Updatable Public Key Encryption in the context of Secure Messaging

Calvin Abou Haidar

THÈSE
en vue de l'obtention du grade de Docteur, délivré par
l'ÉCOLE NORMALE SUPERIEURE DE LYON

École Doctorale N°512
École Doctorale en Informatique et Mathématiques de Lyon

Discipline : Informatique

Soutenue publiquement le 09/02/2024, par :

Calvin Abou Haidar

Chiffrement à Clé Publique dans le Cadre de la Messagerie Sécurisée

---

Devant le jury composé de :

HOFHEINZ, Dennis	Professeur, ETH Zurich	Rapporteur
BLAZY, Olivier	Professeur des Universités École Polytechnique	Rapporteur
FONTAINE, Caroline	Directrice de Recherche Université Paris-Saclay	Examinatrice
ROSSI, Melissa	Chercheuse, ANSSI	Examinatrice
LOIDREAU, Pierre	Chercheur, Université Rennes 1	Examinateur
STEHLÉ, Damien	Professeur des Universités CryptoLab	Examinateur
PASSELÈGUE, Alain	Chargé de Recherche CryptoLab	Examinateur
SALVY, Bruno	Directeur de Recherche ENS de Lyon	Directeur de Thèse

RÉSUMÉ EN FRANÇAIS

Cette thèse s'intéresse à la construction de primitives cryptographiques, dont l'application principale est la messagerie sécurisée de groupe. Le développement récent et l'adoption de nombreuses applications de messagerie ont vu pour conséquence un intérêt particulier de la communauté cryptographique pour les sujets qui lui sont liés. Ces systèmes possèdent leurs besoins propres. On les utilise très souvent pour avoir des discussions longues, de groupes, vidéos, etc. Ces applications fonctionnent en temps réel et ne peuvent donc pas se permettre d'utiliser des primitives lourdes. On y retrouve le compromis classique entre sécurité et vitesse. Une application de messagerie instantanée qui mettrait plusieurs minutes à recevoir un message, aussi sécurisée qu'elle soit, serait probablement autant utilisée qu'elle est instantanée. Pourtant, l'adoption large d'applications réellement sécurisées est un enjeu pour la protection de la vie privée. En imposant un coût, même moyennement élevé, pour compromettre la conversation d'un individu, on augmente substantiellement celui de la surveillance de masse.

Les conversations s'établissant sur plusieurs années, la messagerie sécurisée rentre dans le régime de la cryptographie de long terme. On s'intéresse ici à la notion de "forward secrecy" ou *confidentialité persistante* qui vise à garantir à un utilisateur que son historique de conversation est protégé, quand bien même son appareil viendrait d'être compromis. Cette notion correspond à un besoin bien réel. Il est facile d'imaginer un scénario dans lequel une entreprise/un gouvernement enregistrerait tous les messages sortant de l'appareil d'une personne avec pour but de les déchiffrer plus tard une fois l'appareil compromis. Dans le cas d'une application à confidentialité persistante, l'attaquant se retrouverait avec les clés secrètes de l'appareil de la victime, mais serait tout de même incapable de déchiffrer les anciens messages. Naturellement, une telle propriété implique une évolution des clés secrètes utilisées. Si les mêmes clés sont conservées tout au long de l'utilisation, compromettre ces clés revient à compromettre tous les messages. Une manière simple d'obtenir la confidentialité persistante est de remettre à jour régulièrement ses clés en calculant de nouvelles et en envoyant les informations nécessaires aux correspondants.

On s'intéresse ici à une nouvelle méthode proposée par Dodis et al. en 2021 dans [39]. Il s'agit cette fois, non pas de recalculer de nouvelles clés, mais de profiter de la linéarité de certains systèmes de chiffrement à clé publique pour re-randomiser les clés publiques. En chiffrant l'aléa utilisé pour re-randomiser une clé publique sous cette même clé, on offre la possibilité à n'importe qui de mettre à jour n'importe quelle clé publique. D'abord étrange, cette propriété s'avère pratique dans le cadre de systèmes complexes utilisant des clés publiques partagées. Dans leur article, Dodis et al. proposent une définition formelle de cette nouvelle primitive appelée "Updatable

Public Key Encryption" (UPKE), qu'on pourrait cavalièrement traduire en *Chiffrement à Clé Publique Évolutive*. Leurs constructions sont avant tout théoriques, en cela qu'elles manipulent des objets de tailles déraisonnables au vu du contexte.

Dans le Chapitre 4, on présente la première construction efficace d'UPKE. Ce travail est le fruit de l'article [4]. On y reprend le schéma de chiffrement à clé publique de Elgamal-Pailler introduit dans [27], pour montrer qu'il est possible de l'étendre en un UPKE sécurisé. On montre ensuite, en utilisant le paradigme de Naor-Yung [70], qu'il est possible, à relativement faible coût, de rendre cet UPKE sécurisé contre des adversaires actifs et ayant accès à un oracle de déchiffrement (sécurité CCA).

Dans le Chapitre 5, on présente une seconde construction, dont l'efficacité est moindre que la première, mais qui reste du même ordre de grandeur. Cette fois-ci, notre construction se base sur une hypothèse post-quantique (LWE), ce qui en fait la première construction efficace reposant sur de telles hypothèses. Ce travail est le fruit de l'article [50]. On y introduit une nouvelle hypothèse appelée "Adaptive Extended LWE", qui étend l'hypothèse "Extended LWE" en prenant en compte une intéraction supplémentaire. On réduit ensuite LWE à notre nouvelle hypothèse. Adaptive Extended LWE nous permet de construire un UPKE en étendant le schéma de chiffrement KYBER [22], récemment standardisé par le NIST. Comme autre contribution importante, on adapte la technique classique de Fujisaka-Okamoto [46] au cadre de l'UPKE, ce qui nous permet d'obtenir sans coût supplémentaire la sécurité contre des adversaires passifs ayant accès à un oracle de déchiffrement.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

PKE Public Key Encryption

UPKE Updatable Public Key Encryption

KEM Key Encapsulation Mechanism

UKEM Updatable Key Encapsulation Mechanism

NIZK Non-Interactive Zero Knowledge

LWE Learning With Errors

MLWE Module Learning With Errors

RSA Rivest Shamir Adlemann

DCR Decision Composite Residuosity

ROM Random Oracle Model

FO Fujisaki-Okamoto

NY Naor-Yung

CPA Chosen Plaintext Attack

CCA Chosen Ciphertext Attack

Part I

INTRODUCTION AND BACKGROUND

1

GENERAL INTRODUCTION

ABOUT CRYPTOGRAPHERS, CRYPTOGRAPHY AND THEIR WORLD

Les dieux avaient condamné Sisyphe à rouler sans cesse
un rocher jusqu'au sommet d'une montagne d'où la pierre
retombait par son propre poids. Ils avaient pensé avec
quelque raison qu'il n'est pas de punition plus terrible
que le travail inutile et sans espoir.

— Albert Camus, Le mythe de Sisyphe

The pursuit of advancements in the field of cryptography can be likened to a Sisyphean task, wherein researchers continually strive to overcome an ever-escalating mountain of challenges and complexities. The relentless march of technological progress, coupled with the ever-evolving strategies of malicious actors, ensures that cryptographers are engaged in a perpetual battle to safeguard sensitive information and communications.

Much like Sisyphus' boulder, the security breakthroughs in cryptography inevitably suffer setbacks from ingenious attempts to breach them, fueling – hopefully – a perpetual cycle of innovation. It is, however, less challenging to imagine cryptographers happy, for their passion is intimately linked to this ceaseless task.

The following thesis focuses on going up, pushing the boulder a bit further, until gravity – or *cryptanalysts*, as we call them – makes its work.

Before going up, it is only natural to start by assessing our current whereabouts. This work is about building basic tools for cryptography, called *primitives*. Primitives are the fundamental blocks of cryptography, which allow the constructions of sophisticated protocols used in day-to-day digital life. Any complex system can be analyzed as a network of interacting, simple(r) components. The essential nature of modularity is well known by engineers, who are concerned by the ever-growing complexity of their creations and the impact this growth has on their ability to understand them. Akin to craftsmanship, building cryptographic primitives demands an intimate mastery of the raw materials at play. Cryptographic assumptions serve the cryptographer much as wood, stone or ivory do for the sculptor, determining the array of techniques at their disposal and bestowing various properties to their works.

1.1 MODERN PUBLIC-KEY ENCRYPTION

Public-key encryption – also called asymetric encryption – is based on the hypothesis that it is possible to securely share private data using an insecure channel without any prior exchange. More informally, a natural consequence arising from the existence of public key encryption is the possibility for two persons that never met nor talked before to meet in any public space and start a conversation that no other eavesdropping party can understand. This statement is highly non trivial and has deep implications. Indeed, the existence of encryption schemes preceded public key cryptography and, until the 70's, relied on pre-existing exchanges to establish a shared secret. For any two parties to have such a secure conversation in public, they would have had to meet before and agree on some secret information that only them would know. A good analogy would be to consider some kind of secret language that would be devised beforehand and then used in public to convey private information. While this is easily conceivable, stating the possibility to devise such a language without any pre-shared secret is a rather surprising declaration. In public key cryptography, users usually hold a key pair consisting of a private key, which they keep to themselves, and public key, whose purpose is to be known by other users.

The hypothetical nature of the existence of public key encryption is of importance. The security of all existing schemes rely on assumptions made by cryptographers that ultimately rely on empirical experiments. To illustrate this, let us consider a very simple cryptographic assumption:

Assumption (Factoring problem - Informal) *Let* p, q *be prime numbers and* $N = p \cdot q$. *Given only* N*, it is hard to find* p *or* q.

While simplified, this assumption is still quite informative. It is clear that this assumption does not hold for small numbers. Claiming that it is hard to recover 3 and 5 given only 15 would be rather bold. The hidden complexity resides in the "hard" part of the assumption. What do we mean by "hard"? Any reasonable cryptographer would now go on a painful – but interesting – rant about complexity theory and its subtelties in the context of cryptography. Let us skip this part and say: a problem is hard if takes a long time to solve. Disappointing, but definitely enough for our needs. In the previous problem, considering large enough primes does the work. It would take a long time to factor a product of 1000-digit primes. We know this by estimating the time it would take to factor this number using the *best known* method for factoring integers, revealing the empirical side of cryptographic hardness. The security of cryptosystems based on any assumption is thus tied to the most effective method to break it and to the computation hardware available.

Cryptography being a pessimistic field by essence, theoretical breakthroughs must also be taken into account to judge of a system's security. Advances in the theory of quantum computing revealed the existence of a theoretically efficient algorithm to solve the factoring problem, by leveraging quantum mechanics properties with no

known classical counterparts. This algorithm, presented by Peter Shor in [77], is the first algorithm provably solving the factoring problem efficiently. Even though constructing powerful enough quantum computers to carry out the computations appears to be itself a daunting challenge, this breakthrough spurred the quest of finding cryptographic assumptions resisting even the attacks of quantum computers.

1.2 AGREEING: KEY EXCHANGE

While public key encryption has the unprecented property of not relying on preshared secrets, it also suffers from a major drawback: it involves heavier operations and communication costs than its symmetric counterpart. Resorting to hybrid encryptions schemes solves this issue. A public key phase allows the parties to agree on some common secret information, which is immediatly used to switch to symmetric encryption. In their seminal work of 1976 [36], Diffie and Hellman provide a way to implement this public key phase, by introducing a Key Exchange Protocol. A Key Exchange Protocol (KE) allows two parties to agree on a common symmetric encryption key using their public keys. Their KE, dubbed "DH", after their initials, relies upon the hardness of the *Diffie–Hellman* problem. While simple to understand, proving security of Key Agreement protocols has been challenging. Security proofs of a protocol first require to design a security model that captures the behaviour of real-world adversaries. The multiple possible interactions between the different components of a protocol make such models complex, and full security proofs notouriously hard to achieve. We refer to [61] for a discussion around the necessary modularity in Key Agreement protocols construction, in order to achieve simpler security proofs. They provide a modular framework for constructing and analyzing security of such protocols, extending the works of [11, 12, 16].

The Diffie–Hellman problem states that, given an element g of a cyclic group $\mathbb{G}$ of order p and g^a, g^b for a, b chosen integers modulo p, it is hard to distinguish the value g^{ab} from uniform.

The DH key agreement scheme has been extended through the years. Blake-Wilson *et al.* introduce in [16] the triple DH protocol (3-DH), which introduces ephemeral keys sampled by each parties during the protocol, on top of the long term static keys used in DH. These ephemeral keys are used in order to achieve several security properties, some of which we discuss later on in this chapter. In 2016, Marlinspike and Perrin [67] introduce the extended triple Diffie-Hellman key exchange protocol "X3DH", following the works of [16, 61]. Their protocol was meant to be used in the context of secure messaging, as the key exchange phase for their messaging application Signal. It is now also used by other major messaging applications such as Whatsapp or Facebook Messenger. A recent work of Hashimoto *et al.* [51] (in the lines of [26]) proposed a replacement for X3DH, based on quantum-resistant assumptions. This lead to the construction of "PQXDH" [60] (Post-Quantum Extended Diffie-Hellman), very recently adopted by Signal.

1.3 SECURE MESSAGING AND THE SIGNAL PROTOCOL

Once they have agreed on a secret, users are able to communicate securely. Their conversation is then handled using the secure messaging protocol of their application. As shown in the survey of Ermoshina *et al.* [42], secure messaging has been implemented in different flavours. However, the Signal protocol seems to have evolved to be the standard protocol in the messaging market, as most of its competitors either adopted the protocol itself or some variants.

A lot of cryptographic effort has been spent in studying, improving, or attacking the Signal protocol (see [4, 14, 29, 30, 37] for instance). Influenced by the Off-The-Record (OTR) [21] communication protocol, the double ratchet protocol is the cornerstone of Signal's messaging session. It is based on a generalization of the old concept of re-keying – also known now as symmetric ratcheting – which aims at extending a symmetric secret's lifetime by deterministically deriving a new secret that will be used in the future. Informally, if k is a secret known by both parties communicating and f is a deterministic function satisfying some security properties, rather than using k directly, users can derive new keys $k_1, k_2, \ldots$ from k, by applying f. These subkeys can then be used in cryptographic operations instead of the original key k. Repeating this process allows users to avoid using the same key several times during their communication, while also updating their shared keys synchronously, since f is deterministic. Abdalla and Bellare show in [1], that re-keying, conformally to our intuition, improves a system's security if properly used. They achieve this by proving that a "secure" re-keying process acts like stateful pseudorandom generator, which implies that generated subkeys look independent. Merlinspike and Perrin adapt the idea of symmetric ratcheting (or re-keying) to the public key setting. Their idea is to alternate between symmetric ratcheting phases and public key ratcheting phases, using DH to non-interactively establish a new secret once a public key is updated. Users A (holding public key pk_A) and B (holding public key pk_B) would use DH to agree on a secret k. Then the first user, say A, to send a message would use symmetric ratcheting with k to send messages. Once B wants to respond, it updates its public key to pk'_B, performs a new DH round to establish a new symmetric secret k' that it will use to send its messages.

This simple idea has several pleasant security consequences. First of all, if either of the parties' secret state is exposed, the symmetric ratcheting phase allows to protect any previously sent messages as the symmetric key is always being updated and it should not be possible to invert the updates. Secondly, updating the public keys allows both parties to recover if they are compromised by an adversary that remains passive (meaning that the adversary does not try to send messages to interfere with the protocol execution). The first property is called *forward secrecy* and the second *post-compromise security*.

We are only interested here in those two properties as they are related to the contribution of this thesis. Of course, many other properties can have some importance such as anonomity, deniablity...

1.4 FORWARD-SECRECY AND POST-COMPROMISE SECURITY

Ensuring that past communications are protected or recovering after compromise are not new ideas. Such considerations in public key cryptography can be traced back to the end of the 90's, where several works such as [10, 28, 73] started to introduce pritimives satisfying related properties. Authors of the OTR protocol emphasize in [21] the difference between classical usecases of cryptography and its use in secure messaging. This stems from the fact that, being used in daily communication, secure messaging consists of multiple long-lived sessions. Careful care then needs to be taken in which values are important for the protocols' security, how long they live and how much they are actually used during different sessions.

In a recent work, Blazy *et al.* [17] provide fine-grained security models and a metric to analyze post-compromise security in a context related to secure messaging. This is a considerable task, as different scenarios can be considered in which the difficulty of achieving either properties varies. In group messaging, one might consider adversaries that fully control the network or not, that potentially collude with group members at some point or not... This makes for a variety of scenarios. This matters as, for instance, it appears obvious that no post-compromise security can be achieved against an active adversary that has fully compromised any member of a group, as it can hijack the communication to impersonate that party. They also take into account the *lifetime* of secret keys used in the protocol, of which we underlined the importance earlier. Another interesting property to consider in post-compromise security is the healing speed, i.e. how many messages need to be exchanged in order to ensure recovery after a leak. The metric given in [17] allows to compare healing speeds in several 2-party protocols. Such comparisons are useful in order to find out protocols suited to the security needs. Critical systems might require fast post-compromise security, even with some cost efficiency penalty, while others would tolerate a slower healing speed.

By lifetime of a value, we refer to the scope of its usage (is it re-used in several sessions? Specific to a session?).

While simpler to achieve, forward-secrecy has also its complexities. Even though Signal provides a very efficient solution by leveraging its symmetric layer, it cannot be applied everywhere. In situations where no shared secret is established, forward-secrecy must rely solely on the properties of the public key schemes involved. The first construction of forward secure public key primitives were proposed by Bellare and Miner [10] (signature) and Canetti *et al.* [28] (encryption). We defer the discussion about the Canetti *et al.* construction to the technical introduction, but an important aspect to underline is that their solution, while of theoretical interest, is highly inefficient. This is yet another gap between symmetric and asymetric cryptography, as no real efficient equivalent of re-keying seems to exist in the public key world. The basic intuition behind achieving forward secrecy is presented in [10]: any secret value has to evolve during time in order to protect previously sent data. Note that different primitives might not have the same needs in terms of forward secrecy. Holding a signature key for long is not a security risk in terms of forward-secrecy, as compromise of that

Signature and encryption here act in an opposite ways in some sense. Signature key leakage is a problem for future communications (impersonation), while encryption key leakage a problem for past communications.

signature key has no effect on previous signatures. Leakage of a long-term encryption key actually allows to decrypt all previously encrypted messages. Signatures are considered in [10], introducing a notion of forward-secrecy akin to timestamping. A signature key is associated with a tag T, which is bound to this key in the verification process. The next key gets assigned another tag, which is publicly computable from the previous one. The signer can then erase the previous key, ensuring that the key associated to tag T can no longer be compromised. Encryption has a more natural definition of forward secrecy. Each public key that is used to encrypt data needs to evolve in order to ensure protection of previously sent messages. As noted earlier, key evolution is implemented in the Signal protocol by symmetric ratcheting for symmetric keys and by sampling new public keys for public key encryption.

This thesis focuses on forward secrecy in the context of secure messaging. Our main focus is the efficient construction of a recently introduced primitive called Updatable Public Key Encryption (UPKE). UPKE allows to adopt a new and simple strategy of public key evolution by leveraging homomorphic properties of some public key encryption schemes. While not yet implemented in real-world systems, we believe that this primitive can have an impact and that efforts in developping it to better understand the associated security properties are meaningful.

2

TECHNICAL INTRODUCTION

As seen earlier, forward secrecy requires key updates. Forward secure encryption schemes (FS-PKE) generate an initial keypair $(\mathsf{pk}_0, \mathsf{sk}_0)$ and provide a way to derive a chain of subsequent new keypairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2), \ldots$ where each public key can be publically derived from the previous one. The notion of FS-PKE is actually very strong, as it implies non-interactive updates of the keypairs and a polynomially bounded number of possible updates. This gives a hint about why no efficient construction of FS-PKE is known to this day. The first construction of FS-PKE comes from [28] and relies on hierarchical identity-based encryption (HIBE) [54]. A consequence of the existence of unbounded HIBE is a public key version of re-keying. Indeed, in an unbounded HIBE scheme, any secret key sk_l at level l allows to derive new keys at level $l+1$. The level 0 key is the master secret key msk. These keys always correspond to the same master public key mpk together with the current identity id_j that is being used, for $j \geqslant 0$ (here, we assume that id_{j+1} can be publically derived from id_j). One can instantiate an FS-PKE keypair chain by taking $\mathsf{pk}_0 = (\mathsf{mpk}, \mathsf{id}_0), \mathsf{pk}_1 = (\mathsf{mpk}, \mathsf{id}_1), \ldots$ and then derive secret keys by starting with $\mathsf{sk}_0 = \mathsf{msk}$ and deriving sk_l from sk_{l-1} for any $l > 0$. While functional, this approach inherits the major inefficiencies of the HIBE constructions. For now, HIBE schemes remain mostly of theoretical interest and relying on them does not seem very promising.

Another interesting approach is given in [35], where forward secrecy is achieved through an efficient instantiation of puncturable encryption using a probabilistic data structure (bloom filter). This comes at the cost of allowing key sizes to grow linearly with the number of messages to protect, as the solution of [35] relies on a user generating a finite set of keys which get progressively punctured. It relies on the Boneh-Franklin identity based encryption (IBE) scheme [18], in which the master secret key is used to generate multiple identity keys. The master secret key is then removed and the user is left with a set of identity keys, which get progressively deleted as soon as they are used to decrypt a message. Here, the public keys pk_i always correspond to the master public key pk, and the secret keys form a chain of strict inclusions $\mathsf{sk}_0 \supsetneq \mathsf{sk}_1 \supsetneq \mathsf{sk}_2 \ldots$ One drawback is the size of the original key set sk_0, as it must contain at least as many keys as the number of messages to be protected. In [35], deletion is handled through a probabilistic structure, which needs to work with a large number of keys to compensate for the potential errors. This becomes an even worse concern in the lattice setting, where even the most efficient IBE [40] would end up needing to generate gigabytes of secret keys.

Recently introduced in [5, 39], Updatable Public Key Encryption (UPKE) provides another approach which this times relaxes the non-interactivity of FS-PKE. Allowing interactions in the update process gives hopes to achieve more efficient constructions. In a UPKE, users are allowed to update any keypair by running an update algorithm. This algorithm relies on some randomness sampled by the caller and generates a public key and some private update information. This update information needs to be sent to the owner of the keypair, which then proceeds to an update of its secret key accordingly.

Formally, UPKE extends the syntax of PKE by adding new algorithms (UpdatePk, UpdateSk). Given any public key pk_t for some epoch $t > 0$ as input, the UpdatePk algorithm outputs a new public key pk_{t+1} together with an update message up. The update message contains the relevant private data that should be processed by the owner of pk_t in order to compute the secret key associated to pk_{t+1}. In practice, the update message contains an encryption under pk_t of the random coin r used by the updater to run UpdatePk. The security property of UPKE captures the idea that any honest update should protect previously sent messages even if the updated secret key leaks. This is formaly defined in the IND-CR-CPA security notion. It states that even if the adversary is given the ability to choose the randomness for all updates (CR stands for Chosen Randomness) but the last one, which is honestly made, leaking the final secret key does not compromise security of the scheme. IND-CR-CPA was formalized in [39], but a related notion is already present in [57]. To capture the ability of the adversary to make updates, adversary is given access to an update oracle, which takes as input a random coin r, and runs the update algorithm UpdatePk with input the public key of the current epoch and uses randomness r. Note that this captures the behaviour of *honest* adversaries, which actually run the UpdatePk algorithm for updates. The CCA variant IND-CR-CCA is analogous to IND-CCA, where the adversary is given access to a decryption oracle. IND-CR-CCA is however a strange hybrid where the adversary has access to a decryption oracle, meaning that more care needs to be given to regular ciphertexts to achieve security, but updates are still honestly made. In practice, the possibility of updating a key not owned by the updater can be puzzling and it seems much easier to craft malicious updates than to have access to a decryption oracle. Stronger security models are required in order to prevent ill-usages. The notion of Chosen Update (CU), is introduced in [39] for that purpose. IND-CU-CPA for instance, compared to IND-CR-CPA, captures the scenarios where the adversary can craft *any* update. It also extends the syntax of UPKE to add a VerifyUpdate algorithm that allows to publicly check than an update is wellformed. Such an algorithm is essential in practice, as it avoids the owner of the key that was updated to have to confirm that the update was honest. This is particularly useful in the context of secure messaging as the updatee may not even be online when the update is performed, which means that a malicious update would remain undetected until they return. This type of *denial-of-*

service attack, even without any effect on security, could break the correctness for the owner of the key and provide for a malicious way to kick users out of group.

While recent, this primitive has been constructed under several assumptions. The first construction was proposed in [57] and relies on the Computational Diffie-Hellman (CDH) assumption in the Random Oracle Model (ROM). This is still by far the most efficient construction one can get with pre-quantum assumptions. The first instantiations of UPKE in the standard model were given in [39], under the Decisional Diffie-Hellman (DDH) and Learning With Errors (LWE) assumptions. These two constructions are however quite inefficient as they rely on bit-by-bit encryptions for updates and require statistical leftover hash lemmas. The DDH construction is based on the BHHO [20] scheme, while the LWE one is based on the Dual-Regev [47, 75] cryptosystem. Both work with secret keys of the form $\mathbf{s} \in \{0,1\}^\ell$ and updates of the same form $\mathbf{r} \in \{0,1\}^\ell$, for ℓ a security parameter. After a single update, the secret then becomes $\mathbf{s}+\mathbf{r} \in \{0,1,2\}^\ell$. In both schemes, the public key can be seen as $f(\mathbf{s})$, for f a public function that is homorphic for the relevant group operation (we consider only $+$ to illustrate this). An update then consists of the updated public key $f(\mathbf{s})+f(\mathbf{r}) = f(\mathbf{s}+\mathbf{r})$ and a bit-by-bit encryption of $\mathbf{s}$. The encryption of $\mathbf{r}$ is the update information that is sent to the owner of the secret key $\mathbf{s}$ for it to accordingly update its keypair. Both schemes were shown to achieve circular security (CS) and leakage resilience (LR).

To prove IND-CR-CPA security of the schemes, Dodis *et al.* [39] introduce a new security property that the underlying PKE scheme should retain IND-CPA security even after the adversary is given $\mathbf{s}+\mathbf{r}$ for $\mathbf{r}$ uniformly sampled in $\{0,1\}^\ell$ and a bit-by-bit encryption of $\mathbf{s}$. They call it CS+LR security, as $\mathbf{s}+\mathbf{r}$ can be seen as a leakage on the original key $\mathbf{s}$ and it is given at the same time as an encryption of the secret key $\mathbf{s}$. CS+LR security actually corresponds to IND-CR-CPA security in which the adversary decides to make no update. Once a scheme is proven to be CS+LR secure, one can handle IND-CR-CPA security by making a reduction to CS+LR security. The reduction works by showing that the updates made by the adversary can be handled on the fly using homorphic properties of the PKE, assuming it has such properties.

While the Dual-Regev based construction of [39] provides a post-quantum UPKE, their construction is mainly of theoretic interest. The estimates to instantiate their scheme achieving 128 bits of security gives updates of size 360 kB and ciphertexts of 33 kB. Real world applicability is clearly out of reach for sizes of this order of magnitude.

Upgrading IND-CR-CPA to IND-CR-CCA/IND-CU-CCA security is done through generic transforms in [39]. However, they rely on one-time strong True-Simulation f-Extractable Non Interactive Zero Knowledge (f-tSE NIZK) argument [38]. While instantiable in the standard model, it is unknown whether or not it is possible to have efficient constructions of those NIZKs.

Dodis et al. [38] show that building a CCA secure encryption scheme together with a NIZK for some specific language in NP suffices to build a f-tSE NIZK

This raises the following questions:

Q1. Is it possible to construct an efficient IND-CR-CPA UPKE scheme in the standard model ?

Q2. Is it possible to construct an efficient post-quantum IND-CR-CPA UPKE scheme ?
Q3. Is it possible to efficiently achieve IND-CU-CCA security ?

We answer those questions affirmatively, by providing two constructions of efficient IND-CR-CPA secure UPKEs and several efficient transforms that allow to achieve IND-CU-CCA security.

2.1 A CONSTRUCTION UNDER THE DCR ASSUMPTION

Chapter 4 presents our first construction of an efficient IND-CR-CPA UPKE under the Decisional Composite Residuosity (DCR) assumption in the standard model. We then extend the construction to obtain a practical IND-CR-CCA secure UPKE in the Random Oracle Model. Finally, we adapt the extension to achieve IND-CU-CCA which this time requires the additional Strong-RSA assumption for its security, still in the Random Oracle Model.

2.1.1 *Building an IND-CR-CPA UPKE*

We start with the Elgamal-Paillier PKE scheme [27]. This scheme is particularly interesting for two reasons. The first is that it has a message space than can be taken to be arbitrarily large via a parameter $\zeta \geqslant 1$. We consider an RSA modulus $N = PQ$, for primes P, Q of the form $P = 2p + 1, Q = 2q + 1$, with p, q that are also primes. Damgård *et al.* show in [34] that the Elgamal-Paillier cryptosystem can be set up such that it works over $\mathbb{Z}_{N^{\zeta+1}}$, with message space $\mathbb{Z}_{N^\zeta}$. For the IND-CR-CPA construction, we use only $\zeta = 1$, but we take $\zeta = 2$ for the IND-CU-CCA construction. The second reason is that Malkin *et al.* showed in [66] that this scheme is circular secure, and even Key Dependent Message (KDM) secure with a slight modification of the construction.

