matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ is assumed to be a public parameter and sampled uniformly at random.

We note that our construction can easily be made symmetric, in the sense that A can encapsulate using B's public key by changing the order of the operands in matrix multiplication in Encaps, such that the dimensions match. Then, Decaps can be modified similarly such that B can decapsulate the resulting ciphertext using A's public key and its own secret key.

7.8.4 Security analysis

Lemma 7.8.3 (Correctness)**.** *Let χ be a symmetric distribution around* 0 *and δ_corr be the following probability:*

$$\Pr\left[|\langle \mathbf{s}, \mathbf{d}\rangle + e + f| > \frac{q}{2^{B+2}} : \mathbf{s}, \mathbf{d} \leftarrow\!\!\$\ \chi^{2n}, e, f \leftarrow\!\!\$\ \chi\right]. \tag{7.2}$$

Then, sKEM *defined in Figure 7.11 is $(\bar{n}^2\delta_\mathsf{corr})$-correct.*

Proof. Suppose $(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{A}) \leftarrow\!\!\$\ \mathsf{KeyGenA}(1^\lambda)$ and $(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{B}) \leftarrow\!\!\$\ \mathsf{KeyGenB}(1^\lambda)$. In addition, let

$$(\mathbf{K}, \mathsf{ct}) \leftarrow\!\!\$\ \mathsf{Encaps}(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{B}) \quad \text{and} \quad \mathbf{K}' \leftarrow\!\!\$\ \mathsf{Decaps}(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathsf{ct}).$$

We want to prove that $\mathbf{K} = \mathbf{K}'$. By definition of encapsulation, we know that $\mathbf{K} = \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$ where $\mathsf{ct} = \mathsf{HelpRec}(\mathbf{V})$ and

$$\mathbf{V} = \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B} = \mathbf{S}_\mathsf{B}\mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} + \mathbf{E}_\mathsf{B}.$$

Thus, by Lemma 7.8.2, $\mathbf{K} = \lfloor \mathbf{V} \rceil_{2q,2^B}$. On the other hand,

$$\mathbf{V}' = \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A} = \mathbf{S}_\mathsf{B}\mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$$

which implies that $\mathbf{V} - \mathbf{V}' = \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} + \mathbf{E}_\mathsf{B} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} - \mathbf{F}_\mathsf{A}$. If $\|\mathbf{V} - \mathbf{V}'\|_\infty < \frac{q}{2^{B+2}}$ then by Lemma 7.8.2 we must have

$$\mathbf{K}' = \mathsf{Rec}(2\mathbf{V}', \mathsf{HelpRec}(\mathbf{V})) = \lfloor \mathbf{V} \rceil_{2q,2^B} = \mathbf{K}$$

so correctness holds. Now, using the fact that χ is symmetric around 0, the probability $\|\mathbf{V} - \mathbf{V}'\|_\infty > \frac{q}{2^{B+2}}$ can be upper-bounded using the union bound as follows:

$$\Pr\left[\|\mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} + \mathbf{E}_\mathsf{B} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} - \mathbf{F}_\mathsf{A}\|_\infty > \frac{q}{2^{B+2}}\right] \le \bar{n}^2 \cdot \Pr\left[|\mathbf{s}_0^T\mathbf{d}_0 + \mathbf{s}_1^T\mathbf{d}_1 + e + f| > \frac{q}{2^{B+2}}\right],$$

where $\mathbf{s}_0, \mathbf{s}_1, \mathbf{d}_0, \mathbf{d}_1 \leftarrow\$ \chi^n$ and $e, f \leftarrow\$ \chi$. This concludes the proof. □

OW-CPA security

Next, we focus on proving the OW-CPA security of our construction.

Lemma 7.8.4 (OW-CPA Security)**.** *Let χ be a symmetric distribution over $[-\gamma, \gamma]$ for any $\gamma > 0$. Let* sKEM *be the split-KEM defined in Figure 7.11. Then, for any efficient adversary $\mathscr{A}$, there exist efficient adversaries $\mathscr{B}$ and $\mathscr{B}'$ such that*

$$\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{sKEM}}(\mathscr{A}) \le 2^{-B\bar{n}^2} + \bar{n} \cdot \left(\mathsf{Adv}^{\text{lwe}}_{n,n,\chi,q}(\mathscr{B}) + \mathsf{Adv}^{\text{lwe}}_{n+\bar{n},n,\chi,q}(\mathscr{B}')\right).$$

Proof. Let $\mathscr{A}$ be an efficient adversary against the OW-CPA game. We prove the statement using the hybrid games described explicitly in Figure 7.12.

Game Γ_1: This is the standard OW-CPA game.

Game Γ_2: Instead of computing $\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$, the experiment samples $\mathbf{B}_\mathsf{A} \leftarrow \mathbb{Z}_q^{n\times\bar{n}}$. One can naturally build an efficient adversary, which can solve the $\mathsf{LWE}_{n,n,\chi,q}$ problem with probability at least $\frac{1}{\bar{n}}|\Pr[\Gamma^2] - \Pr[\Gamma^1]|$. Hence, we deduce that this probability is negligible.

Game Γ_3: Here, the experiment computes the values $\mathbf{B}_\mathsf{B}$ and $\mathbf{V}$ differently. Namely, instead of computing:

$$\begin{bmatrix}\mathbf{B}_\mathsf{B} & \mathbf{V}\end{bmatrix} := \mathbf{S}_\mathsf{B}\begin{bmatrix}\mathbf{A} & \mathbf{B}_\mathsf{A}\end{bmatrix} + \begin{bmatrix}\mathbf{D}_\mathsf{B} & \mathbf{E}_\mathsf{B}\end{bmatrix},$$

it samples

$$\begin{bmatrix}\mathbf{B}_\mathsf{B} & \mathbf{V}\end{bmatrix} \leftarrow\$ \mathbb{Z}_q^{\bar{n}\times(n+\bar{n})}.$$

Thus, one can naturally construct an efficient reduction which solves $\mathsf{LWE}_{n+\bar{n},n,\chi,q}$ with probability at least $\frac{1}{\bar{n}}|\Pr[\Gamma^2] - \Pr[\Gamma^3]|$.

Finally, it is easy to see that in Γ_3 the matrix $\mathbf{V}$ is actually uniformly random over $\mathbb{Z}_q$.

$\Gamma_1(\mathscr{A})$	$\Gamma_2(\mathscr{A})$	$\Gamma_3(\mathscr{A})$
$\mathbf{S}_{\mathsf{A}}, \mathbf{D}_{\mathsf{A}} \leftarrow_{\$} \chi(\mathbb{Z}_q^{n\times\bar{n}})$	$\mathbf{B}_{\mathsf{A}} \leftarrow \mathbb{Z}_q^{n\times\bar{n}}$	$\mathbf{B}_{\mathsf{A}} \leftarrow \mathbb{Z}_q^{n\times\bar{n}}$
$\mathbf{F}_{\mathsf{A}} \leftarrow_{\$} \chi^{\bar{n}\times\bar{n}}$	$\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times n}$	$\mathbf{B}_{\mathsf{B}} \leftarrow_{\$} \mathbb{Z}_q^{n\times\bar{n}}$
$\mathbf{B}_{\mathsf{A}} \leftarrow \mathbf{A}\mathbf{S}_{\mathsf{A}} + \mathbf{D}_{\mathsf{A}}$	$\mathbf{F}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times\bar{n}}$	$\mathbf{V} \leftarrow_{\$} \mathbb{Z}_q^{\bar{n}\times\bar{n}}$
$\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times n}$	$\mathbf{B}_{\mathsf{B}} \leftarrow \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$	$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{F}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times\bar{n}}$	$\mathbf{E}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times\bar{n}}$	$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$\mathbf{B}_{\mathsf{B}} \leftarrow \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$	$\mathbf{V} \leftarrow \mathbf{S}_{\mathsf{B}}\mathbf{B}_{\mathsf{A}} + \mathbf{E}_{\mathsf{B}}$	$\mathbf{K}' \leftarrow_{\$} \mathscr{A}(\mathbf{A}, \mathbf{B}_{\mathsf{A}}, \mathbf{B}_{\mathsf{B}}, \mathsf{ct})$
$\mathbf{E}_{\mathsf{B}} \leftarrow_{\$} \chi^{\bar{n}\times\bar{n}}$	$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$	**return** $1_{\mathbf{K}=\mathbf{K}'}$
$\mathbf{V} \leftarrow \mathbf{S}_{\mathsf{B}}\mathbf{B}_{\mathsf{A}} + \mathbf{E}_{\mathsf{B}}$	$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$	
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$	$\mathbf{K}' \leftarrow_{\$} \mathscr{A}(\mathbf{A}, \mathbf{B}_{\mathsf{A}}, \mathbf{B}_{\mathsf{B}}, \mathsf{ct})$	
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$	**return** $1_{\mathbf{K}=\mathbf{K}'}$	
$\mathbf{K}' \leftarrow_{\$} \mathscr{A}(\mathbf{A}, \mathbf{B}_{\mathsf{A}}, \mathbf{B}_{\mathsf{B}}, \mathsf{ct})$		
return $1_{\mathbf{K}=\mathbf{K}'}$		

Figure 7.12: Security games for the proof of Lemma 7.8.4. The lines in blue highlight the main differences from the previous game.

Hence by Lemma 7.8.1, for the adversary $\mathscr{A}$, which is given ct, the key $\mathbf{K}$ looks uniformly random. Therefore, the probability of guessing the key is bounded by $2^{-\bar{n}^2 B}$. $\square$

Deniability

We will use the (transposed) matrix version of ELWE where the secrets and errors are now matrices. In particular, we will be interested in the problem of distinguishing between

$$(\mathbf{A}, \mathbf{S}\mathbf{A} + \mathbf{E} \bmod q, \mathbf{Z}, \mathbf{W}, \mathbf{S}\mathbf{Z} + \mathbf{E}\mathbf{W} \bmod q)$$

and

$$(\mathbf{A}, \mathbf{T}, \mathbf{Z}, \mathbf{W}, \mathbf{S}\mathbf{Z} + \mathbf{E}\mathbf{W} \bmod q)\ ,$$

where $\mathbf{S} \leftarrow_{\$} \chi^{\bar{n}\times m}$, $\mathbf{E} \leftarrow_{\$} \chi^{\bar{n}\times n}$ and $\mathbf{T} \leftarrow_{\$} \mathbb{Z}_q^{\bar{n}\times n}$. This problem can be reduced to ELWE with reduction loss $\bar{n}$ via a standard hybrid argument.

We are ready to prove deniability of the split-KEM based on Extended-LWE. Intuitively, matrices $(\mathbf{S}, \mathbf{E}) := (\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}})$ will be the secret and error constructed by party B, which are hidden from the adversary, while $(\mathbf{Z}, \mathbf{W}) := (\mathbf{D}_{\mathsf{A}}, \mathbf{S}_{\mathsf{A}})$ will be the error and the secret generated by **A** which are given as input to the simulator. The key observation is that the additional hint provided as $\mathbf{S}\mathbf{Z} + \mathbf{E}\mathbf{W} \bmod q$ will be used to simulate the "shared key" $\mathbf{V}$ (before applying the reconciliation function).

Theorem 7.8.2 (Deniability)**.** *Let* sKEM *be the split-KEM defined in Figure 7.11 and* Sim *as defined in Figure 7.13. Then, for any efficient adversary $\mathscr{A}$, there exist efficient adversaries $\mathscr{B}$*

and $\mathscr{B}'$ *such that*

$$\mathsf{Adv}^{\mathsf{deny}}_{\mathsf{sKEM},\mathsf{Sim}}(\mathscr{A}) \leq \bar{n} \cdot \left(\mathsf{Adv}^{\mathsf{elwe}}_{n,n,\bar{n},\chi,q}(\mathscr{B}) + \mathsf{Adv}^{\mathsf{lwe}}_{n,n,\chi,q}(\mathscr{B}')\right) .$$

Proof. We proceed with a sequence of games detailed in Figure 7.14.

Game Γ_1: This is the standard *real* deniability experiment, which we recall here. First, $\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\$ \chi^{n\times\bar{n}}$, $\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\$ \chi^{\bar{n}\times n}$, and $\mathbf{F}_\mathsf{A}, \mathbf{F}_\mathsf{B} \leftarrow\$ \chi^{\bar{n}\times\bar{n}}$ are sampled. Then, the public keys $\mathbf{B}_\mathsf{A} = \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$ and $\mathbf{B}_\mathsf{B} = \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$ are computed. The encapsulation algorithm samples $\mathbf{E}_\mathsf{B} \leftarrow\$ \chi^{\bar{n}\times\bar{n}}$ and sets $\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B}$. Finally, the experiment runs $\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$ and $\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$ and eventually outputs

$$(\mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B}, \mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$$

to the adversary $\mathscr{A}$.

Game Γ_2: The experiment is identical to the previous one, apart from the fact that now $\mathbf{V}$ is explicitly computed as $\mathbf{V} = \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}$. Clearly, $\Pr[\Gamma^1] = \Pr[\Gamma^2]$ since

$$\begin{aligned}\mathbf{V} &= \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}\\ &= \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + (\mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B})\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}\\ &= \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B}.\end{aligned}$$

Game Γ_3: Here, the experiment follows Γ_2 with the only difference being that the experiment samples $\mathbf{B}_\mathsf{B}$ uniformly at random from $\mathbb{Z}_q^{\bar{n}\times n}$ instead of computing $\mathbf{B}_\mathsf{B} = \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$.

Lemma 7.8.5. *There exists an efficient algorithm* $\mathscr{B}$ *that solves the* $\mathsf{ELWE}_{n,n,\bar{n},\chi,q}$ *problem with probability at least* $\frac{1}{\bar{n}}|\Pr[\Gamma^3] - \Pr[\Gamma^2]|$.

Proof. We provide a reduction $\mathscr{B}$ to the (transposed) matrix-version of the Extended-LWE problem as described above. Namely, the reduction is given a tuple of matrices $(\mathbf{A}, \mathbf{B}, \mathbf{Z}, \mathbf{W}, \mathbf{H})$. Then, it sets $\mathbf{S}_\mathsf{A} := -\mathbf{W}$, $\mathbf{D}_\mathsf{A} := \mathbf{Z}$ and $\mathbf{B}_\mathsf{B} := \mathbf{B}$. Further, the reduction samples $\mathbf{F}_\mathsf{A} \leftarrow\$ \chi^{\bar{n}\times\bar{n}}$ and computes

$$\mathbf{B}_\mathsf{A} := \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A} \quad \text{and} \quad \mathbf{V} := \mathbf{H} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B} \ ,$$

where $\mathbf{E}_\mathsf{B} \leftarrow\$ \chi^{\bar{n}\times\bar{n}}$. Finally, the reduction runs $\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$ and $\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$, and outputs $(\mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B}, \mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$ to the adversary.

Suppose the input tuple received by $\mathscr{B}$ is a true Extended-LWE instance, i.e. $\mathbf{B}_\mathsf{B} = \mathbf{B} = \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$ for $\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\$ \chi^{\bar{n}\times n}$. This implies that $\mathbf{H} = \mathbf{S}_\mathsf{B}\mathbf{Z} + \mathbf{D}_\mathsf{B}\mathbf{W} = \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A}$ and hence

$$\mathbf{V} = \mathbf{H} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B} = \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}.$$

This implies that when the input tuple is the Extended-LWE instance then $\mathscr{B}$ perfectly sim-

$\mathsf{Sim}(\mathbf{A}, \mathbf{B}_{\mathsf{B}}, \mathbf{S}_{\mathsf{A}}, \mathbf{D}_{\mathsf{A}}, \mathbf{F}_{\mathsf{A}})$

$\mathbf{S}_{\mathsf{sim}}, \mathbf{D}_{\mathsf{sim}} \leftarrow\!\!\$\ \chi^{\bar{n}\times n}$
$\mathbf{E}_{\mathsf{sim}} \leftarrow\!\!\$\ \chi^{\bar{n}\times \bar{n}}$
$\mathbf{V}_{\mathsf{sim}} \leftarrow \mathbf{S}_{\mathsf{sim}}\mathbf{D}_{\mathsf{A}} - \mathbf{D}_{\mathsf{sim}}\mathbf{S}_{\mathsf{A}} + \mathbf{B}_{\mathsf{B}}\mathbf{S}_{\mathsf{A}} + \mathbf{E}_{\mathsf{sim}}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V}_{\mathsf{sim}})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}_{\mathsf{sim}}, \mathsf{ct})$
return $(\mathbf{K}, \mathsf{ct})$

Figure 7.13: Simulator for the deniability game.

ulates the output of Γ_2 [6]. On the other hand, if $\mathbf{B}_{\mathsf{B}}$ is uniformly random then $\mathscr{B}$ perfectly simulates the output of Γ_3. Finally the statement follows by further reducing the matrix-version of ELWE to the standard one. □

Game Γ_4: First, we rename the variables $(\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}}, \mathbf{E}_{\mathsf{B}}) := (\mathbf{S}_{\mathsf{sim}}, \mathbf{D}_{\mathsf{sim}}, \mathbf{E}_{\mathsf{sim}})$. Further, instead of picking $\mathbf{B}_{\mathsf{B}}$ uniformly at random, the experiment now samples alternative secrets/errors $\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}} \leftarrow\!\!\$\ \chi^{\overline{n}\times n}$ for B and sets $\mathbf{B}_{\mathsf{B}} := \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$. The rest is identical as in Γ_3.

Lemma 7.8.6. *There exists an efficient algorithm $\mathscr{B}'$ that solves the $\mathsf{LWE}_{n,n,\chi,q}$ problem with probability at least $\frac{1}{\bar{n}}|\Pr[\Gamma^4] - \Pr[\Gamma^3]|$.*

Proof. We describe a reduction $\mathscr{B}$ which solves the matrix-version of LWE. Then, the reduction to plain LWE follows by a hybrid argument. First, $\mathscr{B}$ is given a tuple $(\mathbf{A}, \mathbf{B})$ where either $\mathbf{B} = \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$ for short $\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}}$ or $\mathbf{B}$ is uniformly random. In either case, only given $\mathbf{A}$ and $\mathbf{B}$, the reduction $\mathscr{B}$ can simulate the rest of Γ_3 (and Γ_4). If $\mathbf{B} = \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$ then this becomes Γ_4, and when $\mathbf{B}_{\mathsf{B}}$ is uniformly random then $\mathscr{B}$ simulates Γ_3. □

Finally, we present the simulator in Figure 7.13. Γ_4 can now be alternatively described in the following way. The experiment first samples $\mathbf{S}_{\mathsf{A}}, \mathbf{D}_{\mathsf{A}} \leftarrow\!\!\$\ \chi^{n\times\overline{n}}$ and $\mathbf{S}_{\mathsf{B}}, \mathbf{D}_{\mathsf{B}} \leftarrow\!\!\$\ \mathbb{Z}_q^{\overline{n}\times n}$ and $\mathbf{F}_{\mathsf{A}} \leftarrow\!\!\$\ \chi^{\bar{n}\times\bar{n}}$. Further, the public keys are defined as $\mathbf{B}_{\mathsf{A}} = \mathbf{A}\mathbf{S}_{\mathsf{A}} + \mathbf{D}_{\mathsf{A}}$ and $\mathbf{B}_{\mathsf{B}} = \mathbf{S}_{\mathsf{B}}\mathbf{A} + \mathbf{D}_{\mathsf{B}}$. Finally, it runs $(\mathbf{K}, \mathsf{ct}) \leftarrow\!\!\$\ \mathsf{Sim}(\mathbf{A}, \mathbf{B}_{\mathsf{B}}, \mathbf{S}_{\mathsf{A}}, \mathbf{D}_{\mathsf{A}}, \mathbf{F}_{\mathsf{A}})$ and outputs $(\mathbf{A}, \mathbf{B}_{\mathsf{A}}, \mathbf{B}_{\mathsf{B}}, \mathbf{S}_{\mathsf{A}}, \mathbf{D}_{\mathsf{A}}, \mathbf{F}_{\mathsf{A}}, \mathbf{K}, \mathsf{ct})$. Hence, Γ^4 is the same as the simulated deniability game. This concludes the proof. □

Decaps-OW-CPA Security

Finally, we show that our split-KEM satisfies the decaps-OW-CPA security notion (see Definition 7.5.6).