The scheme starts by computing a random generator g of the subgroup of $\mathbb{Z}_{N^{\zeta+1}}$ of order $\phi(N)/4 = pq$. Note that this order is unknown to the users, as it can be used to factor N and thus break security. In this subgroup, computing a discrete logarithm is hard. However, there exists another subgroup of order N^ζ generated by $T = 1 + N$, in which computing a discrete logarithm can easily be done using an algorithm given in [34]. A secret key consists of a random element $x \hookleftarrow \mathcal{U}([0, (N-1)/4])$ and the corresponding public key is $h = g^x \bmod N^{\zeta+1}$. Encrypting a message $m \in \mathbb{Z}_{N^\zeta}$ is done by sampling $t \hookleftarrow \mathcal{U}([0, (N-1)/4])$ and outputing the ciphertext

$$(c_0, c_1) = (g^t \bmod N^{\zeta+1}, h^t T^m \bmod N^{\zeta+1}).$$

For $\zeta = 1$, finding x such that $y = T^x \bmod N^2$ can be achieved by taking $(y-1)/N$, as $T^x = 1 + xN \bmod N^2$

To decrypt, it suffices to notice that $c_1 \cdot c_0^{-x} = T^m \bmod N^{\zeta+1}$, from which m can be recovered by computing the discrete logarithm.

To perform an update on a public key pk with this scheme, one can sample a random element $r \hookleftarrow \mathcal{U}([0, (N-1)/4])$, compute $up = \mathsf{Enc}(pk, r)$ and set the new public

key $\mathsf{pk}' = g^r \cdot \mathsf{pk}$. Notice how the size of the plaintext space already gives a clear advantage compared to the schemes given in [39], as the update now consists of a single ciphertext instead of a sequence of bit-by-bit encryptions. Updating the secret key is done by decrypting the ciphertext to recover r and set the new secret key sk' to be $\mathsf{sk} + r \in \mathbb{Z}$.

To prove security, we follow the steps of [39] by first proving CS+LR security. We actually consider a slightly different version of CS+LR security, where an adversary for IND-CPA additionally receives a leakage $\mathsf{sk} + r \in \mathbb{Z}$ and an encryption $\mathsf{Enc}(\mathsf{pk}, r)$ (instead of $\mathsf{Enc}(\mathsf{pk}, \mathsf{sk})$). This is still equivalent to the original definition when the underlying scheme is message-homorphic, but this modification makes our scheme more efficient for reasons detailed in Section 4.2.2. In the CS+LR game, the adversary receives a public key pk, a challenge ciphertext under pk together with a leakage $\mathsf{sk} + r$ and $\mathsf{Enc}(\mathsf{pk}, r)$, for $r \hookleftarrow \mathcal{U}([0, (N-1)/4])$. We start by leveraging the algebraic structure of the scheme to show that it is equivalent to work with $\mathsf{pk}' = g^{-r}$ instead of pk. This implies that the only information given to the adversary that depends on the original secret key sk is the leakage $\mathsf{sk} + r$. Taking sk to be exponentially bigger than r allows us to use a smudging lemma to argue that sk statistically hides r in the leakage $\mathsf{sk} + r$. This means that the leakage can be replaced by a uniform element, which leaks no information about r. Notice that now, as we work with pk' that depends only on r, we are left to prove a standard form of circular security, which can be done à la [66].

This simple construction is already quite efficient. A ciphertext consists of only 2 group elements and an update is a single ciphertext. For 128-bit security, we consider 3072-bit primes. A ciphertext/update is then of size 1.5 kB, which is a considerable improvement compared to the BHHO based construction. Indeed, with BHHO, a ciphertext is about 41 kB. Bit-by-bit encryption makes the update size even worse with at least 52 MB.

2.1.2 *Upgrading to IND-CR-CCA/IND-CU-CCA*

In order to upgrade our scheme's security we rely on the Naor-Yung paradigm [70]. It allows us to achieve CCA security simply by adding a ciphertext per encryption and a NIZK proof. This works by extending the public parameters of the scheme with a supplementary generator h_d of the subgroup of order pq. This additional generator works as a public key whose associated secret key should not be known by any party. To encrypt a message m under a public key pk, one then computes two ciphertexts $c_0 = \mathsf{Enc}(\mathsf{pk}, m)$ and $c_1 = \mathsf{Enc}(h_d, m)$, together with a NIZK proof π that both ciphertexts encrypt the same plaintext. In the security proof, the decryption oracle is handled by using h_d as a backdoor. By setting $h_d = g^x$ for some uniform x, the reduction can decrypt any wellformed ciphertext (c_0, c_1, π) as it has the guarantee (through soundness of the NIZK) that c_0 and c_1 encrypt the same message and it knows the

secret key to decrypt c_1. Notice that h_d is a public key that is never updated. In practice, this should not be a problem as the secret key associated to h_d is not known.

On a practical point of view, the NIZK proof is generated by compiling a Σ-protocol for plaintext equality with the Fiat-Shamir heuristic [44]. A proof then consists of 4 group elements and 4 integers. This amounts to a ciphertext of 8.3 kB for the CCA variant. While too much for real-world scenarios, this is still in an acceptable range, especially compared to the BHHO construction. We show later on how to achieve CCA security without this size blow-up.

The same rationale can be applied to achieve CU security. Indeed, by devising a sigma protocol for the language of wellformed updates (which we again compile to a NIZK proof system) and performing the same Naor-Yung paradigm transformation for the updates, we are able to reduce IND-CU-CCA security to IND-CR-CCA security. The NIZK proof of wellformedness allows us to argue that a witness exists (here, a witness is the randomness r used for the update), and the double encryption with NIZK Naor-Yung transform allow to extract that witness r and give it as input to the update oracle of the IND-CR-CCA game. At first glance, this only costs an additional NIZK proof. However, the major drawback of this technique is that we are compelled to work with $\zeta = 2$, which induces a blow-up in the group size. Indeed, we adapt a technique from [27] that is used to prove that a ciphertext encrypts the discrete logarithm of a given element of the group. Doing so introduces a soudness gap, as we are only able to prove membership of the witness (the randomness r) in a superset of $[-(N-1)/4, (N-1)/4]$. For this superset to fit in the plaintext space, we work in $\mathbb{Z}_{N^3}$.

2.2 A CONSTRUCTION UNDER THE LWE ASSUMPTION

In Chapter 5, we present our second efficient construction of UPKE. The main motivitation was to provide another efficient UPKE but this time based on a post-quantum assumption. The efficiency of the LWE based construction of [39] heavily suffered from their use of flooding techniques to prove security. Flooding relies on the use of super-polynomial modulus-to-noise ratio, allowing to remove unwanted noise by arguing that the statistical discrepancy introduced by the removal is negligeable. The challenge is then twofold. We first need, as for the previous construction, to find a lattice-based PKE scheme that has circular-security and leakage-resilience properties, and a sufficiently large plaintext space. Second, we need to avoid the use of noise flooding, in order to be able to work with a polynomial modulus, which is absolutely necessary to hope to achieve reasonable sizes.

2.2.1 *Building a post-quantum IND-CR-CPA UPKE*

A natural idea is to start with the recently standardized KYBER [22] PKE/KEM. It offers the advantages given by the structured lattice assumption Module Learning

With Errors (MLWE) [24, 62], small sizes and leeway to fix security parameters. We here however consider the scheme on the ring of integers $\mathbb{Z}$, which simplifies our study.

In [24], MLWE is actually introduced as Generalized Learning With Errors

In this integer version of KYBER, we fix two global parameters $q > p > 0$. A public key is an LWE sample consisting of a uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ and a vector $\mathbf{b} = \mathbf{As} + \mathbf{e} \in \mathbb{Z}_q^n$, with $\mathbf{s}, \mathbf{e}$ small vectors sampled from a discrete Gaussian distribution. An encryption of a message $\boldsymbol{\mu} \in \mathbb{Z}_p$ is computed as

$$\mathsf{ct}_0 = \mathbf{XA} + \mathbf{E},\ \mathsf{ct}_1 = \mathbf{Xb} + \mathbf{f} + \boldsymbol{\mu} \cdot \lfloor q/p \rfloor,$$

where $\mathbf{f}, \mathbf{X}$ and $\mathbf{E}$ are sampled from a discrete Gaussian distribution. Decryption can be done by rounding $\mathsf{ct}_1 - \mathsf{ct}_0\mathbf{s}$, using the relation:

$$\begin{aligned}\mathsf{ct}_1 - \mathsf{ct}_0\mathbf{s} &= \mathbf{Xb} + \mathbf{f} - (\mathbf{XA} + \mathbf{E})\mathbf{s} + \boldsymbol{\mu} \cdot \lfloor q/p \rfloor \\ &= \mathbf{Xe} + \mathbf{f} - \mathbf{Es} + \boldsymbol{\mu} \cdot \lfloor q/p \rfloor\end{aligned}$$

where the term $\mathbf{Xe} + \mathbf{f} - \mathbf{Es}$ is small.

Updating a public key $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ is done by sampling $\mathbf{r}$ and $\boldsymbol{\eta}$ from a Gaussian distribution and setting the new public key to be $(\mathbf{A}, \mathbf{b} + \mathbf{Ar} + \boldsymbol{\eta})$. The update information consists only of an encryption of $\mathbf{r}$. Notice that for the same reason that $\mathbf{e}$ is not needed for decryption, the vector $\boldsymbol{\eta}$ does not need to be transmitted to the key owner. The updated secret key is thus $\mathbf{s} + \mathbf{r}$. Unlike DCR, LWE inherently carries geometrical constraints. As noted earlier, correctness of decryption depends on the size of $\mathbf{Xe} + \mathbf{f} - \mathbf{Es}$, which grows with the key $\mathbf{s}$ and the noise $\mathbf{e}$. The sizes of both elements grow with updates, which means that it necessary to control the number of updates in order to guarantee correctness.

As a solution, we introduce a parameter k which controls the number of updates allowed in the scheme. Note that the size of the module q depends on k. For 128-bit security, having $k = 2^5$ updates allowed gives a scheme whose ciphertext size is 1.8 kB, roughly losing a 2.3 factor compared to KYBER's 0.8 kB ciphertexts. This difference is due to the necessary increased size of the modulus q. An update is a bit heavier, as it is an encryption of a vector, with 5.5 kB.

Sizes are given in the tables of Section 2.3.

2.2.2 *IND-CR-CPA security & a necessary new assumption*

As opposed to the previous construction, we tackle IND-CR-CPA security directly instead of going through CS+LR. An adversary for IND-CR-CPA first receives a public key $\mathsf{pk}_0 = (\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$. It then can make several update queries using random coins $(\mathbf{r}_0, \boldsymbol{\eta}_0), (\mathbf{r}_1, \boldsymbol{\eta}_1) \ldots$ and after asks for a challenge for a plaintext pair $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1)$ of its choice at epoch $\mathtt{chall}$. This challenge is computed as an encryption of one of the two plaintexts under the public key

$$\mathsf{pk}_{\mathtt{chall}} = (\mathbf{A}, \mathbf{b} + \mathbf{A}\Delta^{\mathbf{r}}_{\mathtt{chall}} + \Delta^{\boldsymbol{\eta}}_{\mathtt{chall}}),$$

While probably feasible, the CS+LR approach heavily relied on homomorphic properties the scheme. The presence of noise in LWE schemes can make homomorphism tricky to handle, as we will see later on.

where $\Delta^{\mathbf{r}}_{chall} = \sum_{i=1}^{chall} \mathbf{r}_i$ and $\Delta^{\eta}_{chall} = \sum_{i=1}^{chall} \eta_i$. The adversary can then again perform updates, until it wants a leak of the secret key at epoch $last$. Before the challenger sends the secret key to the adversary, it performs a final update using randomness $(\mathbf{r}^*, \eta^*)$ and sends

$$\begin{aligned} \mathsf{sk}^* &= \mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^* \\ \mathsf{pk}^* &= (\mathbf{A}, \mathbf{b} + \mathbf{A}(\Delta^{\mathbf{r}}_{i} + \mathbf{r}^*) + \Delta^{\eta}_{last} + \eta^*) \\ \mathsf{up}^* &= \mathsf{Enc}(\mathsf{pk}_{last}, \mathbf{r}^*) \end{aligned}$$

to the adversary. The latter must guess which plaintext was encrypted given this information.

We start by observing that up^* can be homorphically transformed into an encryption of $-\mathbf{s}$ using $\mathsf{sk}^* = \mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$, as $\Delta^{\mathbf{r}}_{last}$ is known. This amounts to prove circular-security with the additional information given by the leakage. We then notice that for any ciphertext

$$(\mathsf{ct}_0, \mathsf{ct}_1) = (\mathbf{XA} + \mathbf{E}, \mathbf{Xb} + \mathbf{f} + \mu \cdot \lfloor q/p \rfloor)$$

there is an algebraic relation between ct_0 and ct_1, that can be written as

$$\mathsf{ct}_1 = \mathsf{ct}_0 \mathbf{s} + \mathbf{f} + \mathbf{Xe} - \mathbf{Es} + \lfloor q/p \rfloor \cdot \mu.$$

While different, the Dual-Regev scheme used in [39] shares a similar agebraic structure than ours.

At this step, noise flooding is used in [39] to argue that by taking the noise $\mathbf{f}$ large enough compared to $\mathbf{Xe} - \mathbf{Es}$, one can compute ct_1 as

$$\mathsf{ct}_1 = \mathsf{ct}_0 \mathbf{s} + \mathbf{f} + \lfloor q/p \rfloor \cdot \mu,$$

removing the term $\mathbf{Xe} - \mathbf{Es}$. This allows them to prove security directly from LWE, given that replacing ct_0 by a random group element turns ct_1 into an LWE sample. We deviate from their analysis from this point, since we aim to avoid noise flooding and have to deal with the original relation between ct_0 and ct_1.

Our objective is to prove pseudorandomness of ct_0 and ct_1 even in the presence of the cross term $\mathbf{Xe} - \mathbf{Es}$. Using only LWE appears to be insufficient, since the cross term contains information about both the secret $\mathbf{X}$ and noise $\mathbf{E}$ matrices. This can be seen as a particular instance of the Extended LWE problem [71], which claims that pseudorandomness of an LWE instance $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ still holds given the additional information $\langle \mathbf{z}, \mathbf{e} \rangle \bmod q$, with $\mathbf{z}$ a small vector chosen by the adversary *before* receiving the matrix $\mathbf{A}$. However, in our case, both key $\mathbf{s}$ and noise $\mathbf{e}$ involved may contain information (coming from the updates) chosen by the adversary *after* seeing the matrix $\mathbf{A}$. This compells us to introduce a new assumption: Adaptive Extended LWE (AextLWE). It states that even if an adversary can choose $\mathbf{z}$ after seeing $\mathbf{A}$, the Extended LWE problem remains hard. There is, however, a small difference with Extended LWE, as we take the hint to be $\langle \mathbf{z}, \mathbf{e} \rangle + \mathbf{g} \bmod q$, with $\mathbf{g}$ a small Gaussian vector. We then prove that LWE reduces to this new assumption using discrete Gaussian convolution, similar to [58].

It is important that $\mathbf{z}$ is chosen before $\mathbf{A}$, as in [25], the reduction from LWE to Extended LWE works by sampling $\mathbf{A}$ with a distribution depending on $\mathbf{z}$.

The reduction proceeds as follows: Given an LWE instance $(\mathbf{A}, \mathbf{b})$, first send $\mathbf{A}$ to the AextLWE adversary to receive its choice of small hint vector $\mathbf{z}$. In response, sample an additional error $\mathbf{e}'$ and Gaussian term g' from a well-chosen distribution that depends on the small vector $\mathbf{z}$ chosen by the adversary, and return $\mathbf{b}' = \mathbf{b} + \mathbf{e}'$ and a hint $h = \langle \mathbf{z}, \mathbf{e}' \rangle + g'$.

One can rewrite the hint as

$$h = \langle \mathbf{z}, \mathbf{e} + \mathbf{e}' \rangle - \langle \mathbf{z}, \mathbf{e} \rangle + g' = \langle \mathbf{z}, \mathbf{e} + \mathbf{e}' \rangle + g$$

for $g = -\langle \mathbf{z}, \mathbf{e} \rangle + g'$. If the vector $\mathbf{b}$ is equal to $\mathbf{As} + \mathbf{e}$, as we have $\mathbf{b}' = \mathbf{As} + (\mathbf{e} + \mathbf{e}')$ and $h = \langle \mathbf{z}, \mathbf{e} + \mathbf{e}' \rangle + g$, it suffices to show that the joint distribution of $\mathbf{e} + \mathbf{e}'$ and g is a spherical Gaussian. This is achieved by applying a convolution lemma to the sum

$$\begin{pmatrix} \mathbf{e} + \mathbf{e}' \\ -\mathbf{z}^\mathsf{T}\mathbf{e} + g' \end{pmatrix} = \begin{pmatrix} \mathbf{Id} \\ -\mathbf{z}^\mathsf{T} \end{pmatrix} \mathbf{e} + \begin{pmatrix} \mathbf{e}' \\ g' \end{pmatrix},$$

which is possible if the standard deviation is larger by a factor of $\|\mathbf{z}\|_2$. The analysis for the case where of uniform $\mathbf{b}$ is identical.

2.2.3 *A Fujisaki-Okamoto transform for UPKE*

The technique used to achieve CCA security with the DCR construction had the disadvantage to cause a blow-up in the size of ciphertexts mainly due to the NIZK proof that had to be generated. We provide an alternative, by building an adaptation of the Fujisaki-Okamoto [46] to UPKEs. We actually introduce the notion of Updatable Key Encapsulation Mechanism (UKEM) and provide an FO transform that allows to build a IND-CR-CCA UKEM in the ROM, from a IND-CR-CPA UPKE. Note that the built UKEM has the same efficiency as the underlying UPKE. This FO transform can thus be used to enhance the result presented in Chapter 4.

Encapsulating under a public key pk is done by computing a ciphertext ct of a uniform message m, with respect to a random coin computed using a hash function G, modeled as a random oracle, with inputs pk and m. This derandomization step binds a ciphertext to its content and the key that it was encrypted under. The encapsulated key is $\mathsf{H}(\mathsf{ct}, \mathsf{m})$, where H is another hash function, also modeled as a random oracle. Decapsulation is done by decrypting the ciphertext ct to recover the message m. Once decrypted, one must perform a check that the randomness used was indeed $\mathsf{G}(\mathsf{pk}, \mathsf{m})$. This is done by rerunning the encryption algorithm with this randomness and checking equality with ct. If the check passes, the output key is $\mathsf{H}(\mathsf{ct}, \mathsf{m})$. The update procedures of the UKEM are exactly the same as those of the underlying UPKE. The main difference with the FO transform for PKEs is that the target public key is used in the derandomization step. This is used in the security proof, in order to prevent the adversary from abusing the update mechanism.

2.2.4 *A generic transform to achieve CU security*

Finally, we generalize the idea applied to the DCR construction by proving that the double encryption and NIZK Naor-Yung paradigm provides a generic transform to achieve CU security. Specifically, we prove that given an IND-CR-CCA UKEM, we can build an IND-CU-CCA UKEM. In terms of efficiency, the ciphertext size remains unchanged. Only the updates are impacted, as we use a second encryption and a NIZK proof that both ciphertexts decrypt to the same message.

2.3 PERFORMANCES

To summarize our results, we give concrete sizes achieved by our constructions and compare them to the state of the art. For the sake of completeness, we give sizes for IND-CR-CPA constructions in Table 1 and for IND-CR-CCA constructions in Table 2 while the latter is quasi identical to the former due to our FO transform. This is to put the emphasis on the efficiency achieved in the standard model for IND-CR-CPA and also show that our FO transform allows to lift all previous constructs to IND-CR-CCA for free in the ROM. Notice that in Table 2, we give the two IND-CR-CCA variants of the DCR construct, one using the Naor-Yung paradigm and the other with the FO transform, even though the first is technically a UPKE while the second is a UKEM. Finally, we present the sizes of IND-CU-CCA constructions in Table 3. While definitely being the most efficient pre-quantum scheme in the ROM, the UPKE from [57] is not easily turned into an IND-CU-CCA one. Their construction heavily relies on the ROM, even for IND-CR-CPA security, which makes our NIZK based approach difficult to apply. Even though our generic CU transform applies to our LWE based scheme, we do not give explicit sizes as some care should be taken in building the NIZK proofs of plaintext equality. Such efficient proofs rely on recent frameworks from [64], which seem to be evolving quite fast and are still complex to handle practically.

2.4 PUBLICATIONS

This thesis is based on the following publications:

- **[49] Updatable Public Key Encryption from DCR: Efficient Constructions With Stronger Security**

  C. Abou Haidar, B. Libert, A. Passelègue. CCS 2022.

- **[50] Efficient Updatable Public-Key Encryption from Lattices**

  C. Abou Haidar, A. Passelègue, D. Stehlé. Asiacrypt 2023.

	λ	ROM	k	|ct|	|up|
CDH-based from [57]	128	Yes	∞	64B	64B
LWE-based from [39]	120	No	2^5	33KB	360KB
DDH-based from [39]	128	No	∞	41KB	52MB
DCR-based construction	128	No	∞	1.5KB	1.5KB
LWE-based construction	128	No	2^5	1.8KB	5.4KB
	128	No	2^{10}	3.0KB	12KB
	116	No	2^{15}	5.8KB	12KB
	128	No	2^{20}	9.1KB	27KB

Table 1: Sizes for IND-CR-CPA UPKE schemes. Sizes are given with respect to bit-security λ.

	λ	k	|ct|	|up|
CDH-based from [57] (+ FO)	128	∞	64B	64B
LWE-based from [39]	120	2^5	33KB	360KB
DCR-based construction (+ NY)	128	∞	8.3KB	1.5KB
DCR-based construction (+ FO)	128	∞	1.5KB	1.5KB
LWE-based construction	128	2^5	1.8KB	5.4KB
	128	2^{10}	3.0KB	12KB
	116	2^{15}	5.8KB	12KB
	128	2^{20}	9.1KB	27KB

Table 2: Sizes for IND-CR-CCA UPKE/UKEM schemes, in the ROM. Sizes are given with respect to bit-security λ.

	λ	k	|ct|	|up|
DCR-based construction (+ NY)	128	∞	11KB	13KB
DCR-based construction (+ FO)	128	∞	2.0 KB	13KB

Table 3: Sizes for IND-CU-CCA UPKE/UKEM schemes, in the ROM. Sizes are given with respect to bit-security λ.

3

PRELIMINARIES

This chapter recalls the definitions related to Updatable Public Key Encryption (UPKE) together with several definitions and results needed in the following chapters.

3.1 NOTATIONS

For a distribution $\mathcal{S}$, we note $s \hookleftarrow S$ the fact that s is sampled using distribution $\mathcal{S}$. For a random variable X, we write $X \sim \mathcal{S}$ if X follows the distribution $\mathcal{S}$. We let $\mathcal{B}(p)$ denote the Bernouilli distribution of parameter p. We write $a \approx_\delta b$ for $a, b, \delta > 0$ if there exists $\varepsilon < \delta$ such that $|a - b| = \varepsilon$.

We say an algorithm is PPT if it is probabilistic and polynomial-time. We use log to denote the logarithm in base 2 and ln to denote the logarithm in base e.

3.2 UPDATABLE PUBLIC KEY ENCRYPTION

We recall the syntax of Updatable Public Key Encryption and adapt the underlying IND-CR-CPA security notion defined in [39], with a minor modification: we define correctness and security with a bound on the number of updates. This is motivated by the fact that, in LWE-based schemes, updates might make the key slightly larger and then after a (large but polynomial) number of updates, correctness of decryption is no longer guaranteed.

Definition 1. *(Updatable Public Key Encryption) An updatable public key encryption scheme consists of a tuple* $\mathsf{UPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *of PPT algorithms with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input a security parameter* 1^λ *and outputs a pair* $(\mathtt{pk}, \mathtt{sk})$.
- $\mathsf{Enc}(\mathtt{pk}, m)$ *takes as input a public key* $\mathtt{pk}$ *and a message* m *and outputs a ciphertext* $\mathtt{ct}$.
- $\mathsf{Dec}(\mathtt{sk}, \mathtt{ct})$ *takes as input a secret key* $\mathtt{sk}$ *and a ciphertext* $\mathtt{ct}$ *and outputs a message* m'.
- $\mathsf{UpdatePk}(\mathtt{pk})$ *takes as input a public key* $\mathtt{pk}$ *and outputs an update* $\mathtt{up}$ *and a new public key* $\mathtt{pk}'$.
- $\mathsf{UpdateSk}(\mathtt{sk}, \mathtt{up})$ *takes as input a secret key* $\mathtt{sk}$ *and an update* $\mathtt{up}$ *and outputs a new secret key* $\mathtt{sk}'$.

(k, δ)-***Correctness***: *Let* $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *be a key pair and* $k > 0$ *be an integer. For* $t < k$, *define*

$$(up_{t+1}, pk_{t+1}) \leftarrow \mathsf{UpdatePk}(pk_t) \text{ and } sk_{t+1} \leftarrow \mathsf{UpdateSk}(sk_t, up_{t+1}).$$

The UPKE scheme is said to be (k, δ)*-correct, for* $\delta > 0$, *if for all messages* m *and* $t \leqslant k$

$$\mathbb{P}\left[\mathsf{Dec}(sk_t, \mathsf{Enc}(pk_t, m)) \neq m\right] < \delta,$$

where the probability is over the coins of the underlying algorithms.

3.3 IND-CR-CPA SECURITY OF UPKE

We give the definition from [39] which we adapt to the bounded number of updates setting by adding a parameter k for the number of updates.

Definition 2 (k-IND-CR-CPA security)**.** *Let* $k > 0$ *be an integer and* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *be a UPKE scheme. Let* $\mathcal{R}$ *be the randomness space of* UpdatePk. *We give the* k-IND-CR-CPA *security game in Figure 1.*
The advantage of $\mathcal{A}$ *in winning the above game is*

$$\mathsf{Adv}^{\text{IND-CR-CPA}}_{\text{UPKE}}(\mathcal{A}) = \left|\Pr\left[\beta = \beta'\right] - \frac{1}{2}\right|.$$

A UPKE scheme is k-IND-CR-CPA*-secure if for all PPT attackers* $\mathcal{A}$*, the advantage of winning* $\mathsf{Adv}^{\text{IND-CR-CPA}}_{\text{UPKE}}(\mathcal{A})$ *is negligible.*

Parameters: λ, k.

GAME($\mathcal{A}$):
- $t = 0$; ▷ Epoch counter
- $\beta \hookleftarrow \mathcal{U}(\{0,1\})$;
- $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
- $(m_0^*, m_1^*, st) \leftarrow \mathcal{A}^{\mathcal{O}_{up}}(pk_0)$;
- $c^* \leftarrow \mathsf{Enc}(pk_t, m_\beta^*)$;
- $st \leftarrow \mathcal{A}^{\mathcal{O}_{up}}(c^*, st)$;
- $r^* \hookleftarrow \mathcal{U}(\mathcal{R})$;
- $(pk^*, up^*) \leftarrow \mathsf{UpdatePk}(pk_t, r^*)$;
- $sk^* \leftarrow \mathsf{UpdateSk}(sk_t, up^*)$;
- $\beta' \leftarrow \mathcal{A}(pk^*, sk^*, up^*, c^*, st)$;

$\mathcal{A}$ wins if $\beta = \beta'$.

$\mathcal{O}_{up}(r)$:
- $t = t + 1$;
- **if** $t > k$
  - **return** $\perp$;
- $(pk_t, up_t) \leftarrow \mathsf{UpdatePk}(pk_{t-1}; r)$;
- $sk_t \leftarrow \mathsf{UpdateSk}(sk_{t-1}, up_t)$;

Figure 1: k-IND-CR-CPA security game.

3.4 IND-CR-CCA SECURITY OF UPKE

Definition 3 (k-IND-CR-CCA UPKE security in the ROM). *Let* (KeyGen, Enc, Dec, UpdatePk, UpdateSk) *be a UPKE. Let* $\mathcal{R}$ *denote the randomness space of* UpdatePk. *We give the game for* k-IND-CR-CCA *security for an adversary that has access to a random oracle* H *in Figure 3.*

Parameters: λ, k.

GAME($\mathcal{A}$):
- $t = 0$; ▷ Epoch counter
- $\beta \hookleftarrow \mathcal{U}(\{0,1\})$;
- $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
- $(m_0^*, m_1^*, st) \leftarrow \mathcal{A}^{\mathcal{O}_{up}, \mathcal{O}_{dec}, \mathsf{H}}(pk_0)$;
- $c^* \leftarrow \mathsf{Enc}(pk_t, m_\beta^*)$;
- $pk^{chall} = pk_t$;
- $st \leftarrow \mathcal{A}^{\mathcal{O}_{up}, \mathcal{O}_{dec}, \mathsf{H}}(c^*, st)$;
- $r^* \hookleftarrow \mathcal{U}(\mathcal{R})$;
- $(up^*, pk^*) \leftarrow \mathsf{UpdatePk}(pk_t, r^*)$;
- $sk^* \leftarrow \mathsf{UpdateSk}(sk_t, up^*)$;
- $\beta' \leftarrow \mathcal{A}^{\mathsf{H}}(pk^*, sk^*, up^*, c^*, st)$;
- $\mathcal{A}$ wins if $\beta = \beta'$.

$\mathcal{O}_{up}(r)$:
- $t = t + 1$;
- **if** $t > k$
  - **return** $\perp$;
- $(pk_t, up_t) \leftarrow \mathsf{UpdatePk}(pk_{t-1}; r)$;
- $sk_t \leftarrow \mathsf{UpdateSk}(sk_{t-1}, up_t)$;

$\mathcal{O}_{dec}(c)$:
- **if** $pk_t = pk^{chall} \wedge c = c^*$
  - **return** $\perp$;
- **return** $\mathsf{Dec}(sk_t, c)$.

Figure 2: UPKE k-IND-CR-CCA security game in the ROM.