Lemma 7.8.7 (decaps-OW-CPA Security)**.** *Let* sKEM *be the split-KEM defined in Figure 7.11, m be such that the ciphertext space of the split-KEM is $\{0,1\}^m$, and χ be a probability distribution*

[6] We used the fact that χ is symmetric around 0 to argue that $\mathbf{S}_{\mathsf{A}} := -\mathbf{W}$ is correctly distributed.

$\Gamma_1(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\; \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathsf{pk}_\mathsf{A} = (\mathbf{A}, \mathbf{B}_\mathsf{A})$
$\mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A})$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times n}$
$\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
$\mathsf{pk}_\mathsf{B} = (\mathbf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\bar{n}\times\bar{n}}$
$\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{B}_\mathsf{A} + \mathbf{E}_\mathsf{B}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$b \leftarrow\!\!\$\; \mathscr{A}(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$
return b

$\Gamma_2(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\; \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathsf{pk}_\mathsf{A} = (\mathbf{A}, \mathbf{B}_\mathsf{A})$
$\mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A})$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times n}$
$\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
$\mathsf{pk}_\mathsf{B} = (\mathbf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\bar{n}\times\bar{n}}$
$\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$b \leftarrow\!\!\$\; \mathscr{A}(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$
return b

$\Gamma_3(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\; \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathsf{pk}_\mathsf{A} = (\mathbf{A}, \mathbf{B}_\mathsf{A})$
$\mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A})$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times n}$
$\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbb{Z}_q^{\bar{n}\times n}$
$\mathsf{pk}_\mathsf{B} = (\mathbf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\bar{n}\times\bar{n}}$
$\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{B}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$b \leftarrow\!\!\$\; \mathscr{A}(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$
return b

$\Gamma_4(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\; \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathsf{pk}_\mathsf{A} = (\mathbf{A}, \mathbf{B}_\mathsf{A})$
$\mathsf{sk}_\mathsf{A} = (\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A}, \mathbf{F}_\mathsf{A})$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times n}$
$\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\; \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
$\mathsf{pk}_\mathsf{B} = (\mathbf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{E}_\mathsf{sim} \leftarrow\!\!\$\; \chi^{\bar{n}\times\bar{n}}$
$\mathbf{S}_\mathsf{sim}, \mathbf{D}_\mathsf{sim} \leftarrow\!\!\$\; \chi^{\overline{n}\times n}$
$\mathbf{V} \leftarrow \mathbf{S}_\mathsf{sim}\mathbf{D}_\mathsf{A} - \mathbf{D}_\mathsf{sim}\mathbf{S}_\mathsf{A} + \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{E}_\mathsf{sim}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$b \leftarrow\!\!\$\; \mathscr{A}(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathbf{K}, \mathsf{ct})$
return b

Figure 7.14: Security games for the proof of Theorem 7.8.2. The lines in blue highlight the main differences from the previous game. The lines in gray correspond to the simulator defined in Figure 7.13.

over $[-\gamma, \gamma]$ *symmetric around* 0 *for any* $\gamma > 0$. *Then, for any efficient adversary* $\mathscr{A}$, *there exist efficient adversaries* $\mathscr{B}$ *and* $\mathscr{B}'$ *such that*

$$\mathsf{Adv}^{\text{decaps-ow-cpa}}_{\mathsf{sKEM}}(\mathscr{A}) \leq 2^m \cdot \left(\delta_{\mathsf{cpa}}^{\bar{n}^2} + \bar{n} \cdot \mathsf{Adv}^{\mathsf{lwe}}_{n,n,\chi,q}(\mathscr{B}) + \bar{n} \cdot \mathsf{Adv}^{\mathsf{lwe}}_{n+\bar{n},n,\chi,q}(\mathscr{B}')\right) ,$$

where

$$\delta_{\mathsf{cpa}} := \max_{\substack{\mathsf{ct} \in \{0,1\} \\ u \in \mathbb{Z}_{2^B}}} \Pr_{w \leftarrow \$ \mathbb{Z}_q} [\mathsf{Rec}(2w, \mathsf{ct}) = u] \ . \tag{7.3}$$

Proof. Let $\mathscr{A}$ be an efficient adversary against the decaps-OW-CPA game. We prove the statement using the hybrid games described explicitly in Figure 7.15.

Game Γ_1: This is the standard decaps-OW-CPA game corresponding to the sKEM in Figure 7.11.

Game Γ_2: In this game, the ciphertext ct is not given to the adversary anymore. Note that the first phase adversary outputting $\mathbf{B}$ is now useless and it can be removed, along with the operations needed to compute ct. Given the ciphertext space is $\{0,1\}^m$ for some $m \in \mathbb{Z}$, we have $\Pr[\Gamma^2] \geq \frac{1}{2^m} \Pr[\Gamma^1]$ as any adversary in Γ_2 can simulate the view of an adversary in Γ_1 by guessing ct.

Game Γ_3: In this game, the only change is that instead of computing $\mathbf{B}_\mathsf{B} = \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$, it is picked uniformly at random from $\mathbb{Z}_q^{\bar{n} \times n}$. The indistinguishability between Γ_3 and Γ_2 follows directly from $\mathsf{LWE}_{n,n,\chi,q}$.

Game Γ_4: Now, instead of computing $\mathbf{B}_\mathsf{A}$ and $\mathbf{V}'$ as:

$$\begin{bmatrix} \mathbf{B}_\mathsf{A} \\ \mathbf{V}' \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B}_\mathsf{B} \end{bmatrix} \mathbf{S}_\mathsf{A} + \begin{bmatrix} \mathbf{D}_\mathsf{A} \\ \mathbf{F}_\mathsf{A} \end{bmatrix},$$

the experiment samples $\mathbf{B}_\mathsf{A} \leftarrow\$ \mathbb{Z}_q^{n \times \bar{n}}$ and $\mathbf{V}' \leftarrow\$ \mathbb{Z}_q^{\bar{n} \times \bar{n}}$ uniformly at random. Then, the reduction executes Lines 4 to 6 of Γ_4. Clearly there is an efficient adversary which solves $\mathsf{LWE}_{n+\bar{n},n,\chi,q}$ with probability at least $\frac{1}{\bar{n}} |\Pr[\Gamma^4] - \Pr[\Gamma^3]|$.

Finally, since $\mathbf{V}'$ is uniformly random, the probability that any adversary wins Γ_4, i.e. $\mathbf{K}_\mathsf{A} = \mathbf{K}'_\mathsf{A}$, can be upper-bounded by $\delta_{\mathsf{cpa}}^{\bar{n}^2}$ by definition of δ_{cpa}. Collecting the probabilities concludes the proof. □

$\Gamma_1(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\ \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times n}$
$\mathbf{F}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
$\mathsf{st}, \mathbf{B} \leftarrow\!\!\$\ \mathscr{A}(\mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{E}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
$\mathbf{V} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{B} + \mathbf{E}_\mathsf{B}$
$\mathsf{ct} \leftarrow \mathsf{HelpRec}(\mathbf{V})$
$\mathbf{K} \leftarrow \mathsf{Rec}(2\mathbf{V}, \mathsf{ct})$
$\mathbf{K}'_\mathsf{A}, \mathsf{ct}' \leftarrow\!\!\$\ \mathscr{A}(\mathsf{st}, \mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B}, \mathsf{ct})$
$\mathbf{V}' \leftarrow \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$
$\mathbf{K}_\mathsf{A} \leftarrow \mathsf{Rec}(2\mathbf{V}', \mathsf{ct}')$
$\mathbf{return}\ 1_{\mathbf{K}_\mathsf{A} = \mathbf{K}'_\mathsf{A}}$

$\Gamma_2(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\ \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathbf{S}_\mathsf{B}, \mathbf{D}_\mathsf{B} \leftarrow\!\!\$\ \chi^{\overline{n}\times n}$
$\mathbf{B}_\mathsf{B} \leftarrow \mathbf{S}_\mathsf{B}\mathbf{A} + \mathbf{D}_\mathsf{B}$
$\mathbf{K}'_\mathsf{A}, \mathsf{ct}' \leftarrow\!\!\$\ \mathscr{A}(\mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{V}' \leftarrow \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$
$\mathbf{K}_\mathsf{A} \leftarrow \mathsf{Rec}(2\mathbf{V}', \mathsf{ct}')$
$\mathbf{return}\ 1_{\mathbf{K}_\mathsf{A} = \mathbf{K}'_\mathsf{A}}$

$\Gamma_3(\mathscr{A})$

$\mathbf{S}_\mathsf{A}, \mathbf{D}_\mathsf{A} \leftarrow\!\!\$\ \chi(\mathbb{Z}_q^{n\times\overline{n}})$
$\mathbf{F}_\mathsf{A} \leftarrow\!\!\$\ \chi^{\overline{n}\times\overline{n}}$
$\mathbf{B}_\mathsf{A} \leftarrow \mathbf{A}\mathbf{S}_\mathsf{A} + \mathbf{D}_\mathsf{A}$
$\mathbf{B}_\mathsf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{\overline{n}\times n}$
$\mathbf{K}'_\mathsf{A}, \mathsf{ct}' \leftarrow\!\!\$\ \mathscr{A}(\mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{V}' \leftarrow \mathbf{B}_\mathsf{B}\mathbf{S}_\mathsf{A} + \mathbf{F}_\mathsf{A}$
$\mathbf{K}_\mathsf{A} \leftarrow \mathsf{Rec}(2\mathbf{V}', \mathsf{ct}')$
$\mathbf{return}\ 1_{\mathbf{K}_\mathsf{A} = \mathbf{K}'_\mathsf{A}}$

$\Gamma_4(\mathscr{A})$

$\mathbf{B}_\mathsf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times\bar{n}}$
$\mathbf{B}_\mathsf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{\bar{n}\times n}$
$\mathbf{V}' \leftarrow\!\!\$\ \mathbb{Z}_q^{\bar{n}\times\bar{n}}$
$\mathbf{K}'_\mathsf{A}, \mathsf{ct}' \leftarrow\!\!\$\ \mathscr{A}(\mathbf{A}, \mathbf{B}_\mathsf{A}, \mathbf{B}_\mathsf{B})$
$\mathbf{K}_\mathsf{A} \leftarrow \mathsf{Rec}(2\mathbf{V}', \mathsf{ct}')$
$\mathbf{return}\ 1_{\mathbf{K}_\mathsf{A} = \mathbf{K}'_\mathsf{A}}$

Figure 7.15: Security games for the proof of Lemma 7.8.7. The lines in blue highlight the main differences from the previous game.

$\mathsf{KeyGen}_{\mathsf{sKEM}}(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}_{\mathsf{sKEM}_0}(1^\lambda)$
$\mathbf{return}\ (\mathsf{pk}, \mathsf{sk})$

$\mathsf{Decaps}_{\mathsf{sKEM}}(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, (\mathsf{ct}, t))$

$K'_0 \leftarrow \mathsf{Decaps}_{\mathsf{sKEM}_0}(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, (\mathsf{ct}, t))$
$\mathbf{if}\ H'(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K'_0) \neq t:$
$\quad \mathbf{return}\ \perp$
$\mathbf{return}\ H(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K'_0)$

$\mathsf{Encaps}_{\mathsf{sKEM}}(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{B})$

$K_0, \mathsf{ct} \leftarrow\!\!\$\ \mathsf{Encaps}_{\mathsf{sKEM}_0}(\mathsf{pk}_\mathsf{A}, \mathsf{sk}_\mathsf{B})$
$t \leftarrow H'(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K_0)$
$K \leftarrow H(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K_0)$
$\mathbf{return}\ K, (\mathsf{ct}, t)$

Figure 7.16: $\mathsf{T}_{\mathsf{CH}}^{\mathsf{skem}}$ transform for split-KEMs. We assume that pk_B can be derived from sk_B or is contained in it.

7.8.5 Building a UNF-1KCA and IND-1BatchCCA split-KEM

We have proven so far that the modified version of FrodoKEX given above is decaps-OW-CPA and OW-CPA. We show now that any scheme satisfying both these properties can easily be transformed into a UNF-1KCA and IND-1BatchCCA split-KEM in the ROM and QROM. The construction is actually very similar to the T_{CH} transform introduced in Chapter 6 translated to the split-KEM setting. We present it in Figure 7.16. Then, the following theorem states the security guarantees of the resulting split-KEM.

Theorem 7.8.3. *Let* sKEM_0 *be any split-KEM and* $\mathsf{sKEM} := \mathsf{T}_{\mathsf{CH}}(\mathsf{sKEM}_0)$ *be the split-KEM obtained from applying the* T_{CH} *transform (Figure 7.16) to* sKEM_0. *Then, in the ROM, we have that for any efficient* UNF-1KCA *adversary* $\mathscr{A}$, *one can build efficient* $\mathscr{B}$ *and* $\mathscr{C}$ *adversaries s.t.*

$$\mathsf{Adv}^{\mathrm{unf-1kca}}_{\mathsf{sKEM}}(\mathscr{A}) \le \frac{q_{H'}^2+1}{2^s} + (q_H + q_{H'} + 1)\cdot \mathsf{Adv}^{\mathrm{decaps-ow-cpa}}_{\mathsf{sKEM}_0}(\mathscr{C})\,,$$

where q_H *and* $q_{H'}$ *are the number of queries made by* $\mathscr{A}$ *to the random oracles* H *and* H', *respectively, and* s *is the output size of both random oracles. In the QROM, the bound becomes*

$$\mathsf{Adv}^{\mathrm{unf-1kca}}_{\mathsf{sKEM}}(\mathscr{A}) \le \frac{8(q_H+q_{H'})^2}{2^{2s}} + \epsilon + 2(2(q_H+q_{H'})+1)^2\cdot \mathsf{Adv}^{\mathrm{decaps-ow-cpa}}_{\mathsf{sKEM}_0}(\mathscr{B})\,,$$

where $\epsilon := \frac{2}{2^s} + 8\sqrt{2/2^s} + \frac{40e^2(q_{H'}+2)^3+2}{2^s}$.

Proof. For the sake of brevity, we provide the proof in Appendix C. □

Similarly, we have that the $\mathsf{T}^{\mathsf{skem}}_{\mathsf{CH}}$ transform makes an IND-1BatchCCA scheme out of an OW-CPA one, which is stated in the following theorem.

Theorem 7.8.4. *Let* sKEM_0 *be any split-KEM and* $\mathsf{sKEM} := \mathsf{T}_{\mathsf{CH}}(\mathsf{sKEM}_0)$ *be the split-KEM obtained from applying the* T_{CH} *transform (Figure 7.16) to* sKEM_0. *Then, in the ROM, we have that for any efficient* IND-1BatchCCA *adversary* $\mathscr{A}$, *one can build efficient* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}^{\mathrm{ind-1batchcca}}_{\mathsf{sKEM}}(\mathscr{A}) \le \frac{q_{H'}^2+d}{2^s} + 2(q_H + q_{H'} + d)\cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{sKEM}_0}(\mathscr{B})\,,$$

where q_H *and* $q_{H'}$ *are the number of queries made by* $\mathscr{A}$ *to the random oracles* H *and* H', *respectively,* s *is the output size of both random oracles, and* d *is the number of tuples submitted to the IND-1BatchCCA oracle BatchDec. In the QROM, the previous bound becomes*

$$\mathsf{Adv}^{\mathrm{ind-1batchcca}}_{\mathsf{sKEM}}(\mathscr{A}) \le \delta + \epsilon_1 + \epsilon_2 + \epsilon_3 + 2(q_H + d + q_{H'})\sqrt{2\mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{sKEM}_0}(\mathscr{B})}\,,$$

where δ *is the correctness error,* $\epsilon_1 = \frac{40e^2(q_{H'}+d+1)^3+2}{2^s}$, $\epsilon_2 = 8d(d+2q_{H'}+1)\sqrt{2/2^s}$ *and* $\epsilon_3 = \frac{4d}{2^s}$.

Proof. As the proof is nearly identical to the proof of IND-qCCA security of the T_{CH} transform for PKE/KEM (c.f. Theorems 6.4.1 and 6.4.2), we defer it to Appendix D. □

n	$\bar{n}$	q	B	χ	$\lvert t\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert$
1452	8	31751	4	$\mathcal{U}(\{-1,1\})$	64B	21.3KB	72B

Table 7.1: Concrete parameters for our lattice-based split-KEM, where $\mathcal{U}(\{-1,1\})$ denotes the uniform distribution over $\{-1,1\}$. We note that in practice, we do not need to include the whole matrix $\mathbf{A}$ in the public key pk, but rather the seed for the pseudorandom function to generate it (as is the case in this table). The ciphertext ct comprises the original split-KEM ciphertext (8B) and the tag t (64B).

$\bar{n}^2\delta_{\mathsf{corr}}$ (7.2)	δ_{elwe} (7.1)	δ_{cpa} (7.3)
2^{-48}	2^{-46}	$2^{-3.9996}$

Table 7.2: Correctness and security terms.

We call the split-KEM obtained from applying the $\mathsf{T}^{\mathsf{skem}}_{\mathsf{CH}}$ transform on our modified version of Frodo $\mathsf{FrodoKEX+}$.

7.8.6 Concrete instantiation

In Table 7.1 we propose a parameter set for $\mathsf{FrodoKEX+}$ where we aim for 256-bit security before applying the transform and 128-bit (resp. 64-bit) security after the transform assuming 2^{64} random oracle (resp. quantum random oracle) queries. In addition, we give the security terms in Table 7.2. We show in the following how these parameters were computed, where we set $(B,\bar{n}) = (4,8)$.

Correctness error and security loss

One of the main challenges in instantiating our FrodoKEX variant is computing δ_{corr} and δ_{elwe} from Equations 7.2 and 7.1. They are related to the correctness error and the security loss of ELWE. First, we recall that we set χ to be a uniform distribution over the set $\{-1,1\}$. Clearly, it is symmetric around 0 and has standard deviation equal to 1.

Another useful property of this distribution is that a product XY, where $X, Y \leftarrow\$ \chi$, still follows the distribution of χ. Based on this observation, we have

$$\delta_{\mathsf{corr}} = \Pr_{\substack{X_1,\ldots,X_{2n+1}\leftarrow\$\chi \\ E\leftarrow\$\chi}}\left[\left|\sum_{i=1}^{2n+1} X_i + E\right| > \frac{q}{2^{B+2}}\right].$$

We can directly compute this term using Laurent polynomials. Namely, define

$$P(X) := \Pr_{X\leftarrow\$\chi}[X=1]\cdot X + \Pr_{X\leftarrow\$\chi}[X=-1]\cdot X^{-1} = \frac{1}{2}\cdot\left(X + X^{-1}\right).$$

Then, using the convolution properties, we observe that the probability of $X_1 + \ldots + X_{2n+2} = k$, for some $-2n-2 \le k \le 2n+2$, is equal to the k-th coefficient of the polynomial $P(X)^{2n+2}$. Hence, we calculate δ_{corr} by computing $P(X)^{2n+2}$ and summing all the k-th coefficients, such that $2n+2 \ge |k| > \frac{q}{2^{B+2}}$.

We now turn into computing δ_{elwe}. The first step is the analysis of the following random variable $\mathbf{v} = \frac{1}{2} \cdot (e-d) \cdot \mathbf{z}$, where $e, d \leftarrow\!\!\$\, \chi$ and $\mathbf{z} \leftarrow\!\!\$\, \chi^{\bar{n}}$. We denote this distribution as $\mathcal{V}$. By simple calculation we get:

$$\Pr[\mathbf{v} = \mathbf{a}] = \Pr\left[\frac{1}{2} \cdot (e-d) \cdot \mathbf{z} = \mathbf{a}\right] = \begin{cases} \frac{1}{2} \text{ if } \mathbf{a} = \mathbf{0} \\ \frac{1}{2^{\bar{n}+1}} \text{ if } \mathbf{a} \in \{-1, 1\}^{\bar{n}} \\ 0 \text{ otherwise} \end{cases} .$$

Then, the multivariate Laurent polynomial corresponding to $\mathbf{v}$ has an elegant form:

$$P(X_1, \ldots, X_{\bar{n}}) = \frac{1}{2} + \frac{1}{2^{\bar{n}+1}} \prod_{i=1}^{\bar{n}} (X_i + X_i^{-1}).$$

As before, we observe that δ_{elwe} is the probability that for $\mathbf{v}_1, \ldots, \mathbf{v}_{n+m} \leftarrow\!\!\$\, \mathcal{V}$,

$$2 \cdot (\mathbf{v}_1 + \ldots + \mathbf{v}_{n+m}) = \mathbf{0} \pmod{q} \iff \mathbf{v}_1 + \ldots + \mathbf{v}_{n+m} = \mathbf{0}^{7}.$$

In terms of the newly defined Laurent polynomials, δ_{elwe} is the constant coefficient of:

$$P(X_1, \ldots, X_{\bar{n}})^{n+m} = \textstyle\sum_{j=0}^{n+m} \binom{n+m}{j} \frac{1}{2^{n+m-j}} \cdot \frac{1}{2^{(\bar{n}+1)j}} \cdot \prod_{i=1}^{\bar{n}} (X_i + X_i^{-1})^j.$$

We now look at the constant coefficient of each of the $n+m+1$ terms of the sum. The first observation is that

$$(X_i + X_i^{-1})^j = \sum_{k=0}^{j} \binom{j}{k} X_i^{(j-k)} X_i^{-k} = \sum_{k=0}^{j} \binom{j}{k} X_i^{(j-2k)}.$$

Hence, the constant coefficient of the expression above is 0 if j is odd, and $\binom{j}{j/2}$ when j is even. Consequently, the constant coefficient of $\prod_{i=1}^{\bar{n}} (X_i + X_i^{-1})^j$ is either 0, for odd j, or $\binom{j}{j/2}^{\bar{n}}$ for even j. Hence, we conclude that

$$\delta_{\mathsf{elwe}} = \sum_{j \text{ even}} \binom{n+m}{j} \frac{1}{2^{n+m-j}} \cdot \frac{1}{2^{(\bar{n}+1)j}} \cdot \binom{j}{\frac{j}{2}}^{\bar{n}}$$

which can then be computed efficiently for our parameters. Finally, δ_{cpa} can be straightforwardly computed for small primes, such as $\approx 2^{15}$. In our case, we make sure that $\delta_{\mathsf{cpa}}^{\bar{n}^2} \approx 2^{-256}$ for the decaps-OW-CPA proof.

[7] This holds as long as $n+m < q/2$ since then no modulo overflow occurs.

Hardness of LWE

We measure the hardness following the methodology used for the original FrodoKEX [Bos+16] for fair comparison, and refer to it for more details on the attacks. Here, the main bottleneck of setting the parameters is the reduction loss between ELWE and plain LWE. Taking this into account for the parameters proposed above, we aim for 307-bit classical LWE security.

We consider the primal and dual BKZ attacks [SE94; CN11]. As a subroutine, the BKZ algorithm with block-size b uses an algorithm for the shortest vector problem (SVP) in lattices of dimension b. As in Frodo [Bos+16], for precautionary purposes we only count the cost of one such call (even though in practice it will run the SVP sub-algorithm polynomially many times). The lower-bound on the time complexity of one call is given by about $b2^{cb}$ CPU cycles, where $c \approx 0.292$ for classical attacks, and $c \approx 0.265$ for quantum attacks (see Laarhoven [Laa16, Section 14.2.10]). For 307-bit classical security, this corresponds to the block size being 1018, and the root Hermite factor being ≈ 1.0020 (in the quantum setting these parameters correspond to 279 bits of security). Further, we estimate the hardness of LWE against known attacks using the LWE estimator by Albrecht et al. [APS15]. Namely, we run the estimator under both "sieving" and "enumeration", and set the final root Hermite factor δ as the largest root Hermite factor returned by the program.

7.9 Benchmarks, Comparison and Discussion

Hereafter, we refer to the scheme by Brendel et al. [Bre+22] as SPQR, and we refer to the deniable (i.e. with ring signatures) scheme by Hashimoto et al. [Has+22] as HKKP.

7.9.1 Benchmarks

Security of non-standard primitive. As K-Waay, SPQR and HKKP can each be implemented using, except for a single primitive in each case, only (soon to be) standardised primitives, we wish to compare the security of the non-standard primitives. In the case of K-Waay it is a split-KEM, here implemented using a variant of FrodoKEX passed through the T_{CH} transform (that we call FrodoKEX+), and in the case of both HKKP and SPQR it is a ring signature (RS), or a designated verifier signature (DVS) derived from a ring signature. The authors of both SPQR and HKKP proposed possible implementations for the RS without picking one in particular. The most efficient one for a ring of size 2 that has an existing C implementation is Raptor [LAZ19] which we use for the benchmarks below. Other candidates would be Falafl [BKP20] or DualRing-LB [Yue+21].

We present in Table 7.3 a summary of the security claims, approximate leading factor in the bounds in the (Q)ROM, and assumptions for these non-standard primitives. We note that none of these primitives are proven secure in the standard model and all are based on lattices.

First, we note that parameters for these RS schemes are picked *before* the reduction in the

Scheme	Cl. (C)	Cl. (Q)	ROM bound	QROM bound	Assumption
FrodoKEX+	128	64	$(q_H + d)/2^{192}$	$(q_H + d)/2^{128}$	LWE
Raptor [LAZ19]	114	103	?	✗	NTRU
DualRing-LB [Yue+21]	(128)	(64)	?	✗	MSIS, MLWE
Falafl [BKP20]	128	64	?	✗	MSIS, MLWE

Table 7.3: Security comparison between FrodoKEX+and several post-quantum RS. 'Cl.' stands for claimed number of security bits. DualRing-LB's authors do not seem to make a clear security claim, we thus assume NIST level I. '?' indicates that no bound is explicitly given for the security, '✗' indicates that no proof is provided in the QROM.

ROM. That is, a primitive P based on lattices is built, parameters are chosen such that P satisfies the security claim, then P is used to build a RS in the ROM, which incurs a loss factor that usually depends on the number of queries to the random oracle q_H. In particular, it is common to have at least a q_H factor in the security bound (e.g. if the adversary can make 2^{64} queries to the RO, the security level is reduced by 64 bits). Therefore, the claimed security level does not match the *provable* security level. In the QROM, the security loss is usually greater: square root and q_H^2 or q_H^3 losses are quite common, however these schemes are not proven secure in this model.

We chose the opposite approach in designing a split-KEM with a conservative assumption (i.e., plain LWE) and parameters. Therefore, FrodoKEX+ with our proposed parameters achieves 128 (resp. 64) bits of classical (resp. quantum) security *after* the (Q)ROM proof. We put the (approximate) highest terms of both the ROM and QROM security bounds in Table 7.3. These satisfy our security claims as long as $q_H + d \leq 2^{64}$, where d is the number of public key/ciphertext tuples allowed in the IND-1BatchCCA game. We note that in K-Waay, d corresponds to the number of distinct users trying to communicate with an offline receiver after all prekeys have run out, thus should typically be small.

The reason behind the approximations and lack of QROM proofs for PQ ring signatures is likely the youth of the field and the speed at which it is evolving. Still, we believe it is worth noting as it makes any comparison between our protocol and previous ones quite difficult.

Benchmarking. The protocols we benchmarked are: our own implementation of the current X3DH protocol, a witness protocol made only with PQ KEMs, and a signature scheme similar to the *non-deniable* variant of HKKP, Brendel et al.'s [Bre+22] construction SPQR using PQ KEMs, a signature scheme and a DVS, Hashimoto et al.'s [Has+22] construction HKKP using PQ KEMs, a signature scheme and a RS, and our scheme K-Waay using PQ KEMs, a signature scheme and FrodoKEX+ as the split-KEM.

We picked Kyber512 as the KEM, both Falcon-512 and Dilithium2 as signatures, and Raptor as the ring signature. We implemented both HKKP and SPQR with signed prekeys as is the case in Signal's implementation of X3DH. That is, a PQ signature key pair is part of the long-term

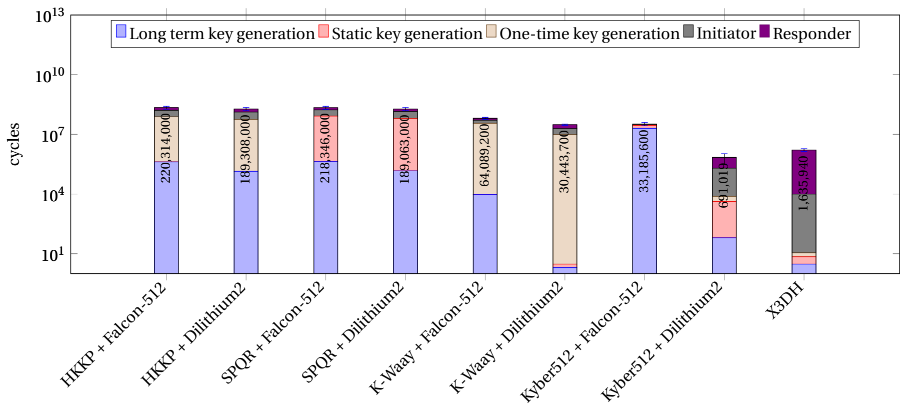


Figure 7.17: Speed benchmark for X3DH protocols

key, and ephemeral keys uploaded to server are signed with it. Note that this is make explicit in K-Waay as the ephemeral keys are signed with the long-term one. The authors of HKKP show that this is not necessary in their protocol, however not doing so weakens perfect forward secrecy.

We built the different protocols in C using the `liboqs` library[8] for Kyber, Falcon, and Dilithium, the Raptor implementation provided by the authors[9], and a modified version of the `lwe-frodo` library[10] with scaled parameters to properly simulate FrodoKEX+. More precisely, the modulus was set to the first power of 2 larger than the modulus in FrodoKEX+, the addition of the noise during decapsulation was also added, and the noise distributions were modified to match the ones of FrodoKEX+. We did not optimise the scheme in any way (e.g. by using AVX instructions or parallelisation) and we leave this as future work. For the sake of completeness, we also provide a reference implementation of FrodoKEX+ in Rust[11] for the interested reader. All benchmarks were run on a virtual machine running Ubuntu 22.04 with 2 cores of an Intel i7-9750H running at 2.60GHz and 4GB of RAM allocated.

Speed. For the speed benchmark, we measured how many cycles each protocol takes in one execution. We summarise our results in a logarithmic graph on Figure 7.17 (note that the internal division of the bars is linear).

Depending on the choice of KEM and signature scheme, our protocol K-Waay is between 3 and 6 times faster than the previous proposals even with our relatively conservative parameter choice. In our protocol K-Waay using Dilithium2, most cycles are spent in the ephemeral key generation, while using Falcon makes the static key generation as expensive as the ephemeral key one. Overall, one can see that Falcon, while more compact than Dilithium2, has a great

[8] https://github.com/open-quantum-safe/liboqs
[9] https://github.com/zhenfeizhang/raptor
[10] https://github.com/lwe-frodo/lwe-frodo
[11] https://github.com/lehugueni/frodokexp-rust

Scheme	|lpk|	|prek|	|ct|
K-Waay + Dilithium	2112	24520	1632
K-Waay + Falcon	1697	22790	1632
HKKP [Has+22]	1700	1700	4056
HKKP [Has+22] + Dilithium2	3012	4120	4056
HKKP [Has+22] + Falcon	2597	2390	4056
SPQR [Bre+22]	3400	1632	4824
SPQR [Bre+22] + Dilithium2	4712	4052	4824
SPQR [Bre+22] + Falcon	4297	2322	4824

Table 7.4: Size comparison in bytes between K-Waay instantiated with FrodoKEX+, HKKP [Has+22] and SPQR [Bre+22]. We also computed the sizes for both HKKP and SPQR implemented with signed prekey bundles.

impact on efficiency. For instance, K-Waay with Dilithium2 is faster than the non-deniable scheme using Kyber and Falcon.

Apart from Falcon, we see that the most time-consuming primitives are the non-standard ones, i.e., ring signatures and split-KEM. Hence, we see that the KEM+SIG protocol (HKKP's baseline proposal that does not provide deniability) performs even better than X3DH, which shows once again that lattice-based schemes can be faster than their classical counterparts. Interestingly, X3DH is the only construction that spends more time in sending and receiving than generating keys. Finally, we note that our protocol's Send and Receive (i.e. BatchReceive with a single input message) procedures are very fast.

Data size. In Table 7.4, we provide for each scheme the size of the long-term keys, the prekeys (output by Init in our DAKE syntax), and the ciphertext output by the sender. We computed for both HKKP and SPQR the size with and without long-term signatures. We see that K-Waay compares well in terms of long-term public key and ciphertext size as both are smaller than in HKKP and SPQR with signed prekeys. However, the prekeys are much larger as one could expect from a LWE-based scheme and due to our conservative choice of parameters.

7.9.2 Advantages, limitations, and discussion

Running out of ephemeral keys. The main disadvantage of our protocol is that running out of ephemeral keys requires the receiver to abort if *any* of the sessions that used the same prekey is bogus. If this happens, then a malicious party could mount some kind of denial of service (DoS) attack against the user that was offline for too long by sending a bogus split-KEM ciphertext. There is an obvious trade-off between the risk of such an attack happening and the number of ephemeral keys uploaded on the server, thus the storage required on the server. We leave the analysis and the mitigation of such a threat as future work, but we believe that if a reasonable amount of prekeys are uploaded, creating fake accounts is difficult (e.g. by requiring a phone number as in Signal), and/or users are online often enough, such an attack would be difficult to mount. Furthermore, several practical mitigations are possible. For instance, if the receiver (i.e. the victim) received a bogus ciphertext among the n ciphertexts sent for the same prekey while offline, they can restart K-Waay with the n parties but as the initiator, which will probably succeed. The victim could also send n new prekeys to the n initiators directly, making sure the protocol will succeed at the next iteration. This would make the attack less useful as it could only *delay* communication and not prevent it.

We also think it is worth mentioning that the trick we propose might be easy to mis-implement. In particular, it is crucial that no information about which split-KEM ciphertext failed leaks if such a situation occurs. That is, all precautions should be taken such that such leakage via side-channels is prevented.

split-KEM instead of ring signatures. The first advantage of using our protocol over existing ones is the use of an ephemeral primitive instead of a long-term one, the former being often more efficient as the security requirements are less strict. In addition, the use of a primitive similar to a post-quantum KEM allows us to leverage the extensive literature on the topic and existing safe/optimised implementations. This also gives good security guarantees as post-quantum KEMs have been heavily scrutinised as part of the NIST standardisation process. For example, as mentioned above, our proposed lattice-based implementation is based on a key-exchange variant of FrodoKEM, which is itself the PQ KEM recommended by the German Federal Office for Information Security (BSI) [Inf23]. Overall we think that a split-KEM such as FrodoKEX+ is more mature and closer to being usable in practice than ring signatures.

On the necessity of modifying FrodoKEX. Currently, our split-KEM differs from the original FrodoKEX in two aspects: (i) the modulus for our construction has to be prime in order for our reduction from Extended-LWE to LWE to hold, and (ii) we have to introduce additional masking terms to prove decaps-OW-CPA security. However, we believe that both changes are artefacts of the security proofs, and the original FrodoKEX split-KEM should be (up to a reasonable security loss) deniable.

There are alternative reduction techniques from Extended-LWE to LWE in the literature [Bou+21; Bra+13], which do not rely on having an odd modulus at the cost of using discrete Gaussian error distributions with large parameters. It is thus an interesting research problem to efficiently reduce Extended-LWE to LWE for even modulus with small reduction loss.

As for our second main modification, it is unclear how to argue decaps-OW-CPA security without the additional masking terms.

Deniability. While the signature on the ephemeral public keys might give the impression that our protocol is less deniable than X3DH or previous PQ alternatives, this is actually not the case. The reason is that prekey bundles in these protocols are signed as well, but this detail is abstracted away in the analysis (i.e. it is assumed that all parties have received and authenticated all public keys before the protocol actually starts). While this kind of analysis allows for strong deniability claims, in practice these protocols do not satisfy something stronger than some kind of peer-deniability. The exception is the ring signature based variant by Hashimoto et al. [Has+22], where the prekey bundle is not necessarily signed. However, in this variant, the authors can only prove the security of their protocol in a weaker model (i.e. it satisfies a weaker notion of forward secrecy). Overall, if deniability should not come at the price of security, peer-deniability seems like the best notion one can achieve in these DAKEs.