The advantage of $\mathcal{A}$ *in winning the above game is*

$$\mathsf{Adv}^{\text{IND-CR-CCA}}_{\text{UPKE}}(\mathcal{A}) = \left| \Pr\left[\beta = \beta'\right] - \frac{1}{2} \right|.$$

A UPKE scheme is k-IND-CR-CCA*-secure if for all PPT attackers* $\mathcal{A}$*, the advantage of winning* $\mathsf{Adv}^{\text{IND-CR-CCA}}_{\text{UPKE}}(\mathcal{A})$ *is negligible.*

Compared to the original definition of [39], a slight modification is made to the decryption oracle. The condition in the safety check is $pk^{chall} = pk_t \wedge c = c^*$ instead of $t = chall \wedge c = c^*$. This is to handle a problem underlined in [8], where the authors show that an attack is possible in the original definition of [39]. If the adversary is able to make an update that does not change the key (using what they call "non-influential randomness"), then by making such an update, the original safety check can be trivially bypassed. Note that this can typically be achieved by considering two consecutive updates of the public key, where the second one is taken to be "the opposite" of the first, so as to come back to the original key. The solution of [8] is to bind ciphertexts to epochs using the random oracle, making each ciphertexts generated at an epoch t only decryptable at this epoch with the oracle. This makes the trivial attack impossible as

even if the adversary is able to come back to the challenge key, the challenge ciphertext cannot be decrypted by the oracle as the epoch has changed.

We circumvent this problem using this additional check in the security model, but the solution from [8] could also be applied here.

3.5 UPDATABLE KEY ENCAPSULATION MECHANISM

We introduce the KEM variant of UPKE, which we term Updatable KEM or UKEM. Defining the KEM equivalent of UPKE seems particularly relevant considering that UPKE was introduced as a group messaging primitive, hence requiring real-world efficiency.

We adapt the definitions of IND-CR-CCA and IND-CU-CCA security notions defined by [39] for UPKEs.

Definition 4 (Updatable KEM (UKEM))**.** *An updatable KEM is a tuple* $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *of algorithms with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input a security parameter* 1^λ *and outputs a pair* $(\mathtt{pk}, \mathtt{sk})$.
- $\mathsf{Encaps}(\mathtt{pk})$ *takes as input a public key* $\mathtt{pk}$ *and outputs an encapsulation* $\mathtt{c}$ *and a key* K.
- $\mathsf{Decaps}(\mathtt{sk}, \mathtt{c})$ *takes as input a secret key* $\mathtt{sk}$ *and an encapsulation* $\mathtt{c}$ *and outputs a key* K'.
- $\mathsf{UpdatePk}(\mathtt{pk})$ *takes as input a public key* $\mathtt{pk}$ *and outputs an update* $\mathtt{up}$ *and a new public key* $\mathtt{pk}'$.
- $\mathsf{UpdateSk}(\mathtt{sk}, \mathtt{up})$ *takes as input a secret key* $\mathtt{sk}$ *and an update* $\mathtt{up}$ *and outputs a new secret key* $\mathtt{sk}'$.

(k, δ)***-Correctness****: Let* $(\mathtt{pk}_0, \mathtt{sk}_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *be a key pair and* $k > 0$ *be an integer. For* $t < k$*, define*

$$(\mathtt{up}_{t+1}, \mathtt{pk}_{t+1}) \leftarrow \mathsf{UpdatePk}(\mathtt{pk}_t) \text{ and } \mathtt{sk}_{t+1} \leftarrow \mathsf{UpdateSk}(\mathtt{sk}_t, \mathtt{up}_{t+1}).$$

The UKEM scheme is said to be (k, δ)*-correct, for* $\delta > 0$*, if for all* $t \leqslant k$

$$\mathbb{P}\left[\mathsf{Decaps}(\mathtt{sk}_t, \mathtt{c}_t) \neq \mathsf{K}_t \mid (\mathtt{c}_t, \mathsf{K}_t) \leftarrow \mathsf{Encaps}(\mathtt{pk}_t)\right] < \delta \ ,$$

where the probability is over the coins of the underlying algorithms.

The notion of k-IND-CR-CCA security corresponds to a variant of k-IND-CR-CPA where the adversary is given access to a decapsulation oracle. We define k-IND-CR-CCA in the Random Oracle Model (ROM), as we make use of the Fujisaki-Okamato transform in Section 5.4 in order to build our IND-CR-CCA UKEM.

Parameters: λ, k.

GAME($\mathcal{A}$):
$t = 0$; ▷ Epoch counter
$\beta \hookleftarrow \mathcal{U}(\{0,1\})$;
$(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
$st \leftarrow \mathcal{A}^{\mathcal{O}_{up},\mathcal{O}_{dec},\mathsf{H}}(pk_0)$;
$(c^*, K^*) \leftarrow \mathsf{Encaps}(pk_t)$;
if $\beta = 1$
  $K^* = \mathcal{U}(\mathcal{K})$;
$pk^{chall} = pk_t$;
$st \leftarrow \mathcal{A}^{\mathcal{O}_{up},\mathcal{O}_{dec},\mathsf{H}}(c^*, st)$;
$r^* \hookleftarrow \mathcal{U}(\mathcal{R})$;
$(up^*, pk^*) \leftarrow \mathsf{UpdatePk}(pk_t, r^*)$;
$sk^* \leftarrow \mathsf{UpdateSk}(sk_t, up^*)$;
$\beta' \leftarrow \mathcal{A}^{\mathsf{H}}(pk^*, sk^*, up^*, c^*, st)$;
$\mathcal{A}$ wins if $\beta = \beta'$.

$\mathcal{O}_{up}(r)$:
$t = t + 1$;
if $t > k$
  return $\perp$;
$(pk_t, up_t) \leftarrow \mathsf{UpdatePk}(pk_{t-1}; r)$;
$sk_t \leftarrow \mathsf{UpdateSk}(sk_{t-1}, up_t)$;

$\mathcal{O}_{dec}(c)$:
if $pk_t = pk^{chall} \wedge c = c^*$
  return $\perp$;
return $\mathsf{Decaps}(sk_t, c)$.

Figure 3: k-IND-CR-CCA security game in the ROM. Note that if $\beta = 0$, then the value of the key K^* is the output of Encaps.

Definition 5 (k-IND-CR-CCA KEM security in the ROM)**.** *Let* (KeyGen, Encaps, Decaps, UpdatePk, UpdateSk) *be a UKEM with key space* $\mathcal{K}$*. Let* $\mathcal{R}$ *denote the randomness space of* UpdatePk*. The game for* k-IND-CR-CCA *security for an adversary that has access to a random oracle* H *is given in Figure 3.*

The advantage of $\mathcal{A}$ *in winning the above game is*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}}(\mathcal{A}) = \left| \Pr\left[\beta = \beta'\right] - \frac{1}{2} \right|.$$

A UKEM scheme is k-IND-CR-CCA*-secure if for all PPT attackers* $\mathcal{A}$*, the advantage of winning* $\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}}(\mathcal{A})$ *is negligible.*

Notice that compared to the IND-CR-CCA definition for UPKE given in [39], we add a check in the $\mathcal{O}_{dec}$ oracle that the current public key pk_t is different from the challenge public key pk^{chall}. This disallows trivial attacks in which an adversary might make carefully chosen updates that would cancel out in order to get back to the challenge public key and issue a decryption query on the challenge. Another approach to solve this is given in [8], which generalizes the one considered in [49].

3.6 IND-CU-CCA SECURITY FOR UPKE/UKEM

In order to define the stronger k-IND-CU-CCA security notions for UKEM/UPKE, we add an algorithm VerifyUpdate to the UKEM/UPKE syntax that allows a user to check

the validity of an update. Specifically, the algorithm $\mathsf{VerifyUpdate}(\mathsf{pk}, (\mathsf{pk}', \mathsf{up}))$ takes as input the current epoch public key pk and a proposed update $(\mathsf{pk}', \mathsf{up})$ and returns a Boolean value. k-IND-CU-CCA security aims to guarantee security against adversaries who make malicious updates.

Definition 6 (k-IND-CU-CCA UKEM/UPKE security in the ROM)**.** *Let* (KeyGen, Encaps, Decaps,UpdatePk,UpdateSk,VerifyUpdate) *be a UKEM (resp.* (KeyGen, Enc, Dec, UpdatePk, UpdateSk,VerifyUpdate) *be a UPKE). The security game for* IND-CU-CCA *is identical to the* IND-CR-CCA *game, except for the modified* $\mathcal{O}_{\mathsf{up}}(\cdot)$ *oracle. We present the modified* $\mathcal{O}_{\mathsf{up}}$ *oracle in Figure 4.*

$\mathcal{O}_{\mathsf{up}}(\mathsf{pk}', \mathsf{up})$:
  if $\mathsf{VerifyUpdate}(\mathsf{pk}_t, (\mathsf{pk}', \mathsf{up})) = \bot$
    return $\bot$;
  $\mathsf{pk}_{t+1} = \mathsf{pk}'$;
  $\mathsf{sk}_{t+1} \leftarrow \mathsf{UpdateSk}(\mathsf{sk}_t, \mathsf{up}_{t+1})$;
  $t = t + 1$.

Figure 4: k-IND-CU-CCA security game in the ROM.

A UKEM (resp. UPKE) scheme is k-IND-CU-CCA*-secure if for all PPT attackers* $\mathcal{A}$*, its advantage of winning* $\mathsf{Adv}^{\mathsf{IND\text{-}CU\text{-}CCA}}_{\mathsf{UKEM}}(\mathcal{A})$ *(resp.* $\mathsf{Adv}^{\mathsf{IND\text{-}CU\text{-}CCA}}_{\mathsf{UPKE}}(\mathcal{A})$*) is negligible.*

For both UPKE and UKEM, we omit the parameter k in k-IND-CR-CPA/ IND-CR-CCA/ IND-CU-CCA if it is clear from context.

3.7 NON-INTERACTIVE ZERO KNOWLEDGE PROOFS

Finally, we recall standard definitions of zero-knowledge proofs [48]. Non-Interactive Zero Knowledge Proofs are used afterwards to upgrade security from IND-CR-CPA to IND-CU-CCA.

INTERACTIVE PROOF SYSTEMS. An interactive proof system (IPS) for a language $\mathcal{L}$ is a two-party protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where the former wants to convince the latter that a statement x belongs to $\mathcal{L}$. An IPS is *zero-knowledge* when this can be achieved without having $\mathcal{P}$ leak any information beyond the fact that $x \in \mathcal{L}$. This is formalized by requiring the existence of a zero-knowledge simulator $\mathcal{S}$ that produces transcripts indistinguishable from real conversations between $\mathcal{P}$ and $\mathcal{V}$. An IPS provides *soundness* when even an unbounded $\mathcal{P}$ cannot trick $\mathcal{V}$ into accepting a proof of a false statement. When the soundness condition only holds for polynomial-time cheating provers, an IPS is called an *argument*. A zero-knowledge proof/argument system is non-interactive (NIZK) when a proof/argument consists of a single message from $\mathcal{P}$ to $\mathcal{V}$.

We now recall the definition of Σ-protocols [32].

Definition 7. *Let an NP language $\mathcal{L}$ associated with a relation $\mathcal{R}$. A 3-move interactive proof system $\Pi = (\mathcal{P}, \mathcal{V})$ is a Σ-protocol for $\mathcal{L}$ if it satisfies the following:*

- ***3-Move Form:*** *$\mathcal{P}$ and $\mathcal{V}$ proceed as follows: (i) $\mathcal{P}$ inputs $(x, w) \in \mathcal{R}$, computes $(a, st) \leftarrow \mathcal{P}(x, w)$ and sends a to $\mathcal{V}$; (ii) $\mathcal{V}$ sends back a random challenge c; (iii) $\mathcal{P}$ sends a response $z = \mathcal{P}(x, w, a, c, st)$ to $\mathcal{V}$; (iv) On input of (a, c, z), $\mathcal{V}$ outputs 1 or 0.*
- ***Completeness:*** *If $(x, w) \in \mathcal{R}$ and $\mathcal{P}$ honestly computes (a, z) for a challenge c, then $\mathcal{V}(x, (a, c, z))$ outputs 1 with probability $1 - \mathsf{negl}(\lambda)$.*
- ***Special soundness:*** *There exists a PPT knowledge extractor $\mathcal{E}$ that, for any statement x, on input of two accepting transcripts (a, c, z) and (a, c', z') with $c \neq c'$, outputs a witness w' such that $(x, w') \in \mathcal{R}$.*
- ***Special honest-verifier zero-knowledge:*** *There is a PPT simulator $\mathcal{S}$ that inputs $x \in \mathcal{L}$ and a challenge $c \in \mathcal{C}$. It outputs $(a, z) \leftarrow \mathcal{S}(x, c)$ such that (a, c, z) is indistinguishable from a real transcript (for $(x, w) \in \mathcal{R}$) with challenge c.*

The special soundness property can be relaxed to hold in the computational sense. In this case, we require that no PPT cheating prover be able to find two transcripts (a, c, z), (a, c', z'), with $c \neq c'$, for which the extractor fails to compute a witness.

In order to prove plaintext equalities via the Naor-Yung paradigm, we require a Σ-protocol satisfying the following property.

Definition 8 (Quasi-unique responses, [43])**.** *Let $\lambda \in \mathbb{N}$ a security parameter. A Σ-protocol $(\mathcal{P}, \mathcal{V})$ has* quasi-unique responses *if, for any PPT algorithm $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{V}(x, a, c, z) = \mathcal{V}(x, a, c, z') = 1 \wedge z \neq z' \mid (x, a, c, z, z') \leftarrow \mathcal{A}(1^\lambda)] = \mathsf{negl}(\lambda)(\lambda).$$

Removing Interaction. In the random oracle model, the Fiat-Shamir heuristic [44] provides an efficient way to build NIZK proofs by collapsing interactive Σ-protocols. The interaction between $\mathcal{P}$ and $\mathcal{V}$ is removed by replacing the latter's challenge with a hash value $H(x, a)$ computed by the prover, where H is a hash function modeled as a random oracle. The resulting NIZK protocol is denoted by $(\mathcal{P}^H, \mathcal{V}^H)$. In the zero-knowledge simulation, the simulator $\mathcal{S}$ is allowed to program the random oracle at arbitrary points. As in [43], we model it as a stateful algorithm that operates in two modes: a mode $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ that handles random oracle queries (typically via lazy sampling); and a second mode $(\pi, st) \leftarrow \mathcal{S}(2, st, x)$ that simulates actual proofs. The different calls to $\mathcal{S}(1, \cdot, \cdot)$ and $\mathcal{S}(2, \cdot, \cdot)$ share a common state st which is updated after each operation.

Definition 9 (NIZK in the ROM). *Let an NP language $\mathcal{L}$ associated with a relation $\mathcal{R}$. Let $(\mathcal{S}_1, \mathcal{S}_2)$ denote oracles such that $\mathcal{S}_1(q_i)$ returns the first output of $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ and $\mathcal{S}_2(x, w)$ returns the first output of $(\pi, st) \leftarrow \mathcal{S}(2, st, x)$ if $(x, w) \in \mathcal{R}$. A non-interactive protocol $(\mathcal{P}^H, \mathcal{V}^H)$ is zero-knowledge for $\mathcal{L}$ in the random oracle model if there exists a PPT simulator $\mathcal{S}$ such that, for any PPT distinguisher $\mathcal{D}$, we have*

$$|\Pr[\mathcal{D}^{H(\cdot),\mathcal{P}^H(\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot,\cdot)}(1^\lambda) = 1]| = \mathsf{negl}(\lambda),$$

where oracles $\mathcal{P}$ and $\mathcal{S}_2$ both output $\perp$ on input of $(x, w) \notin \mathcal{R}$.

It is known that, in the random oracle model, the Fiat-Shamir transform compiles Σ-protocols into NIZK protocols.

Theorem 1. *Let λ be a security parameter. Consider a Σ-protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L}$ in NP. Let H be a function that ranges over the challenge space of Π. In the random oracle model, the proof system $\Pi^{FS} = (\mathcal{P}^H, \mathcal{V}^H)$ derived from Π by applying the Fiat-Shamir transform is a NIZK argument system for $\mathcal{L}$ if the special soundness of Π holds (in the statistical or computational sense). Moreover, if the special honest-verifier zero-knowledge property of Π holds in the statistical sense (resp. computational), then Π^{FS} provides statistical (resp. computational) ZK.*

Simulation-soundness. The soundness property of a NIZK proof ensures that no cheating prover $\mathcal{P}^*$ can come up with a convincing proof for a false statement. The notion of *simulation-soundness* [76] requires that soundness be preserved even if a cheating prover is allowed to see simulated proofs (generated by the NIZK simulator) for possibly false statements chosen by $\mathcal{P}^*$. In the random oracle model, we use a definition of simulation-soundness given in [43].

Definition 10 (Unbounded simulation-soundness). *Let $\mathcal{L}$ an NP language. Consider a proof system $(\mathcal{P}^H, \mathcal{V}^H)$ for $\mathcal{L}$ where the zero-knowledge simulator is denoted by $\mathcal{S}$. Denote by $(\mathcal{S}_1, \mathcal{S}_2)$ the oracles such that $\mathcal{S}_1(q_i)$ returns the first output of $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ and $\mathcal{S}_2(x)$ returns the first output of $\mathcal{S}(2, st, x)$ (possibly with $x \notin \mathcal{L}$). A protocol $(\mathcal{P}^H, \mathcal{V}^H)$ is (unbounded) simulation-sound with respect to $\mathcal{S}$ in the random oracle model if, for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot)}(1^\lambda) : (x^*, \pi^*) \notin \mathcal{T} \wedge x^* \notin \mathcal{L} \wedge \mathcal{V}^{\mathcal{S}_1}(x^*, \pi^*) = 1] = \mathsf{negl}(\lambda),$$

where $\mathcal{T}$ is the list of pairs (x_i, π_i) such that x_i was queried to the simulator and π_i was the latter's response.

Fouque and Pointcheval [45] showed that, in the random oracle model, certain Σ-protocols are turned into simulation-sound NIZK proofs by applying the Fiat-Shamir heuristic. Their proof applies to a restricted family of IPS. In [43], Faust *et al.* gave a more general result that is used in our CCA-secure constructions.

Theorem 2 ([43]). *Consider a non-trivial Σ-protocol $(\mathcal{P}, \mathcal{V})$ for an NP language $\mathcal{L}$, which satisfies the quasi-unique response property of Definition 8. In the random oracle model, the proof system $(\mathcal{P}^H, \mathcal{V}^H)$ derived from $(\mathcal{P}, \mathcal{V})$ via the Fiat-Shamir transform is a simulation-sound NIZK.*

We note that, unlike the result of Fouque and Pointcheval [74], Theorem 2 does not rely on the forking lemma [74] and thus provides tighter concrete security bounds.

Part II

CONSTRUCTIONS

4

A CONSTRUCTION UNDER THE DCR ASSUMPTION

This chapter presents our first construction, whose security is based on the Decision Composite Residuosity assumption. Our main goal is to achieve efficiency for IND-CR-CPA security in the standard model and obtain an IND-CU-CCA construction with reasonable sizes.

4.1 UPKE CONSTRUCTION WITH DCR

4.1.1 *Hardness Assumptions*

We rely on two number theoretic assumptions that imply the hardness of factoring.

Definition 11 ([72]). *Let an RSA modulus* $N = PQ$*, for primes* P, Q*, and let an integer* $\zeta \geqslant 1$*. The* ζ*-Decision Composite Residuosity (*ζ*-DCR) assumption holds if, for any PPT adversary* $\mathcal{A}$*, we have*

$$\mathbf{Adv}^{\mathrm{dcr}}_{\zeta,\mathcal{A}}(\lambda) = \Big|\Pr\left[\mathcal{A}\left(N, r^{N^\zeta} \bmod N^{\zeta+1}\right) = 1 \mid r \hookleftarrow \mathcal{U}(\mathbb{Z}^*_N)\right] - \Pr\left[\mathcal{A}(N, z) = 1 \mid z \hookleftarrow \mathcal{U}(\mathbb{Z}^*_{N^{\zeta+1}})\right]\Big| = \mathsf{negl}(\lambda),$$

Damgård and Jurik [34] showed that all DCR assumptions are equivalent (up to a polynomial loss) for any $\zeta \in \mathsf{poly}(\lambda)$.

When $N = PQ$ is a product of safe primes $P = 2p + 1$ and $Q = 2q + 1$ (i.e., where p and q are also primes), the DCR assumption can equivalently be defined as follows.

Definition 12 ([52]). *For a safe-prime product* $N = PQ$*, let* $T = 1 + N$*. The Decision Composite Residuosity (DCR) assumption asserts that, for any PPT adversary* $\mathcal{A}$*, we have*

$$\mathbf{Adv}^{\mathrm{dcr}}_{\zeta,\mathcal{A}}(\lambda) = \Big|\Pr\left[\mathcal{A}\left(N, g, g^r \bmod N^{\zeta+1}\right) = 1\right] - \Pr\left[\mathcal{A}\left(N, g, T \cdot g^r \bmod N^{\zeta+1}\right) = 1\right]\Big| = \mathsf{negl}(\lambda),$$

where $r \hookleftarrow \mathcal{U}(\mathbb{Z}_n)$ *and* $g = \mu^{N^\zeta} \bmod N^{\zeta+1}$*, where* $\mu \hookleftarrow \mathcal{U}(\mathbb{Z}^*_N)$*.*

In our IND-CU-CCA secure UPKE construction, we also rely on the Strong RSA assumption to prove the soundness of a non-interactive argument system in the random oracle model.

Definition 13 ([9]). *Let a safe-prime product* $N = PQ$ *where* P, Q *are primes of at least* $l(\lambda)$ *bits for some polynomial* $l : \mathbb{N} \to \mathbb{N}$. *The Strong RSA assumption states that, for any PPT algorithm* $\mathcal{A}$, *we have*

$$\Pr[y = x^e \bmod N \wedge e > 1 \mid y \hookleftarrow \mathcal{U}(\mathbb{Z}_N^*),\ (x, e) \leftarrow \mathcal{A}(N, y)] = \mathsf{negl}(\lambda)$$

Looking ahead, we rely (in the proof of Lemma 6) on the Strong RSA assumption in the subgroup of $2N^\zeta$-th residues in $\mathbb{Z}^*_{N^{\zeta+1}}$. However, an algorithm that computes a non-trivial e-th root of $g = \mu^{2N^\zeta} \in \mathbb{Z}^*_{N^{\zeta+1}}$, for a random $\mu \in \mathbb{Z}_N^*$, can be used to solve the usual Strong RSA problem in the subgroup $\mathbb{QR}_N$ of quadratic residues in $\mathbb{Z}_N^*$ as long as e is restricted to be co-prime to N. Given $v = \mu^2 \bmod N$, for some $\mu \in \mathbb{Z}_N^*$, one can set $g = v^{N^\zeta} \bmod N^{\zeta+1}$. From an adversary that outputs $w \in \mathbb{Z}^*_{N^{\zeta+1}}$ and $e > 1$ such that $\gcd(e, N) = 1$ and $g = w^e \bmod N^{\zeta+1}$, one can apply Shamir's trick to compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha e + \beta N^\zeta = 1$, which yields $v = (w^\beta \cdot v^\alpha)^e \bmod N^{\zeta+1}$. Then, $w' \triangleq w^\beta \cdot v^\alpha \bmod N$ satisfies $v = w'^e \bmod N$.

4.1.2 *Useful Lemmas*

In the next section, we rely on the following lemma, which was proven by Kitagawa *et al.* [59] (based on earlier ideas from [23, 65]) in the context of key-dependent message security [15].

Definition 14. *(Interactive vector game from [59].) Let* $\zeta \geqslant 1$ *be an integer and* κ *be a polynomial of* λ. *We define the following* $\mathrm{IV}_{\zeta,\kappa}$ *game between a challenger and an adversary* $\mathcal{A}$.

- *The challenger chooses a challenge bit* $b \hookleftarrow \mathcal{U}(\{0,1\})$ *and generates* $(N, P, Q, T, g) \leftarrow \mathsf{GGen}(1^\lambda, \zeta)$. *The challenger generates the values* $\alpha_i \hookleftarrow \mathcal{U}([0, (N-1)/4])$ *and computes* $g_i \leftarrow g^{\alpha_i} \bmod N^{\zeta+1}$ *for every* $i \in [\kappa]$, *and sends* N, g, *and* $g_1, \ldots, g_\kappa$ *to* $\mathcal{A}$.
- *Adversary* $\mathcal{A}$ *can adaptively make sample queries* $(a_1, \ldots, a_\kappa) \in \mathbb{Z}^\kappa_{N^\zeta}$ *to the challenger. The challenger generates* $r \hookleftarrow \mathcal{U}([0, (N-1)/4])$ *and computes* $e_i \leftarrow T^{b \cdot a_i} \cdot g_i^r \bmod N^{\zeta+1}$ *for every* $i \in [\kappa]$. *The challenger then returns* $(e_1, \ldots, e_\kappa)$ *to* $\mathcal{A}$.
- *Adversary* $\mathcal{A}$ *outputs* $b' \in \{0,1\}$ *and wins if* $b' = b$.

We remind the following result from [65].

Lemma 1. *(DCR Interactive Vector Lemma for* $\kappa = 1, 2$, *adapted from [65]). For* $\kappa = 1, 2$, *no polynomial time adversary can have non-negligible advantage in* $\mathrm{IV}_{\zeta,\kappa}$ *with a polynomial number of queries, under the DCR assumption.*

In the next section, we also rely on the following standard smudging lemma.

Lemma 2 (Smudging lemma). *Let* B_1, B_2 *be positive integers,* $a \hookleftarrow \mathcal{U}([-B_1, B_1]), b \hookleftarrow \mathcal{U}([-B_2, B_2])$ *with* $B_2/B_1 = \mathsf{negl}(\lambda)$. *Then, the distribution of* $(a + b, a)$ *is statistically indistinguishable from that of* (a, b).

4.2 A DCR-BASED IND-CR-CPA-SECURE UPKE

In this section, we construct a UPKE scheme that we prove to be IND-CR-CPA secure assuming the DCR assumption. We follow a similar roadmap as in [39] and start by constructing a circular-secure and leakage-resilient (CS+LR) PKE scheme, for a well-chosen leakage function, under the DCR assumption.

We then extend this construction into a UPKE scheme by appending the additional two key update algorithms, and reduce the security of the resulting scheme to the CS+LR security of the underlying PKE scheme.

4.2.1 *A DCR-Based CR+LR Secure PKE*

Definition 15 (CS+LR security, adapted from [39])**.** *Let* $(\mathsf{GGen}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme with message space* $\mathcal{M}$ *and integer secret keys. For* $B \in \mathbb{Z}$*, we say that the scheme is* B*-CS+LR secure if, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have:*

$$\left| \Pr[\mathsf{pp} \leftarrow \mathsf{GGen}(1^\lambda);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp});\ b \hookleftarrow \mathcal{U}(\{0,1\}); r \hookleftarrow \mathcal{U}([-B, B]);\ (m_0, m_1, st) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}) : \mathcal{A}_2(st, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_b), \mathsf{Enc}(\mathsf{pk}, r), \mathsf{sk} + r) = b] - \frac{1}{2} \right| = \mathsf{negl}(\lambda),$$

We refer to the above quantity as the advantage of $\mathcal{A}$ *against CS+LR security, denoted by* $\mathbf{Adv}^{\mathrm{cs+lr}}_{\mathcal{A}}$*.*

DIFFERENCE WITH THE DODIS ET AL. DEFINITION. Our notion of CS+LR security defers slightly from the one defined by Dodis *et al.* in the sense that it is actually not an extension of circular-security. Here, the adversary is given a randomized leakage of the secret key $\mathsf{sk} + r$ and an encryption of r, whereas the definition of [39] gives of an encryption of (a function of) sk. We could modify our construction to rely on standard CS+LR security, where an encryption of sk is revealed instead of an encryption of sk. We opt to do the opposite as it allows reducing the size of update messages in our UPKE construction. Indeed, in our security proof we use smudging on the sum $\mathsf{sk} + r$, in which sk and r play a symmetric part, allowing us to chose which of the two should be exponentially larger than the other.

We now describe our DCR-based B-CS+LR scheme.

- $\mathsf{GGen}(1^\lambda, B)$: Generate two λ-bit safe primes $P = 2p + 1, Q = 2q + 1$, where p, q are also primes, and set $N = PQ$. Let $\zeta \geqslant 1$, choose $\mu \hookleftarrow \mathcal{U}(\mathbb{Z}^*_N)$, and define a generator g of $\mathbb{Z}^*_{N^{\zeta+1}}$ of the subgroup of order $n = \varphi(N)/4 = pq$ by setting $g = \mu^{2N^\zeta} \bmod N^{\zeta+1}$. Define $\mathsf{pp} = (N, g, \zeta, B)$ for the public parameters. We also set $T = 1 + N$.

- $\mathsf{KGen}(pp)$: Sample a random $x \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$. Set

$$\mathsf{pk} = h = g^x \bmod N^{\zeta+1}, \mathsf{sk} = x.$$

- $\mathsf{Enc}(\mathsf{pk}, m)$: Pick $t \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and return

$$ct = \left(g^t \bmod N^{\zeta+1},\ T^m \cdot h^t \bmod N^{\zeta+1}\right).$$

- $\mathsf{Dec}(\mathsf{sk}, ct)$: Parse ct as (c_0, c_1) and return

$$m = \mathsf{DLog}\left(c_1 \cdot (c_0)^{-x} \bmod N^{\zeta+1}\right)$$

When $\zeta > 1$, the decryption algorithm can use the Damgård-Jurik technique [34] to efficiently recover the message $m \in \mathbb{Z}_{N^\zeta}$ from $(1+N)^m \bmod N^{\zeta+1}$, even when m is arbitrarily large.

Theorem 3. *Under the DCR assumption, the above PKE construction provides* B*-CS+LR security for any* $B \in \mathbb{N}$ *such that the distributions* $\{u \bmod pq \mid u \hookleftarrow \mathcal{U}([-B, B])\}$ *and* $\mathcal{U}(\mathbb{Z}_{pq})$ *are statistically close. In particular, it holds for* $B = (N-1)/4$.