We wished to provide a transparent model for peer-deniability, where the upload of signed ephemeral keys is made explicit. We also strengthen the deniability definition of Brendel et al. [Bre+22] by allowing the exposure of one of the parties (i.e. the receiving one, which would be the malicious party trying to frame the sender). While our protocol satisfies our stronger (in terms of key exposure) notion of deniability, we believe both previous PQ X3DH alternatives satisfy it as well. Indeed, in these schemes, the ephemeral keys are KEM and RS keys only, which are deniable. Hence, exposing these should not harm deniability.

Hashimoto et al. [Has+22] consider a strong notion of deniability where the adversary is malicious (i.e. can arbitrarily deviate from the protocol) and show how to modify HKKP s.t. it is secure against such a threat. However, such deniability comes at the expense of NIZKs, which are complex, expensive and are not always proven secure in the QROM when random oracles are used. Moreover, as in other deniable systems against malicious adversaries, non-falsifiable assumptions (i.e. knowledge-type assumptions) are required to prove the security. In addition, it seems difficult to defend against adversaries actively trying to frame a given user in messaging in practice [GPA19; CCH23]; for example, an adversary could also simply ask questions that would identify the victim with good probability. Because of these reasons, we do not consider such a notion of deniability here.

To contextualise our results, we remark here that cryptographic deniability, which is targeted by this work and all previous work on deniable X3DH key exchange, translates to deniability on a *system* level only if the application preserves deniability. For example, we observe in another work not included in this thesis [CCH23] that Signal as currently deployed does not provide this kind of 'practical' deniability for ordinary users. Suppose Bob is trying to frame Alice and hands over their phone, that contains a transcript of communication between Alice and Bob, to a judge. Because Signal authenticates users (either directly or indirectly through Signal sealed sender [Lun18]), unless Bob was able to modify their phone (which depends on the technical expertise of Bob), the judge can deduce that the conversation plausibly took place as in the transcript, regardless of the cryptographic protocols employed underneath.

An optimisation. As presented in Section 7.7, the K-Waay protocol generates a signature for each ephemeral public key uploaded. This can easily be optimised by signing the whole prekey bundle containing several ephemeral keys. This way, the server needs to store only one PQ signature for each user. The downside is that now each user needs to download the whole bundle to verify the signature. This offers a trade-off between data stored at the server and sent to clients.

8 Conclusion

We saw in this thesis the challenges that quantum computing poses to cryptography. In particular, it seems that the design space of efficient post-quantum public-key encryption schemes is somewhat narrow, as all candidates proposed to NIST for standardisation share a similar structure. More precisely, they use a FO-like transform to guarantee IND-CCA security. We studied in this dissertation how these transforms impact security and efficiency, and why it is hard to get rid of them. We also explored the realm of post-quantum protocols and how primitives weaker than IND-CCA KEMs can be employed to construct them. In this last chapter, we briefly summarise the content of this thesis and discuss further directions that could be explored.

We started by studying misuse attacks against several NIST candidates in Chapter 3. To no surprise, the schemes based on lattices and codes in the Hamming metric were no more resistant to KR-PCA than the candidates we previously studied [Băe+19]. That is, a few thousand queries to the plaintext-checking oracle are needed to recover the key. These results have been most impactful in the field of side-channel attacks, with several of them building upon our techniques (e.g. [Uen+22; Xag+21]). Overall, numerous plaintext-checking attacks have been proposed and this topic of research has been extensively studied since the publication of our results (e.g. [Qin+21; Raj+23; Azo+22]). A more interesting direction for future work would be to analyse further the resistance of rank-based schemes to KR-PCA, as we merely showed that the learning problem was hard but other types of attacks might be more efficient.

In Chapter 4, we showed how to generalise the concept of FO-like transforms by introducing FO-like combiners that take several PKEs as input and output a hybrid IND-CCA KEM. We also analysed how random oracles can be combined in our construction, as for practical reasons hash functions are often nested in FO implementations, which sometimes creates difficulties in security proofs (e.g. [GMP22]). Finally, we studied how hybrid schemes based on the NIST round 2 proposals would perform. We only scratched the surface there, and it would be interesting to benchmark these hybrid schemes and others in real-life scenarios. Overall, we believe that the devastating cryptanalyses of SIDH and Rainbow highlight the relevance of hybrid cryptography and combiners.

In the subsequent chapter, we translated Gertner et al.'s impossibility result to the post-quantum setting. That is, we showed that there is no shielding black-box reduction from IND-CCA to IND-CPA, even when the reduction algorithm and the adversaries are quantum. Going from there, the obvious open question is to extend such an impossibility result to the general case, i.e. to show that there is no black-box reduction from IND-CCA to IND-CPA in the standard model. This problem has been around for decades now and solving it would be a major result in theoretical cryptography.

Next, in Chapter 6, we showed that there are very efficient transforms that take a CPA-secure PKE and output an IND-qCCA KEM. Compared to FO-induced KEMs, those output by our transform can be twice as efficient at decryption due to the absence of re-encryption. We did not provide a QROM proof for the second transform we introduced (i.e. T_{H}), but since the publication of the corresponding paper such a result was produced by Jiang et al. [JMZ23] for the IND-1CCA case. A proof for the generic IND-qCCA case is still missing. One could also aim at reducing the loss in the security bounds. In the second part of the same chapter, we proved that a CPA-secure KEM can be used in the post-quantum variant of TLS 1.3. Our demonstration is only valid in the ROM and a proof in the QROM is left as future work. In addition, the security loss induced by our proof is huge, thus reducing it to make the result practically relevant would be of interest.

Finally, in Chapter 7, we introduced K-Waay, the first PQ asynchronous DAKE that does not rely on ring signatures. Instead, we revisited the notion of split-KEM [Bre+21] by augmenting it with additional security properties and we used it as the main building block of our protocol. We then proposed an instantiation of the split-KEM using a modified version of the Frodo key-exchange [Bos+16]. In order to prove our split-KEM deniable, we had to rely on the Extended-LWE problem, which we showed to be as hard as plain LWE up to a loss factor. To make the latter as small as possible, our construction uses an odd modulus, which impacts the efficiency. Hence, providing a more efficient reduction for even modulus would be interesting future work. In practice, the most efficient LWE attacks do not consider the structure of the modulus, so intuitively this should translate to the Extended-LWE setting. More generally, proving the original FrodoKEX scheme provides deniability and decaps-OW-CPA security is left as an open problem.

The main drawback of our split-KEM FrodoKEX+ is that the public keys are quite large. Therefore, another interesting line of work would be to build a more efficient split-KEM based on, e.g., structured lattices. Indeed, the problem of Ring/Module-LWE with hints (equivalent to Extended-LWE in these structures) has already been analysed [Bou+21; Mer+22]. However, the problem is that in our case the hints are composed of the multiplication of both secret keys, informally. Thus, in the ring setting that hint would be at least a single polynomial, which would contain $O(\lambda)$ coefficients, where λ is the security parameter. In turn, this would probably make the reduction loss much larger.

On a more practical note, K-Waay could probably be optimised in several ways. First, the parameters we propose for the split-KEM are very conservative (e.g. we guarantee more than 200 bits of security for deniability), therefore one could propose smaller parameters for better memory/storage efficiency. Also, it would be informative to benchmark K-Waay and other protocols in real-life conditions, and to implement additional ring signature schemes to get a more complete comparison. One could also try to find other applications of IND-1BatchCCA and of the type of transforms introduced and used throughout Chapters 6 and 7.

A Hashed DH is IND-1CCA

We prove here that Diffie-Hellman with hashed key as used in TLS 1.3 is a IND-1CCA KEM in the ROM, if the CDH assumption holds.

Theorem A.0.1. *Let* DH *be the Hashed Diffie-Hellman key-exchange modelled as a KEM,* $\mathbb{Z}_p^*$ *be the associated group for a safe prime* p*, and* g *be a generator of a subgroup* $\mathbb{G}$ *of* $\mathbb{Z}_p^*$ *s.t. the order of* $\mathbb{G}$ *is prime. In addition, let the hash function* H *be modelled as a RO. Then, for all ppt adversaries* $\mathscr{A}$ *making at most* q_H *queries to the RO, there exists a CDH solver* $\mathscr{B}$ *s.t.*

$$\mathsf{Adv}_{\mathsf{DH}}^{\text{ind-1cca}}(\mathscr{A}) \leq q_H(q_H+1) \cdot \mathsf{Adv}_{\mathbb{G}}^{\text{cdh}}(\mathscr{B}) ,$$

where $\mathscr{B}$ *runs approximately in the same time as* $\mathscr{A}$*.*

Proof. The idea of the proof is similar to the previous ones. First, we notice that (contrary to PQ schemes), the only ciphertext that decrypts to the challenge key in DH in a group of prime order is the challenge ciphertext. Since the latter cannot be queried to the decapsulation oracle, in the IND-1CCA game the adversary can only recover one RO value associated to another key. Since the RO is perfectly hiding, this does not give much information to the adversary. Then, in the CDH reduction $\mathscr{B}$, one can simulate the decapsulation oracle for $\mathscr{A}$ by always returning a random value. The only issue is if the corresponding value matches a query to the random oracle. However, as this happens at most once, $\mathscr{B}$ can guess whether it will happen and at which query (e.g. by sampling a value i in $\{0,\ldots,q_H\}$). If the guess is correct the simulation is perfect. Finally, $\mathscr{A}$ can only distinguish the real and random keys if it queries the CDH solution to the RO.

Formally, we proceed with a short sequence of games presented in Figure A.1. We assume w.l.o.g. that each query $\mathscr{A}$ makes to the RO H is unique.

$\underline{\Gamma^0:}$ This is the IND-1CCA game with DH expressed as a KEM. I.e. we identify the public key with g^a, the secret key with a, the challenge ciphertext with g^b, and the key as $H(g^{ab})$. Also, we assume the decapsulation oracle only accepts elements of the subgroup $\mathbb{G}$ as inputs.

$\Gamma^{0-2}(\mathscr{A})$

$(g^a, a) \leftarrow\$ \mathsf{Gen}$
$\beta \leftarrow\$ \{0,1\}$
$(g^b, g^{ab}) \leftarrow\$ \mathsf{Encaps}(g^a)$
$K_0 \leftarrow H(g^{ab})$
$K_1 \leftarrow\$ \{0,1\}^n$
$\beta' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}}(g^a, g^b, K_\beta)$
if query **: abort** $/\!/ \Gamma^2$
return $1_{\beta'=\beta}$

Oracle $\mathscr{O}^{\mathsf{Dec}}(g^x \in \mathbb{G})$

if $g^x = g^b$ **: abort**
if more than 1 query :
  return $\perp$
$\sigma' \leftarrow (g^x)^a$
if $\sigma' = g^{ab}$ **: abort** $/\!/ \Gamma^1\text{-}\Gamma^2$
return $H(\sigma')$

$H(\sigma)$

if $\sigma = g^{ab}$: $\mathsf{query} \leftarrow$ **true** $/\!/ \Gamma^2$
if $\exists h$ s.t. $(\sigma, h) \in \mathscr{L}_H$:
  return h
$h \leftarrow\$ \{0,1\}^n$
$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{(\sigma, h)\}$
return h

Figure A.1: Sequence of games for the proof of Thorem A.0.1.

$\mathscr{B}(g, g^a, g^b)$

$K \leftarrow\$ \mathscr{K}$
$K_{\mathsf{Dec}} \leftarrow \perp$
$\mathscr{L}_H \leftarrow \emptyset$
$i \leftarrow\$ \{0, \ldots, q_H + 1\}$
$b' \leftarrow \mathscr{A}^{\mathscr{O}^{\mathsf{Dec}}, H}(g^a, g^b, K)$
$\sigma \leftarrow\$ \mathscr{L}_H$
return σ

$H(\sigma)$

$h \leftarrow\$ \{0,1\}^n$
if i-th query :
  if $K_{\mathsf{Dec}} \neq \perp$: $h \leftarrow K_{\mathsf{Dec}}$
  else : $K_{\mathsf{Dec}} \leftarrow h$
$\mathscr{L}_H \leftarrow \mathscr{L}_H \cup \{\sigma\}$
return h

Oracle $\mathscr{O}^{\mathsf{Dec}}(g^x \in \mathbb{G})$

if more than 1 query :
  return $\perp$
if $K_{\mathsf{Dec}} = \perp$:
  $K_{\mathsf{Dec}} \leftarrow\$ \{0,1\}^n$
return K_{Dec}

Figure A.2: CDH adversary $\mathscr{B}$ for the proof of Theorem A.0.1. We assume all queries to H are fresh (e.g. we do not check whether an identical previous query was made in H).

Note that w.l.o.g the game aborts if the adversary queries the challenge ciphertext to the decapsulation oracle.

$\underline{\Gamma^1}$: This is the same as Γ^0, except we abort if on input g^x, the decapsulation oracle computes g^{ax} s.t. $g^{ax} = g^{ab}$. Now, since $\mathbb{G}$ is a subgroup of prime order of $\mathbb{Z}_p^*$, this happens iff $x = b \Rightarrow g^x = g^b$. Since decapsulation queries on the challenge ciphertext g^b are disallowed, Γ^0 and Γ^1 are identical.

$\underline{\Gamma^2}$: As in other proofs, we abort if the challenge seed g^{ab} is queried by the adversary to the RO.

We call this event query. We have

$$|\Pr[\Gamma^1 \Rightarrow 1] - \Pr[\Gamma^2 \Rightarrow 1]| \leq \Pr[\mathsf{query}] \ .$$

We give in Figure A.2 a CDH adversary $\mathscr{B}$ s.t. $\mathscr{B}$ wins with probability at least $\frac{1}{q_H+1}\Pr[\mathsf{query}]$. Note that in Γ^2, as long as query does not happen, the decapsulation oracle and the random oracle H always return fresh values sampled uniformly at random unless:

1. The decapsulation oracle returns $H(g^{ab})$. However, by the condition enforced since Γ^1, this cannot happen.
2. The decapsulation oracle returns $H(g^{ax})$ for some x, and $H(g^{ax})$ is later queried by $\mathscr{A}$, or the other way around. Let i be s.t. $H(g^{ax})$ was the i-th query made to H by the adversary and let $i = 0$ if no such case happen. If $i > 0$, then the i-th query to H must return the same value as the result of the decapsulation oracle. In our reduction, we let $\mathscr{B}$ guess i in advance and thus the simulation is perfect with probability $\frac{1}{q_H+1}$.

Hence, if $\mathscr{B}$ guessed the correct i, the simulation of game Γ^2 is perfect and if query happens, $\mathscr{B}$ can recover g^{ab} in the list of queries made to H. However, as it cannot check which value is correct, it outputs a random query made to H and succeeds with probability $\frac{1}{q_H}$. Overall, we have

$$\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathscr{B}) \geq \frac{1}{q_H(q_H+1)}\Pr[\mathsf{query}] \ .$$

Collecting the probabilities concludes the proof. □

Remark. In the proof, for simplicity, we used the fact that DH is performed in a subgroup of prime order. We note that it is always the case in TLS 1.3 (the list of supported groups is given in RFC 7919 [Gil16]).

B Proof of Theorem 7.8.1

Proof. We prove the statement by introducing a sequence of LWE-type games Γ_i.

Game Γ_1: This is the standard $\mathsf{ELWE}_{n,m,\bar{n},\chi,q}$ game. The adversary $\mathscr{A}$ wins this game with probability $\Pr[\Gamma_1]$.

Game Γ_2: Here, we consider the ELWE-type game where the secret vector is uniformly random. Namely, the challenger samples the public $\mathbf{A} \leftarrow\!\$\; \mathbb{Z}_q^{(n+m)\times m}$, secret $\mathbf{s} \leftarrow\!\$\; \mathbb{Z}_q^m$, error $\mathbf{e} \leftarrow\!\$\; \mathbb{Z}_q^{n+m}$ as well as the hint matrix $\mathbf{Z} \leftarrow\!\$\; \chi^{\bar{n}\times(n+m)}$. Then it flips a bit $b \leftarrow\!\$\; \{0,1\}$. If $b = 0$ then the challenger computes

$$\mathbf{t} := \mathbf{A}\mathbf{s} + \mathbf{e}$$

and otherwise it samples $\mathbf{t} \leftarrow\!\$\; \mathbb{Z}_q^{n+m}$. The challenger outputs $(\mathbf{A}, \mathbf{t}, \mathbf{Z}, \mathbf{Z}\mathbf{e})$.

Lemma B.0.1. *For every efficient adversary $\mathscr{A}$, there is an efficient adversary $\mathscr{B}$ such that* $\Pr[\Gamma^2] \geq \Pr[\Gamma^1] - \mathsf{negl}(n)$.

Proof. The reduction follows similarly as in the one by Applebaum et al. [App+09]. Suppose the algorithm $\mathscr{B}$ is given a tuple $(\mathbf{A}, \mathbf{t}, \mathbf{Z}, \mathbf{h})$ from Γ_2. With probability at most $1/q^{(n+m)-m-1} \leq 1/q^{n-1}$, the matrix $\mathbf{A}$ is not full-rank. Let us exclude that case and assume without loss of generality that we can write

$$\mathbf{A} := \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}, \quad \mathbf{Z} := \begin{bmatrix} \mathbf{Z}_0 & \mathbf{Z}_1 \end{bmatrix}, \quad \text{and} \quad \mathbf{t} := \begin{bmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \end{bmatrix}$$

where $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ and the matrix $\mathbf{A}_0 \in \mathbb{Z}_q^{m\times m}$, which contains the first m rows of $\mathbf{A}$, is invertible. Thus, define $\mathbf{A}' := \mathbf{A}_1\mathbf{A}_0^{-1} \in \mathbb{Z}_q^{n\times m}$, and $\mathbf{t}' := \mathbf{A}'\mathbf{t}_0 - \mathbf{t}_1 \in \mathbb{Z}_q^n$. Then, it runs $\mathscr{A}$ on input

$$\left(\mathbf{A}', \mathbf{t}', \mathbf{Z}_0, -\mathbf{Z}_1, \mathbf{h}\right)$$

and returns what $\mathscr{A}$ outputs.

Suppose that $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{s} \in \mathbb{Z}_q^m$ and $\mathbf{e} := (\mathbf{e}_0, \mathbf{e}_1) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^n$. Then

$$\mathbf{t}' = \mathbf{A}'\mathbf{t}_0 - \mathbf{t}_1 = \mathbf{A}_1\mathbf{A}_0^{-1}(\mathbf{A}_0\mathbf{s} + \mathbf{e}_0) - (\mathbf{A}_1\mathbf{s} + \mathbf{e}_1) = \mathbf{A}'\mathbf{e}_0 - \mathbf{e}_1 \ ,$$

which is a valid LWE instance since χ is symmetric around 0. Also, if $\mathbf{A}$ is uniformly random among all nonsingular matrices, then $\mathbf{A}'$ and $\mathbf{B}'$ are statistically close to uniformly random matrices over $\mathbb{Z}_q$. As for the hints, note that

$$\mathbf{h} = \mathbf{Z}_0\mathbf{e}_0 + \mathbf{Z}_1\mathbf{e}_1 = \mathbf{Z}_0\mathbf{e}_0 + (-\mathbf{Z}_1)(-\mathbf{e}_1),$$

so $\mathbf{h}$ is a well-formed hint for Γ_1.

On the other hand, if $\mathbf{t}$ is uniformly random, then so is $\mathbf{t}'$. It can be argued similarly as before that all the other components follow the distribution for $b = 1$. □

Game Γ_3: We consider the knapsack version of ELWE. Here, the challenger samples the public $\mathbf{G} :=\leftarrow\$ \mathbb{Z}_q^{n\times(n+m)}$, secret $\mathbf{e} \leftarrow\$ \mathbb{Z}_q^{n+m}$ and the hint matrix $\mathbf{Z} \leftarrow\$ \chi^{\bar{n}\times(n+m)}$. Then it flips a bit $b \leftarrow\$ \{0,1\}$. If $b = 0$ then the challenger computes $\mathbf{t} := \mathbf{G}\mathbf{e}$, and otherwise it samples $\mathbf{t} \leftarrow\$ \mathbb{Z}_q^n$. Finally, the challenger outputs $(\mathbf{G}, \mathbf{t}, \mathbf{Z}, \mathbf{Z}\mathbf{e})$.

Lemma B.0.2. *For every efficient adversary $\mathscr{A}$, there is an efficient adversary $\mathscr{B}$ such that* $\Pr[\Gamma^3] \geq \Pr[\Gamma^2] - \mathsf{negl}(n)$.

Proof. The reduction is similar to the proof of Micciancio and Mol [MM11, Lemma 4.9]. Suppose the algorithm $\mathscr{B}$ is given a tuple $(\mathbf{G}, \mathbf{t}, \mathbf{Z}, \mathbf{h})$ from Γ_3. Then, $\mathscr{B}$ can construct a randomised matrix $\mathbf{A} \in \mathbb{Z}_q^{(n+m)\times m}$ whose columns generate the kernel of $\mathbf{G}$. In particular, if $\mathbf{G}$ is uniformly random, then so are $(\mathbf{A}, \mathbf{B})$, up to the constraint that they are nonsingular. Then, $\mathscr{B}$ computes any solution $\mathbf{r}$ such that $\mathbf{G}\mathbf{r} = \mathbf{t}$. Finally, it samples a uniformly random $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^m$ and runs $\mathscr{A}$ on input

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{r}, \mathbf{Z}, \mathbf{h})$$

and returns what $\mathscr{A}$ outputs.

Suppose that $\mathbf{G}\mathbf{e} = \mathbf{t} = \mathbf{G}\mathbf{r}$. By definition of the matrix $\mathbf{A}$, $\mathbf{G}(\mathbf{r} - \mathbf{e}) = \mathbf{0}$ implies that there exists some vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{r} - \mathbf{e} = \mathbf{A}\mathbf{x}$. Thus,

$$\mathbf{A}\mathbf{s} + \mathbf{r} = \mathbf{A}(\mathbf{s} + \mathbf{x}) + \mathbf{e}$$

which is a valid LWE instance since $\mathbf{s} + \mathbf{x}$ is still uniformly random over $\mathbb{Z}_q^m$. As for the hints, we still have $\mathbf{h} = \mathbf{Z}\mathbf{e}$ and thus $\mathscr{B}$ correctly simulates Γ_2 for $b = 0$. The case $b = 1$ follows by arguing that $\mathbf{t}$ is uniformly random and if $\mathbf{G}$ is nonsingular then $\mathbf{r}$ must be uniformly random. □

Game Γ_4: This game is a plain knapsack LWE problem. The challenger samples the public $\mathbf{G} \leftarrow\$ \mathbb{Z}_q^{n\times(n+m)}$ and a secret $\mathbf{e} \leftarrow\$ \mathbb{Z}_q^{n+m}$. Then it flips a bit $b \leftarrow\$ \{0,1\}$. If $b = 0$ then the challenger computes $\mathbf{t} := \mathbf{Ge}$, and otherwise it samples $\mathbf{t} \leftarrow\$ \mathbb{Z}_q^n$. Finally, the challenger outputs $(\mathbf{G}, \mathbf{t})$.

Lemma B.0.3. *For every efficient adversary $\mathscr{A}$, there is an efficient adversary $\mathscr{B}$ such that* $\Pr[\Gamma^4] \geq \delta_{\mathsf{elwe}} \cdot \Pr[\Gamma^3]$.

Proof. We follow the proof strategy from Alperin-Sheriff and Peikert [AP12, Theorem 1]. Suppose the algorithm $\mathscr{B}$ is given a tuple $(\mathbf{G}_0, \mathbf{G}_1, \mathbf{t})$ from Γ_4. Then, it samples $\mathbf{Z} \leftarrow\$ \chi^{\bar{n}\times(n+m)}$, $\mathbf{d} \leftarrow\$ \chi^{n+m}$, and a matrix $\mathbf{V} \leftarrow\$ \mathbb{Z}_q^{n\times\bar{n}}$. Further, it sets

$$\mathbf{G}' := \mathbf{G} - \mathbf{VZ} \quad \text{and} \quad \mathbf{t}' := \mathbf{t} - \mathbf{VZd}.$$

Finally, it runs $\mathscr{A}$ on input

$$\left(\mathbf{G}', \mathbf{t}', \mathbf{Z}, \mathbf{Zd}\right)$$

and returns what $\mathscr{A}$ outputs.

Clearly, if $\mathbf{G}$ (resp. $\mathbf{t}$) is uniformly random then so is $\mathbf{G}'$ (resp. $\mathbf{t}'$). Hence, the case $b = 1$ follows directly. Suppose $b = 0$ and thus $\mathbf{t} = \mathbf{Ge}$. Then, we have

$$\mathbf{t}' = \mathbf{t} - \mathbf{VZd} = \mathbf{Ge} - \mathbf{VZd} = \mathbf{G}'\mathbf{e} + \mathbf{V}(\mathbf{Ze} - \mathbf{Zd}).$$

Hence, if $\mathbf{Ze} = \mathbf{Zd}$ then $([\mathbf{G}_0' \; \mathbf{G}_1], \mathbf{t}', \mathbf{Z}, \mathbf{Zd})$ is indeed a valid knapsack ELWE tuple. This happens exactly with probability at most δ_{elwe} by definition. Otherwise, $\mathbf{V}(\mathbf{Ze} - \mathbf{Zd})$ is a uniformly random vector over $\mathbb{Z}_q$, and so is $\mathbf{t}'$. Thus, the tuple output by $\mathscr{B}$ follows the case $b = 1$ for Γ_3. The statement now follows by simple calculation. □

Game Γ_5: Here, we consider the plain LWE game. Recall that the challenger samples the public $\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{(n+m)\times m}$, secret $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^m$, error $\mathbf{e} \leftarrow\$ \mathbb{Z}_q^{n+m}$. Then it flips a bit $b \leftarrow\$ \{0,1\}$. If $b = 0$ then the challenger computes $\mathbf{t} := \mathbf{As} + \mathbf{e}$, and otherwise it samples $\mathbf{t} \leftarrow\$ \mathbb{Z}_q^{n+m}$. At the end, the challenger outputs $(\mathbf{A}, \mathbf{t})$.

Lemma B.0.4. *For every efficient adversary $\mathscr{A}$, there is an efficient adversary $\mathscr{B}$ such that* $\Pr[\Gamma^5] \geq \Pr[\Gamma^4] - \mathsf{negl}(n)$.

Proof. The reduction is identical to the one of Micciancio and Mol [MM11, Lemma 4.8] which we recall for completeness. Suppose the algorithm $\mathscr{B}$ is given a tuple $(\mathbf{A}, \mathbf{t})$ from Γ_5. If $\mathbf{A}$ is full-rank, then $\mathscr{B}$ can construct a (randomised) matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times(n+m)}$ whose rows generate all the vectors $\mathbf{x}$ such that $\mathbf{x}^T\mathbf{A} = \mathbf{0}$. Also, if $\mathbf{A}$ is chosen at random among all full-rank matrices, then $\mathbf{G}$ is also distributed statistically close to a uniformly random. Then, $\mathscr{B}$ outputs $(\mathbf{G}, \mathbf{Gt})$ to $\mathscr{A}$ and returns what $\mathscr{A}$ outputs.

Suppose $b = 0$ and $\mathbf{t} = \mathbf{As} + \mathbf{e}$. Then $\mathbf{Gt} = \mathbf{GAs} + \mathbf{e} = \mathbf{Ge}$, which is the correct instance of Γ_4 for $b = 0$. On the other hand, if $\mathbf{t}$ is uniformly random, then so is $\mathbf{Gt}$. □

The statement of the theorem now follows by combining all the previous lemmas using reduction composition. □

C Proof of Theorem 7.8.3

First, we recall the measure-and-reprogram lemma of Jiang et al. [JMZ23].

Lemma C.0.1 (Lemma 3.1 [JMZ23])**.** *Let $H : \{0,1\}^\ell \mapsto \{0,1\}^s$ be a quantum random oracle and $\mathscr{A}^H$ be a quantum algorithm that makes q quantum queries to H and outputs (x, z), where x and z are classical. Furthermore, we assume the i^*-th query of $\mathscr{A}$ to H is classical and equal to x, for some $i^* \in [q_H]$. In addition, let $V(x, y, z)$ be some predicate s.t. $V(x, y, z) = 1$ implies that y was output on $\mathscr{A}$'s i^*-th query to H.*
Then, there exists an algorithm $\mathscr{S}_{i^*}$ (see Figure C.1), that takes some $\Theta \in \{0,1\}^s$ as input and is such that

$$\Pr\left[V(x, H(x), z) = 1 : (x, z) \leftarrow \mathscr{A}^H\right] \leq 2(2q_H + 1)^2 \Pr\left[V(x, \Theta, z) = 1 : (x, z) \leftarrow \mathscr{S}_{i^*}^{\mathscr{A}}(\Theta)\right] + \frac{8q_H^2}{2^s},$$

where the probabilities are taken over the randomness of the algorithms, the random oracle H, and the sampling of Θ at random.

Informally, the previous lemma states that if some adversary $\mathscr{A}^H$ can satisfy a predicate with probability p, one can build another algorithm $\mathscr{S}^{\mathscr{A}}$ that does not query H on the i^*-th query (but uses its input instead) but that can satisfy the predicate with probability $\approx \frac{p}{q_H^2}$.

C.1 Proof in the QROM

We proceed with a sequence of games that is detailed in Figure C.2. The proof uses the extractable RO-simulator of Don et al. [Don+22] (see Definition 2.3.1).

Game Γ_0: This is the UNF-1KCA game with $\mathsf{sKEM} := \mathsf{T}_{\mathsf{CH}}(\mathsf{sKEM}_0)$ written explicitly. In addition, the RO used to compute the tag corresponding to ct (i.e. $t = H_1(\mathsf{pk}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K_\mathsf{B})$) is different from the one used to compute the tag for ct' (i.e. $t_c = H_2(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}', \mathscr{K}_\mathsf{A})$). Note that since $(\mathsf{pk}, \mathsf{ct}) \neq (\mathsf{pk}_\mathsf{A}, \mathsf{ct}')$ for the adversary to win, both oracles can be separated in this way.

$\mathscr{S}_{i^*}^{H,\mathscr{A}}(\Theta)$

```
(j, b) <-$ ({0, ... q_H − 1} \ {i*} × {0,1}) ∪ {(q_H, 0)}
q <- 1
(x, z) <-$ A^{H'} and
x' <- measure A's j+1-th query input register
return (x, z)
```

$H'(x)$

```
if q = i* :
  return Θ
if q < j + b + 1 :
  return H(x)
else
  if x = x' :
    return Θ
  else :
    return H(x)
```

Figure C.1: Algorithm $\mathscr{S}$ for Lemma C.0.1.

Game Γ_1: The game is the same as the previous one, except we use the simulated RO for H_2, and we use the extractor on t' (the tag output by the adversary) at the end. Note that this does not change anything to the probability of success of the game.

Game Γ_2: Now the game outputs 0 if the values extracted are different than $(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A})$. For the game to return 1, t_c must be equal to t', so let's assume it is the case. Hence, Γ^2 and Γ^1 differ only if $\mathsf{S.Ext}(t_c) \neq (\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A})$ and $H_2(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A}) = \mathsf{t}_c$. By Lemma 2.3.2, this happens with probability at most $\epsilon := \frac{2}{2^s} + 8\sqrt{2/2^s} + \frac{40e^2(q_{H'}+2)^3+2}{2^s}$. Hence, we have

$$\Pr[\Gamma_1] - \Pr[\Gamma_2] \leq \epsilon\,.$$

Game Υ_1: We see that if an adversary $\mathscr{A}$ wins Γ_2, one can build an adversary $\mathscr{B}$ that wins the game Υ_1 defined in Figure C.2. The reduction works simply by $\mathscr{B}$ running $\mathscr{A}$, simulating H_2 with the simulated RO, and running the extractor on t' at the end. Therefore, we have

$$\Pr[\Gamma_2] \leq \Pr[\Upsilon_1]\,.$$

In addition, note that one can consider oracles H and H_1 as *one* oracle $H^* := H_1 \otimes H$ with images in $\{0,1\}^{2s}$ that can be accessed $q_H + q_{H'}$ times by the adversary.

Game Υ_2: We change the game such that (t, K) are picked at random and the oracle used is now $\hat{H}$ instead of $H^* := H_1 \otimes H$. Now, let's consider a game $\mathscr{C}$ that runs Γ^1 and outputs $(x = (\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B}), z = ((t, K), K_\mathsf{A}, K'_\mathsf{A})$. In addition, let $V(x, y, z) := 1_{z_1 = y} \wedge 1_{z_2 = z_3}$. Clearly, we have that

$$\Pr[\Upsilon_1] \leq \Pr[V(x, H^*(x), z) : (x, z) \leftarrow\$ \mathscr{C}^{H^*}]$$

as V is satisfied iff $K_\mathsf{A} = K'_\mathsf{A}$. Also, note that the condition $z_1 = y$ in the predicate is always satisfied by the definition of z_1 itself. Therefore, one can apply Lemma C.0.1 with i^* equal to the query to H^* made by the game (i.e. $(t,K) \leftarrow H^*(\mathsf{pk},\mathsf{pk}_\mathsf{B},\mathsf{ct},K_\mathsf{B})$) and we get

$$\Pr[\Upsilon_1] \le 2(2(q_H+q_{H'})+1)^2 \Pr\left[V(x,(t,K),z)=1 : (x,z) \leftarrow \mathscr{S}_{i^*}^{\mathscr{A}}((t,K)\right] + \frac{8(q_H+q_{H'})^2}{2^{2s}}$$

where (t,K) is sampled at random and $\mathscr{S}_{i^*}$ is the algorithm shown in Figure C.1. By inspection, one can see that if the output of $\mathscr{S}_{i^*}$ satisfies the predicate V then Υ_2 would output 1. Therefore, we have

$$\Pr[\Upsilon_1] \le 2(2(q_H+q_{H'})+1)^2 \Pr[\Upsilon_2] + \frac{8(q_H+q_{H'})^2}{2^{2s}}.$$

Finally, one can see that if $\mathscr{A}$ wins Υ_2, one can build an adversary $\mathscr{B}$ s.t. $\mathscr{B}$ wins the decaps-OW-CPA game against sKEM_0. That is, the first phase of $\mathscr{B}$ runs the first phase of $\mathscr{A}$ and outputs the same public key pk. Then, in the second phase, $\mathscr{B}$ runs $\mathscr{A}^{\hat{H}}$ with its own input $(\mathsf{pk}_\mathsf{A},\mathsf{pk}_\mathsf{B},\mathsf{ct})$ and random tag and key (t,K). In addition, note that $\mathscr{B}$ can perfectly simulate $\hat{H}$. Finally, $\mathscr{B}$ outputs the same as the adversary $\mathscr{A}$. If $K_\mathsf{A} = K'_\mathsf{A}$ then $\mathscr{B}$ wins the decaps-OW-CPA game. Hence, we have that

$$\Pr[\Upsilon_2] \le \Pr[\text{decaps-OW-CPA}_{\mathsf{sKEM}_0}(\mathscr{B}) \Rightarrow 1] .$$

Collecting the probabilities concludes the proof. □

C.1.1 Proof in the ROM

The proof follows a similar idea as the one in the QROM.

Game Γ_0: This is the same as the UNF-1KCA game with $\mathsf{sKEM} = \mathsf{T}_\mathsf{CH}(\mathsf{sKEM}_0)$, except we assume there is no collision on H'. Thus, Γ^0 is the same as UNF-1KCA except with probability at most $\frac{q_{H'}^2}{2^s}$.