Proof. We prove the result using a sequence of four games.

Game_0: This is the original CS+LR game obtained by encrypting the message m_b, for some $b \hookleftarrow \mathcal{U}(\{0,1\})$. The adversary is given pk, $\mathsf{Enc}(\mathsf{pk}, m_b)$, $\mathsf{Enc}(\mathsf{pk}, r)$ and the leakage $L(x; r) = x + r \in \mathbb{Z}$, where $\mathsf{pk} = g^x \bmod N^{\zeta+1}$ for $x \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$ and $r \hookleftarrow \mathcal{U}([-B, B])$.

Game_1: We change the generation of $\mathcal{A}_2$'s input

$$\mathsf{pk},\ \mathsf{Enc}(\mathsf{pk}, m_b),\ \mathsf{Enc}(\mathsf{pk}, r),\ x + r. \tag{1}$$

We first compute

$$\mathsf{pk}',\ \mathsf{Enc}(\mathsf{pk}', m_b),\ \mathsf{Enc}(\mathsf{pk}', r),\ x + r, \tag{2}$$

where $r \hookleftarrow \mathcal{U}([-B, B])$ and $\mathsf{pk}' = g^{-r} \bmod N^{\zeta+1}$. Then, we compute $\mathsf{pk} = \mathsf{pk}' \cdot g^{(x+r)}$, while $\mathsf{Enc}(\mathsf{pk}, m_b) = (g^t, T^{m_b} \cdot \mathsf{pk}^t)$ is computed from $\mathsf{Enc}(\mathsf{pk}', m_b) = (g^t, T^{m_b} \cdot \mathsf{pk}'^t)$ as

$$\mathsf{Enc}(\mathsf{pk}, m_b) = (g^t, T^{m_b} \cdot \mathsf{pk}'^t \cdot (g^t)^{(x+r)}).$$

$\mathsf{Enc}(\mathsf{pk}, r)$ can be computed from $\mathsf{Enc}(\mathsf{pk}', r)$ in a similar way. While the distribution of (1) is statistically close to that of Game_1 (in particular, the distribution of pk' is statistically uniform in the subgroup generated by g thanks to the constraint on r), the new distribution does not contain any information about x, except in the leakage $x + r \in \mathbb{Z}$, since (1) can be computed from (2).

Game_2: In this game, we replace the leakage $L(x;r) = x + r$ by a random element $u \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$ in (2). Since $g^{-r} \bmod N^{\zeta+1}$, $\mathsf{Enc}(g^{-r}, m_b)$, $\mathsf{Enc}(g^{-r}, r)$ do not depend on x and since

$$\Delta(\mathcal{U}([-2^\lambda B, 2^\lambda B]) * \mathcal{U}([-B, B]), \mathcal{U}([-2^\lambda B, 2^\lambda B])) \leqslant \frac{B}{2^\lambda B} = \mathsf{negl}(\lambda)$$

by Lemma 2, it follows that Game_2 and Game_1 are statistically indistinguishable.

Game_3: In this game, we replace $\mathsf{Enc}(g^{-r}, r) = (g^t, T^r(g^{-r})^t)$ by a pair $(Tg^t, (g^{-r})^t)$ in (2). The DCR assumption guarantees that $g^t \approx_c Tg^t$, as t is unknown to the adversary. Hence, assuming DCR, we have

$$(g^t, T^r(g^t)^{-r}) \approx_c \left(Tg^t, T^r(Tg^t)^{-r}\right) = \left(Tg^t, (g^{-r})^t\right),$$

and distributions from Game_3 and Game_2 are computationally indistinguishable. In Game_3, $\mathcal{A}_2$'s input is thus generated from a tuple

$$\mathsf{pk}' = g^{-r},\ \mathsf{Enc}(g^{-r}, m_b),\ (Tg^t, (g^{-r})^t),\ u, \tag{3}$$

where $r \hookleftarrow \mathcal{U}([-B, B])$, $u \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$.

Game_4: We finally replace

$$\mathsf{Enc}(g^{-r}, m_b) = (g^k, T^{m_b} \cdot (g^{-r})^k) \text{ and } (Tg^t, (g^{-r})^t)$$

from the distribution of (3) by elements

$$\mathsf{Enc}(g^{-r}, 0) = (g^k, (g^{-r})^k) \text{ and } (g^t, (g^{-r})^t).$$

Using the Interactive Vector game $\mathrm{IV}_{\zeta,2}$ for $(g_1, g_2) = (g, g^{-r} \bmod N^{\zeta+1})$ with two queries $(0, m_b)$ and $(1, 0)$ (where the IV challenger replies using the randomness k, t sampled from $\mathcal{U}(\mathbb{Z}_{(N-1)/4})$), it is straightforward to prove that these two distributions are computationally indistinguishable via Lemma 1, assuming that DCR holds. Note that this step again requires the distribution of $g^{-r} \bmod N^{\zeta+1}$ to be statistically close to the distribution

$$\{g^\alpha \bmod N^{\zeta+1} \mid \alpha \hookleftarrow \mathcal{U}((N-1)/4)\},$$

as guaranteed by the constraint on B in the statement of Theorem 3.

We finally observe that the distribution Game_4 is independent of $b \in \{0, 1\}$. Indeed, g, $g^{-r} \bmod N^{\zeta+1}$, $(g^k, (g^{-r})^k)$, and $(g^t, (g^{-r})^t)$ do not carry any information about m_b. This concludes the proof of Theorem 3. □

4.2.2 *A DCR-Based IND-CR-CPA-Secure UPKE*

To construct our UPKE, we extend the CS+LR PKE scheme described in Section 4.2.1 with algorithms $(\mathsf{UpdatePk}, \mathsf{UpdateSk})$, defined as follows. We set $B = (N-1)/4$ for our UPKE scheme, which satisfies the requirements of Theorem 3. In the description below, we include the current epoch number in each public key as it simplifies our security proofs in the CCA-secure extensions later on.

- $\mathsf{GGen}(1^\lambda)$: Generate two λ-bit safe primes $P = 2p+1, Q = 2q+1$, where p, q are also primes, and set $N = PQ$. Choose $\mu \hookleftarrow \mathcal{U}(\mathbb{Z}_N)$, and obtain a generator g of the subgroup of order $n = \varphi(N)/4 = pq$ by setting $g = \mu^{2N^\zeta} \bmod N^{\zeta+1}$. Define $pp = (N, g)$ as public parameters.
- $\mathsf{KGen}(pp)$: Sample a random $x \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$ and compute $h = g^x \bmod N^{\zeta+1}$. Let $\tau = 0$ the current epoch. Set $\mathsf{pk} = (\tau, h)$, $\mathsf{sk} = (\tau, x)$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: Pick $t \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and return
$$ct = (g^t \bmod N^{\zeta+1}, (1+N)^m h^t \bmod N^{\zeta+1}).$$
- $\mathsf{Dec}(\mathsf{sk}, ct)$: Parse ct as (c_0, c_1) and sk as (τ, x). Return
$$m = \mathsf{DLog}\left(c_1 \cdot c_0^{-x} \bmod N^{\zeta+1}\right).$$
- $\mathsf{UpdatePk}(\mathsf{pk})$: Parse pk as (τ, h).
  1. Sample $r \hookleftarrow \mathcal{U}([-B, B])$ and compute
  $$h' = h \cdot g^r \bmod N^{\zeta+1}.$$
  2. Compute $up \leftarrow \mathsf{Enc}(\mathsf{pk}, r)$ and set $\tau \leftarrow \tau + 1$.

  Return $(up, (\tau, h'))$.
- $\mathsf{UpdateSk}(\mathsf{sk}, up)$: Given the secret key $\mathsf{sk} = (\tau, x)$, compute $r \leftarrow \mathsf{Dec}(\mathsf{sk}, up) \in \mathbb{Z}_{N^\zeta}$. Let $\bar{r} = \min(r, N^\zeta - r)$ and let $b_r \in \{0,1\}$ such that $\bar{r} = (1-b_r)\cdot r + b_r \cdot (N^\zeta - r) \bmod N^\zeta$.

  Return $\mathsf{sk}' = (\tau+1, x')$, where $x' = x + (-1)^{b_r} \cdot r \in \mathbb{Z}$.

Theorem 4. *Under the DCR assumption, the above UPKE construction provides IND-CR-CPA security.*

Proof. Let $\mathcal{A}$ be an adversary in the IND-CR-CPA security game of our UPKE. We build an adversary $\mathcal{B}$ against the CS+LR security of the PKE scheme in Section 4.2.1 with chosen leakage $L(x; r) = x + r \in \mathbb{Z}$ for $r \hookleftarrow \mathcal{U}([-B, B])$ and $x \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$.

- Our adversary $\mathcal{B}$ receives from its challenger a public key $\mathsf{pk} = h = g^x \bmod N^{\zeta+1}$ associated with a secret key x.
- Adversary $\mathcal{B}$ implicitly sets $x_0 = x$ and sets the epoch τ to 0 by forwarding $\mathsf{pk}_0 = (0, h)$ to $\mathcal{A}$.
- When $\mathcal{A}$ queries oracle $\mathcal{O}_{\mathtt{upd}}(\delta_i)$, $\mathcal{B}$ just bookkeeps δ_i.
- At some point, $\mathcal{B}$ receives the challenge messages m_0^*, m_1^* from $\mathcal{A}$. Adversary $\mathcal{B}$ forwards m_0^*, m_1^* to its challenger, using them as challenge messages. In response, $\mathcal{B}$ receives a leakage $z = L(x; r) = x + r \in \mathbb{Z}$, and ciphertexts $c = \mathsf{Enc}(\mathsf{pk}, m_b^*)$ and $c' = \mathsf{Enc}(\mathsf{pk}, r)$. In order to build a challenge ciphertext for $\mathcal{A}$, adversary $\mathcal{B}$ needs to compute $c^* = \mathsf{Enc}(\mathsf{pk}_\ell, m_b^*)$, where ℓ is the epoch at which $\mathcal{A}$ sent its challenge messages. We have $\mathsf{pk}_\ell = (\ell, g^{x+\sum_{i=1}^{\ell}\delta_i} \bmod N^{\zeta+1})$. Parsing $c = (c_0, c_1)$ and defining $\delta = \sum_{i=1}^{\ell}\delta_i$ (computed over $\mathbb{Z}$), $\mathcal{B}$ simply computes

$$c^* = \left(c_0, c_0^{\delta} \cdot c_1 \bmod N^{\zeta+1}\right)$$

  and forwards c^* as challenge ciphertext to $\mathcal{A}$.
- Then $\mathcal{A}$ resumes its queries of $\mathcal{O}_{\mathtt{upd}}(\cdot)$ and $\mathcal{B}$ keeps registering them until the last query $\mathcal{O}_{\mathtt{upd}}(\delta_{\ell'})$ is made.
- Adversary $\mathcal{B}$ has to send the final secret key $\mathsf{sk}^* = (\ell' + 1, x^*)$, which is the result of all update queries made by $\mathcal{A}$ and a last honestly-generated update (of which the randomness is not known to $\mathcal{A}$). Letting $\delta' = \sum_{i=1}^{\ell'}\delta_i \in \mathbb{Z}$, the reduction $\mathcal{B}$ uses the leakage randomness r as the randomness of the final update, which is hidden from $\mathcal{A}$'s view. Namely, adversary $\mathcal{B}$ sets $x^* = z + \delta' = x + r + \delta$ (over $\mathbb{Z}$) and defines the new public for the next epoch to be $\mathsf{pk}^* = (\ell' + 1, g^{x^*} \bmod N^{\zeta+1})$. To finalize the update message, adversary $\mathcal{B}$ can simply update the public key used in the encryption of the c' from g^{x_0} to $g^{x_0+\delta}$. This is done using the same technique as for the challenge ciphertext. Namely, given $c' = \mathsf{Enc}(\mathsf{pk}, r) = (c_0', c_1')$, adversary $\mathcal{B}$ simply sets

$$\mathsf{up}^* = \left(c_0', {c_0'}^{\delta'} \cdot c_1' \bmod N^{\zeta+1}\right)$$

  and then sends $(\mathsf{pk}^*, \mathsf{sk}^*, \mathsf{up}^*)$ to $\mathcal{A}$.
- When $\mathcal{A}$ finally halts with some output $b' \in \{0, 1\}$, adversary $\mathcal{B}$ outputs the same bit b'.

By inspection, it is clear that $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s challenger in the IND-CR-CPA security game and succeeds whenever $\mathcal{A}$ does. Theorem 4 follows. □

4.3 FROM CR-CPA TO CR-CCA/CU-CCA SECURITY IN THE ROM

We now enhance our UPKE scheme so that it becomes IND-CR-CCA secure. To achieve that we need to build Non Interactive Zero Knowledge (NIZK) arguments that allow to perform the Naor-Yung transformation for our IND-CPA-secure PKE. We start by constructing a Σ-protocol for proving plaintext equality in Section 4.3.1, which we transform into a simulation-sound NIZK arguments via the Fiat-Shamir transform and use to build an IND-CR-CCA secure UPKE scheme in Section 4.3.2.

We then further upgrade the scheme to achieve IND-CU-CCA security in Section 4.3.4, by adding NIZK arguments that updates are well-formed. The latter NIZK arguments are obtained in a similar way by building a Σ-protocol in Section 4.3.3 and applying the Fiat-Shamir transform.

4.3.1 *Proofs of Plaintext Equality*

In order to achieve IND-CR-CCA security in the ROM, we apply the Naor-Yung transformation [70] in the random oracle model. Let public parameters $pp = (N, g, \zeta)$, where $N = PQ$ is a safe-prime product and $g \in \mathbb{Z}^*_{N^{\zeta+1}}$ generates the subgroup of $2N^\zeta$-th residues in $\mathbb{Z}^*_{N^{\zeta+1}}$. Let $h = g^x \bmod N^{\zeta+1}$, $h_d = g^{x_d} \bmod N^{\zeta+1}$ be two public keys, for some underlying secret keys $x, x_d \in \mathbb{Z}_{2^{2\lambda}(N-1)/4}$.

We need to prove membership of the language

$$\begin{aligned}
\mathcal{L}_{NY} = \Big\{ & (h, h_d, C_0, C_1, D_0, D_1) \in (\mathbb{Z}^*_{N^{\zeta+1}})^6 \\
& \exists t_c, t_d \quad \in \mathbb{Z}_{pq},\ m \in \mathbb{Z}_{N^\zeta} : \\
& \quad C_0^2 = g^{2\cdot t_c} \bmod N^{\zeta+1} \wedge C_1^2 = (1+N)^{2m} \cdot h^{2\cdot t_c} \bmod N^{\zeta+1} \\
& \qquad \wedge D_0^2 = g^{2\cdot t_d} \bmod N^{\zeta+1} \wedge D_1^2 = (1+N)^{2m} \cdot h_d^{2\cdot t_d} \bmod N^{\zeta+1} \Big\},
\end{aligned}$$

The Σ-protocol for $\mathcal{L}_{NY}$ is standard and goes as follows.

- $\mathcal{P}$ picks $t'_c, t'_d \hookleftarrow \left[0, 2^{2\lambda} \cdot \frac{N-1}{4}\right]$, $m' \hookleftarrow \mathbb{Z}_{N^\zeta}$ and sends

$$\begin{aligned}
C'_0 &= g^{2\cdot t'_c} \bmod N^{\zeta+1}, \\
C'_1 &= (1+N)^{2m'} \cdot h^{2\cdot t'_c} \bmod N^{\zeta+1}, \\
D'_1 &= g^{2\cdot t'_d} \bmod N^{\zeta+1}, \\
D'_1 &= (1+N)^{2m'} \cdot h_d^{2\cdot t'_d} \bmod N^{\zeta+1}.
\end{aligned}$$

- $\mathcal{V}$ sends a random challenge $c \hookleftarrow \mathcal{U}([0, 2^\lambda - 1])$.
- $\mathcal{P}$ computes $\tilde{t}_c = t'_c + c \cdot t_c$, $\tilde{t}_d = t'_d + c \cdot t_d$ (over $\mathbb{Z}$) and $\tilde{m} = m' + c \cdot m \bmod N^\zeta$. If $\tilde{t}_c, \tilde{t}_d \in \left[0, 2^{2\lambda} \cdot \frac{N-1}{4}\right]$, it sends $\tilde{t}_c, \tilde{t}_d, \tilde{m}$ to $\mathcal{V}$. Otherwise, it aborts.

- $\mathcal{V}$ first verifies that $\tilde{t}_c, \tilde{t}_d \in \left[0, 2^{2\lambda} \cdot \frac{N-1}{4}\right]$, then that

$$\begin{aligned} C_0' &= C_0^{-2c} \cdot g^{2\cdot\tilde{t}_c} \bmod N^{\zeta+1}, \\ C_1' &= C_1^{-2c} \cdot (1+N)^{2\tilde{m}} \cdot h^{2\cdot\tilde{t}_c} \bmod N^{\zeta+1}, \\ D_0' &= D_0^{-2c} \cdot g^{2\cdot\tilde{t}_d} \bmod N^{\zeta+1}, \\ D_1' &= D_1^{-2c} \cdot (1+N)^{2\tilde{m}} \cdot h_d^{2\cdot\tilde{t}_d} \bmod N^{\zeta+1}, \end{aligned} \tag{4}$$

  Verifier $\mathcal{V}$ accepts if all checks succeed, and rejects otherwise.

Lemma 3. *The above Σ-protocol provides statistical soundness.*

Proof. Let us assume that a prover can come up with two valid transcripts

$$((C_0', C_1', D_0', D_1), c_1, (\tilde{t}_{c,1}, \tilde{t}_{d,1}, \tilde{m}_1))$$
$$((C_0', C_1', D_0', D_1), c_2, (\tilde{t}_{c,2}, \tilde{t}_{d,2}, \tilde{m}_2)).$$

We show that $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$.

We define $\Delta c = c_1 - c_2$, $\Delta m = \tilde{m}_1 - \tilde{m}_2 \bmod N^\zeta$, $\Delta t_c = \tilde{t}_{c,1} - \tilde{t}_{c,2}$, $\Delta t_d = \tilde{t}_{d,1} - \tilde{t}_{d,2}$ with $\Delta t_c, \Delta t_d \in [-2^{2\lambda} \cdot \frac{N-1}{4}, 2^{2\lambda} \cdot \frac{N-1}{4}]$. From the verification equations (4), we have

$$\begin{aligned} C_0^{2\Delta c} &= g^{2\cdot\Delta t_c} \bmod N^{\zeta+1}, \\ C_1^{2\Delta c} &= (1+N)^{2\Delta m} \cdot h^{2\cdot\Delta t_c} \bmod N^{\zeta+1}, \\ D_0^{2\Delta c} &= g^{2\cdot\Delta t_d} \bmod N^{\zeta+1}, \\ D_1^{2\Delta c} &= (1+N)^{2\Delta m} \cdot h_d^{2\cdot\Delta t_d} \bmod N^{\zeta+1}, \end{aligned} \tag{5}$$

Since $\Delta c < \min(p, q)$, we must have $\gcd(\Delta c, pq) = 1$ and $\gcd(\Delta c, N) = 1$. Then, if we raise all members of (5) to the power $\hat{c} = \Delta c^{-1} \bmod N^\zeta pq$, we obtain

$$\begin{aligned} C_0^2 &= g^{2\cdot\Delta t_c\cdot\hat{c}} \bmod N^{\zeta+1}, \\ C_1^2 &= (1+N)^{2\Delta m\cdot(\Delta c^{-1} \bmod N^\zeta)} \cdot h^{2\cdot\Delta t_c\cdot\hat{c}} \bmod N^{\zeta+1}, \\ D_0^2 &= g^{2\cdot\Delta t_d\cdot\hat{c}} \bmod N^{\zeta+1}, \\ D_1^2 &= (1+N)^{2\Delta m\cdot(\Delta c^{-1} \bmod N^\zeta)} \cdot h_d^{2\cdot\Delta t_d\cdot\hat{c}} \bmod N^{\zeta+1}, \end{aligned}$$

since $\hat{c} \bmod N^\zeta = \Delta c^{-1} \bmod N^\zeta$. The above equalities show that $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$,[1] as claimed. □

Lemma 4. *The Σ-protocol Σ_{NY} is statistically special honest-verifier zero-knowledge.*

1 By applying the same arguments as in the proof of Lemma 6, we can show that, unless the cheating prover is able to compute a non-trivial root of $g \in \mathbb{Z}^*_{N^{\zeta+1}}$, Δc divides both Δt_c and Δt_d, so that witnesses t_c and t_d are extractable. However, we do not rely on this property as we only aim at soundness (rather than extractability).

Proof. In the special honest verifier setting, the simulator is given a challenge c and has to produce a properly distributed accepting transcript for $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$. We can uniformly sample $\tilde{t}_c, \tilde{t}_d \hookleftarrow \mathcal{U}([0, 2^{2\lambda} \cdot \frac{N-1}{4}])$, $\tilde{m} \hookleftarrow \mathcal{U}(\mathbb{Z}_{N^\zeta})$ and then set

$$\begin{aligned} C_0' &= C_0^{-2c} \cdot g^{2\cdot\tilde{t}_c} \bmod N^{\zeta+1} \\ C_1' &= C_1^{-2c} \cdot (1+N)^{2\tilde{m}} \cdot h^{2\cdot\tilde{t}_c} \bmod N^{\zeta+1} \\ D_0' &= D_0^{-2c} \cdot g^{2\cdot\tilde{t}_d} \bmod N^{\zeta+1} \\ D_1' &= D_1^{-2c} \cdot (1+N)^{2\tilde{m}} \cdot h_d^{2\cdot\tilde{t}_d} \bmod N^{\zeta+1} \end{aligned}$$

We then return $(C_0', C_1', D_0', D_1', c, \tilde{t}_c, \tilde{t}_d, \tilde{m})$.

This is a valid transcript as it made to satisfy all conditions of the verification step. Since $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$, we know that $\exists t_c, t_d \in \mathbb{Z}_{pq},\ m \in \mathbb{Z}_{N^\zeta}$ such that

$$\begin{aligned} C_0^2 &= g^{2\cdot t_c} \bmod N^{\zeta+1} \\ C_1^2 &= (1+N)^{2m} \cdot h^{2\cdot t_c} \bmod N^{\zeta+1} \\ D_0^2 &= g^{2\cdot t_d} \bmod N^{\zeta+1} \\ D_1^2 &= (1+N)^{2m} \cdot h_d^{2\cdot t_d} \bmod N^{\zeta+1} \end{aligned}$$

It follows that

$$\begin{aligned} C_0' &= g^{2(\tilde{t}_c - c t_c)} \bmod N^{\zeta+1} \\ C_1' &= (1+N)^{2(\tilde{m} - cm)} \cdot h^{2(\tilde{t}_c - c t_c)} \bmod N^{\zeta+1} \\ D_0' &= g^{2(\tilde{t}_d - c t_d)} \bmod N^{\zeta+1} \\ D_1' &= (1+N)^{2(\tilde{m} - cm)} \cdot h_d^{2(\tilde{t}_d - c t_d)} \bmod N^{\zeta+1} \end{aligned}$$

Since $|c \cdot t_c|, |c \cdot t_d| < 2^\lambda \cdot \frac{N-1}{4}$, Lemma 2 implies

$$(t_c' + c \cdot t_c, t_d' + c \cdot t_d) \approx_s \mathcal{U}([0, 2^{2\lambda} \cdot \frac{N-1}{4}]) \times \mathcal{U}([0, 2^{2\lambda} \cdot \frac{N-1}{4}])$$

and we also have $\{\tilde{m}' + cm \bmod N^\zeta \mid \tilde{m} \hookleftarrow \mathcal{U}(\mathbb{Z}_{N^\zeta})\} \approx_s \mathcal{U}(\mathbb{Z}_{N^\zeta})$. Since (C_0', C_1', D_0', D_1') is uniquely determined by the (true) statement and the whole challenge-response pair $(c, (\tilde{t}_c, \tilde{t}_d, \tilde{m}))$ in the verification equations, our simulated transcript is statistically close to a real transcript. □

The Σ-protocol is used to prove plaintext equalities in order to apply the Naor-Yung paradigm in the random oracle model. In order to obtain the simulation-soundness property by applying Theorem 2, we need the following lemma.

Lemma 5. *Under the assumption that factoring N is hard, the above Σ-protocol has quasi-unique responses.*

Proof. Towards a contradiction, let as assume that an adversary can find two valid transcripts $(C'_0, C'_1, D'_0, D'_1, c, \tilde{t}_{c,0}, \tilde{t}_{d,0}, \tilde{m}_0)$ and $(C'_0, C'_1, D'_0, D'_1, c, \tilde{t}_{c,1}, \tilde{t}_{d,1}, \tilde{m}_1)$ for some statement $(h, h_d, C_0, C_1, D_0, D_1)$, with

$$(\tilde{t}_{c,0}, \tilde{t}_{d,0}, \tilde{m}_0) \neq (\tilde{t}_{c,1}, \tilde{t}_{d,1}, \tilde{m}_1).$$

We first assume that $\tilde{t}_{c,0} \neq \tilde{t}_{c,1}$. By the first equation of the verification equations (4), we have

$$C'_0 = C_0^{-2c} \cdot g^{2\cdot\tilde{t}_{c,0}} \bmod N^{\zeta+1} = C_0^{-2c} \cdot g^{2\cdot\tilde{t}_{c,1}} \bmod N^{\zeta+1}$$

so that $g^{2\cdot(\tilde{t}_{c,0}-\tilde{t}_{c,1})} = 1 \bmod N^{\zeta+1}$. Since $\tilde{t}_{c,0} \neq \tilde{t}_{c,1}$, this implies $\tilde{t}_{c,0} - \tilde{t}_{c,1} = 0 \bmod n$. Given a non-trivial multiple of $n = pq$, Miller's algortihm [69] allows computing a non-trivial factor of N with high probability.

The case where $\tilde{t}_{d,0} \neq \tilde{t}_{d,1}$ is similar. We now assume that $\tilde{t}_{d,0} = \tilde{t}_{d,1}$ and $\tilde{t}_{c,0} = \tilde{t}_{c,1}$ but $\tilde{m}_0 \neq \tilde{m}_1$. From the verification equations (4), we obtain

$$\begin{aligned} D'_1 &= D_1^{-2c} \cdot (1+N)^{2\tilde{m}_0} \cdot h_d^{2\cdot\tilde{t}_{d,0}} \bmod N^{\zeta+1} \\ &= D_1^{-2c} \cdot (1+N)^{2\tilde{m}_1} \cdot h_d^{2\cdot\tilde{t}_{d,1}} \bmod N^{\zeta+1} \end{aligned}$$

and $(1+N)^{2\tilde{m}_0-2\tilde{m}_1} = 1 \bmod N^{\zeta+1}$, which implies that $2\tilde{m}_0 = 2\tilde{m}_1 \bmod N^{\zeta}$. However, this is impossible if the quasi-uniqueness is broken since $\gcd(2, N) = 1$ and $\tilde{m}_0, \tilde{m}_1 \in \mathbb{Z}_{N^\zeta}$. □

4.3.2 *IND-CR-CCA secure UPKE*

We now upgrade our IND-CR-CPA scheme in the following way. Since we just aim at IND-CR-CCA (rather than IND-CU-CCA) security in this section, we can set $B = (N-1)/4$ and $\zeta \geqslant 1$.

- $\mathsf{GGen}(1^\lambda, B)$:
  1. Generate two λ-bit safe primes $P = 2p+1, Q = 2q+1$, where p, q are also primes, and set $N = PQ$. Choose an integer $\zeta \geqslant 1$.
  2. Choose $\mu, \mu_d \hookleftarrow \mathcal{U}(\mathbb{Z}_N)$ at random. Then, compute generators of the subgroup of order $n = \varphi(N)/4 = pq$ by setting $g = \mu^{2N^\zeta} \bmod N^{\zeta+1}$ and $h_d = \mu_d^{2N^\zeta} \bmod N^{\zeta+1}$.
  3. Choose a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ that is modeled as a random oracle in the analysis.

  Define $pp = (N, \zeta, g, h_d, H)$ as public parameters.
- $\mathsf{KGen}(pp)$: Sample a random $x \hookleftarrow \mathcal{U}([-2^\lambda B, 2^\lambda B])$. Define the key pair (sk, pk) by setting $sk = (\tau, x)$ and $pk = (\tau, h)$, where $\tau = 0$ is a counter and $h = g^x \bmod N^{\zeta+1}$.

- $\mathsf{Enc}(\mathsf{pk}, m)$: To encrypt $m \in \mathbb{Z}_{N^\zeta}$ under the public key $\mathsf{pk} = (\tau, h) \in \mathbb{N} \times \mathbb{Z}^*_{N^{\zeta+1}}$, conduct the following steps.

  1. Pick $t_c, t_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and compute

  $$\begin{aligned}(C_0, C_1) &= \left(g^{t_c} \bmod N^{\zeta+1}, (1+N)^m \cdot h^{t_c} \bmod N^{\zeta+1}\right)\\ (D_0, D_1) &= \left(g^{t_d} \bmod N^{\zeta+1}, (1+N)^m \cdot h_d^{t_d} \bmod N^{\zeta+1}\right).\end{aligned}$$

  2. Using (t_c, t_d, m), generate a NIZK proof π that

  $$(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY},$$

  where $\mathcal{L}_{NY}$ is the language defined in Section 4.3.1. This proof is of the form

  $$\begin{aligned}\pi = (C_0', C_1', D_0', D_1', c, \tilde{t}_c, \tilde{t}_d, \tilde{m})&\\ \in \{0,1\}^\lambda \times \left[0, 2^{2\lambda} \cdot (N-1)/4\right]^2 \times \mathbb{Z}_{N^\zeta},&\end{aligned}$$

  with $c = H\big(\tau, (h, h_d, C_0, C_1, D_0, D_1), (C_0', C_1', D_0', D_1')\big)$, and where the tuple (C_0', C_1', D_0', D_1') satisfies (4).