Game Γ_1: In this game, we return 0 if $\mathscr{A}$ did not query $H'(\mathsf{pk}_\mathsf{A},\mathsf{pk}_\mathsf{B},\mathsf{ct}',K_\mathsf{A})$. As we can assume $(\mathsf{pk},\mathsf{ct}) \neq (\mathsf{pk}_\mathsf{A},\mathsf{ct}')$, this changes the probability of $\mathscr{A}$ winning only if $\mathscr{A}$ outputs $t' = H'(\mathsf{pk}_\mathsf{A},\mathsf{pk}_\mathsf{B},\mathsf{ct}',K_\mathsf{A})$ without having made the oracle query. Since the query was not made, one can actually lazy sample the value of $H'(\mathsf{pk}_\mathsf{A},\mathsf{pk}_\mathsf{B},\mathsf{ct}',K_\mathsf{A})$ after $\mathscr{A}$ returns t', and the probability both values are equal is $\frac{1}{2^s}$. Hence,

$$\Pr[\Gamma_0] - \Pr[\Gamma_1] \le \frac{1}{2^s} .$$

Game Υ_1: If Γ_1 outputs 1, it means $\mathscr{A}$ outputs (ct,t') s.t. $((\mathsf{pk}_\mathsf{A},\mathsf{pk}_\mathsf{B},\mathsf{ct}',K_\mathsf{A}),t')$ is in the list of queries made by the $\mathscr{A}$. Hence, if that happens, one can find ct' and K_A s.t. $\mathsf{Decaps}(\mathsf{pk}_\mathsf{B},\mathsf{sk}_\mathsf{A},\mathsf{ct}') = K_\mathsf{A}$ by running $(\mathsf{ct}',t') \leftarrow\!\!\$\ \mathscr{A}$ and looking for t' in the list of queries (note that we assume there is no collision). Therefore, it means one can build an adversary

$\Gamma_0(\mathscr{A})$

$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGen_A}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGen_B}(1^\lambda)$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{pk_A}, \mathsf{pk_B})$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$t \leftarrow H_1(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$K \leftarrow H(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$(\mathsf{ct}', t') \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{st}, \mathsf{pk_A},$
$\qquad \mathsf{pk_B}, (\mathsf{ct}, t), K)$
$\textbf{if } (\mathsf{pk_A}, \mathsf{ct}') = (\mathsf{pk}, \mathsf{ct}) : \textbf{return } 0$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$t_c \leftarrow H_2(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A})$
$\textbf{if } t_c \neq t' : \textbf{return } 0$
$\textbf{return } 1$

$\Gamma_1(\mathscr{A})$

$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGen_A}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGen_B}(1^\lambda)$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{pk_A}, \mathsf{pk_B})$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$t \leftarrow H_1(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$K \leftarrow H(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$(\mathsf{ct}', t') \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{st}, \mathsf{pk_A},$
$\qquad \mathsf{pk_B}, (\mathsf{ct}, t), K)$
$\textbf{if } (\mathsf{pk_A}, \mathsf{ct}') = (\mathsf{pk}, \mathsf{ct}) : \textbf{return } 0$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$t_c \leftarrow H_2(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A})$
$\textbf{if } t_c \neq t' : \textbf{return } 0$
$(\mathsf{pk}_1^\star, \mathsf{pk}_2^\star, \mathsf{ct}^\star, K_\mathsf{A}^\star) \leftarrow \mathsf{S.Ext}(t')$
$\textbf{return } 1$

$\Gamma_2(\mathscr{A})$

$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGen_A}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGen_B}(1^\lambda)$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{pk_A}, \mathsf{pk_B})$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$t \leftarrow H_1(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$K \leftarrow H(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$(\mathsf{ct}', t') \leftarrow\$ \mathscr{A}^{H,H_1,H_2}(\mathsf{st}, \mathsf{pk_A},$
$\qquad \mathsf{pk_B}, (\mathsf{ct}, t), K)$
$(\mathsf{pk}_0^\star, \mathsf{pk}_1^\star, \mathsf{ct}^\star, K_\mathsf{A}^\star) \leftarrow \mathsf{S.Ext}(t')$
$\textbf{if } (\mathsf{pk_A}, \mathsf{ct}') = (\mathsf{pk}, \mathsf{ct}) : \textbf{return } 0$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$t_c \leftarrow H_2(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A})$
$\textbf{if } t_c \neq t' : \textbf{return } 0$
$\textbf{if } (\mathsf{pk}_1^\star, \mathsf{pk}_2^\star, \mathsf{ct}^\star, K_\mathsf{A}^\star) \neq (\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}', K_\mathsf{A}) :$
$\quad \textbf{return } 0$
$\textbf{return } 1$

$\Upsilon_1(\mathscr{A})$

$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGen_A}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGen_B}(1^\lambda)$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}^{H,H_1}(\mathsf{pk_A}, \mathsf{pk_B})$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$(t, K) \leftarrow H^*(\mathsf{pk}, \mathsf{pk_B}, \mathsf{ct}, K_\mathsf{B})$
$in \leftarrow (\mathsf{st}, \mathsf{pk_A}, \mathsf{pk_B}, (\mathsf{ct}, t), K)$
$K_\mathsf{A}', \mathsf{ct}' \leftarrow\$ \mathscr{A}^{H,H_1}(in)$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$\textbf{if } (\mathsf{pk_A}, \mathsf{ct}') = (\mathsf{pk}, \mathsf{ct}) \textbf{ or } K_\mathsf{A} \neq K_\mathsf{A}' :$
$\quad \textbf{return } 0$
$\textbf{return } 1$

$\Upsilon_2(\mathscr{A})$

$(j, b) \leftarrow\$ \big(\{0, \dots q_H - 1\} \times \{0,1\}\big) \cup \{(q_H, 0)\}$
$x' \leftarrow$ measure $\mathscr{A}$'s $j+1$-th query input register
$q \leftarrow 0$
$\mathsf{pk_A}, \mathsf{sk_A} \leftarrow\$ \mathsf{KeyGen_A}(1^\lambda)$
$\mathsf{pk_B}, \mathsf{sk_B} \leftarrow\$ \mathsf{KeyGen_B}(1^\lambda)$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk_A}, \mathsf{sk_B})$
$\mathsf{st}, \mathsf{pk} \leftarrow\$ \mathscr{A}^{\hat{H}}(\mathsf{pk_A}, \mathsf{pk_B})$
$(K_\mathsf{B}, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk}, \mathsf{sk_B})$
$(t, K) \leftarrow\$ \{0,1\}^{2s}$
$K_\mathsf{A}', \mathsf{ct}' \leftarrow\$ \mathscr{A}^{\hat{H}}(\mathsf{st}, \mathsf{pk_A}, \mathsf{pk_B}, (\mathsf{ct}, t), K)$
$K_\mathsf{A} \leftarrow \mathsf{Decaps}(\mathsf{pk_B}, \mathsf{sk_A}, \mathsf{ct}')$
$\textbf{if } K_\mathsf{A} \neq K_\mathsf{A}' :$
$\quad \textbf{return } 0$
$\textbf{return } 1$

$\hat{H}(x)$

$q \leftarrow q + 1$
$\textbf{if } q < j + b + 1 :$
$\quad \textbf{return } H^*(x)$
$\textbf{else}$
$\quad \textbf{if } x = x' :$
$\qquad \textbf{return } (t, K)$
$\quad \textbf{else} :$
$\qquad \textbf{return } H^*(x)$

Figure C.2: Sequence of games for the proof of Theorem 7.8.3. H^* is defined as $H_1 \otimes H_1$.

that wins the game Υ_1 in Figure C.2, and we have

$$\Pr[\Gamma_1] \leq \Pr[\Upsilon_1] \, .$$

Game Υ_2: We modify the game s.t. the tag t and the key given to the adversary are picked uniformly at random as shown in Figure C.2. Both games are indistinguishable unless $\mathscr{A}$ queries $(\mathsf{pk}, \mathsf{pk}_\mathsf{B}, \mathsf{ct}, K_\mathsf{B})$ to H or H'. Then, an adversary $\mathscr{B}$ playing Υ_2 can perfectly simulate $\mathscr{A}$'s view in Υ_1 if it guesses correctly which query it is going to be and if such a query is going to happen. Overall, $\mathscr{B}$ can make a correct guess with probability $\frac{1}{q_{H'}+q_H+1}$. If that happens though, one can build an OW-CPA adversary $\mathscr{B}$ against sKEM_0 that runs $\mathscr{A}$ and picks a random query made by $\mathscr{A}$ to H or H'. Hence, we have

$$\Pr[\Upsilon_1] \leq (q_{H'} + q_H + 1) \Pr[\Upsilon_2] \, .$$

Finally, one can see that Υ_2 is the same as the decaps-OW-CPA for sKEM_0 if we omit the random values K and t and the more restrictive winning condition $(\mathsf{pk}_\mathsf{A}, \mathsf{ct}') \neq (\mathsf{pk}, \mathsf{ct})$. Hence, one can build an adversary $\mathscr{C}$ such that

$$\Pr[\Upsilon_2] \leq \Pr[\text{decaps-OW-CPA}_{\mathsf{sKEM}_0}(\mathscr{C}) \Rightarrow 1] \, .$$

□

D Proof of Theorem 7.8.4

D.1 Proof in the ROM

Proof. As stated in the main body of this thesis, the idea of the proof is very similar to the IND-qCCA proof of the T_{CH} transform presented in Chapter 6 and is the following. Either all tags in the decapsulation query are valid and thus they are the form $H'(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_i, \mathsf{ct}_i, K'_i)$, or the oracle returns $\perp$. Then, if they are valid, with high probability the adversary queried $(\mathsf{pk}_\mathsf{A}, \mathsf{pk}_i, \mathsf{ct}_i, K'_i)$ to H' and thus K'_i can be recovered from the list of queries to the RO, i.e. the decapsulation oracle can be simulated without the knowledge of sk_A. In other words, the only information leaked by a query to the decapsulation oracle is whether *all* tags are valid or not, i.e. 1 bit of information, which is not sufficient to break the OW-CPA game. We prove this formally with a sequence of hybrid games.

Game Γ_0: This is the IND-1BatchCCA game with $\mathsf{sKEM} = \mathsf{T}_{\mathsf{CH}}(\mathsf{sKEM}_0)$.

Game Γ_1: We modify the previous game s.t. we abort if the adversary finds any collision when querying H'. We have that

$$\Pr[\Gamma_0] - \Pr[\Gamma_1] \le \frac{q_{H'}^2}{2^n}$$

where q'_H is the number of queries the adversary makes to H'.

Game Γ_2: We modify the game s.t. it aborts if $\mathrm{BatchDec}(\{(\mathsf{pk}_i, (\mathsf{ct}_i, t_i))\}_{i=1}^d)$ does not return $\perp$ but one of the tags t_i was not obtained through an adversary's query to H'. The probability

$\underline{\mathcal{O}(\mathcal{L}_{H'}, \{(\mathsf{pk}_i, (\mathsf{ct}_i, t_i))\}_{i=1}^d)}$

for $i \in [d]$:
  $K'_i \leftarrow \mathsf{Decaps}(\mathsf{pk}_i, \mathsf{sk}_\mathsf{A}, \mathsf{ct}_i)$
  if $((\mathsf{pk}_i, \mathsf{ct}_i, K'_i), t_i) \notin \mathcal{L}_{H'}$: **return** 0
return 1

Figure D.1: Oracle $\mathcal{O}$ used in the proof of Theorem 7.8.4.

that some tag t_i is valid but $H'(\mathsf{pk_A}, \mathsf{pk}_i, \mathsf{ct}_i, K'_i)$ (with $(\mathsf{pk_A}, \mathsf{pk}_i, \mathsf{ct}_i, t_i) \neq (\mathsf{pk_A}, \mathsf{pk}_B, \mathsf{ct}^*, t_i^*)$) was not queried by the adversary is $\frac{1}{2^n}$. Hence, overall we have

$$\Pr[\Gamma_1] - \Pr[\Gamma_2] \leq \frac{d}{2^n} \ .$$

Game Γ_3: We now change the game as follows. We record all queries to H' of the form $(\mathsf{pk_A}, \cdot, \cdot, \cdot)$ made by the adversary in a list $\mathcal{L}_{H'} = \{((\mathsf{pk}_j, \mathsf{ct}_j, K_j), h_j)\}_{j=1}^{q_{H'}}$ s.t. $H'(\mathsf{pk_A}, \mathsf{pk}_j, \mathsf{ct}_j, K_j) = h_j$ for all $j \in [q_{H'}]$. Then, the BatchDec oracle is modified as follows. If some tag t_i is s.t. for all $K \in \mathcal{K}$ $((\mathsf{pk}_i, \mathsf{ct}_i, K), t_i) \notin \mathcal{L}_{H'}$ then $\perp$ is returned. Then, $\mathcal{O}(\mathcal{L}_{H'}, \{(\mathsf{pk}_i, (\mathsf{ct}_i, t_i))\}_{i=1}^{d}) \to r$ is queried, where $\mathcal{O}$ is defined in Figure D.1. If $r = 0$ BatchDec outputs $\perp$, otherwise it outputs $H(\mathsf{pk_A}, \mathsf{pk}_i, \mathsf{ct}_i, K_i)$ for all $i \in [d]$, where K_i is s.t. $((\mathsf{pk}_i, \mathsf{ct}_i, K_i), t_i) \in \mathcal{L}_{H'}$. Note that all these modifications are only syntactical as $\mathcal{O}$ outputs 1 iff for all $i \in [d]$, K_i is (the unique) value in $\mathcal{L}_{H'}$ s.t. $H'(\mathsf{pk_A}, \mathsf{pk}_i, \mathsf{ct}_i, K_i := \mathsf{Decaps}(\mathsf{pk}_i, \mathsf{sk_A})) = t_i$. Hence, we have

$$\Pr[\Gamma_2] = \Pr[\Gamma_3] \ .$$

Game Γ_4: We replace the challenge tag t^* and the real key K_0 by random values. This change can only be noticed if the adversary or the BatchDec oracle queries $H(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}^*, K^*)$ or $H'(\mathsf{pk_A}, \mathsf{pk_B}, \mathsf{ct}^*, K^*)$ at some point in the game. Let QUERY be this event. We show that if QUERY occurs, then one can break the OW-CPA security of sKEM_0 with high probability. The reduction works as follows. The OW-CPA adversary $\mathcal{B}$ receives a challenge ciphertext ct^* and public keys $\mathsf{pk_A}, \mathsf{pk_B}$ from its own challenger. Next, it samples random values K, t^* and passes all these to the IND-1BatchCCA adversary $\mathcal{A}$. Then, $\mathcal{B}$ can simulate everything in BatchDec (except the oracle call to $\mathcal{O}$) by recording $\mathcal{A}$'s queries to H'. In order to simulate $\mathcal{O}$, $\mathcal{B}$ samples a bit r at random instead, which succeeds with probability $\frac{1}{2}$. Finally, it samples at random a query made by $\mathcal{A}$ to H or H' or a query made to H by itself, and it outputs the key K that was part of this query. Overall, the simulation is correct with probability $\frac{1}{2}$ and if QUERY occurs $\mathcal{B}$ recovers K^* with probability $\frac{1}{q_H + q_{H'} + d}$. Hence,

$$\Pr[\Gamma_3] - \Pr[\Gamma_4] \leq \Pr[\mathsf{QUERY}] \leq 2(q_H + q_{H'} + d)\mathsf{Adv}_{\mathsf{sKEM}}^{\text{ow-cpa}}(\mathcal{A}) \ .$$

Finally, we see that the adversary's view is independent of b in Γ_4, therefore $\Pr[\Gamma_4] = \frac{1}{2}$. This concludes the proof. □

D.2 Proof in the QROM

Proof. As in most of the QROM proofs presented in this thesis, we use the extractable RO-simulator by Don et al. [Don+22] (c.f. Definition 2.3.1) and we proceed with a sequence of hybrid games. Again, the proof is nearly identical to the QROM IND-qCCA proof of $\mathsf{T_{CH}}$ (c.f. Theorem 6.4.2) and we refer the reader to it for a detailed explanation of the game transitions.

Game Γ_0: This is the IND-1BatchCCA game with $\mathsf{sKEM} = \mathsf{T}_{\mathsf{CH}}(\mathsf{sKEM}_0)$. We also assume that the adversary only makes queries of the form $(\mathsf{pk}_\mathsf{A}, \cdot, \cdot, \cdot)$ to the oracles. This has no consequence on the winning probability of the adversary as other type of queries are independent of the key.

Game Γ_1: We modify the BatchDec oracle s.t. it returns $\perp$ whenever the list of $(\mathsf{pk}_i, \mathsf{ct}_i, t_i)$ in the query contains $(\mathsf{pk}_i, \mathsf{ct}_i) = (\mathsf{pk}_\mathsf{B}, \mathsf{ct}^*)$ (and thus $t_i \neq t^*$). This change has no impact except if $\mathsf{Decaps}(\mathsf{pk}_\mathsf{B}, \mathsf{sk}_\mathsf{A}, \mathsf{ct}^*) \neq K_0$, where K_0 is the challenge real key. In turn, this would imply that ct^* is an incorrect ciphertext. Hence,

$$\Pr[\Gamma_0] - \Pr[\Gamma_1] \leq \delta \ .$$

Game Γ_2: Now, we split the random oracle H' into two oracles H'_0 and H'_1 s.t.

$$H'(\mathsf{pk}_\mathsf{A}, \mathsf{pk}, \mathsf{ct}, K) := \begin{cases} H'_0(K), & \text{if } (\mathsf{pk}, \mathsf{ct}) = (\mathsf{pk}_\mathsf{B}, \mathsf{ct}^*) \\ H'_1(\mathsf{pk}, \mathsf{ct}, K), & \text{otherwise} \end{cases}$$

and we give the adversary access to H'_0, H'_1 instead of H'. We also switch to the RO simulator instead of using H'_1. In addition, at the end of the game, the challenger calls the extractor on all tags t_i queried as part of the call to the BatchDec oracle to obtain extracted values $(\mathsf{pk}^\mathsf{e}_i, \mathsf{ct}^\mathsf{e}_i, K^\mathsf{e}_i)$, $i \in [d]$. Note that H'_0 is never called as part of a BatchDec query due to the modification in the previous game. These changes have no impact on the success of the game and thus

$$\Pr[\Gamma_1] = \Pr[\Gamma_2] \ .$$

Game Γ_3: We abort whenever the decapsulation oracle does not return $\perp$ but the extracted values $(\mathsf{pk}^\mathsf{e}_i, \mathsf{ct}^\mathsf{e}_i, K^\mathsf{e}_i)$ are not equal to $\perp$ or $(\mathsf{pk}_i, \mathsf{ct}_i, K'_i)$, where $K'_i = \mathsf{Decaps}(\mathsf{pk}_i, \mathsf{sk}_\mathsf{A}, \mathsf{ct}_i)$. By Property 8 of the extractable oracle, we have

$$\Pr[\Gamma_2] - \Pr[\Gamma_3] \leq \frac{40e^2(q_{H'} + d + 1)^3 + 2}{2^n} \ .$$

Game Γ_4: We move the extraction to the BatchDec oracle, right after the corresponding tag verification. By Property 4 of the extractable oracle, we have

$$\Pr[\Gamma_3] - \Pr[\Gamma_4] \leq 8d(d + q_{H'})\sqrt{2/2^n} \ .$$

Game Γ_5: We modify the BatchDec oracle s.t. we abort if all tag checks pass, i.e. $H'(\mathsf{pk}_i, \mathsf{ct}_i, K'_i) = t_i, \forall i \in [d]$ but some extracted value is equal to $\perp$, i.e. $(\mathsf{pk}^\mathsf{e}_i, \mathsf{ct}^\mathsf{e}_i, K^\mathsf{e}_i) = \perp$ for some $i \in [d]$. By Property 7 of the extractable oracle we have

$$\Pr[\Gamma_4] - \Pr[\Gamma_5] \le d\frac{2}{2^n}\ .$$

Game Γ_6: We modify the BatchDec oracle s.t. the queries to H' made for the tag verification are made after the corresponding extraction. By Property 8 of the extractable oracle we have

$$\Pr[\Gamma_5] - \Pr[\Gamma_6] \le 8d\sqrt{2/2^n}\ .$$

Game Γ_7: We modify the previous game as follows. Let r be a bit set to 1 iff for all $i \in [d]$ $(\mathsf{pk}_i^{\mathsf{e}}, \mathsf{ct}_i^{\mathsf{e}}) = (\mathsf{pk}_i, \mathsf{ct}_i)$ and $\mathsf{Decaps}(\mathsf{pk}_i^{\mathsf{e}}, \mathsf{sk}_{\mathsf{A}}, \mathsf{ct}_i^{\mathsf{e}}) = K_i^{\mathsf{e}}$. Then, we change BatchDec s.t. it returns $\perp$ if $r = 0$, otherwise it returns $H(\mathsf{pk}_{\mathsf{A}}, \mathsf{pk}_i, \mathsf{ct}_i, K_i^{\mathsf{e}})$ for all $i \in [d]$. In addition, the tag verification is now skipped. We argue this affects only negligibly the advantage of the adversary compared to the previous game:

- If BatchDec returns $H(\mathsf{pk}_{\mathsf{A}}, \mathsf{pk}_i, \mathsf{ct}_i, K_i')$, $i \in [d]$ in Γ_6, then by the previous changes we know that $(\mathsf{pk}_i^{\mathsf{e}}, \mathsf{ct}_i^{\mathsf{e}}, K_i^{\mathsf{e}}) = (\mathsf{pk}_i, \mathsf{ct}_i, K_i')$ for all $i \in [d]$, therefore BatchDec returns $H(\mathsf{pk}_{\mathsf{A}}, \mathsf{pk}_i, \mathsf{ct}_i, K_i')$, $i \in [d]$ in Γ_7 as well.
- If BatchDec returns $H(\mathsf{pk}_{\mathsf{A}}, \mathsf{pk}_i, \mathsf{ct}_i, K_i^{\mathsf{e}})$, $i \in [d]$ in Γ_7, we know that $(\mathsf{pk}_i^{\mathsf{e}}, \mathsf{ct}_i^{\mathsf{e}}, K_i^{\mathsf{e}}) = (\mathsf{pk}_i, \mathsf{ct}_i, K_i')$. In addition, for each $i \in [d]$, $t_i = H(\mathsf{pk}_i^{\mathsf{e}}, \mathsf{ct}_i^{\mathsf{e}}, K_i^{\mathsf{e}})$ with probability $1 - \frac{2}{2^n}$ by Property 6 of the extractable oracle. Therefore, the tag verification would pass in Γ_6 with high probability and BatchDec would return the same values in that game as well.