  Output the ciphertext $\mathsf{ct} = (C_0, C_1, D_0, D_1, \pi)$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Parse ct as $(C_0, C_1, D_0, D_1, \pi)$ and sk as $(\tau, x) \in \mathbb{N} \times \mathbb{Z}$. Return $\perp$ if π is not a verifying NIZK argument that $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$. Otherwise, return

  $$m = \mathsf{DLog}\left(C_1^2 \cdot C_0^{-2x} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^\zeta.$$

- $\mathsf{UpdatePk}(\mathsf{pk})$: To update $\mathsf{pk} = (\tau, h) \in \mathbb{N} \times \mathbb{Z}^*_{N^{\zeta+1}}$,

  1. Choose $r \hookleftarrow \mathcal{U}([-B, B])$ and compute $h' = h \cdot g^r \bmod N^{\zeta+1}$.
  2. Choose $k \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and compute

  $$up = (U, V) = \left(g^k, (1+N)^r \cdot h^k \bmod N^{\zeta+1}\right)$$

  and set $\tau' = \tau + 1$.

  Return $\left(up = (U, V), \mathsf{pk}' = (\tau', h')\right)$.

- $\mathsf{UpdateSk}(\mathsf{sk}, up)$: Given $\mathsf{sk} = (\tau, x) \in \mathbb{N} \times \mathbb{Z}$ and an update message

  $$\left(up = (U, V), \mathsf{pk}' = (\tau', h')\right),$$

  return $\perp$ if $\tau' \neq \tau + 1$. Otherwise, conduct the following steps.

  1. Return $\perp$ if $(U, V) \notin (\mathbb{Z}^*_{N^{\zeta+1}})^2$ or $h \notin \mathbb{Z}^*_{N^{\zeta+1}}$.

2. Compute
$$r = \mathsf{DLog}\left(V^2 \cdot U^{-2x} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^{\zeta}.$$
3. Let $\bar{r} = \min(r, N^{\zeta} - r)$ and let $b_r \in \{0,1\}$ such that
$$\bar{r} = (1 - b_r) \cdot r + b_r \cdot (N^{\zeta} - r) \bmod N^{\zeta}.$$
Compute $x' = x + (-1)^{b_r} \cdot r$ over $\mathbb{Z}$.
4. Return $\mathsf{sk}' = (\tau + 1, x') \in \mathbb{N} \times \mathbb{Z}$.

Theorem 5. *The above construction provides IND-CR-CCA security assuming that: (i) The DCR assumption holds; (ii) The NIZK proof for $\mathcal{L}_{NY}$ provides simulation-soundness.*

Proof. The proof proceeds with a sequence of games. For each i, we call W_i the event that the adversary $\mathcal{A}$ outputs 0 in Game_i.

Game_0: This is the original IND-CR-CCA game where the challenger's bit is $b = 0$. The challenger initially generates public parameters pp containing (N, ζ, g, h_d). It generates a key pair $(\mathsf{pk}_0, \mathsf{sk}_0) = (g^x \bmod N^{\zeta+1}, x)$ and gives pk_0 to $\mathcal{A}$. It handles update queries and decryption queries using the real secret key sk_i at any epoch i. At stage 2, $\mathcal{A}$ outputs (m_0^*, m_1^*) and obtains $c^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$ of the form
$$\begin{aligned}(C_0^*, C_1^*) &= \left(g^{t_c^*} \bmod N^{\zeta+1}, (1+N)^{m_b^*} \cdot h_\ell^{t_c^*} \bmod N^{\zeta+1}\right),\\ (D_0^*, D_1^*) &= \left(g^{t_d^*} \bmod N^{\zeta+1}, (1+N)^{m_b^*} \cdot h_d^{t_d^*} \bmod N^{\zeta+1}\right)\end{aligned}$$
where $h_\ell = g^{x_\ell} \bmod N^{\zeta+1}$ denotes the public key at epoch ℓ. At stage 6, adversary $\mathcal{A}$ obtains (sk^*, up^*), where
$$up^* = (U^*, V^*) = \left(g^{k^*} \bmod N^{\zeta+1}, (1+N)^{r^*} \cdot h_{\ell'}^{k^*} \bmod N^{\zeta+1}\right),$$
where $r^* \leftarrow \mathcal{U}([-B, B])$ and $h_{\ell'} \in \mathbb{Z}_{N^{\zeta+1}}$ is the public key at stage 5. Finally, adversary $\mathcal{A}$ ouputs a bit $b' \in \{0,1\}$ and wins if $b' = 0$. The latter event is called W_0.

Game_1: In this game, we modify the generation of public parameters and now set $h_d = g^{x_d} \bmod N^{\zeta+1}$, where $x_d \leftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$. Since h_d remains statistically uniform in the subgroup of $2N^{\zeta}$-th residues, the distribution of pp is statistically close to that of Game_0.

We have $|\Pr[W_1] - \Pr[W_0]| \leqslant 2^{-\lambda}$.

Game_2: We change the decryption oracle. For a decryption query $ct = (C_0, C_1, D_0, D_1, \pi)$ at epoch i, the challenger does no longer use the current secret key sk_i. Instead, it uses x_d to decrypt (D_0, D_1) by computing
$$m = \mathsf{DLog}\left(D_1^2 \cdot D_0^{-2x_d} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^{\zeta}.$$

Clearly, Game_2 is perfectly indistinguishable from Game_1 until the event that $\mathcal{A}$ queries $\mathcal{O}_{\mathtt{dec}}(\cdot)$ on a ciphertext for which (C_0, C_1) and (D_0, D_1) decrypt to distinct messages although π is a valid proof for $\mathcal{L}_{\mathrm{NY}}$. However, Lemma 3 and the Fiat-Shamir heuristic ensure that, in the random oracle model, this can only occur with negligible probability: Concretely, for a false statement and any first prover message, a valid response exists for at most one challenge. We have $|\Pr[W_2] - \Pr[W_1]| \leqslant Q_H \cdot 2^{-\lambda}$, where Q_H is the number of H-queries.

Game_3: We modify the challenge $\mathtt{ct}^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. The NIZK proof π^* that $(h_\ell, h_d, C_0^*, C_1^*, D_0^*, D_1^*) \in \mathcal{L}_{\mathrm{NY}}$ is now simulated by programming the random oracle H and using the NIZK simulator of Lemma 4. By Lemma 4, $|\Pr[W_3] - \Pr[W_2]| \leqslant Q_H \cdot 2^{-\lambda}$ as the distribution of π^* is statistically unchanged.

Game_4: We change the distribution of $\mathtt{ct}^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. In this game, the challenger computes

$$\begin{aligned}(C_0^*, C_1^*) &= \left(g^{t_c^*} \bmod N^{\zeta+1}, (1+N)^{m_1^*} \cdot h_\ell^{t_c^*} \bmod N^{\zeta+1}\right),\\ (D_0^*, D_1^*) &= \left(g^{t_d^*} \bmod N^{\zeta+1}, (1+N)^{m_0^*} \cdot h_d^{t_d^*} \bmod N^{\zeta+1}\right)\end{aligned}$$

where $t_c, t_d \leftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and $h_\ell = g^{x_\ell} \bmod N^{\zeta+1}$ is the public key at epoch ℓ.

The IND-CR-CPA security of the UPKE scheme in Section 4.2.2 implies – via a reduction that proceeds identically to that in the proof Theorem 4 – the indistinguishability of Game_4 and Game_3 under the DCR assumption. We have $|\Pr[W_4] - \Pr[W_3]| \leqslant \mathbf{Adv}^{\mathrm{DCR}}(\lambda)$.

Game_5: We change again the decryption oracle. For a decryption query of the form $\mathtt{ct} = (C_0, C_1, D_0, D_1, \pi)$ at any epoch i, the challenger does no longer decrypt (D_0, D_1) using x_d. Instead, it comes back to decrypting the pair (C_0, C_1) using the current secret key sk_i, by computing

$$m = \mathsf{DLog}\left(C_1^2 \cdot C_0^{-2 \cdot \mathsf{sk}_i} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^{\zeta}.$$

We note that Game_5 is perfectly indistinguishable from Game_4 until $\mathcal{A}$ queries $\mathcal{O}_{\mathtt{dec}}(\cdot)$ on a ciphertext $\mathtt{ct}$ where (C_0, C_1) and (D_0, D_1) decrypt to distinct messages even though π is a verifying proof for $\mathcal{L}_{\mathrm{NY}}$.[2] Here, the simulation-soundness of the NIZK construction presented in Section 4.3.1 – which holds in the random oracle model assuming that factoring is hard when we apply the Fiat-Shamir heuristic – ensures that this only occurs with negligible probability if the DCR assumption holds. We have $|\Pr[W_5] - \Pr[W_4]| \leqslant \mathbf{Adv}^{\text{sim-sound}}(\lambda)$.

2 This includes the case of a decryption query on the challenge ciphertext $\mathtt{ct}^*$ for an epoch $i > \ell$. We included the epoch number among the inputs of the hash function H to cover this case: If $\mathcal{A}$ can come up with a valid proof π' for $(C_0^*, C_1^*, D_0^*, D_1^*)$ at a later epoch $i > \ell$, we have $\pi' \neq \pi^*$ with probability $1 - 2^{-\lambda}$.

Game_6: We change again the distribution of the computed challenge ciphertext $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$. In the challenge phase, the challenger now computes

$$\begin{aligned}(C_0^*, C_1^*) &= \left(g^{t_c^*} \bmod N^{\zeta+1}, (1+N)^{m_1^*} \cdot h_\ell^{t_c^*} \bmod N^{\zeta+1}\right),\\ (D_0^*, D_1^*) &= \left(g^{t_d^*} \bmod N^{\zeta+1}, (1+N)^{m_1^*} \cdot h_d^{t_d^*} \bmod N^{\zeta+1}\right)\end{aligned}$$

where $t_c, t_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and $h_\ell = g^{x_\ell} \bmod N^{\zeta+1}$ is the public key at epoch ℓ. The IND-CPA security of the Elgamal-Paillier PKE scheme ensures that Game_6 is indistinguishable from Game_5 under the DCR assumption: i.e., $|\Pr[W_6] - \Pr[W_5]| \leqslant \mathbf{Adv}^{\mathrm{DCR}}(\lambda)$.

Game_7: In the generation of the challenge $ct^* = (C_0^*, C_1^*, D_0^*, D_1^*, \pi^*)$, we change again the generation of π^*, which is now computed as a real proof using the witnesses $t_c, t_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$. By Lemma4, we have $|\Pr[W_7] - \Pr[W_6]| \leqslant Q_H \cdot 2^{-\lambda}$ as the distribution of π^* is statistically close to that of Game_6.

Game_8: Here, we change again the generation of public parameters. Instead of setting $h_d = g^{x_d} \bmod N^{\zeta+1}$, where $x_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$, we get back to sampling h_d uniformly in the subgroup of $2N^\zeta$-th residues in $\mathbb{Z}^*_{N^{\zeta+1}}$. The distribution of pp is statistically close to that of Game_7 and we have $|\Pr[W_7] - \Pr[W_6]| \leqslant 2^{-\lambda}$.

In Game_8, we are exactly in the real game of Definition 5 when the challenger's bit is $b = 1$. Moreover, by combining the above, we find that that Game_0 and Game_8 are indistinguishable under the DCR assumption as the latter implies the simulation-soundness of our NIZK argument for $\mathcal{L}_{NY}$. □

4.3.3 *Arguments of Well-formedness for Update Ciphertexts*

	|sk|	|pk|	|ct|	|up|	IND-*	ROM	Assumption
Jost *et al.*	32B	64B	64B	64B	CR-CPA	Yes	CDH
Dodis *et al.*	160B	41KB	41KB	52MB	CR-CPA	No	DDH
This work	580B	760B	1.5KB	1.5KB	CR-CPA	No	DCR ($\zeta = 1$)
	580B	760B	8.2KB	1.5KB	CR-CCA	Yes	DCR ($\zeta = 1$)
	580B	1.2KB	11KB	13KB	CU-CCA	Yes	DCR ($\zeta = 2$)

Table 4: Comparison of key/ciphertext/update sizes for existing UPKE schemes with 128-bit strength security

To achieve IND-CU-CCA security, we rely on proofs that the updates are well-formed. We obtain these proofs by using a classical Schnorr-like proof in hidden-order groups.

Given g, h of order pq in $\mathbb{Z}^*_{N^{\zeta+1}}$, where $\zeta \geqslant 2$. Let $B = (N-1)/4$. We need to prove membership of the language

$$\begin{aligned}\mathcal{L}_{WFU} = \big\{ \big((h,h') \in (\mathbb{Z}^*_{N^{\zeta+1}})^2,\ up = (U,V)\big) \mid \exists k \in [0,B],\\ r \in [-B,B] :\ U = g^k \bmod N^{\zeta+1}\\ \wedge V = (1+N)^r \cdot h^k \bmod N^{\zeta+1} \wedge (h/h')^2 = g^r \bmod N^{\zeta+1}\big\},\end{aligned}$$

which is the language of well-formed updates. In the proof of soundness, however, we can only guarantee membership of

$$\begin{aligned}\bar{\mathcal{L}}_{WFU} \quad = \quad & \big\{ \big((h,h') \in (\mathbb{Z}^*_{N^{\zeta+1}})^2,\ up = (U,V)\big) \mid \\ & \exists k \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B],\ r \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B] :\\ & U^2 = g^{2k} \bmod N^{\zeta+1} \wedge (h/h')^2 = g^{2r} \bmod N^{\zeta+1}\\ \wedge \quad & V^2 = (1+N)^{2r} \cdot h^{2k} \bmod N^{\zeta+1}\big\},\end{aligned}$$

so that we have a soundness slack as the actual witnesses (k, r) lives in $[0, B] \times [-B, B]$.

We assume that $\zeta \geqslant 2$, so that the interval $[0, 2^{2\lambda}B]$ fits in the message space $\mathbb{Z}_{N^\zeta}$ of the encryption scheme. We define the following Σ-protocol Σ_{WFU} for $(\mathcal{L}_{WFU}, \bar{\mathcal{L}}_{WFU})$:

- $\mathcal{P}$ picks $r', k' \hookleftarrow \left[-2^{2\lambda}B, 2^{2\lambda}B\right]$, and sends

$$\begin{aligned} U' &= g^{2k'} \bmod N^{\zeta+1},\\ V' &= (1+N)^{2r'} \cdot h^{2k'} \bmod N^{\zeta+1},\\ H &= g^{2r'} \bmod N^{\zeta+1}\end{aligned}$$

  to the verifier.

- $\mathcal{V}$ sends a random challenge $c \hookleftarrow \mathcal{U}([0, 2^\lambda - 1])$.

- $\mathcal{P}$ computes responses $\tilde{k} = k' + ck$ and $\tilde{r} = r' + cr$ over $\mathbb{Z}$. If $\tilde{k}, \tilde{r} \in \left[-2^{2\lambda}B, 2^{2\lambda}B\right]$, it sends $\tilde{k}, \tilde{r}$ to $\mathcal{V}$, else it aborts.

- $\mathcal{V}$ first verifies that $\tilde{k}, \tilde{r} \in \left[-2^{2\lambda}B, 2^{2\lambda}B\right]$, then that

$$\begin{aligned} U' &= U^{-2c} \cdot g^{2\tilde{k}} \bmod N^{\zeta+1},\\ V' &= V^{-2c} \cdot (1+N)^{2\tilde{r}} \cdot h^{2\tilde{k}} \bmod N^{\zeta+1},\\ H &= (h/h')^{-2c} \cdot g^{2\tilde{r}} \bmod N^{\zeta+1}\end{aligned} \tag{6}$$

  It accepts if all this checks are correct, and rejects otherwise.

The special-soundness property can be proven under the Strong RSA assumption, by adapting a technique from Camenisch and Shoup [27]. One difference is that, here, the encrypted discrete logarithm lives in $[-(N-1)/4, (N-1)/4]$ (instead of $\mathbb{Z}_\rho$ for a public prime order $\rho < 2^\lambda pq$ in [27, Section 5.2]). Our use of a larger message space $\mathbb{Z}_{N^\zeta}$ with $\zeta \geqslant 2$ allows the proof to go through by adapting techniques from [27] and [46].

Lemma 6. *The above Σ-protocol provides soundness for the language $\bar{\mathcal{L}}_{WFU}$ under the Strong RSA assumption.*

Proof. Let us assume that a prover can come up with two different accepting transcripts $((U', V', H), c_1, (\tilde{k}_1, \tilde{r}_1))$, $((U', V', H), c_2, (\tilde{k}_2, \tilde{r}_2))$. We show that, unless the Strong RSA assumption is false, it must be that

$$((h, h'), (U, V)) \in \bar{\mathcal{L}}_{WFU}.$$

We define $\Delta c = c_1 - c_2$, $\Delta k = \tilde{k}_1 - \tilde{k}_2$, $\Delta r = \tilde{r}_1 - \tilde{r}_2$ with $\Delta k \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B]$, $\Delta r \in \left[-2^{2\lambda+1}B, 2^{2\lambda+1}B\right]$. From the verification equations (6), we have

$$\begin{aligned} U^{2\Delta c} &= g^{2\Delta k} \bmod N^{\zeta+1}, \\ V^{2\Delta c} &= (1+N)^{2\Delta r} \cdot h^{2\Delta k} \bmod N^{\zeta+1}, \\ (h/h')^{2\Delta c} &= g^{2\Delta r} \bmod N^{\zeta+1}. \end{aligned} \tag{7}$$

Using the same arguments as [27, Theorem 4], we first show that, unless the prover can compute a non-trivial root of $g \in \mathbb{Z}^*_{N^{\zeta+1}}$, Δc divides both Δr and Δk. Let us first assume that $d = \gcd(\Delta c, \Delta r) < \Delta c$. Then, there exist efficiently computable integers $\alpha, \beta \in \mathbb{Z}$ such that $\alpha\Delta c + \beta\Delta r = d$ and the last equation of (7) implies

$$g^{2d} = \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{2\Delta c} \bmod N^{\zeta+1},$$

which in turn yields

$$g^2 = \psi \cdot \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{2(\Delta c/d)} \bmod N^{\zeta+1}, \tag{8}$$

for some $\psi \in \mathbb{Z}^*_{N^{\zeta+1}}$ such that $\psi^d = 1 \bmod N^{\zeta+1}$. This equality implies that $\mathrm{ord}(\psi) | d$. Since $\psi \in \mathbb{Z}^*_{N^{\zeta+1}}$, we have $\mathrm{ord}(\psi) | 2N^{\zeta}pq$. Then, since we also have $d|\Delta c$ and that $\gcd(\Delta c, N^{\zeta}pq) = 1$, we conclude that $\mathrm{ord}(\psi) = 2$. Since we cannot have $\psi = -1$ (as -1 is not a square in $\mathbb{Z}^*_{N^{\zeta+1}}$ when $P = Q = 3 \bmod 4$), we have either $\psi = 1$, or a non-trivial factor of $N = PQ$ is revealed by computing $\gcd(\psi + 1, N)$ (because $(\psi+1)(\psi-1) = 0 \bmod N^{\zeta+1}$). Unless the factoring assumption (and thus the Strong RSA assumption) is broken, we thus have

$$g^2 = \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{2(\Delta c/d)} \bmod N^{\zeta+1}. \tag{9}$$

Then, we distinguish three cases. If $\Delta c/d$ is even, we have

$$g = \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{\Delta c/d} \bmod N^{\zeta+1}. \tag{10}$$

since squaring is a permutation in the subgroup of squares when $P = Q = 3 \bmod 4$, and $\mu \triangleq g^{\alpha} \cdot (h/h')^{\beta} \bmod N^{\zeta+1}$ is then a non-trivial root of g. If $\Delta c/d$ is odd and

$$g \neq \pm \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{\Delta c/d} \bmod N^{\zeta+1},$$

we obtain a non-trivial factor of $N = PQ$. Finally, if $\Delta c/d$ is odd and

$$g = \pm \left(g^{\alpha} \cdot (h/h')^{\beta}\right)^{\Delta c/d} \bmod N^{\zeta+1}, \tag{11}$$

then $g = \left(\pm g^{\alpha} \cdot (h/h')^{\beta}\right)^{\Delta c/d} \bmod N^{\zeta+1}$ and we also have a root $\mu \triangleq \pm g^{\alpha} \cdot (h/h')^{\beta} \bmod N^{\zeta+1}$ of g.

We can show in the same way that the Strong RSA assumption is broken if Δc does not divide Δk.

We now assume that $\Delta c|\Delta r$ and $\Delta c|\Delta k$. Since $\Delta c < \min(p, q)$, we necessarily have $\gcd(\Delta c, pq) = 1$ and $\gcd(\Delta c, N) = 1$. Then, if we define $\hat{c} = \Delta c^{-1} \bmod N^{\zeta} pq$, the above arguments imply

$$\begin{aligned} U^2 &= g^{2\Delta k\hat{c}} \bmod N^{\zeta+1} = g^{2(\Delta k/\Delta c)} \bmod N^{\zeta+1}, \\ V^2 &= (1+N)^{2\Delta r\cdot\hat{c}} \cdot h^{2\Delta k\hat{c}} \bmod N^{\zeta+1} \\ &= (1+N)^{2(\Delta r/\Delta c)} \cdot h^{2(\Delta k/\Delta c)} \bmod N^{\zeta+1}, \\ (h/h')^2 &= g^{2(\Delta r\cdot\hat{c})} \bmod N^{\zeta+1} \\ &= g^{2(\Delta r/\Delta c)} \bmod N^{\zeta+1}, \end{aligned}$$

which means that $((h, h'), (U, V)) \in \bar{\mathcal{L}}_{WFU}$ since we have $\Delta r/\Delta c \in [-2^{2\lambda+1}B, 2^{2\lambda+1}B]$. □

Lemma 7. *The Σ-protocol Σ_{WFU} satisfies the special honest verifier zero-knowledge property for the language $\mathcal{L}_{WFU}$.*

Proof. In the special honest verifier setting, the simulator is given the challenge c and has to simulate an accepting transcript with the proper distribution for $h', (U, V) \in \mathcal{L}_{WFU}$. We can uniformly sample $\tilde{k}, \tilde{r} \leftarrow \mathcal{U}([-2^{2\lambda}B, 2^{2\lambda}B])$ and then set

$$\begin{aligned} U' &= U^{-2c} \cdot g^{2\tilde{k}} \bmod N^{\zeta+1}, \\ V' &= V^{-2c} \cdot (1+N)^{2\tilde{r}} \cdot h^{2\tilde{k}} \bmod N^{\zeta+1}, \\ H &= (h/h')^{-2c} \cdot g^{2\tilde{r}} \bmod N^{\zeta+1}. \end{aligned}$$

We then return $(U', V', H, c, \tilde{k}, \tilde{r})$.

This is a valid transcript as it satisfies (6). Moreover, for any true statement in the language $(h', (U, V)) \in \mathcal{L}_{WFU}$, we know that $\exists k \in [0, B]$, $r \in [-B, B]$ such that

$$\begin{aligned} U^2 &= g^{2k} \bmod N^{\zeta+1}, \\ V^2 &= (1+N)^{2r} \cdot h^{2k} \bmod N^{\zeta+1}, \\ (h/h')^2 &= g^{2r} \bmod N^{\zeta+1}. \end{aligned}$$

We thus have:

$$\begin{aligned} U' &= g^{2\tilde{k}-2cr} \bmod N^{\zeta+1}, \\ V' &= (1+N)^{2\tilde{r}-2cr} \cdot h^{2\tilde{k}-2ck} \bmod N^{\zeta+1}, \\ H &= g^{2\tilde{r}-2cr} \bmod N^{\zeta+1} \end{aligned}$$

Using Lemma 2, we know that the distributions of $\tilde{k}, \tilde{r}$ in a real interaction are statistically uniform in $\left[-2^{2\lambda}B, 2^{2\lambda}B\right]$ as $|c \cdot r|, |c \cdot k| < 2^{\lambda} \cdot B$ and $(2^{\lambda} \cdot B)/(2^{2\lambda} \cdot B) = \mathsf{negl}(\lambda)(\lambda)$. Since (U', V', H) is uniquely determined by the statement and the triple $(c, \tilde{k}, \tilde{r})$ in the verification equations (6), our simulated transcript is statistically indistinguishable from a real transcript. □

4.3.4 *IND-CU-CCA-secure UPKE*

The main differences with our IND-CR-CCA construction presented in Section 4.3.2 are that: (i) Each update message up comes with a non-interactive argument showing that it was properly generated; (ii) secret keys are chosen in a larger interval over the integers. In addition, ciphertexts are computing using a somewhat larger modulus as we need $\zeta \geqslant 2$ in order to ensure the soundness of our Σ-protocol in Section 4.3.3.

- $\mathsf{GGen}(1^{\lambda})$:
  1. Choose λ-bit safe primes $P = 2p + 1, Q = 2q + 1$ and set $N = PQ$. Choose an integer $\zeta \geqslant 2$.
  2. Choose $\mu, \mu_d, \mu'_d \hookleftarrow \mathcal{U}(\mathbb{Z}_N)$ at random. Then, compute generators of the subgroup of order $n = \varphi(N)/4 = pq$ by setting $g = \mu^{2N^{\zeta}} \bmod N^{\zeta+1}$, $h_d = \mu_d^{2N^{\zeta}} \bmod N^{\zeta+1}$, and $h'_d = \mu_d'^{2N^{\zeta}} \bmod N^{\zeta+1}$.
  3. Choose hash functions $H, H' : \{0,1\}^* \to \{0,1\}^{\lambda}$ that will be modeled as random oracles in the analysis.

  Define $pp = (N, \zeta, g, h_d, h'_d, H, H')$ as public parameters.

- $\mathsf{KGen}(pp)$: Sample a random $x \hookleftarrow \mathcal{U}([-2^{\lambda}B, 2^{\lambda}B])$. Define the key pair (sk, pk) by setting $sk = (\tau, x)$ and $pk = (\tau, h)$ where $\tau = 0$ and $h = g^x \bmod N^{\zeta+1}$.

- $\mathsf{Enc}(pk, m)$: To encrypt $m \in \mathbb{Z}_{N^{\zeta}}$ under $pk = (\tau, h) \in \mathbb{N} \times \mathbb{Z}^*_{N^{\zeta+1}}$, conduct the following steps.
  1. Pick $t_c, t_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$ and compute

$$\begin{aligned} (C_0, C_1) &= \left(g^{t_c} \bmod N^{\zeta+1}, (1+N)^m \cdot h^{t_c} \bmod N^{\zeta+1}\right) \\ (D_0, D_1) &= \left(g^{t_d} \bmod N^{\zeta+1}, (1+N)^m \cdot h_d^{t_d} \bmod N^{\zeta+1}\right). \end{aligned}$$

2. Using witnesses (t_c, t_d, m), generate a NIZK proof π that

$$(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY},$$

where $\mathcal{L}_{NY}$ is the language defined in Section 4.3.1. This proof π is of the form

$$(c, \tilde{t}_c, \tilde{t}_d, \tilde{m}) \in \{0,1\}^\lambda \times \left[0, 2^{2\lambda} \cdot (N-1)/4\right]^2 \times \mathbb{Z}_{N^\zeta},$$

with $c = H\big(\tau, (h, h_d, C_0, C_1, D_0, D_1), (C_0', C_1', D_0', D_1')\big)$, and where the tuple (C_0', C_1', D_0', D_1') satisfies (4).

Output the ciphertext $ct = (C_0, C_1, D_0, D_1, \pi)$.

- $\mathsf{Dec}(\mathsf{sk}, ct)$: Parse ct as $(C_0, C_1, D_0, D_1, \pi)$ and sk as $(\tau, x) \in \mathbb{N} \times \mathbb{Z}$. Return $\perp$ if π is not a verifying NIZK argument that $(h, h_d, C_0, C_1, D_0, D_1) \in \mathcal{L}_{NY}$. Otherwise, return

$$m = \mathsf{DLog}\left(C_1^2 \cdot C_0^{-2x} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^\zeta.$$

- $\mathsf{UpdatePk}(\mathsf{pk})$: To update $\mathsf{pk} = (\tau, h) \in \mathbb{Z}^*_{N^{\zeta+1}}$,

1. Pick $r \hookleftarrow \mathcal{U}(\{-B, \ldots, B\})$, set $h' = h \cdot g^r \bmod N^{\zeta+1}$.
2. Compute $(U_0, V_0, U_1, V_1, \pi)$ as

$$\begin{aligned}(U_0, V_0) &= \left(g^{t_c} \bmod N^{\zeta+1}, (1+N)^r \cdot h^{t_c} \bmod N^{\zeta+1}\right)\\ (U_1, V_1) &= \left(g^{t_d} \bmod N^{\zeta+1}, (1+N)^r \cdot {h_d'}^{t_d} \bmod N^{\zeta+1}\right),\end{aligned}$$

for random $t_c, t_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$, and π a proof that $(h, h_d', U_0, V_0, U_1, V_1) \in \mathcal{L}_{NY}$ computed using (t_c, t_d, r), as described in Enc. Note however that here the public key h_d' replaces h_d in the Enc algorithm.

3. Using witness (t_c, r), generate a NIZK argument π_{up} that $(h, h', U_0, V_0) \in \mathcal{L}_{WFU}$, where $\mathcal{L}_{WFU}$ is the language of Section 4.3.3. This argument is of the form

$$\pi_{up} = (c_{up}, \tilde{k}, \tilde{r}) \in \{0,1\}^\lambda \times \left[-2^{2\lambda}B, 2^{2\lambda}B\right]^2,$$

with

$$c_{up} = H'\big(\tau, (h, h', U_0, V_0), (U', V', H)\big),$$

and where (U', V', H) satisfy (6).

Return $\big(up = (U_0, V_0, U_1, V_1, \pi, \pi_{up}), \mathsf{pk}' = (\tau + 1, h')\big)$.

- $\mathsf{UpdateSk}(\mathsf{sk}, up)$: Given $\mathsf{sk} = (\tau, x) \in \mathbb{N} \times \mathbb{Z}$ and

$$\big(up = (U_0, V_0, U_1, V_1, \pi, \pi_{up}), \mathsf{pk}' = (\tau', h')\big),$$

return $\perp$ if $\tau' \neq \tau + 1$. Otherwise,

1. Return $\perp$ if $(U_0, V_0, U_1, V_1) \notin (\mathbb{Z}^*_{N^{\zeta+1}})^4$ or $h' \notin \mathbb{Z}^*_{N^{\zeta+1}}$.
2. Return $\perp$ if $\pi = (c, \tilde{t}_c, \tilde{t}_d, \tilde{m}) \in \{0,1\}^\lambda \times \left[0, 2^{2\lambda} \cdot \frac{N-1}{4}\right]^2 \times \mathbb{Z}_{N^\zeta}$ does not parse properly or if it is not a valid NIZK argument that $(h, h'_d, U_0, V_0, U_1, V_1) \in \mathcal{L}_{NY}$.
3. Return $\perp$ if $\pi_{up} = (c_{up}, \tilde{k}, \tilde{r}) \in \{0,1\}^\lambda \times [-2^{2\lambda}B, 2^{2\lambda}B]^2$ does not parse properly or if it is not a valid NIZK argument that $(h, h', U_0, V_0) \in \mathcal{L}_{WFU}$.
4. Compute
$$r = \mathsf{DLog}\left(V^2 \cdot U^{-2x} \bmod N^{\zeta+1}\right) \cdot 2^{-1} \bmod N^\zeta.$$
5. Let $\bar{r} = \min(r, N^\zeta - r)$ and let $b_r \in \{0,1\}$ such that
$$\bar{r} = (1 - b_r) \cdot r + b_r \cdot (N^\zeta - r) \bmod N^\zeta.$$
Compute $x' = x + (-1)^{b_r} \cdot r$ over $\mathbb{Z}$.

Return $\mathsf{sk}' = (\tau + 1, x') \in \mathbb{N} \times \mathbb{Z}$.

- $\mathsf{VerifyUpdate}(h = \mathsf{pk}, \mathsf{up}, \mathsf{pk}')$: Let $\mathsf{up} = (U_0, V_0, U_1, V_1, \pi, \pi_{up})$. Return $\perp$ if π is not a valid NIZK proof for the statement
$$(h, h_d, (U_0, V_0, U_1, V_1)) \in \mathcal{L}_{NY}.$$
Then, verify that π_{up} is a valid NIZK argument that
$$(h, \mathsf{pk}', U_0, V_0) \in \mathcal{L}_{WFU},$$
in which case return 1, else return $\perp$.

We now prove that the scheme provides IND-CU-CCA-security in the ROM assuming that the Strong RSA assumption holds and that the scheme of Section 4.3.2 provides IND-CR-CCA security.

Theorem 6. *The above construction provides IND-CU-CCA security assuming that: (i) The DCR assumption holds; (ii) The NIZK proof for $\mathcal{L}_{NY}$ provides simulation-soundness; (iii) The NIZK argument for $\mathcal{L}_{WFU}$ provides computational soundness*

Proof. The proof uses with a sequence of games. For each i, W_i denotes the event that the adversary $\mathcal{A}$ outputs 0 in Game_i.

Game_0: This is the original IND-CU-CCA game where the challenger's bit is $b = 0$.

Game_1: We replace the proof π^*_{up} in the final update message
$$\mathsf{up}^* = (U^*_0, V^*_0, U^*_1, V^*_1, \pi^*, \pi^*_{up})$$
by a simulated NIZK proof obtained by programming H'. We have $|\Pr[W_1] - \Pr[W_0]| \leqslant 2^{-\lambda}$ as the distribution of π^*_{up} is statistically unchanged. We note that $\mathcal{A}$ can only see

this simulated proof at the very end of the game, *after* having submitted all its update queries. Therefore, $\mathcal{A}$ never gets to generate a proof for $\mathcal{L}_{WFU}$ after having seen this simulated proof.

Game_2: In this game, we replace the proof π^* of plaintext equality in the final update $up^* = (U_0^*, V_0^*, U_1^*, V_1^*, \pi^*, \pi_{up}^*)$ by a simulated proof, which is obtained by programming the random oracle H. By Lemma 4, we have $|\Pr[W_2] - \Pr[W_1]| \leqslant 2^{-\lambda}$.

Game_3: We now change the (U_1^*, V_1^*) components of up^* to be an encryption of 0 under the public key h_d. Thanks to the standard IND-CPA security of the Elgamal-Paillier PKE scheme, these two games are indistinguishable under the DCR assumption, and we have $|\Pr[W_3] - \Pr[W_2]| \leqslant \mathbf{Adv}^{\mathrm{DCR}}(\lambda)$.

Game_4: We modify the generation of public parameters and now choose h_d' as $h_d' = g^{x_d'} \bmod N^{\zeta+1}$, where $x_d' \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$. Since the element h_d' remains statistically uniform in the subgroup of $2N^{\zeta}$-th residues, the distribution of pp is statistically close to that of Game_0. We have $|\Pr[W_4] - \Pr[W_3]| \leqslant 2^{-\lambda}$.

Game_5: We introduce the following check: When the adversary makes an update query $(U_0, V_0, U_1, V_1, \pi, \pi_{up}, pk' = ((i+1), h'))$, the challenger uses the secret key sk_i to decrypt (U_0, V_0) and check that the underlying plaintext $r \in \mathbb{Z}$ (as decrypted at step 4 of UpdateSk) is such that $h' = h_i \cdot g^{(-1)^{b_r} \cdot r} \bmod N^{\zeta+1}$, where $h_i \in \mathbb{Z}_{N^{\zeta+1}}^*$ is part of pk_i. It halts if it is not the case. We denote by E_5 the event that it halts. Game_5 and Game_4 are identical until event E_6 occurs and we have $|\Pr[W_5] - \Pr[W_4]| \leqslant \Pr[E_5]$. The computational soundness of our Σ-protocol for $\bar{\mathcal{L}}_{WFU}$ (together with the soundness of Fiat-Shamir in the ROM) guarantees that $\Pr[E_5] \leqslant \mathbf{Adv}^{\mathrm{SRSA}}(\lambda)$. Here, note that the fact that the simulated proof π_{up}^* is revealed only at the very end of the game plays a crucial role and allows relying on the standard soundness rather than on simulation-soundness.

Game_6: This game is identical to Game_5 except that, at each query to $\mathcal{O}_{upd}(\cdot)$, we use x_d' instead of the actual secret sk_i of the current epoch i. Namely, when the adversary makes an update query $(U_0, V_0, U_1, V_1, \pi, \pi_{up}, pk')$ that passes VerifyUpdate, the challenger does no longer use sk_i to decrypt (U_0, V_0) at step 4 of UpdateSk. Instead, it uses x_d' to decrypt (U_1, V_1). Game_6 proceeds identically to Game_5 until the event E_6 that an $\mathcal{O}_{upd}(\cdot)$-query involves pairs (U_0, V_0), (U_1, V_1) that decrypt to distinct values although π verifies. The statistical soundness of π (which is guaranteed in the ROM when Fiat-Shamir is applied to our Σ-protocol in Section 4.3.1) ensures that $\Pr[E_6] \leqslant 2^{-\lambda}$. We insist that we do not rely on simulation-soundness here since E_6 occurs before $\mathcal{A}$ is allowed to see up^* (and thus before it gets to see a simulated proof).

Game_7: This final game is identical to the previous game except that the challenger encrypts m_1 instead of m_0. Lemma 8 shows that these two games are indistinguishable under the IND-CR-CCA security of our construction from Section 4.3.2.

We have now switched the challenge ciphertext to be an encryption of m_1^* instead of m_0^*, and the claim follows by using the same sequence of hybrid games backwards to go back to the IND-CU-CCA security game where the challenger's bit is $b = 1$. □

Lemma 8. Game_6 *and* Game_7 *are computationally indistinguishable if the scheme of Section 4.3.2 provides IND-CR-CCA security.*

Proof. Assuming that there exists a PPT adversary $\mathcal{A}$ that can distinguish between Game_6 and Game_7, we build a PPT adversary $\mathcal{B}$ against the IND-CR-CCA security of the construction in Section 4.3.2. Algorithm $\mathcal{B}$ receives as input public parameters pp and pk_0 from its challenger. It appends h'_d to the public parameters pp, where it computes h'_d as $h'_d = g^{x'_d} \bmod N^{\zeta+1}$ by sampling $x'_d \hookleftarrow \mathcal{U}(\mathbb{Z}_{(N-1)/4})$, and forwards $(\mathsf{pp}' \triangleq \mathsf{pp} \cup \{h'_d\}, \mathsf{pk}_0)$ to $\mathcal{A}$. $\mathcal{B}$ has access to a random oracle H'' while $\mathcal{A}$ has access to two random oracles H, H'. Except at the end of this reduction, $\mathcal{B}$ uses H'' to answer oracle H-queries while H' is honestly simulated.

During stages 2 and 4, when $\mathcal{A}$ makes a decryption query, $\mathcal{B}$ submits the same query to its decryption oracle and relays the result to $\mathcal{A}$. When $\mathcal{A}$ makes an update query $(U_0, V_0, U_1, V_1, \pi, \pi_{up}, \mathsf{pk}')$, $\mathcal{B}$ uses its knowledge of x'_d to decrypt (U_1, V_1). Thanks to the two tweaks introduced in Game_5 and Game_6, $\mathcal{B}$ is guaranteed to recover the correct randomness r used by $\mathcal{A}$ to generate its update Then, $\mathcal{B}$ submits this r to its own update oracle in the IND-CR-CCA game.

At stage 3, when $\mathcal{A}$ submits a pair of challenge messages (m_0^*, m_1^*), adversary $\mathcal{B}$ submits the same pair to its IND-CR-CCA challenger. It then relays the latter's challenge ciphertext c^* to $\mathcal{A}$.

Finally, at stage 6 of the IND-CR-CCA game, $\mathcal{B}$ obtains a tuple $(\mathsf{pk}^* = (\ell^*, h^*), \mathsf{sk}^*, up^*)$, where $up^* = (U_0^*, V_0^*)$. It then generates an encryption (U_1^*, V_1^*) of 0 under key h'_d. Let $h_{\ell'}$ the public key at the last epoch ℓ' before moving to pk^*. $\mathcal{B}$ generates a simulated proof π^* for the statement $(h_{\ell'}, h'_d, U_0^*, V_0^*, U_1^*, V_1^*) \in \mathcal{L}_{NY}$ by programming the random oracle H. Note that H must be programmed on an input which is unpredictable to $\mathcal{A}$ (as it is chosen by $\mathcal{B}$'s challenger), so that a collision on H occurs with negligible probability $\leqslant Q_H \cdot 2^{-\lambda}$. Finally, adversary $\mathcal{B}$ generates a simulated proof π_{up}^* for the statement $(h_{\ell'}, h^*, U_0^*, V_0^*)$ by programming the random oracle H'. It then sends $\mathcal{A}$ the tuple $(\mathsf{pk}^*, \mathsf{sk}^*, (U_0^*, V_0^*, U_1^*, V_1^*, \pi^*, \pi_{up}^*))$. When $\mathcal{A}$ halts and outputs $b' \in \{0, 1\}$, adversary $\mathcal{B}$ outputs the same b'.

The above reduction shows

$$|\Pr[W_7] - \Pr[W_6]| \leqslant \mathbf{Adv}^{\text{cr-cca}}(\lambda) + Q_H \cdot 2^{-\lambda},$$

where $\mathbf{Adv}^{\text{cr-cca}}(\lambda)$ denotes the advantage in the proof of Theorem 5. □

4.4 IMPLEMENTATION AND PERFORMANCES

The main advantage of our scheme resides in that its parameters make it close to practical use, in terms of key size, ciphertext size, and running time. In this section, we compare the key, ciphertext and updates sizes to the previous UPKE constructed by Dodis *et al.* in [39] and by Jost *et al.* in [56]. For the sake of completeness, we also present the running times of our implementation.

4.4.1 *Key/Ciphertext/Update Sizes*

Table 4 compares the different key/ciphertext/update sizes achieved by the existing UPKE construction and their security assumptions for a 128-bit strength security. One can remark that we avoid the quadratic blowup suffered by the Dodis *et al.* scheme in the update sizes thanks to DCR. This allows us to construct the first IND-CR-CPA secure UPKE with both reasonable key, ciphertext and update sizes in the standard model.

Jost *et al.*'s scheme remains the most efficient to date. However, its IND-CR-CPA security fully relies on the ROM and it does not extend to provide the stronger security notions that we are able to achieve with DCR.

IND-*	Encryption	Decryption	Update
CR-CPA (112 bits)	0.021 s	0.020 s	0.042 s
CR-CPA (128 bits)	0.062 s	0.062 s	0.127 s
CR-CCA (112 bits)	0.045 s	0.042 s	0.085 s
CR-CCA (128 bits)	0.092 s	0.091 s	0.202 s

Table 5: Benchmarks on our implementation of our IND-CR-CPA and IND-CR-CCA schemes

4.4.2 *Running Time*

We implemented our IND-CR-CPA and IND-CR-CCA schemes in C/C++ as a proof of concept. Our implementation relies on the GMP[3] library to handle computations on big integers. One should note that, being only a proof of concept, the implementation is heavily optimizable. Our benchmarks were made on an Apple M1 CPU with 8 cores running at 3Ghz, under macOS 12. They are presented in Table 5. The code was compiled with clang (clang-1205.0.22.9) with the optimization flag -O3. The running time of each function was estimated by taking the mean running time of 1000 evaluations. For the IND-CR-CPA version, encryptions, decryptions and updates sensibly

3 GMP https://gmplib.org

have the same running time. Simply because an update consists of an encryption and a decryption, and that decryptions require the same operations as encryptions.

The first gap appears when introducing NIZKs for Naor-Yung. As updates do not require proofs, they are a bit faster than encryptions. IND-CR-CCA security comes at the cost of losing a factor 20 in efficiency in our benchmarks.

5

A CONSTRUCTION UNDER LATTICE ASSUMPTIONS

This chapter presents our second construction of UPKE. We construct the first efficient post-quantum IND-CR-CPA scheme together with a FO transform for UPKE. Also, we prove that the Naor-Yung paradigm approach used to achieve CU security is generic.

5.1 PRELIMINARIES

We start by giving out the mathematical background and some useful lemmas needed in this chapter.

We use bold upper case letters to denote matrices ($\mathbf{A}$), bold lower case letters for vectors ($\mathbf{a}$) and italic letters for scalars (a). For any vector $\mathbf{x} = (x_1, \ldots, x_n)$, we use the ℓ_2-norm $\|\mathbf{x}\|_2 = \sqrt{\sum x_i^2}$, the ℓ_1-norm $\|\mathbf{x}\|_1 = \sum |x_i|$ and the ℓ_∞-norm $\|\mathbf{x}\|_\infty = \max |x_i|$. For any matrix $\mathbf{A} = (\mathbf{a}_1 \| \ldots \| \mathbf{a}_n)$, we define $\|\mathbf{F}\|_2 = \max \|\mathbf{a}_i\|_2$, $\|\mathbf{F}\|_1 = \max \|\mathbf{a}_i\|_1$ and $\|\mathbf{F}\|_\infty = \max \|\mathbf{a}_i\|_\infty$. We let $\lfloor \cdot \rfloor$ denote the floor function and $\lfloor \cdot \rceil$ denote the rounding to the closest integer with ties being rounded up, which are extended to vectors by considering their coefficient-wise application. For $\mathbf{x} \in \mathbb{Q}^n$ and $q > p > 0$, we write $\lfloor \mathbf{x} \rceil_{p,q}$ for $\lfloor p/q \cdot \mathbf{x} \bmod q \rceil$. In this work, the modulus q will always be implicit and omitted.

We use the convolution product to express the distribution of a sum of random variables, which we remind below as well as some additional basic operations and properties of probability distributions and discrete Gaussian distributions.

Definition 16 (Convolution)**.** *Let $m \in \mathbb{N}$. Let $\mathcal{S}_1, \mathcal{S}_2$ be two probability distribution on $\mathbb{Z}^m$. We define the convolution product $\mathcal{S}_1 * \mathcal{S}_2$ as:*

$$\mathcal{S}_1 * \mathcal{S}_2(x) = \sum_{y \in \mathbb{Z}^m} \mathcal{S}_1(x-y)\mathcal{S}_2(y).$$

If $X \sim \mathcal{S}_1$ and $Y \sim \mathcal{S}_2$ are independent random variables, then $X + Y \sim \mathcal{S}_1 * \mathcal{S}_2$.

Definition 17 (Statistical distance)**.** *Let $\mathcal{S}_1, \mathcal{S}_2$ be two distributions on $\mathbb{Z}^n$. We define the statistical $\Delta(\mathcal{S}_1, \mathcal{S}_2)$ as:*

$$\Delta(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{2} \sum_{x \in \mathbb{Z}^n} |\mathcal{S}_1(x) - \mathcal{S}_2(x)| \ .$$

5.1.1 *Gaussian distributions*

We give the definition of Gaussian distribution and several useful lemmas that are used afterwards.

Definition 18 (Gaussian distribution)**.** *Let* $m \in \mathbb{N}$*. For any symmetric positive-definite matrix* $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$*, define the function* $g_{\mathbf{\Sigma}} : \mathbb{R}^m \to \mathbb{R}$ *as*

$$\rho_{\mathbf{\Sigma}}(\mathbf{x}) = \exp\left(-\pi \frac{\mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x}}{2}\right).$$

We define the Gaussian distribution on $\mathbb{Z}^m$ *with center parameter* $\mathbf{c}$ *and covariance matrix parameter* $\mathbf{\Sigma}$ *as* $\mathcal{D}_{\mathbb{Z}^m,\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) = \rho_{\mathbf{\Sigma}}(\mathbf{x}-\mathbf{c})/\rho_{\mathbf{\Sigma}}(\mathbb{Z}^m - \mathbf{c})$*. We will also use, for* $\sigma > 0$*, the notation* $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ *to denote* $\mathcal{D}_{\mathbb{Z}^m,\sigma^2\mathbf{Id},\mathbf{0}}$*. Additionally, we will let* $\mathcal{D}_{\mathbb{Z}^{m\times n},\sigma}$ *denote the distribution obtained by sampling* n *vectors from* $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ *and viewing them as the columns of a matrix in* $\mathbb{Z}^{m\times n}$*.*

Lemma 9 (Gaussian tail-bound, [33, Lemma 2.13])**.** *Let* $\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m,\sigma}$*, then for all* $t > 1$*, we have*

$$\mathbb{P}\left[\|\mathbf{x}\|_2 \geqslant t\sigma\sqrt{\frac{m}{2\pi}}\right] \leqslant e^{-\frac{m}{2}(1-t)^2} \ .$$

Lemma 10 (Gaussian convolution, [19, Lemma 4.12])**.** *Let* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}^n$*. Let* $X \sim \mathcal{D}_{\mathbb{Z}^n,\sigma,\mathbf{c}_1}$*,* $Y \sim \mathcal{D}_{\mathbb{Z}^n,\sigma',\mathbf{c}_2}$ *and let* $\mathcal{S}$ *be the distribution followed by* $X+Y$*. Then, if*

$$\left(\frac{1}{\sigma^2} + \frac{1}{\sigma'^2}\right)^{-1/2} > \sqrt{\frac{\ln(2n(1+\frac{1}{\varepsilon}))}{\pi}} \ ,$$

then we have the following inequality

$$\Delta\left(\mathcal{S}, \mathcal{D}_{\mathbb{Z}^n,\sqrt{\sigma^2+\sigma'^2},\mathbf{c}_1+\mathbf{c}_2}\right) < \frac{2\varepsilon}{1-\varepsilon} \ .$$

We now state a discrete Gaussian decomposition result.

Lemma 11 (Gaussian decomposition, instantiated from [68, Lemma 1])**.** *For* $m \geqslant n$*, let* $\mathbf{F} \in \mathbb{Z}^{m\times n}$ *be a matrix and let* $s_1(\mathbf{F})$ *be the largest singular value of* $\mathbf{F}$*. Take* $\sigma, \sigma_1 > 0$*. Let* $\mathbf{e}_1 \sim \mathcal{D}_{\mathbb{Z}^n,\sigma_1}$ *and* $\mathbf{e}_2 \sim \mathcal{D}_{\mathbb{Z}^m,\mathbf{\Sigma}}$ *for*

$$\mathbf{\Sigma} = \sigma^2\mathbf{Id} - \sigma_1^2\mathbf{F}^T\mathbf{F} \ .$$

Then, if $\sigma > \sqrt{2}\sigma_1 s_1(\mathbf{F})$ *and* $\sigma_1 > \sqrt{2\ln(2n(1+1/\varepsilon))/\pi}$*, we have:*

$$\Delta\left(\mathcal{S}, \mathcal{D}_{\mathbb{Z}^m,\sigma}\right) < \frac{2\varepsilon}{1-\varepsilon} \ ,$$

where $\mathcal{S}$ *is the distribution of* $\mathbf{F}\mathbf{e}_1 + \mathbf{e}_2$*.*

In order to apply Lemma 11, one needs to control the ratio $s_1(\mathbf{F})$. This is the purpose of the following result.

Lemma 12 (Adapted from [2, Lemma 8]). *There exists a constant $K > 1$ such that the following holds. For $m \geqslant 2n$, $\sigma > K\sqrt{n}$ and $\mathbf{F} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m\times n},\sigma}$*

$$\mathbb{P}\left[s_1(\mathbf{F}) > K\sigma\sqrt{m}\right] < e^{-m/K} \ ,$$

where $s_1(\mathbf{F})$ denotes the largest singular value of $\mathbf{F}$

We recall the definition of γ-spreadness, which allows to bound the probability that a specific randomness r was used to produce a valid encryption. It is used in Section 5.4 for our FO transform.

Definition 19 (γ-spreadness, adapted from [41, Section 2.1]). *Let $\gamma > 0$. We say that a UPKE $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ is γ-spread if for all m, c and $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, we have*

$$\mathbb{P}\left[\mathsf{Enc}(pk, m) = c\right] \leqslant \gamma.$$

5.2 EXTENDED LWE

We start by recalling the Learning With Errors (LWE) assumption.

Definition 20. *(Learning With Errors - LWE) Let $\lambda \geqslant 0$ be a security parameter. Let $q = q(\lambda), n = n(\lambda), m = m(\lambda) \geqslant 0$, $\mathcal{S}$ be a distribution on $\mathbb{Z}_q^n$ and χ be an error distribution on $\mathbb{Z}^m$. The goal for an adversary $\mathcal{A}$ in the game $\mathsf{LWE}_{q,n,m,\chi}(\mathcal{S})$ is to distinguish between $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ and $(\mathbf{A}, \mathbf{u})$, for $\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m\times n})$, $\mathbf{s} \hookleftarrow \mathcal{S}$, $\mathbf{e} \hookleftarrow \chi^m$ and $\mathbf{u} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m)$. We define the advantage of $\mathcal{A}$ in the LWE game as*

$$\mathsf{Adv}^{\mathsf{LWE}}(\mathcal{A}) := \left|\mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{As} + \mathbf{e}) \to 1\right] - \mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{u}) \to 1\right]\right| \ .$$

To keep notations simple, we write $\mathsf{LWE}_{q,n,m,\sigma}$ with $\sigma > 0$, for $\mathsf{LWE}_{q,n,m,\mathcal{D}_{\mathbb{Z}^m,\sigma}}(\mathcal{U}(\mathbb{Z}_q^n))$.

The extended-LWE assumption claims that pseudorandomness of an LWE instance $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ still holds when the adversary is given an additional hint h computed as $\langle \mathbf{z}, \mathbf{e}\rangle$ mod q for a small $\mathbf{z}$ chosen by the adversary independently of $\mathbf{A}$. We define Adaptive extended-LWE, an adaptive version of this assumption. As the name suggests, it allows the adversary to choose the hint vector $\mathbf{z}$ adaptively, i.e. after having seen the matrix $\mathbf{A}$, which is not allowed in the definition of the extended-LWE from [71]. In Theorem 7, we prove that LWE reduces to this adaptive version.

Definition 21 (Adaptive extended-LWE - AextLWE). *Let $\lambda \geqslant 0$ be a security parameter. Let $q = q(\lambda), n = n(\lambda), m = m(\lambda), B = B(\lambda) \in \mathbb{N}$ and χ be an error distribution on $\mathbb{Z}^m$. The goal for an adversary $\mathcal{A}$ in $\mathsf{AextLWE}_{q,n,m,\chi,B}$ is to distinguish between the case where $\beta = 0$*

and $\beta = 1$ *in the interactive game depicted in Figure 5. We define the advantage of* $\mathcal{A}$ *in the* AextLWE *game as*

$$\mathsf{Adv}^{\mathsf{AextLWE}}(\mathcal{A}) := |\mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{As}+\mathbf{e}, \mathbf{z}, h) \to 1\right] - \mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{u}, \mathbf{z}, h) \to 1\right]| \ ,$$

where the elements are distributed as shown in Figure 5.

To keep notations simple, we write $\mathsf{AextLWE}_{q,n,m,\sigma,B}$*, with* $\sigma > 0$*, for* $\mathsf{AextLWE}_{q,n,m,\mathcal{D}_{\mathbb{Z}^m,\sigma},B}$.

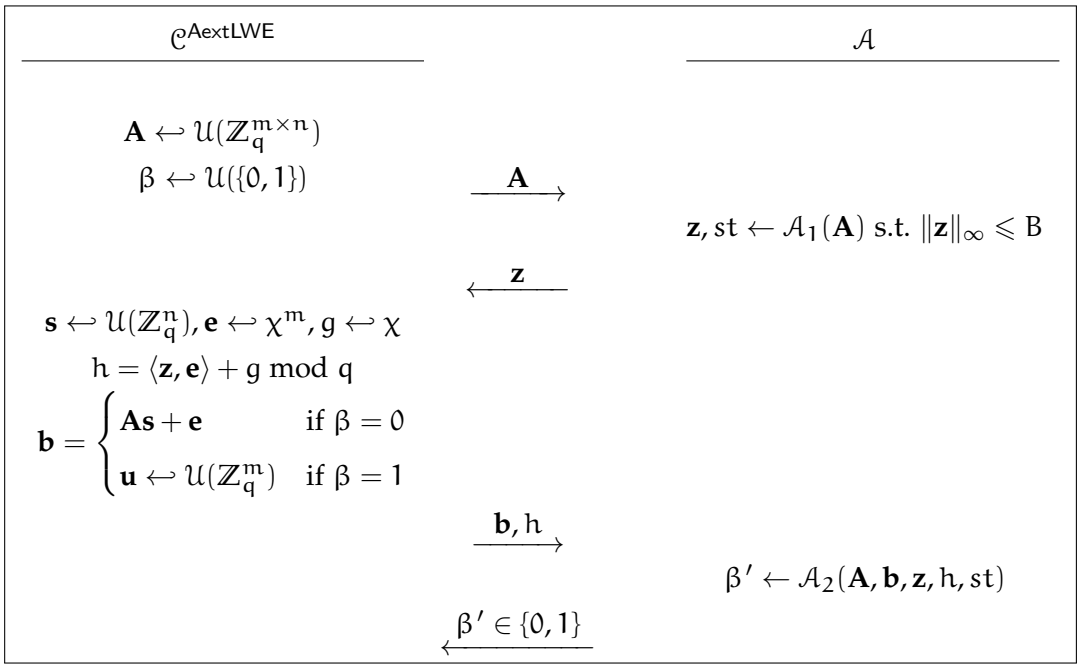

Figure 5: The decision game for $\mathsf{AextLWE}_{q,n,m,\chi}$.

We define the Hermite Normal Form (HNF) variant of Adaptive extended-LWE, based on the normal form reduction from [7, Lemma 2]. Lemma 13 shows that the HNF variant reduces to the standard Adaptive extended-LWE.

Definition 22 (HNF Adaptive extended-LWE - HNF-AextLWE)**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter. Let* $q = q(\lambda), n = n(\lambda), m = m(\lambda), B = B(\lambda) \in \mathbb{N}$ *and* χ *be an error distribution on* $\mathbb{R}^m$*. The goal for an adversary* $\mathcal{A}$ *in* $\mathsf{HNF\text{-}AextLWE}_{q,n,m,\chi,B}$ *is to distinguish between the case where* $\beta = 0$ *and* $\beta = 1$ *in the interactive game depicted in Figure 6. We define the advantage of* $\mathcal{A}$ *in the* HNF-AextLWE *game as*

$$\mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{A}) = |\mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{As}+\mathbf{e}, \mathbf{z}_0, \mathbf{z}_1, h) \to 1\right] - \mathbb{P}\left[\mathcal{A}(\mathbf{A}, \mathbf{u}, \mathbf{z}_0, \mathbf{z}_1, h) \to 1\right]|$$

where the elements are distributed as shown in Figure 6.

To keep the notations simple, we write $\mathsf{HNF\text{-}AextLWE}_{q,n,m,\sigma,B}$*, for* $\sigma > 0$*, to denote* $\mathsf{HNF\text{-}AextLWE}_{q,n,m,\mathcal{D}_{\mathbb{Z}^m,\sigma},B}$.