Overall, we have

$$\Pr[\Gamma_6] - \Pr[\Gamma_7] \le d\frac{2}{2^n}\ .$$

Game Γ_8: Now we move all d queries to H' made in BatchDec to the end of the game. By Property 8 of the extractable oracle, we have

$$\Pr[\Gamma_7] - \Pr[\Gamma_8] \le 8d\, q_{H'}\sqrt{2/2^n}\ .$$

Note that we can now forget about the queries to H' we just moved to the end of the game.

Game Γ_9: We replace the real key K_0 and the challenge tag t^* by random values. We have $\Pr[\Gamma_9] = \frac{1}{2}$. Applying the OW2H lemma on $H \otimes H_0'$, we get

$$\Pr[\Gamma_8] - \Pr[\Gamma_9] \le 2(q_{H'} + q_H + d)\sqrt{\Pr[\Upsilon]}$$

where Υ is the same as Γ_9, except the challenger measures a random query made to $H \otimes H_0'$ and outputs 1 iff the query contains K^*, where K^* is the key encapsulated in ct^*. We can build an OW-CPA adversary $\mathscr{B}$ against sKEM_0 that wins with high probability when Υ outputs 1. The reduction works nearly as in the ROM proof: $\mathscr{B}$ receives a challenge ciphertext ct^* and two public keys $\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}$, then it samples t^* and K^* at random and passes all these values to $\mathscr{A}$. Then, $\mathscr{B}$ can perfectly simulate BatchDec as in Γ_9, except for the bit r that it can guess correctly with probability $\frac{1}{2}$. Finally, $\mathscr{B}$ measures a random query that was made to H or H' in the execution and outputs the corresponding value K. Overall, we have

$$\Pr[\Upsilon] \leq 2\mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{sKEM}}(\mathscr{A}) \,,$$

which concludes the proof. $\square$

Bibliography

[AHV98] William Aiello, Stuart Haber, and Ramarathnam Venkatesan. “New Constructions for Secure Hash Functions”. In: *International Workshop on Fast Software Encryption*. Springer. 1998, pp. 150–167.

[Alw+21] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. “Analysing the HPKE Standard”. In: *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*. Springer. 2021, pp. 87–116.

[AP12] Jacob Alperin-Sheriff and Chris Peikert. “Circular and KDM Security for Identity-Based Encryption”. In: *Public Key Cryptography*. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 334–352.

[App+09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. “Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems”. In: *CRYPTO*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 595–618.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. “On the Concrete Hardness of Learning With Errors”. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203.

[Ara+18] N. Aragon, P. Gaborit, A. Hauteville, and J. Tillich. “A New Algorithm for Solving the Rank Syndrome Decoding Problem”. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. June 2018, pp. 2421–2425. DOI: 10.1109/ISIT.2018.8437464.

[AS16] Gilad Asharov and Gil Segev. “Limits on the Power of Indistinguishability Obfuscation and Functional Encryption”. In: *SIAM Journal on Computing* 45.6 (2016), pp. 2117–2176.

[Azo+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. “Systematic Study of Decryption and Re-Encryption Leakage: The Case of Kyber”. In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer. 2022, pp. 236–256.

[Bad+15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. "Tightly-Secure Authenticated Key Exchange". In: *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Springer, 2015, pp. 629–658. DOI: 10.1007/978-3-662-46494-6_26. URL: https://doi.org/10.1007/978-3-662-46494-6%5C_26.

[Băe+19] Ciprian Băetu, F Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. "Misuse Attacks on Post-Quantum Cryptosystems". In: *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II 38*. Springer. 2019, pp. 747–776.

[Bar+23a] Khashayar Barooti, Daniel Collins, Simone Colombo, Loïs Huguenin-Dumittan, and Serge Vaudenay. "On Active Attack Detection in Messaging With Immediate Decryption". In: *Annual International Cryptology Conference*. Springer. 2023, pp. 362–395.

[Bar+23b] Khashayar Barooti, Alex B. Grilo, Loïs Huguenin-Dumittan, Giulio Malavolta, Or Sattath, Quoc-Huy Vu, and Michael Walter. "Public-Key Encryption with Quantum Keys". In: *Theory of Cryptography*. Ed. by Guy Rothblum and Hoeteck Wee. Springer Nature Switzerland, 2023, pp. 198–227.

[Bau+19] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. "Assessment of the Key-Reuse Resilience of NewHope". In: *Topics in Cryptology – CT-RSA 2019*. Ed. by Mitsuru Matsui. Cham: Springer International Publishing, 2019, pp. 272–292. ISBN: 978-3-030-12612-4.

[BB84] Charles H. Bennett and Gilles Brassard. "Quantum Cryptography: Public Key Distribution and Coin Tossing". In: *Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing*. 1984, pp. 175–179.

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. "Notions of Black-Box Reductions, Revisited". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2013, pp. 296–315.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2004, pp. 171–188.

[BDG20] Mihir Bellare, Hannah Davis, and Felix Günther. "Separate Your Domains: NIST PQC KEMs, Oracle Cloning and Read-Only Indifferentiability". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 3–32.

[Be20] Daniel J. Bernstein and Tanja Lange (editors). *eBACS: ECRYPT Benchmarking of Cryptographic Systems*. https://bench.cr.yp.to. accessed 14 May 2020.

[Bel06a] Mihir Bellare. “Code-Based Game-Playing Proofs and the Security of Triple Encryption”. In: *Advances in Cryptology-Eurocrypt’06* (2006), pp. 409–426.

[Bel06b] Mihir Bellare. “New Proofs for NMAC and HMAC: Security Without Collision-Resistance”. In: *CRYPTO 2006*. Springer. 2006, pp. 602–619.

[Ber+18] Daniel J. Bernstein, Leon Groot Bruinderink, Tanja Lange, and Lorenz Panny. “HILA5 Pindakaas: On the CCA Security of Lattice-Based Encryption with Error Correction”. In: *Progress in Cryptology – AFRICACRYPT 2018*. Ed. by Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Cham: Springer International Publishing, 2018, pp. 203–216. ISBN: 978-3-319-89339-6.

[Bet+19] Slim Bettaieb, Loïc Bidoux, Philippe Gaborit, and Etienne Marcatel. “Preventing Timing Attacks Against RQC Using Constant Time Decoding of Gabidulin Codes”. In: *Post-Quantum Cryptography*. Ed. by Jintai Ding and Rainer Steinwandt. Cham: Springer International Publishing, 2019, pp. 371–386. ISBN: 978-3-030-25510-7.

[Beu22] Ward Beullens. “Breaking Rainbow Takes a Weekend on a Laptop”. In: *Annual International Cryptology Conference*. Springer. 2022, pp. 464–479.

[Bha+23] Karthikeyan Bhargavan, Charlie Jacomme, Franziskus Kiefer, and Rolfe Schmidt. *An Analysis of Signal’s PQXDH*. https://cryspen.com/post/pqxdh/ Accessed: 23.10.23. 2023.

[Bin+17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. “Transitioning to a Quantum-Resistant Public Key Infrastructure”. In: *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*. Springer. 2017, pp. 384–405.

[Bin+19a] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. “Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange”. In: *Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*. Springer. 2019, pp. 206–226.

[Bin+19b] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. “Tighter Proofs of CCA Security in the Quantum Random Oracle Model”. In: *Theory of Cryptography*. Ed. by Dennis Hofheinz and Alon Rosen. Cham: Springer International Publishing, 2019, pp. 61–90. ISBN: 978-3-030-36033-7.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. “Calamari and Falafl: Logarithmic (Linkable) Ring Signatures From Isogenies and Lattices”. In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II*. Springer. 2020, pp. 464–492.

[Ble98] Daniel Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *Advances in Cryptology — CRYPTO '98*. Ed. by Hugo Krawczyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 1–12. ISBN: 978-3-540-68462-6.

[BN13] Ahto Buldas and Margus Niitsoo. "Black-Box Separations and Their Adaptability to the Non-Uniform Model". In: *Australasian Conference on Information Security and Privacy*. Springer. 2013, pp. 152–167.

[Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 41–69. ISBN: 978-3-642-25385-0.

[Bos+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. "Frodo: Take Off the Ring! Practical, Quantum-Secure Key Exchange From LWE". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 1006–1018.

[Bou+21] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. "On the Hardness of Module-Lwe With Binary Secret". In: *CT-RSA*. Vol. 12704. Lecture Notes in Computer Science. Springer, 2021, pp. 503–526.

[BR60] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. "On a Class of Error Correcting Binary Group Codes". In: *Information and Control* 3.1 (1960), pp. 68–79.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 1993, pp. 62–73.

[Bra+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical Hardness of Learning With Errors". In: *CoRR* abs/1306.0281 (2013).

[Bre+17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. *PRF-ODH: Relations, Instantiations, and Impossibility Results*. Cryptology ePrint Archive, Report 2017/517. https://ia.cr/2017/517. 2017.

[Bre+21] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. "Towards Post-Quantum Security for Signal's X3DH Handshake". In: *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*. Springer. 2021, pp. 404–430.

[Bre+22] Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila. "Post-Quantum Asynchronous Deniable Key Exchange and the Signal Handshake". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2022, pp. 3–34.

[BRV20] Fatih Balli, Paul Rösler, and Serge Vaudenay. "Determining the Core Primitive for Optimally Secure Ratcheting". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 621–650.

[BS99] Mihir Bellare and Amit Sahai. "Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 519–536. ISBN: 978-3-540-48405-9.

[Car75] David Carlson. "Matrix Decompositions Involving the Schur Complement". In: *SIAM Journal on Applied Mathematics* 28.3 (1975), pp. 577–587. ISSN: 00361399. URL: http://www.jstor.org/stable/2100380.

[Cas+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24*. Springer. 2018, pp. 395–427.

[CCH23] Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. "Real World Deniability in Messaging". In: *Cryptology ePrint Archive* (2023).

[CD23] Wouter Castryck and Thomas Decru. "An Efficient Key Recovery Attack on SIDH". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 423–447.

[Cel+21] Sofía Celi, Armando Faz-Hernández, Nick Sullivan, Goutam Tamvada, Luke Valenta, Thom Wiggers, Bas Westerbaan, and Christopher A Wood. "Implementing and Measuring KEMTLS". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2021, pp. 88–107.

[CF11] Cas Cremers and Michele Feltz. "One-Round Strongly Secure Key Exchange With Perfect Forward Secrecy and Deniability". In: *Cryptology ePrint Archive, Paper 2011/300* (2011). URL: https://eprint.iacr.org/2011/300.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. "The Random Oracle Methodology, Revisited". In: *Journal of the ACM (JACM)* 51.4 (2004), pp. 557–594.

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *ASIACRYPT*. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 1–20.

[Coh+19] Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. "Highly Efficient Key Exchange Protocols With Optimal Tightness". In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*. Springer. 2019, pp. 767–797.

[Coh+20] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. "A Formal Security Analysis of the Signal Messaging Protocol". In: *J. Cryptol.* 33.4 (2020), pp. 1914–1983. DOI: 10.1007/s00145-020-09360-1. URL: https://doi.org/10.1007/s00145-020-09360-1.

[Col+ar] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. "K-Waay: Fast and Deniable Post-Quantum X3DH Without Ring Signatures". In: *USENIX Security*. 2024 (to appear).

[CPS19] Eric Crockett, Christian Paquin, and Douglas Stebila. "Prototyping Post-Quantum and Hybrid Key Exchange and Authentication in TLS and SSH." In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 858.

[Cra+07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. "Bounded CCA2-secure Encryption". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2007, pp. 502–518.

[CS03] Ronald Cramer and Victor Shoup. "Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack". In: *SIAM Journal on Computing* 33.1 (2003), pp. 167–226.

[CX21] Shujiao Cao and Rui Xue. "Being a Permutation Is Also Orthogonal to One-Wayness in Quantum World: Impossibilities of Quantum One-Way Permutations From One-Wayness Primitives". In: *Theoretical Computer Science* 855 (2021), pp. 16–42.

[CZ24] Cas Cremers and Mang Zhao. "Secure Messaging With Strong Compromise Resilience, Temporal Privacy, and Immediate Decryption". In: *IEEE S&P*. 2024.

[DAn+19a] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. *SABER: Mod-LWR Based KEM*. NIST Round 2 Submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 2019.

[DAn+19b] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. "Timing Attacks on Error Correcting Codes in Post-Quantum Secure Schemes". In: *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*. 2019, pp. 2–9.

[DG21] Hannah Davis and Felix Günther. "Tighter Proofs for the SIGMA and TLS 1.3 Key Exchange Protocols". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2021, pp. 448–479.

[DG22] Samuel Dobson and Steven D. Galbraith. "Post-Quantum Signal Key Agreement From SIDH". In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*. Ed. by Jung Hee Cheon and Thomas Johansson. Vol. 13512. Lecture Notes in Computer Science. Springer, 2022, pp. 422–450. DOI: 10.1007/978-3-031-17234-2_20. URL: https://doi.org/10.1007/978-3-031-17234-2%5C_20.

[DHV20] F Betül Durak, Loïs Huguenin-Dumittan, and Serge Vaudenay. "BioLocker: A Practical Biometric Authentication Mechanism Based on 3D Fingervein". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2020, pp. 62–80.

[Din+17] Jintai Ding, Saed Alsayigh, RV Saraswathy, Scott Fluhrer, and Xiaodong Lin. "Leakage of Signal Function With Reused Keys in RLWE Key Exchange". In: *2017 IEEE international conference on communications (ICC)*. IEEE. 2017, pp. 1–6.

[DJ21] Denis Diemert and Tibor Jager. "On the Tight Security of TLS 1.3: Theoretically Sound Cryptographic Parameters for Real-World Deployments". In: *Journal of Cryptology* 34.3 (2021), pp. 1–57.

[DK05] Yevgeniy Dodis and Jonathan Katz. "Chosen-Ciphertext Security of Multiple Encryption". In: *Theory of Cryptography Conference*. Springer. 2005, pp. 188–209.

[Don+22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Online-Extractability in the Quantum Random-Oracle Model". In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Cham: Springer International Publishing, 2022, pp. 677–706. ISBN: 978-3-031-07082-2.

[Dow+20] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol". In: *Journal of Cryptology* (2020).

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. "MatRiCT$^+$: More Efficient Post-Quantum Private Blockchain Payments". In: *IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 1281–1298.

[FHZ18] Tomas Fabsic, Viliam Hromada, and Pavol Zajac. "A Reaction Attack on LEDApkc". In: *Cryptology ePrint Archive, Report 2018/140* (2018). https://eprint.iacr.org/2018/140.

[FL08] Marc Fischlin and Anja Lehmann. "Multi-Property Preserving Combiners for Hash Functions". In: *Theory of Cryptography Conference*. Springer. 2008, pp. 375–392.

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. "Robust Multi-Property Combiners for Hash Functions Revisited". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2008, pp. 655–666.

[Flu16] Scott Fluhrer. "Cryptanalysis of Ring-LWE Based Key Exchange With Key Share Reuse". In: *Cryptology ePrint Archive, Report 2016/085* (2016). https://eprint.iacr.org/2016/085.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Journal of Cryptology* 26.1 (2013), pp. 80–101. URL: https://doi.org/10.1007/s00145-011-9114-1.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 537–554.

[Gab85] Ernst Gabidulin. "Theory of Codes With Maximum Rank Distance (Translation)". In: *Problems of Information Transmission* 21 (Jan. 1985), pp. 1–12.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. "KEM Combiners". In: *Public-Key Cryptography–PKC 2018: 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I 21*. Springer. 2018, pp. 190–218.

[Gil16] Daniel Gillmor. "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)". In: *Ietf RFC 7919* (2016).

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. "A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 789–815. ISBN: 978-3-662-53887-6.

[GJY19] Qian Guo, Thomas Johansson, and Jing Yang. "A Novel CCA Attack Using Decryption Errors Against LAC". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 82–111.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. "On the (In) Security of the Fiat-Shamir Paradigm". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings*. IEEE. 2003, pp. 102–113.

[GMM07] Yael Gertner, Tal Malkin, and Steven Myers. "Towards a Separation of Semantic and CCA Security for Public Key Encryption". In: *Theory of Cryptography Conference*. Springer. 2007, pp. 434–455.

[GMP22] Paul Grubbs, Varun Maram, and Kenneth G Paterson. "Anonymous, Robust Post-Quantum Public Key Encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 402–432.

[GPA19] Lachlan J Gunn, Ricardo Vieitez Parra, and N Asokan. "Circumventing Cryptographic Deniability With Remote Attestation". In: *Proceedings on Privacy Enhancing Technologies* 3 (2019), pp. 350–369.

[Gün+22] Felix Günther, Simon Rastikian, Patrick Towa, and Thom Wiggers. "KEMTLS With Delayed Forward Identity Protection in (Almost) a Single Round Trip". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2022, pp. 253–272.