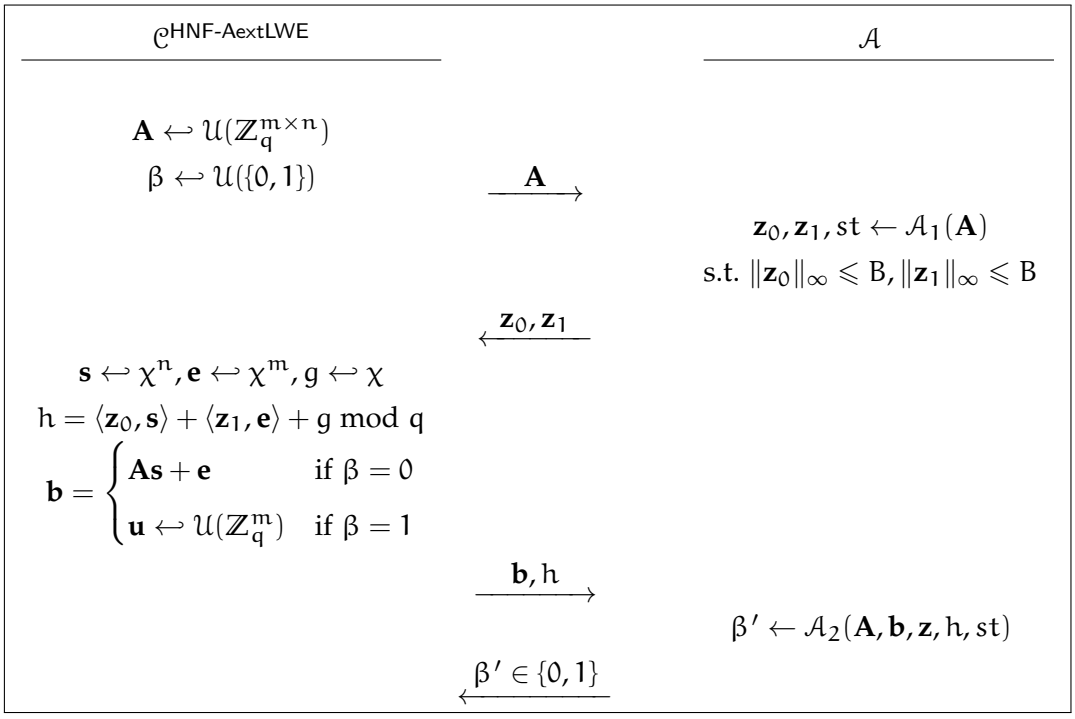

Figure 6: The decision game for $\mathsf{HNF\text{-}AextLWE}_{q,n,m,\chi}$.

Multiple-secret variants. We consider the multiple-secret variants of all our assumptions $\mathsf{Asp} \in \{\mathsf{LWE}, \mathsf{AextLWE}, \mathsf{HNF\text{-}AextLWE}\}$ which consist in considering k distinct secrets for the same public matrix $\mathbf{A}$, thus replacing the secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ by a secret matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times k}$ and the error vector $\mathbf{e}$ by an error matrix $\mathbf{E} \in \mathbb{Z}_q^{m\times k}$. Note that for AextLWE and HNF-AextLWE, the hint $h \in \mathbb{Z}_q$ also becomes a vector $\mathbf{h} \in \mathbb{Z}_q^k$. Also, the multiple-secret variants for AextLWE and HNF-AextLWE could allow for a different $\mathbf{z}$ for each secret, but we restrict ourselves to the case where the $\mathbf{z}$ is the same for all secrets, as it is all we need for our proofs.

Using a hybrid argument, one can show that for every adversary $\mathcal{A}$ for the multiple-secret variant of Asp with k secrets, there exists an adversary $\mathcal{B}$ with a similar runtime against the single-secret problem Asp such that $\mathcal{A}$'s advantage is bounded by $k \cdot \mathsf{Adv}^{\mathsf{Asp}}(\mathcal{B})$.

Lemma 13. *Let* $q \geqslant 25, n \geqslant 1, m \geqslant 16n + 4\log\log q$, *then any adversary* $\mathcal{A}$ *for the game* $\mathsf{HNF\text{-}AextLWE}_{q,n,m',\sigma,B}$, *where* $m' = m - 16n - 4\log\log q$, *running in time* T *can be used to build an adversary* $\mathcal{B}$ *for* $\mathsf{AextLWE}_{q,n,m,\sigma,B}$ *running in time* $\approx T$, *with advantage*

$$\mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{A}) \leqslant 4 \cdot \mathsf{Adv}^{\mathsf{AextLWE}}(\mathcal{B}) \ .$$

Proof. Assume $\mathcal{A}$ is an adversary against HNF-AextLWE. We construct an adversary $\mathcal{B}$ against AextLWE with the claimed advantage as follows.

Adversary $\mathcal{B}$ receives a matrix $\mathbf{A} = \left(\mathbf{A}_0^T \| \mathbf{A}_1^T\right)^T \in \mathbb{Z}_q^{m \times n}$ from the AextLWE challenger, with $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times n}$ and $\mathbf{A}_1 \in \mathbb{Z}_q^{m-n \times n}$. According to [25, Claim 2.13], with probability at least $1 - 2e^{-1} \geqslant 1/4$, there exist n linearly independent rows within the first $16n + 4\log\log q$ rows of $\mathbf{A}$ and an efficient way to find them, so that $\mathcal{B}$ can reorder the matrix so that $\mathbf{A}_0$ is invertible. If it cannot find such n rows, adversary $\mathcal{B}$ aborts. To avoid keeping track of the indices for the reordering, assume that $\mathbf{A}$ is such that $\mathbf{A}_0$ is invertible and denote by $\mathbf{A}_d$ the last $15n + 4\log\log q$ rows of $\mathbf{A}_1$ so that $\mathbf{A}_1 = (\tilde{\mathbf{A}}_1^T \| \mathbf{A}_d^T)^T$ with matrix $\tilde{\mathbf{A}}_1 \in \mathbb{Z}_q^{m' \times n}$.

It then computes $\mathbf{A}^* = -\tilde{\mathbf{A}}_1 \mathbf{A}_0^{-1} \in \mathbb{Z}_q^{m' \times n}$ and sends $\mathbf{A}^*$ to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ responds with the hint vectors $\mathbf{z}_0 \in \mathbb{Z}_q^n, \mathbf{z}_1 \in \mathbb{Z}_q^{m'}$. Then, adversary $\mathcal{B}$ forwards $\mathbf{z} = (\mathbf{z}_0^T \| \mathbf{z}_1^T \| 0^{m-m'-n})^T \in \mathbb{Z}_q^m$ to its challenger and receives a vector $\mathbf{b} = \left(\mathbf{b}_0^T \| \mathbf{b}_1^T \| \mathbf{d}^T\right)^T$ and a hint $h = \langle \mathbf{z}, \mathbf{e} \rangle + g \bmod q$ from the AextLWE challenger, with $\mathbf{b}_0 \in \mathbb{Z}_q^n, \mathbf{b}_1 \in \mathbb{Z}_q^{m'}$, $\mathbf{d} \in \mathbb{Z}_q^{m-m'}$ and $g \hookleftarrow \mathcal{D}_{\mathbb{Z},\sigma}$. It then computes $\mathbf{b}^* = \mathbf{b}_1 + \mathbf{A}^* \mathbf{b}_0$ and sends $(\mathbf{b}^*, h)$ to $\mathcal{A}$. Finally, it receives a response bit β from $\mathcal{A}$, which it forwards to its challenger.

In the case where $\mathbf{b}$ was a uniform vector, as $\mathbf{A}_0$ is an invertible matrix, matrix $\mathbf{A}^*$ is uniform and so is $\mathbf{b}^* = \mathbf{b}_1 + \mathbf{A}^* \mathbf{b}_0$.

If we are in the case where

$$\begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_0 \\ \tilde{\mathbf{A}}_1 \\ \mathbf{A}_d \end{pmatrix} \mathbf{s} + \begin{pmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \\ \mathbf{e}_d \end{pmatrix}$$

for $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathbf{e}_0 \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}, \mathbf{e}_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m'},\sigma}$ and $\mathbf{e}_d \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m-m'},\sigma}$, then

$$\mathbf{b}^* = \tilde{\mathbf{A}}_1 \mathbf{s} + \mathbf{e}_1 - \tilde{\mathbf{A}}_1 \mathbf{A}_0^{-1} \mathbf{A}_0 \mathbf{s} + \mathbf{A}^* \mathbf{e}_0 = \mathbf{A}^* \mathbf{e}_0 + \mathbf{e}_1.$$

Furthermore, the hint is exactly

$$\begin{aligned} \langle \mathbf{z}, \mathbf{e} \rangle + g &= \left\langle \mathbf{z}_0^T \| \mathbf{z}_1^T \| 0^{m-m'}, \mathbf{e}_0^T \| \mathbf{e}_1^T \| \mathbf{e}_d^T \right\rangle + g \\ &= \langle \mathbf{z}_0, \mathbf{e}_0 \rangle + \langle \mathbf{z}_1, \mathbf{e}_1 \rangle + g \bmod q. \end{aligned}$$

Consequently, adversary $\mathcal{A}$ receives a valid HNF Adaptive extended-LWE instance.

Adversary $\mathcal{B}$ runs $\mathcal{A}$ only once and has to compute the reordering which is feasible in time $\mathsf{poly}(\lambda)$. It then has advantage at least $\mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{A})/4$, completing the proof of the lemma. □

We now show that LWE reduces to Adaptive extended-LWE .

Theorem 7. *Let q be a prime, $\varepsilon > 0$ and $n, m, B, \gamma, \sigma \geqslant 0$. Assume that*

$$\sigma > \sqrt{2\ln(2(n+1)(1+1/\varepsilon))/\pi}$$

and $\gamma > \sigma\sqrt{2(1+nB^2)}$. *Then for any adversary* $\mathcal{A}$ *for* $\mathsf{AextLWE}_{q,n,m,\sigma,B}$ *running in time* T, *there exists an adversary* $\mathcal{B}$ *for* $\mathsf{LWE}_{q,n,m,\gamma,B}$ *running in time* $\mathsf{poly}(m, \log q) \cdot T$ *such that:*

$$\mathsf{Adv}^{\mathsf{AextLWE}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B}) + 2\frac{2\varepsilon}{1-\varepsilon} \ .$$

Proof. Let $\mathcal{A}$ be an adversary against $\mathsf{AextLWE}_{q,n,m,\sigma}$. We build an adversary $\mathcal{B}$ against $\mathsf{LWE}_{q,n,m,\gamma}$ as follows. Adversary $\mathcal{B}$ receives from its LWE challenger a tuple $(\mathbf{A}, \mathbf{b})$. It forwards $\mathbf{A}$ to $\mathcal{A}$ and receives a small hint vector $\mathbf{z}$ such that $\|\mathbf{z}\|_\infty \leqslant B$.

It then samples $[\mathbf{e'}^T \| g']^T \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n+1},\mathbf{\Sigma}}$, for some $\mathbf{\Sigma}$ defined later on. It sets $\mathbf{b'} = \mathbf{b} + \mathbf{e'}$ and $h = \langle \mathbf{z}, \mathbf{e'} \rangle + g'$ and sends $(\mathbf{b'}, h)$ to $\mathcal{A}$. Adversary $\mathcal{B}$ receives a final bit from $\mathcal{A}$ which it forwards to its challenger.

Assume we are in the $\beta = 0$ case of the LWE game. Then $\mathbf{b} = \mathbf{As} + \mathbf{e}$ and $\mathbf{b'} = \mathbf{As} + (\mathbf{e} + \mathbf{e'})$. The hint $h = \langle \mathbf{z}, \mathbf{e'} \rangle + g'$ can be rewritten as $h = \langle \mathbf{z}, (\mathbf{e} + \mathbf{e'}) \rangle - \langle \mathbf{z}, \mathbf{e} \rangle + g'$. Notice that if

$$\begin{pmatrix} \mathbf{e} + \mathbf{e'} \\ -\langle \mathbf{z}, \mathbf{e} \rangle + g' \end{pmatrix} = \begin{pmatrix} \mathbf{e} \\ -\mathbf{z}^T\mathbf{e} \end{pmatrix} + \begin{pmatrix} \mathbf{e'} \\ g' \end{pmatrix} \sim \mathcal{D}_{\mathbb{Z}^{n+1},\gamma} \tag{12}$$

this corresponds to the $\beta = 0$ case of the AextLWE game.

It thus suffices to set $\mathbf{\Sigma}$ accordingly. Notice that as $\mathbf{e} \sim \mathcal{D}_{\mathbb{Z}^n,\sigma}$, we have

$$\begin{pmatrix} \mathbf{e} \\ -\mathbf{z}^T\mathbf{e} \end{pmatrix} = \begin{pmatrix} \mathbf{Id} \\ -\mathbf{z}^T \end{pmatrix} \mathbf{e} \sim \mathcal{D}_{\mathbb{Z}^{n+1},\sigma^2\mathbf{F}\mathbf{F}^T}$$

for $\mathbf{F} = [\mathbf{Id} \| -\mathbf{z}]^T$. Let us then take $\mathbf{\Sigma} = \gamma^2\mathbf{Id} - \sigma^2\mathbf{F}\mathbf{F}^T$. Note that $s_1(\mathbf{F})^2 = 1 + \|z\|_2^2 \leqslant 1 + nB^2$. By assumption, we have $\gamma > \sqrt{2}\sigma s_1(\mathbf{F})$. By applying Lemma 11, we get

$$\begin{pmatrix} \mathbf{e} \\ -\mathbf{z}^T\mathbf{e} \end{pmatrix} + \begin{pmatrix} \mathbf{e'} \\ g' \end{pmatrix} \approx_\delta \mathcal{D}_{\mathbb{Z}^{n+1},\gamma} \tag{13}$$

for $\delta = 2\varepsilon/(1-\varepsilon)$.

In the $\beta = 1$ case of the LWE game, the vector $\mathbf{b}$ is uniform and so is $\mathbf{b'}$. The same analysis holds for the distribution of the hint h, so this case matches with the $\beta = 1$ case of the AextLWE game for $\mathcal{A}$. □ □

5.3 IND-CR-CPA UPKE FROM LWE

We now describe a UPKE scheme whose security is based on this newly introduced HNF-AextLWE assumption. As already shown, it is implied by the standard LWE assumption. Our scheme, detailed in Figure 7, avoid noise flooding by taking advantage

of the HNF-AextLWE assumption defined in Section 5.2. We then provide the first efficient UPKE scheme based on lattices. Our construction follows the lines of [63] which underlies Kyber [22].

In contrast, the only prior lattice-based construction, proposed in [39] and based on the Dual-Regev PKE from [47], is highly inefficient: (i) it supports only binary plaintexts, (ii) updates are done via bit-by-bit encryption of the private coins, and (iii) the security analysis relies on noise flooding, which requires a super-polynomial modulus.

Public parameters: $(n, q, p, \sigma, \sigma_c)$

KeyGen(1^λ):
$\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{n\times n})$, $\mathbf{s}, \mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$;
$pk = (\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$, $sk = \mathbf{s}$;
return (pk, sk).

Enc($pk, \boldsymbol{\mu} \in \mathbb{Z}_p^n$):
$\mathbf{X}, \mathbf{E} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma_c}$ and $\mathbf{f} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma_c}$;
return $\mathbf{ct} = (\mathbf{X}\mathbf{A} + \mathbf{E},\ \mathbf{X}\mathbf{b} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} \bmod q)$.

Dec($sk = \mathbf{s}, ct = (\mathbf{ct}_0, ct_1)$):
$\mathbf{v} = ct_1 - \mathbf{ct}_0\mathbf{s}$;
return $\lfloor p/q \cdot \mathbf{v} \rceil_p$.

UpdatePk($pk = (\mathbf{A}, \mathbf{b})$):
$\mathbf{r}, \boldsymbol{\eta} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$;
return $(pk' = (\mathbf{A}, \mathbf{b} + \mathbf{A}\mathbf{r} + \boldsymbol{\eta}),\ up = \mathsf{Enc}(pk, \mathbf{r}))$.

UpdateSk(sk, up):
return $sk' = sk + \mathsf{Dec}(sk, up)$.

Figure 7: LWE-based IND-CR-CPA UPKE construction.

Theorem 8. *Let* $\varepsilon, \delta \in (0,1), k > 0$. *Let* q, p *be primes and* $n, m > 0$ *and* $\sigma, \sigma_c > 0$ *such that*

$$\sigma \geqslant \sqrt{2\ln(2n(1+1/\varepsilon))/\pi} \text{ and } \sigma_c > 2\sigma\sqrt{1 + n((k+1)y\sigma)^2},$$

where $y = \sqrt{-2\log(\delta/(4n))}$.

Assuming the hardness of HNF-AextLWE*, the scheme presented in Figure 7 is* k-IND-CR-CPA *secure. More precisely, for any adversary $\mathcal{A}$ for the* k-IND-CR-CPA *game, there exists an adversary $\mathcal{B}$ for* HNF-AextLWE *running in similar time as $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UPKE}}(\mathcal{A}) \leqslant (2n+8)\cdot\frac{2\varepsilon}{1-\varepsilon} + (2n+1)\cdot\mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{B})\,.$$

Furthermore, assuming $q > 2p\sigma_c\cdot(2y^2\sigma nk + y)$ *and* $p > 2y\sigma$*, the scheme is* (k,δ)*-correct.*

Proof. For the sake readability, we delay the proof of correctness for after the security proof.

We show the IND-CR-CPA security of the scheme. Let us start by defining all the security games.

Game G_0: This is the original IND-CR-CPA game. Adversary $\mathcal{A}$ receives $pk_0 = (\mathbf{A}, \mathbf{b}_0 = \mathbf{As}+\mathbf{e})$ and queries the $\mathcal{O}_{up}(\cdot)$ oracle with randomness $(\mathbf{r}_1,\boldsymbol{\eta}_1),\dots,(\mathbf{r}_{chall},\boldsymbol{\eta}_{chall})$ until it asks for a challenge at epoch $chall$ for a pair of plaintexts $(\boldsymbol{\mu}_0,\boldsymbol{\mu}_1)$. At this epoch, the secret key is $sk_{chall} = \mathbf{s}+\Delta^{\mathbf{r}}_{chall}$ where $\Delta^{\mathbf{r}}_{chall} = \sum_{i=1}^{chall}\mathbf{r}_i$ and the public key is

$$pk_{chall} = \left(\mathbf{A},\ \mathbf{b}_{chall} = \mathbf{A}(\mathbf{s}+\Delta^{\mathbf{r}}_{chall}) + \mathbf{e} + \Delta^{\boldsymbol{\eta}}_{chall}\right),$$

with $\Delta^{\boldsymbol{\eta}}_{chall} = \sum_{i=1}^{chall}\boldsymbol{\eta}_i$. It receives a challenge

$$\mathbf{c}^* = (\mathbf{T}_{chall} = \mathbf{X}_{chall}\mathbf{A} + \mathbf{E}_{chall},\ \mathbf{pad}_{chall} = \mathbf{X}_{chall}\mathbf{b}_{chall} + \mathbf{f}_{chall} + \lfloor q/p\rfloor\cdot\boldsymbol{\mu}_\beta),$$

for $\beta\in\{0,1\}$ uniform.

Then the adversary queries the $\mathcal{O}_{up}(\cdot)$ oracle until the last epoch $last$. At this epoch, the secret key is $sk_{last} = \mathbf{s}+\Delta^{\mathbf{r}}_{last}$, where $\Delta^{\mathbf{r}}_{last} = \sum_{i=1}^{last}\mathbf{r}_i$ and the public key is $pk_{last} = (\mathbf{A}, \mathbf{b}_{last} = \mathbf{A}(\mathbf{s}+\Delta^{\mathbf{r}}_{last}) + \mathbf{e} + \Delta^{\boldsymbol{\eta}}_{last})$, where $\Delta^{\boldsymbol{\eta}}_{last} = \sum_{i=1}^{last}\boldsymbol{\eta}_i$. The challenger samples the final update $\mathbf{r}^*,\boldsymbol{\eta}^* \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ and sends

$$\begin{aligned} up^* &= \mathsf{Enc}(pk_{last},\mathbf{r}^*)\\ &= (\mathbf{T}_{last} = \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last},\\ &\qquad \mathbf{pad}_{last} = \mathbf{X}_{last}\mathbf{b}_{last} + \mathbf{f}_{last} + \lfloor q/p\rfloor\cdot\mathbf{r}^*) \end{aligned}$$

together with $pk^* = (\mathbf{A}, \mathbf{b}_{last} + \mathbf{Ar}^* + \boldsymbol{\eta}^*)$ and $sk^* = \mathbf{s}+\Delta^{\mathbf{r}}_{last}+\mathbf{r}^*$ to the adversary.

Game G_1: In this game we modify the update up^*. Instead of computing it as

$$up^* = (\mathbf{T}_{last} = \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last},\ \mathbf{pad}_{last} = \mathbf{X}_{last}\mathbf{b}_{last} + \mathbf{f}_{last} + \lfloor q/p\rfloor\cdot\mathbf{r}^*),$$

the challenger sets

$$up^* = (\mathbf{T}_{last} = \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last}, \quad \mathbf{pad}_{last} = \mathbf{X}_{last}\mathbf{b}_{last} + \mathbf{f}_{last} + \lfloor q/p \rfloor \cdot (-\mathbf{s})).$$

This modification results in a computationally equivalent game. Indeed adversary receives up^* together with $sk^* = \mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$ with $\Delta^{\mathbf{r}}_{last}$ known to the adversary. This modification is just a subtraction of $\lfloor q/p \rfloor \cdot (\mathbf{s} + \mathbf{r}^*)$ in $\mathbf{pad}_{last}$.

Game G_2**:** In this game, we again modify the update. This time the challenger computes the update up^* as

$$\begin{aligned} \mathbf{T}_{last} &= \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last} - \lfloor q/p \rfloor \cdot \mathbf{Id}, \\ \mathbf{pad}_{last} &= \mathbf{T}_{last}(\mathbf{s}+\Delta^{\mathbf{r}}_{last}) - \mathbf{E}_{last}(\mathbf{s} + \Delta^{\mathbf{r}}_{last}) + \mathbf{X}_{last}(\mathbf{e} + \Delta^{\boldsymbol{\eta}}_{last}) \\ &\quad + \mathbf{f}_{last} + \lfloor q/p \rfloor \cdot \Delta^{\mathbf{r}}_{last}. \end{aligned}$$

Notice that

$$\mathbf{pad}_{last} = \mathbf{X}_{last}\mathbf{b}_{last} + \mathbf{f}_{last} + \lfloor q/p \rfloor \cdot (-\mathbf{s}).$$

Therefore, the only difference with the previous game is that we subtract a publicly computable element $\lfloor q/p \rfloor \cdot \mathbf{Id}$ in $\mathbf{T}_{last}$, which implies that this game is computationally equivalent to the last one.

Game G_3**:** In this game, instead of computing $\mathbf{T}_{last}$ as

$$\mathbf{T}_{last} = \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last} - \lfloor q/p \rfloor \cdot \mathbf{Id}$$

the challenger sets $\mathbf{T}_{last}$ uniformly, i.e., $\mathbf{T}_{last} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n})$.

Lemma 14 below states that games G_2 and G_3 are computationally indistinguishable. The proof relies on the hardness of HNF-AextLWE. In particular, any adversary $\mathcal{B}$ has advantage at most $\mathsf{Adv}(\mathcal{B}) \leqslant n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}$ at distinguishing games G_2 and G_3.

Game G_4**:** Here, instead of having the challenger sample $\mathbf{s}, \mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ at the start of the game, and $\mathbf{r}^*, \boldsymbol{\eta}^* \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ at the end and setting $sk^* = \mathbf{s} + \mathbf{r}^* + \Delta^{\mathbf{r}}_{last}$ and $pk^* = (\mathbf{A}, \mathbf{A}(\mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*) + \mathbf{e} + \Delta^{\boldsymbol{\eta}}_{last} + \boldsymbol{\eta}^*)$, we do the following.

Let us define distributions $\mathcal{S}$, $\mathcal{S}_{\mathbf{t}}$ and $\mathcal{S}_{\tilde{\mathbf{e}}}$ as:

$$\mathcal{S} = \mathcal{D}_{\mathbb{Z}^n,\sigma\sqrt{2}}, \quad \mathcal{S}_{\mathbf{t}} = \mathcal{D}_{\mathbb{Z}^n,\frac{\sigma}{\sqrt{2}},\frac{\mathbf{t}}{2}}, \quad \text{and } \mathcal{S}_{\tilde{\mathbf{e}}} = \mathcal{D}_{\mathbb{Z}^n,\frac{\sigma}{\sqrt{2}},\frac{\tilde{\mathbf{e}}}{2}}.$$

Then, in game G_4, the challenger samples $\mathbf{t}, \tilde{\mathbf{e}} \hookleftarrow \mathcal{S}$ at the beginning of the game, then samples $\mathbf{s} \hookleftarrow \mathcal{S}_{\mathbf{t}}, \mathbf{e} \hookleftarrow \mathcal{S}_{\tilde{\mathbf{e}}}$ and finally sets $sk^* = \mathbf{t} + \Delta^{\mathbf{r}}_{last}$ and $pk^* = (\mathbf{A}, \mathbf{At} + \tilde{\mathbf{e}} + \mathbf{A}\Delta^{\mathbf{r}}_{last} + \Delta^{\boldsymbol{\eta}}_{last})$.

Let $\delta = 2\varepsilon/(1-\varepsilon)$. Lemma 10 shows that this change only induces a statistically negligible bias. Specifically, assuming that we have the inequality $\sigma \geqslant \sqrt{2\ln(2n(1+1/\varepsilon))/\pi}$,

vector $\mathbf{t}$ is within statistical distance at most δ from the distribution of $\mathbf{s}+\mathbf{r}^*$ in game G_3, and the marginal distribution of $\mathbf{s}$ in game G_4 with respect to the adversary's view is:

$$\begin{aligned}
\mathbb{P}[\mathbf{s}=\mathbf{x}] &= \sum_{\mathbf{y}\in\mathbb{Z}^n} \mathbb{P}[\mathbf{s}=\mathbf{x}|\mathbf{t}=\mathbf{y}]\,\mathbb{P}[\mathbf{t}=\mathbf{y}] \\
&= \sum_{\mathbf{y}\in\mathbb{Z}^n} \mathcal{D}_{\mathbb{Z}^n,\frac{\sigma}{\sqrt{2}}}\left(\mathbf{x}-\frac{\mathbf{y}}{2}\right)\mathcal{D}_{\mathbb{Z}^n,\sigma\sqrt{2}}(\mathbf{y}) \\
&= \sum_{\mathbf{y}\in\mathbb{Z}^n} \mathcal{D}_{\mathbb{Z}^n,\sigma\sqrt{2}}(2\mathbf{x}-\mathbf{y})\,\mathcal{D}_{\mathbb{Z}^n,\sigma\sqrt{2}}(\mathbf{y}) \\
&\approx_\delta \mathcal{D}_{\mathbb{Z}^n,2\sigma}(2\mathbf{x}) = \mathcal{D}_{\mathbb{Z}^n,\sigma}(\mathbf{x}).
\end{aligned}$$

The fourth equality comes from applying Lemma 10 for the convolution of two Gaussian distributions with the same standard deviation. The same argument applies for $\tilde{\mathbf{e}}$ and $\mathbf{e}$. Hence any adversary $\mathcal{B}$ has advantage at most $4\delta = 8\varepsilon/(1-\varepsilon)$ in distinguishing games G_3 and G_4.

Game G_5: In this game, we replace $\mathbf{b}_0$ and $up^* = (\mathbf{T}_{last}, \mathbf{pad}_{last})$ by uniform elements. Note that $\mathbf{T}_{last}$ is already uniform since game G_3. Hence, the challenger samples $\mathbf{b}_0, \mathbf{pad}_{last} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$, and sets $pk_0 = (\mathbf{A}, \mathbf{b}_0)$ at the start of the game, and returns $up^* = (\mathbf{T}_{last}, \mathbf{pad}_{last})$ as the last update message.

Lemma 15 below states that this game and the previous one are computationally indistinguishable under the LWE assumption.

Game G_6: This is the final game. Here, the challenger replaces the challenge $\mathbf{c}^*$ to make it uniform: it samples $\mathbf{T}_{chall} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{n\times n})$ and $\mathbf{pad}_{chall} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$, and then sets $\mathbf{c}^* = (\mathbf{T}_{chall}, \mathbf{pad}_{chall})$.

Remember that in game G_5, we have

$$\mathbf{c}^* = (\mathbf{X}_{chall}\mathbf{A} + \mathbf{E}_{chall}, \mathbf{X}_{chall}\mathbf{b}_{chall} + \mathbf{f}_{chall} + \lfloor q/p \rfloor \cdot \mu_\beta).$$

We can rewrite $\mathbf{c}^*$ in a matrix form as:

$$\mathbf{X}_{chall}\left(\mathbf{A}\|\mathbf{b}_{chall}\right) + \left(\mathbf{E}_{chall}\|\mathbf{f}_{chall}\right) + \lfloor q/p \rfloor \cdot \left(\mathbf{o}\|\mu_\beta\right) \tag{14}$$

with $\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{n\times n})$ and $\mathbf{b}_{chall} = \mathbf{b}_0 + \mathbf{A}\Delta^{\mathbf{r}}_{chall} + \Delta^{\eta}_{chall}$. Recall that we have $\mathbf{b}_0 \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$ since game G_5. The last column of Equation (14) is

$$(\mathbf{X}_{chall}\mathbf{b}_0 + \mathbf{f}_{chall}) + (\mathbf{X}_{chall}(\mathbf{A}\Delta^{\mathbf{r}}_{chall} + \Delta^{\eta}_{chall})) + \lfloor q/p \rfloor \cdot \mu_\beta.$$

and can be rewritten as

$$\begin{aligned}
&(\mathbf{X}_{chall}\mathbf{b}_0 + \mathbf{f}_{chall,0}) \\
&+ \mathbf{T}_{chall}\Delta^{\mathbf{r}}_{chall} - \mathbf{E}_{chall}\Delta^{\mathbf{r}}_{chall} + \mathbf{X}_{chall}\Delta^{\eta}_{chall} + \mathbf{f}_{chall,1} \\
&+ \lfloor q/p \rfloor \cdot \mu_\beta\ ,
\end{aligned}$$

where $\mathbf{f}_{chall} = \mathbf{f}_{chall,0} + \mathbf{f}_{chall,1}$ for $\mathbf{f}_{chall,0}, \mathbf{f}_{chall,1} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma_c/\sqrt{2}}$. Consider we are working with standard deviation $\sigma_c/\sqrt{2}$ instead of σ_c. The first term is a multiple-secret LWE sample that is independent of any adversarially chosen value. The second one can be computed from $\mathbf{T}_{chall}$. The next three can be viewed as an HNF-AextLWE hint on the secret $\mathbf{X}_{chall}$ and the error $\mathbf{E}_{chall}$ with hint vector $\mathbf{z}_0 = \Delta^{\eta}_{chall}$ and $\mathbf{z}_1 = \Delta^{r}_{chall}$, which are small vectors. The difference in standard deviation can be handled as in Lemma 15 by adding terms sampled from $\mathcal{D}_{\mathbb{Z}^n,\sigma_c/\sqrt{2}}$ to the matrices $\mathbf{X}_{chall}$ and $\mathbf{E}_{chall}$. Applying Lemma 10 proves this change to be statistically unnoticeable.