[GY06] Maximilien Gadouleau and Zhiyuan Yan. "Properties of Codes With the Rank Metric". In: *CoRR* abs/cs/0610099 (Oct. 2006). DOI: 10.1109/GLOCOM.2006.173.

[Has+21] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. "An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2021, pp. 410–440.

[Has+22] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. "An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable". In: *Journal of Cryptology* 35.3 (2022), p. 17.

[Her05] Amir Herzberg. "On Tolerant Cryptographic Constructions". In: *Cryptographers' Track at the RSA Conference*. Springer. 2005, pp. 172–190.

[HGS99] Chris Hall, Ian Goldberg, and Bruce Schneier. "Reaction Attacks against Several Public-Key Cryptosystem". In: *Information and Communication Security*. Ed. by Vijay Varadharajan and Yi Mu. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 2–12. ISBN: 978-3-540-47942-0.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *Theory of Cryptography Conference*. Springer. 2017, pp. 341–371.

[HL21] Loïs Huguenin-Dumittan and Iraklis Leontiadis. "A Message Franking Channel". In: *Information Security and Cryptology: 17th International Conference, Inscrypt 2021, Virtual Event, August 12–14, 2021, Revised Selected Papers 17*. Springer. 2021, pp. 111–128.

[Hoc59] Alexis Hocquenghem. "Codes Correcteurs d'Erreurs". In: *Chiffers* 2 (1959), pp. 147–156.

[How+03] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. "The Impact of Decryption Failures on the Security of NTRU Encryption". In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 226–246. ISBN: 978-3-540-45146-4.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. "Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?" In: *Advances in Cryptology–CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings 24*. Springer. 2004, pp. 92–105.

[HV20] Loïs Huguenin-Dumittan and Serge Vaudenay. "Classical Misuse Attacks on NIST Round 2 PQC: The Power of Rank-Based Schemes". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2020, pp. 208–227.

[HV21] Loïs Huguenin-Dumittan and Serge Vaudenay. "FO-like Combiners and Hybrid Post-Quantum Cryptography". In: *International Conference on Cryptology and Network Security*. Springer. 2021, pp. 225–244.

[HV22] Loïs Huguenin-Dumittan and Serge Vaudenay. "On IND-qCCA Security in the ROM and Its Applications: CPA Security Is Sufficient for TLS 1.3". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022, pp. 613–642.

[HV24] Loïs Huguenin-Dumittan and Serge Vaudenay. "Impossibility of Post-Quantum Shielding Black-Box Constructions of CCA from CPA". In: *Communications in Cryptology* 1.1 (2024).

[HY20] Akinori Hosoyamada and Takashi Yamakawa. "Finding Collisions in a Quantum World: Quantum Black-Box Separation of Collision-Resistance and One-Wayness". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 3–32.

[Inf23] BSI - German Federal Office for Information Security. *Bsi Tr-01102-1*. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html. 2023.

[IR89] Russell Impagliazzo and Steven Rudich. "Limits on the Provable Consequences of One-Way Permutations". In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. 1989, pp. 44–61.

[Jea+02] Coron Jean-Sébastien, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. "GEM: A Generic Chosen-Ciphertext Secure Encryption Method". In: *Topics in Cryptology — CT-RSA 2002*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 263–276. ISBN: 978-3-540-45760-2.

[JMM19] Daniel Jost, Ueli Maurer, and Marta Mularczyk. "Efficient Ratcheting: Almost-Optimal Guarantees for Secure Messaging". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 159–188.

[JMZ23] Haodong Jiang, Zhi Ma, and Zhenfeng Zhang. "Post-Quantum Security of Key Encapsulation Mechanism Against CCA Attacks With a Single Decapsulation Query". In: *Cryptology ePrint Archive* (2023).

[Kil+23] Eike Kiltz, Jiaxin Pan, Doreen Riepel, and Magnus Ringerud. "Multi-User CDH Problems and the Concrete Security of NAXOS and HMQV". In: *Cryptographers' Track at the RSA Conference*. Springer. 2023, pp. 645–671.

[KM15] Neal Koblitz and Alfred J Menezes. "The Random Oracle Model: A Twenty-Year Retrospective". In: *Designs, Codes and Cryptography* 77 (2015), pp. 587–610.

[Kuc+20] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shi-Feng Sun. "Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 703–728.

[Laa16] Thijs Laarhoven. "Search problems in cryptography: from fingerprinting to lattice sieving". Proefschrift. PhD thesis. Mathematics and Computer Science, Feb. 2016. ISBN: 978-90-386-4021-1.

[LAZ19] Xingye Lu, Man Ho Au, and Zhenfei Zhang. "Raptor: A Practical Lattice-Based (Linkable) Ring Signature". In: *ACNS*. Vol. 11464. Lecture Notes in Computer Science. Springer, 2019, pp. 110–130.

[Lep18] Tancrède Lepoint. "Algorithmic of LWE-based Submissions to NIST Post-Quantum Standardization Effort". In: *Post-Scryptum Spring School* (2018).

[LN22] Vadim Lyubashevsky and Ngoc Khanh Nguyen. "BLOOM: Bimodal Lattice One-Out-of-Many Proofs and Applications". In: *ASIACRYPT (4)*. Vol. 13794. Lecture Notes in Computer Science. Springer, 2022, pp. 95–125.

[Lu+19] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. *Lac*. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 2019.

[Lun18] Joshua Lund. *Technology Preview: Sealed Sender for Signal*. https://signal.org/blog/sealed-sender/. Last visited on 13-09-2023. 2018.

[Mel+19a] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. *Rank Quasi-Cyclic (RQC)*. NIST Round 2 Submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 2019.

[Mel+19b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. *Hamming Quasi-Cyclic (HQC)*. NIST Round 2 Submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. 2019.

[Mer+22] Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. "Efficient Lattice-Based Inner-Product Functional Encryption". In: *Public Key Cryptography (2)*. Vol. 13178. Lecture Notes in Computer Science. Springer, 2022, pp. 163–193.

[MM11] Daniele Micciancio and Petros Mol. "Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions". In: *CRYPTO*. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 465–484.

[MS74] George Marsaglia and George P. H. Styan. "Equalities and Inequalities for Ranks of Matrices". In: *Linear and Multilinear Algebra* 2.3 (1974), pp. 269–292. DOI: 10.1080/03081087408817070.

[NO02] Harumichi Nishimura and Masanao Ozawa. "Computational Complexity of Uniform Quantum Circuit Families and Quantum Turing Machines Communicated by O. Watanabe". In: *Theoretical Computer Science* 276.1-2 (2002), pp. 147–181.

[OP01] Tatsuaki Okamoto and David Pointcheval. "REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform". In: *Topics in Cryptology — CT-RSA 2001*. Ed. by David Naccache. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 159–174. ISBN: 978-3-540-45353-6.

[Pei14] Chris Peikert. "Lattice Cryptography for the Internet". In: *PQCrypto*. Vol. 8772. Lecture Notes in Computer Science. Springer, 2014, pp. 197–219.

[Per+10] Mayana Pereira, Rafael Dowsley, Goichiro Hanaoka, and Anderson CA Nascimento. "Public Key Encryption Schemes With Bounded CCA Security and Optimal Ciphertext Length Based on the CDH Assumption". In: *International Conference on Information Security*. Springer. 2010, pp. 299–306.

[PR18] Bertram Poettering and Paul Rösler. "Towards Bidirectional Ratcheted Key Exchange". In: *Annual International Cryptology Conference*. Springer. 2018, pp. 3–32.

[PR20] Bertram Poettering and Paul Rösler. "Combiners for AEAD". In: *IACR Transactions on Symmetric Cryptology* (2020), pp. 121–143.

[PST20] Christian Paquin, Douglas Stebila, and Goutam Tamvada. "Benchmarking Post-Quantum Cryptography in TLS". In: *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*. Springer. 2020, pp. 72–91.

[QCD19a] Yue Qin, Chi Cheng, and Jintai Ding. "A Complete and Optimized Key Mismatch Attack on NIST Candidate NewHope". In: *European symposium on research in computer security*. Springer. 2019, pp. 504–520.

[QCD19b] Yue Qin, Chi Cheng, and Jintai Ding. "An Efficient Key Mismatch Attack on the NIST Second Round Candidate Kyber". In: *Cryptology ePrint Archive, Report 2019/1343* (2019). https://eprint.iacr.org/2019/1343.

[Qin+21] Yue Qin, Chi Cheng, Xiaohan Zhang, Yanbin Pan, Lei Hu, and Jintai Ding. "A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based Nist Candidate KEMs". In: *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*. Springer. 2021, pp. 92–121.

[Raj+23] Gokulnath Rajendran, Ravi Ravi, Jan-Pieter D'Anvers, Shivam Bhasin, and Anupam Chattopadhyay. "Pushing the Limits of Generic Side-Channel Attacks on Lwe-Based KEMs-Parallel PC Oracle Attacks on Kyber KEM and Beyond". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023.2 (2023), pp. 418–446.

[Reg05] Oded Regev. "On Lattices, Learning With Errors, Random Linear Codes, and Cryptography". In: *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. 2005, pp. 84–93.

[RGK06] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. "Deniable Authentication and Key Exchange". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM, 2006, pp. 400–409. DOI: 10.1145/1180405.1180454. URL: https://doi.org/10.1145/1180405.1180454.

[RTV04] Omer Reingold, Luca Trevisan, and Salil Vadhan. "Notions of Reducibility Between Cryptographic Primitives". In: *Theory of Cryptography Conference*. Springer. 2004, pp. 1–20.

[Sam+19] Simona Samardjiska, Paolo Santini, Edoardo Persichetti, and Gustavo Banegas. "A Reaction Attack Against Cryptosystems Based on LRPC Codes". In: *Progress in Cryptology–LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings 6*. Springer. 2019, pp. 197–216.

[SE94] Claus-Peter Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". In: *Math. Program.* 66 (1994), pp. 181–199.

[Sho01] Victor Shoup. *A Proposal for an ISO Standard for Public Key Encryption (Version 2.1)*. Manuscript. http://shoup.net/papers/. 2001.

[Sho04] Victor Shoup. "Sequences of Games: A Tool for Taming Complexity in Security Proofs". In: *Cryptology Eprint Archive* (2004).

[Sho94] Peter W Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Sim98] Daniel R Simon. "Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?" In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1998, pp. 334–345.

[SM23] Douglas Stebila and Michele Mosca. *OQS OpenSSL*. https://github.com/open-quantum-safe/openssl, August 2023. 2023.

[SSW20a] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1461–1480.

[SSW20b] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "Post-Quantum TLS Without Handshake Signatures". In: *Cryptology ePrint Archive, Report 2020/534* (2020). https://eprint.iacr.org/2020/534.

[SSW21] Peter Schwabe, Douglas Stebila, and Thom Wiggers. "More Efficient Post-Quantum KEMTLS With Pre-Distributed Public Keys". In: *Cryptology ePrint Archive, Report 2021/779* (2021). https://eprint.iacr.org/2021/779.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. "Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model". In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Cham: Springer International Publishing, 2018, pp. 520–551. ISBN: 978-3-319-78372-7.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. "Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms". In: *Theory of Cryptography Conference*. Springer. 2016, pp. 192–216.

[Uen+22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. "Curse of Re-Encryption: A Generic Power/Em Analysis on Post-Quantum KEMs". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022), pp. 296–322.

[UG15] Nik Unger and Ian Goldberg. "Deniable Key Exchanges for Secure Messaging". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM, 2015, pp. 1211–1223. DOI: 10.1145/2810103.2813616. URL: https://doi.org/10.1145/2810103.2813616.

[UG18] Nik Unger and Ian Goldberg. "Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging". In: *Proc. Priv. Enhancing Technol.* 2018.1 (2018), pp. 21–66. DOI: 10.1515/popets-2018-0003. URL: https://doi.org/10.1515/popets-2018-0003.

[Unr15] Dominique Unruh. "Revocable Quantum Timed-Release Encryption". In: *Journal of the ACM (JACM)* 62.6 (2015), pp. 1–76.

[Vat+20] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. "On the Cryptographic Deniability of the Signal Protocol". In: *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part II*. Ed. by Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi. Vol. 12147. Lecture Notes in Computer Science. Springer, 2020, pp. 188–209.

[Vaz98] Umesh Vazirani. "On the Power of Quantum Computation". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 356.1743 (1998), pp. 1759–1768.

[Xag+21] Keita Xagawa, Akira Ito, Rei Ueno, Junko Takahashi, and Naofumi Homma. "Fault-Injection Attacks Against NIST's Post-Quantum Cryptography Round 3 KEM Candidates". In: *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*. Springer. 2021, pp. 33–61.

[Yam+15] Takashi Yamakawa, Shota Yamada, Takahiro Matsuda, Goichiro Hanaoka, and Noboru Kunihiro. "Reducing Public Key Sizes in Bounded CCA-Secure KEMs With Optimal Ciphertext Length". In: *Information Security*. Springer, 2015, pp. 100–109.

[Yue+21] Tsz Hon Yuen, Muhammed F. Esgin, Joseph K. Liu, Man Ho Au, and Zhimin Ding. "DualRing: Generic Construction of Ring Signatures With Efficient Instantiations". In: *CRYPTO (1)*. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 251–281.

[YZ17] Yu Yu and Jiang Zhang. *Lepton: Key Encapsulation Mechanisms From a Variant of Learning Parity With Noise*. NIST Round 1 Submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. 2017.

[Zha+16] Cong Zhang, David Cash, Xiuhua Wang, Xiaoqi Yu, and Sherman SM Chow. "Combiners for Chosen-Ciphertext Security". In: *International Computing and Combinatorics Conference*. Springer. 2016, pp. 257–268.

[Zha12] Mark Zhandry. "Secure Identity-Based Encryption in the Quantum Random Oracle Model". In: *Annual Cryptology Conference*. Springer. 2012, pp. 758–775.

[Zha13] Mark Zhandry. "A Note on the Quantum Collision and Set Equality Problems". In: *arXiv Preprint arXiv:1312.1027* (2013).

[Zha19] Mark Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 239–268.

Curriculum Vitae

Loïs Huguenin-Dumittan

Date of Birth	12.08.1995
Place of Birth	Neuchâtel
Nationality	Swiss

Education

2019-2024	**PhD, Computer and Communication Sciences** Supervision: Prof. Serge Vaudenay Area: Post-Quantum Cryptography LASEC, Ecole Polytechnique Fédérale de Lausanne (EPFL)
2016-2019	**MSc, Communication Systems** Ecole Polytechnique Fédérale de Lausanne (EPFL)
2013-2016	**BSc, Communication Systems** Ecole Polytechnique Fédérale de Lausanne (EPFL)

Work Experience

2019	**Research Engineer** LASEC, Ecole Polytechnique Fédérale de Lausanne (EPFL)
2017-2018	**Software Engineer Intern** ELCA Informatique SA, Lausanne, Switzerland

Experience as Teaching Assistant

2023-2024	**Cryptography and Security**
2020-2021	Prof. Serge Vaudenay
2022-2023	**Advanced Information, Computation, Communication**
	Prof. Tanja Käser
2021-2022	**Information Security and Privacy**
	Prof. Jean-Pierre Hubaux
2021-2022	**Networks out of Control**
	Prof. Patrick Thiran and Prof. Matthias Grossglauser
2020-2021	**Advanced Cryptography**
	Prof. Serge Vaudenay
2019-2020	**Information, Computation, Communication**
	Dr. Mirjana Stojilovic and Dr. Martin Rajman
2019-2023	**Project supervision**
	Supervised 10 Master semester projects/Master theses

Languages

French	Native
English	Fluent
German	Intermediate (Baccalaureate level)
Scottish Gaelic	Intermediate (reading/writing)

Programming Languages

C/C++, Java, Python, Bash. Some knowledge of Rust, Perl, Scala, Sagemath, VHDL, R, and Matlab.

Awards

EDIC PhD Fellowship, EPFL, 2018

Publications

1. Loïs Huguenin-Dumittan and Serge Vaudenay. Impossibility of Post-Quantum Shielding Black-Box Constructions of CCA from CPA. *Communications in Cryptology, Volume 1. IACR, 2024.*

2. Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures. *USENIX Security'24.*

3. Khashayar Barooti, Alex B. Grilo, Loïs Huguenin-Dumittan, Giulio Malavolta, Or Sattath, and Quoc-Huy Vu. Public-Key Encryption with Quantum Keys. In Guy Rothblum and Hoeteck Wee, editors. *Theory of Cryptography – TCC 2023, Lecture Notes in Computer Science, Volume 14372. Springer, 2023.*

4. Khashayar Barooti, Daniel Collins, Simone Colombo, Loïs Huguenin-Dumittan and Serge Vaudenay. On Active Attack Detection in Messaging with Immediate Decryption. In Helena Handschuh and Anna Lysyanskaya, editors. *Advances in Cryptology – CRYPTO 2023, Lecture Notes in Computer Science, Volume 14084. Springer, 2023.*

5. Daniel Collins, Simone Colombo, and Loïs Huguenin-Dumittan. Real World Deniability in Messaging. *Extended abstract of a talk given at RWC 2023. https://eprint.iacr.org/2023/403.pdf*.

6. Loïs Huguenin-Dumittan and Serge Vaudenay. On IND-qCCA Security in the ROM and Its Applications. In Orr Dunkelman and Stefan Dziembowski, editors. *Advances in Cryptology – EUROCRYPT 2022, Lecture Notes in Computer Science, volume 13277. Springer, 2022.*

7. Loïs Huguenin-Dumittan and Serge Vaudenay. FO-like Combiners and Hybrid Post-Quantum Cryptography. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors. *Cryptology and Network Security – CANS 2021, Lecture Notes in Computer Science, volume 13099. Springer, 2021.*

8. Loïs Huguenin-Dumittan and Iraklis Leontiadis. A Message Franking Channel. In Yu Yu and Moti Yung, editors. *Information Security and Cryptology – Inscrypt 2021, Lecture Notes in Computer Science, volume 13099. Springer, 2021.*

9. F. Betül Durak, Loïs Huguenin-Dumittan, and Serge Vaudenay. BioLocker: A Practical Biometric Authentication Mechanism Based on 3D Fingervein. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors. *Applied Cryptography and Network Security – ACNS 2020, Lecture Notes in Computer Science, volume 12147. Springer, 2020.*

10. Loïs Huguenin-Dumittan and Serge Vaudenay. Classical Misuse Attacks on NIST Round 2 PQC. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi,

editors. *Applied Cryptography and Network Security – ACNS 2020, Lecture Notes in Computer Science, volume 12146. Springer, 2020.*

11. Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. Misuse Attacks on Post-quantum Cryptosystems. In Yuval Ishai and Vincent Rijmen, editors. *Advances in Cryptology – EUROCRYPT 2019, Lecture Notes in Computer Science, volume 11477. Springer, 2022.*