The above indicates that the modification between this game and game G_5 can be analyzed by using the multiple-secret variant of $\mathsf{HNF\text{-}AextLWE}_{q,n,n+1,\sigma_c/\sqrt{2},k\gamma\sigma}$ with n secrets and hint vector $\mathbf{z} = [(\Delta^{\eta}_{chall})^T \| (\Delta^{r}_{chall})^T]^T$. Consequently, any adversary $\mathcal{A}$ has advantage at most $n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}} + (2n+1) \cdot 2\varepsilon/(1-\varepsilon)$ in distinguishing between games G_5 and G_6.

Note that in game G_6, the adversary has no information on the challenge μ_β. Hence $\mathsf{Adv}^{G_6}(\mathcal{A}) = 0$. We obtain

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UPKE}}(\mathcal{A}) \leqslant (2n+8) \cdot \frac{2\varepsilon}{1-\varepsilon} + (2n+1) \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}.$$

This completes the proof, up to Lemmas 14 and 15 below. □

Lemma 14. *For any adversary $\mathcal{A}$ that distinguishes between games G_2 and G_3, there exists an efficient algorithm $\mathcal{B}$ for $\mathsf{HNF\text{-}AextLWE}_{q,n,n,\sigma_c,B}$ (for $B = (k+1)\gamma\sigma$), calling $\mathcal{A}$ once, such that*

$$\mathsf{Adv}^{\mathsf{dist}}_{G_2,G_3}(\mathcal{A}) \leqslant n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{B}).$$

Proof. This proof constructs an algorithm $\mathcal{B}$ for the multiple-secret variant of the HNF-AextLWE assumption with n secrets, using a distinguisher $\mathcal{A}$ for games G_2 and G_3.

Algorithm $\mathcal{B}$ receives a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ from the HNF-AextLWE challenger. Then it samples $\mathbf{s}, \mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$ and sets $pk_0 = (\mathbf{A}, \mathbf{b}_0 = \mathbf{A}\mathbf{s} + \mathbf{e})$, forwards pk_0 to $\mathcal{A}$ and acts as $\mathcal{A}$'s challenger until the last update phase where it has to send up^* and sk^* to $\mathcal{A}$. At this stage, algorithm $\mathcal{B}$ knows the sum of all the updates Δ^{r}_{last} and the sum of all the noises used for each updates Δ^{η}_{last} as $\mathcal{A}$ has finished querying the $\mathcal{O}_{up}$ oracle.

The HNF-AextLWE challenger expects small vectors $\mathbf{z}_0, \mathbf{z}_1$ for which to send a hint $\mathbf{h}$. Let $\mathbf{X}_{last} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma_c}$ be the secret matrix and $\mathbf{E}_{last} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma_c}$ be the error matrix sampled by the challenger in the multiple-secret variant of HNF-AextLWE. Algorithm $\mathcal{B}$ sets $\mathbf{z}_0 = \mathbf{e} + \Delta^{\eta}_{last}$ and $\mathbf{z}_1 = -(\mathbf{s} + \Delta^{r}_{last})$.

It then receives from the challenger a matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times n}$ and a hint

$$\mathbf{h} = \mathbf{X}_{last}\mathbf{z}_0 + \mathbf{E}_{last}\mathbf{z}_1 = (\mathbf{X}_{last} \| \mathbf{E}_{last})\mathbf{z} + \mathbf{f}_{last}\ ,$$

where $\mathbf{z} = \left(\mathbf{z}_0^T \| \mathbf{z}_1^T\right)^T$ and $\mathbf{f}_{last} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma_c}$. The matrix $\mathbf{B}$ is either uniform or of the form $\mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last}$.

Adversary $\mathcal{B}$ sets

$$\begin{aligned} up^* &= \left(\mathbf{T}_{last} = \mathbf{B} - \lfloor q/p \rfloor \cdot \mathbf{Id},\ \mathbf{T}_{last}(\mathbf{s} + \Delta^{\mathbf{r}}_{last}) + \mathbf{h} + \lfloor q/p \rfloor \Delta^{\mathbf{r}}_{last}\right) \\ &= \Big(\mathbf{T}_{last}, \mathbf{T}_{last}(\mathbf{s} + \Delta^{\mathbf{r}}_{last}) + \mathbf{X}_{last}(\mathbf{e} + \Delta^{\boldsymbol{\eta}}_{last}) - \mathbf{E}_{last}(\mathbf{s} + \Delta^{\mathbf{r}}_{last}) \\ &\qquad + \mathbf{f}_{last} + \lfloor q/p \rfloor \Delta^{\mathbf{r}}_{last}\Big). \end{aligned}$$

It also sets $pk^* = (\mathbf{A}, \mathbf{b}_0 + \mathbf{A}(\Delta^{\mathbf{r}}_{last} + \mathbf{r}^*) + \Delta^{\boldsymbol{\eta}}_{last} + \boldsymbol{\eta}^*)$ and $sk^* = \mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$, where $\mathbf{r}^*, \boldsymbol{\eta}^* \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$.

The case where $\mathbf{B}$ is uniform corresponds to adversary $\mathcal{A}$ playing game G_3 and the case where $\mathbf{B} = \mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last}$ corresponds to $\mathcal{A}$ playing game G_2. Hence $\mathcal{B}$ has the same advantage as $\mathcal{A}$.

By a hybrid argument, there exists an adversary $\mathcal{B}'$ for the game $\mathsf{HNF\text{-}AextLWE}_{q,n,n,\sigma,B}$ such that the advantage of $\mathcal{B}$ in the multiple-secret variant of HNF-AextLWE with n secrets can be bounded by $n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{B}')$, completing the proof. □

Lemma 15. *For any adversary $\mathcal{A}$ that distinguishes between games G_4 and G_5, there exists an adversary $\mathcal{B}$ for $\mathsf{LWE}_{q,n,2n,\sigma/2}$ calling $\mathcal{A}$ once, such that:*

$$\mathsf{Adv}^{\mathsf{dist}}_{G_4,G_5}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B}) + \frac{6\varepsilon}{1-\varepsilon}.$$

Proof. Let us build an adversary $\mathcal{B}$ for $\mathsf{LWE}_{q,n,2n,\sigma/2}$ that uses any distinguisher $\mathcal{A}$ between games G_4 and G_5.

Adversary $\mathcal{B}$ receives a uniform $\mathbf{B} \in \mathbb{Z}_q^{2n\times n}$ and a vector $\mathbf{c} \in \mathbb{Z}_q^{2n}$ from the LWE challenger. The vector $\mathbf{c}$ is either uniform or computed as an LWE sample with secret $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2}$. Now adversary $\mathcal{B}$ samples $\mathbf{E}_{last}, \mathbf{X}_{last} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma_c}$. It then computes

$$\mathbf{B}' = \mathbf{M}\mathbf{B} + \begin{pmatrix} \mathbf{0} \\ \mathbf{E}_{last} \end{pmatrix}, \text{with } \mathbf{M} = \begin{pmatrix} \mathbf{Id} & \mathbf{0} \\ \mathbf{X}_{last} & \mathbf{Id} \end{pmatrix} \in \mathbb{Z}_q^{2n\times 2n}$$

and parses $\mathbf{B}'$ as $\left(\mathbf{A}^\intercal \| \mathbf{T}_{last}^\intercal\right)^\intercal$. Let $\mathbf{t}, \tilde{\mathbf{e}} \hookleftarrow \mathcal{S} = \mathcal{D}_{\mathbb{Z}^n,\sigma\sqrt{2}}$. After that, it samples elements $\mathbf{s}' \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2,\mathbf{t}/2}$, $\boldsymbol{\eta} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2,\tilde{\mathbf{e}}/2}$ and $\mathbf{f}' \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,(\sigma_c^2-\sigma^2/4)\mathbf{Id}}$ that are used to adjust the standard deviations of the discrete Gaussian distributions involved in the proof. Then it sets $\mathbf{e}' = \left(\boldsymbol{\eta}^\intercal \| \mathbf{f}'^\intercal\right)^\intercal$ and $\mathbf{c}' = \mathbf{M}(\mathbf{c} + \mathbf{e}') + \mathbf{M}\mathbf{B}\mathbf{s}'$ and parses $\mathbf{c}'$ as $\left(\mathbf{b}_0^\intercal \| \mathbf{u}_1^\intercal\right)^\intercal$.

From there, adversary $\mathcal{B}$ runs as $\mathcal{A}$'s challenger and sets $pk_0 = (\mathbf{A}, \mathbf{b}_0)$. At epoch $last$, it computes

$$up^* = (\mathbf{T}_{last}, \mathbf{u}_1 + (\mathbf{T}_{last} - \mathbf{E}_{last} + \lfloor q/p \rfloor \cdot \mathbf{Id})\Delta^{\mathbf{r}}_{last}) + \mathbf{X}_{last}\Delta^{\boldsymbol{\eta}}_{last}.$$

If $\mathcal{A}$ returns G_4 then $\mathcal{B}$ guesses that $\mathbf{c}$ is an LWE sample and if $\mathcal{A}$ returns G_5 it guesses that it is uniform.

If $\mathbf{c}$ is uniform, as $\mathbf{M}$ is invertible, $\mathbf{B}'$ and $\mathbf{c}'$ are also uniformly distributed and adversary $\mathcal{A}$ is playing game G_5.

If $\mathbf{c} = \mathbf{Bs} + (\mathbf{e}^T \| \mathbf{f}^T)^T$, for $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2}$ and $\mathbf{e}, \mathbf{f} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2}$, then

$$\begin{aligned}\mathbf{c}' &= \mathbf{M}\left(\mathbf{Bs} + \begin{pmatrix}\mathbf{e}+\boldsymbol{\eta}\\ \mathbf{f}+\mathbf{f}'\end{pmatrix}\right) + \mathbf{MBs}' \\ &= \begin{pmatrix}\mathbf{A}\\ \mathbf{T}_{last} - \mathbf{E}_{last}\end{pmatrix}(\mathbf{s}+\mathbf{s}') + \begin{pmatrix}\mathbf{e}+\boldsymbol{\eta}\\ \mathbf{X}_{last}(\mathbf{e}+\boldsymbol{\eta}) + \mathbf{f} + \mathbf{f}'\end{pmatrix} = \begin{pmatrix}\mathbf{b}_0\\ \mathbf{u}_1\end{pmatrix}.\end{aligned}$$

Let us set $\bar{\mathbf{s}} = \mathbf{s}+\mathbf{s}'$, $\bar{\mathbf{e}} = \mathbf{e}+\boldsymbol{\eta}$, $\bar{\mathbf{f}} = \mathbf{f}+\mathbf{f}'$. Then, using the equation above, we have the following:

$$\begin{aligned}up^* &= (\mathbf{T}_{last}, \mathbf{u}_1 + (\mathbf{T}_{last} - \mathbf{E}_{last} + \lfloor q/p \rfloor \cdot \mathbf{Id})\Delta^{\mathbf{r}}_{last} + \mathbf{X}_{last}\Delta^{\boldsymbol{\eta}}_{last}) \\ &= (\mathbf{T}_{last}, (\mathbf{T}_{last} - \mathbf{E}_{last})\bar{\mathbf{s}} + \mathbf{X}_{last}(\bar{\mathbf{e}} + \Delta^{\boldsymbol{\eta}}_{last}) + \bar{\mathbf{f}} \\ &\quad + (\mathbf{T}_{last} - \mathbf{E}_{last} + \lfloor q/p \rfloor \cdot \mathbf{Id})\Delta^{\mathbf{r}}_{last}) \qquad (15) \\ &= (\mathbf{T}_{last}, (\mathbf{T}_{last} - \mathbf{E}_{last})(\bar{\mathbf{s}} + \Delta^{\mathbf{r}}_{last}) + \mathbf{X}_{last}(\bar{\mathbf{e}} + \Delta^{\boldsymbol{\eta}}_{last}) \\ &\quad + \bar{\mathbf{f}} + \lfloor q/p \rfloor \cdot \Delta^{\mathbf{r}}_{last}).\end{aligned}$$

Let $\delta = 2\varepsilon/(1-\varepsilon)$, for $\varepsilon \in (0,1)$. As $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2}$ and $\mathbf{s}' \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2,t/2}$, Lemma 10 gives that the distribution of $\bar{\mathbf{s}}$ has statistical distance at most δ from $\mathcal{D}_{\mathbb{Z}^n,\sigma/\sqrt{2},t/2}$. Similarly, errors $\boldsymbol{\eta}$ and $\mathbf{f}'$ were chosen such that $\bar{\mathbf{e}}$ and $\bar{\mathbf{f}}$ are within statistical distance at most δ from $\mathcal{D}_{\mathbb{Z}^n,\sigma/\sqrt{2},\tilde{\mathbf{e}}}$ and $\mathcal{D}_{\mathbb{Z}^n,\sigma_c}$. The equation above shows that up^* is statistically close (at distance at most 3δ) from its value in game G_4, thus $\mathcal{A}$ can be viewed as playing game G_4.

Overall, algorithm $\mathcal{B}$ has advantage at least $\mathrm{Adv}^{\mathrm{dist}}_{G_4,G_5}(\mathcal{A}) - 3\delta$, completing the proof. □

We now prove correctness of the scheme, with notations of Theorem 8.

Proof. Let $(pk = (\mathbf{A}, \mathbf{As}+\mathbf{e}), sk = \mathbf{s}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ be an honestly generated key pair. In order to consider the worst case scenario where k updates to the key have been performed, assume that $\mathbf{s}$ and $\mathbf{e}$ satisfy $\|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leqslant k\gamma\sigma$, for γ a parameter that we set afterwards.

Let $\boldsymbol{\mu} \in \mathbb{Z}_p^n$ and

$$\mathsf{Enc}(pk, \boldsymbol{\mu}) = (\mathbf{XA} + \mathbf{E},\ \mathbf{Xb} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} \bmod q).$$

Then, we have

$$\begin{aligned}\mathsf{Dec}(\mathbf{s}, ct) &= \lfloor ct_1 - \mathbf{ct}_0 \cdot \mathbf{s} \rceil_p \\ &= \lfloor \mathbf{Xb} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} - (\mathbf{XA} + \mathbf{E})\mathbf{s} \rceil_p \\ &= \lfloor \mathbf{Xe} - \mathbf{Es} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} \rceil_p.\end{aligned}$$

We obtain that $\mathsf{Dec}(\mathbf{s}, \mathsf{ct}) = \boldsymbol{\mu}$ if $\|\mathbf{Xe} - \mathbf{Es} + \mathbf{f}\|_\infty < q/(2p)$. By the triangular inequality, it suffices to have $\|\mathbf{Xe}\|_\infty + \|\mathbf{Es}\|_\infty + \|\mathbf{f}\|_\infty < q/(2p)$. By using that $\|\mathbf{Mv}\|_\infty \leqslant \|\mathbf{M}^\mathsf{T}\|_1 \|\mathbf{v}\|_\infty \leqslant \sqrt{n}\|\mathbf{M}^\mathsf{T}\|_2 \|\mathbf{v}\|_\infty$ for any matrix $\mathbf{M} \in \mathbb{Z}^{n\times n}$ and any vector $\mathbf{v} \in \mathbb{Z}^n$, we obtain another sufficient condition:

$$\sqrt{n}\|\mathbf{X}^\mathsf{T}\|_2 \|\mathbf{e}\|_\infty + \sqrt{n}\|\mathbf{E}^\mathsf{T}\|_2 \|\mathbf{s}\|_\infty + \|\mathbf{f}\|_\infty < q/(2p). \tag{16}$$

If we assume that $\|\mathbf{X}^\mathsf{T}\|_2, \|\mathbf{E}^\mathsf{T}\|_2 < y\sqrt{n}\sigma_c$ and $\|\mathbf{f}\|_\infty < y\sigma_c$, for some $y > 0$, then (16) is verified if

$$q > 2p \cdot (2y^2\sigma_c \sigma n k + y\sigma_c).$$

We bound the ℓ_2-norms using Lemma 9 and a union-bound, and the ℓ_∞-norms with Lemma 9 in dimension 1 and a union-bound. Using the independence of the random variables, the assumption we made on the norms are verified with probability at least

$$\begin{aligned}
&\mathbb{P}[\|\mathbf{X}^\mathsf{T}\|_2, \|\mathbf{E}^\mathsf{T}\|_2 < y\sqrt{n}\sigma_c \wedge \|\mathbf{f}\|_\infty < y\sigma_c] \\
&\qquad > \left(1 - ny^n e^{\frac{n}{2}(1-y^2)}\right)^2 \left(1 - 2ne^{-\frac{y^2}{2}}\right) \\
&\qquad > \left(1 - 2ny^n e^{\frac{n}{2}(1-y^2)}\right) \left(1 - 2ne^{-\frac{y^2}{2}}\right) \\
&\qquad > 1 - 4ne^{-\frac{y^2}{2}}.
\end{aligned}$$

In order to achieve (k, δ)-correctness, it suffices to set $y = \sqrt{-2\log(\delta/(4n))}$.

Notice that we implicitly assumed that the norm of the updates were bounded by $y\sigma$. As the plaintext space is $\mathbb{Z}_p^n$, in order to fit a secret key into an encryption, it suffices that $p > 2y\sigma$. □

We also prove γ-spreadness of our scheme. This is needed later on to apply our FO transform. We adapt the proof of [55, Lemma 6] for FrodoKEM to prove the following result.

Lemma 16. *Our UPKE construction is γ-spread.*

Proof. Let $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ be an element of the ciphertext space, $\boldsymbol{\mu}$ be a message and $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$ be a public key. We have:

$$\begin{aligned}
\mathbb{P}\left[\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \boldsymbol{\mu})\right] &\leqslant \mathbb{P}_{\mathbf{X},\mathbf{E}}[\mathsf{ct}_0 = \mathbf{X}\mathbf{A} + \mathbf{E}] \\
&= \sum \mathbb{P}_{\mathbf{X},\mathbf{E}}[\mathsf{ct}_0 = \tilde{\mathbf{X}}\mathbf{A} + \mathbf{E} \wedge \mathbf{X} = \tilde{\mathbf{X}}] \\
&= \sum \mathbb{P}_{\mathbf{E}}[\mathbf{E} = \tilde{\mathbf{X}}\mathbf{A} + \mathsf{ct}_0] \cdot \mathbb{P}\left[\mathbf{X} = \tilde{\mathbf{X}}\right] \\
&\leqslant \sum \mathbb{P}\left[\mathbf{E} = \mathbf{0}\right] \cdot \mathbb{P}\left[\mathbf{X} = \tilde{\mathbf{X}}\right] \\
&= \mathbb{P}\left[\mathbf{E} = \mathbf{0}\right] \\
&= \left(\mathcal{D}_{\mathbb{Z},\sigma}(0)\right)^{n^2}
\end{aligned}$$

where the fourth inequality stems from the fact that the distribution $\mathcal{D}_{\mathbb{Z},\sigma}(x)$ is maximal at $x = 0$.

□

5.4 A UPKE FUJISAKI-OKAMOTO TRANSFORM

In this section, we describe a transform from an IND-CR-CPA UPKE into an IND-CR-CCA UKEM following the Fujisaki-Okamoto [46] technique.

Definition 23 (FO-transform for UPKEs)**.** *Let* UPKE *be a UPKE, and* G *and* H *be two functions modeled as random oracles. We define the transform* FO(UPKE, G, H) *in Figure 8.*

Our FO transform is essentially the $\mathsf{KEM}^{\perp}$ construction from [53]. We add pk to the inputs of the hash function used to determinize the Enc algorithm in order to prevent trivial attacks, given the ability of the adversary to update the key pair.

Theorem 9 (FO transform for UPKEs)**.** *Let* $\gamma, \delta \in (0,1), k > 0$*. Let* UPKE $= (\mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *denote a* γ*-spread and* (k, δ)*-correct* k-IND-CR-CPA *UPKE scheme. Then the UPKE* FO(UPKE, G, H) *is a* (k, δ)*-correct* k-IND-CR-CCA *UKEM in the ROM.*

More precisely, for any adversary $\mathcal{A}$ *for the* k-IND-CR-CCA *UKEM game in the ROM making at most* q_G *queries to oracle* G, q_H *queries to oracle* H *and* q_D *queries to oracle* $\mathcal{O}_{\mathtt{dec}}$*, there exists an adversary* $\mathcal{B}$ *for the* k-IND-CR-CPA *game of* UPKE *with a similar running time such that:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\mathcal{A}) \leqslant q_G \cdot \delta + q_D \cdot \gamma + 2\left(\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{B}) + \frac{q_G + q_H}{|\mathcal{M}|}\right) .$$

The proof of the above theorem follows standard techniques for FO analysis (e.g., [53])

Proof. The (k, δ)-correctness of FO(UPKE, G, H) in the ROM follows from the (k, δ)-correctness of the underlying UPKE scheme, since Encaps runs the Enc algorithm, Decaps runs the Dec algorithm and the underlying KeyGen, UpdatePk algorithms are unchanged.

```
KeyGen = UPKE.KeyGen.

Encaps(pk):
  m ↩ U(M);
  c ← UPKE.Enc(pk, m; G(pk, m));
  K = H(m, c);
  return (c, K).

Decaps(sk, c):
  m' ← UPKE.Dec(sk, c);
  if c ≠ UPKE.Enc(pk, m'; G(pk, m'))
    return ⊥;
  return K' = H(m', c).

UpdatePk = UPKE.UpdatePk.

UpdateSk = UPKE.UpdateSk.
```

Figure 8: Transform FO(UPKE, G, H) for a UPKE using random oracles G, H.

In Algorithm 1, we present the random oracles and decapsulation oracles as they are in the original IND-CR-CCA UKEM game. The idea of the proof is the same as for usual proofs of FO: we modify oracles to allow the challenger to simulate the decapsulation oracle without knowledge of the secret key sk. The additional pk in the inputs of the oracle G allows the challenger to keep track of the ciphertexts known by the adversary for any public key pk, through epochs.

We add a subscript i to the oracle names to refer to the implementation of this oracle in Game i. For instance, oracle G_0 refers to the oracle G in Game 0. When the context is clear, we omit the subscript. We let $\mathcal{K}$ denote the key space and $\mathcal{R}$ the space of the randomness used by algorithm Enc.

Let us define the following sequence of games. Note that, in each game, the challenger initializes all relevant lists $\mathcal{L}_H, \mathcal{L}_G$, or $\mathcal{L}_E$ to $\emptyset$ at the start of the game.

- Game 0: This is the original IND-CR-CCA UKEM game, using oracles as they are described in Algorithm 1.
- Game 1: In this game, we modify both the random oracles and the decapsulation oracle. We replace the oracles of Algorithm 1 by those in Algorithm 2. The main difference is that oracle G on input (pk, m) keeps track of $(pk, m, \mathsf{Enc}(pk, m; r), r)$, where r is the output of $G(pk, m)$. This allows for oracles H and $\mathcal{O}_{dec}$ to know, for every epoch t, if they are queried on valid encapsulations for pk_t.

Algorithmus 1 : Oracles G, H and $\mathcal{O}_{dec}$ for Game 0.

```
1  G_0(pk, m):
2      if ∃r : (pk, m, r) ∈ L_G
3          return r
4      r ↩ U(R);
5      L_G = L_G ∪ {(pk, m, r)};
6      return r

7  H_0(m, c):
8      if ∃K : (m, c, K) ∈ L_H
9          return K
10     K ↩ U(K);
11     L_H = L_H ∪ {(m, c, K)};
12     return K

13 O_dec,0(c):
14     if c = c* ∧ pk_t = pk_chall
15         Abort
16     return Decaps(sk_t, c)
```

Algorithmus 2 : Oracles G, H and $\mathcal{O}_{dec}$ for Game 1. Here pk_t denotes the public key at the current epoch t.

```
1  G_1(pk, m):
2      if ∃r : (pk, m, r) ∈ L_G
3          return r
4      r ↩ U(R);
5      c = Enc(pk, m; r);
6      L_E = L_E ∪ {(pk, m, r, c)};
7      L_G = L_G ∪ {(pk, m, r)};
8      return r
9  H_1(m, c):
10     if ∃K : (m, c, K) ∈ L_H
11         return K
12     K ↩ U(K);
13     if ∃pk, r : (pk, m, c, r) ∈ L_E
14         L_D = L_D ∪ {(pk, m, c, K)};
15     L_H = L_H ∪ {(m, c, K)};
16     return K

17 O_dec,1(c):
18     if c = c* ∧ pk_t = pk_chall
19         abort
20     if ∃m, K : (pk_t, m, c, K) ∈ L_D
21         return K
22     if ∃m, r : (pk_t, m, c, r) ∈ L_E
23         K ↩ U(K);
24         L_H = L_H ∪ {(m, c, K)};
25         L_D = L_D ∪ {(pk_t, m, c, K)};
26         return K
27     return ⊥
```

- Game 2: In this game, the challenger additionally aborts if the adversary makes a query $G(pk, m^*)$ or $H(m^*, c)$ with m^* being the (uniformly random) message used to compute the challenge encapsulation c^*, where pk and c are arbitrary. As the adversary $\mathcal{A}$ cannot learn $H(m^*, c^*)$, no information about it is available to the adversary. Hence $\mathsf{Adv}^{G_2}(\mathcal{A}) = 0$, and Games 1 and 2 are indistinguishable up to the adversary making a query using m^*.

Let us now prove that the above games are indistinguishable in the adversary's view.

Indistinguishability of Games 0 and 1. Compared to G_0, oracle G_1 only performs additional bookkeeping operations. Hence there is no difference between G_0 and G_1 for the adversary. Oracle H_1 might behave differently than H_0 only if a decapsulation query is made to $\mathcal{O}_{dec}$ for a c such that $(pk_t, m, c, r) \in \mathcal{L}_E$ for some (m, r), where t is the current epoch. Consider the case where the adversary makes a query c to the decapsulation oracle $\mathcal{O}_{dec}$ at epoch t:

1. Assume that $\mathcal{O}_{dec,0}(c) = \perp$ and $\mathcal{O}_{dec,1}(c) \neq \perp$: then by the definition of $\mathcal{O}_{dec,1}$, this implies that there exists[1] $(pk_t, m, r, c) \in \mathcal{L}_E$ such that $c = \mathsf{Enc}(pk_t, m; r)$, where $r = G(pk_t, m)$. As we assumed $\mathcal{O}_{dec,0}(c) = \perp$, the original decapsulation function fails on c, hence r is such that we have $\mathsf{Dec}(sk_t, \mathsf{Enc}(pk_t, m; r)) \neq m$. By the (k, δ)-correctness, this happens with probability at most δ.

2. Assume that $\mathcal{O}_{dec,0}(c) \neq \perp$ and $\mathcal{O}_{dec,1}(c) = \perp$: by the definition of $\mathcal{O}_{dec,1}$, this implies that there is no $(pk_t, m, r, c) \in \mathcal{L}_E$, hence $\mathcal{A}$ did not make any query $G(pk_t, m)$ but was able to compute a valid ciphertext of m under pk_t. By γ-spreadness, this happens only with probability at most γ.

3. Assume that $\mathcal{O}_{dec,0}(c) = K$ and $\mathcal{O}_{dec,1}(c) = K'$ for some $K, K' \neq \perp$: by the definition of $\mathcal{O}_{dec,1}$ we know that there exists a tuple $(pk_t, m, r, c) \in \mathcal{L}_E$ such that $c = \mathsf{Enc}(pk_t, m; r)$, and as $\mathcal{O}_{dec,0}(c) \neq \perp$, this is a valid encryption. Hence $K = H_0(m, c)$. We consider the two following sub-cases:

   a) Adversary $\mathcal{A}$ first made the decryption query $\mathcal{O}_{dec}(c)$ without knowing $H(m, c)$. By definition of $\mathcal{O}_{dec,1}(c)$, the challenger samples $K' \hookleftarrow \mathcal{U}(\mathcal{K})$ and adds (m, c, K') to $\mathcal{L}_H$. By definition of H_1, we have $H_1(m, c) = K'$. Thus K' has the same distribution as K and H_1 has the same behaviour as H_0.

   b) Adversary $\mathcal{A}$ already knows $H(m, c)$ as it queried it before to the oracle H. It is then set to a uniformly random value $K' \hookleftarrow \mathcal{U}(\mathcal{K})$. Then, when the adversary makes the decryption query $\mathcal{O}_{dec}(c)$, the definition of $H_1(m, c)$ guarantees that $\mathcal{O}_{dec,1}(c)$ returns K', which has the same distribution as K and H_1 behaves identically to H_0.

1 Note that the only way $\mathcal{O}_{dec,1}(c)$ returns $K \neq \perp$ is that either $(pk_t, m, r, c) \in \mathcal{L}_E$ or that oracle H_1 added (pk_t, m, c, K) to $\mathcal{L}_D$. However, the latter only happens if $(pk_t, m, r, c) \in \mathcal{L}_E$. Thus $\mathcal{O}_{dec,1}(c)$ does not return $\perp$ only if $(pk_t, m, r, c) \in \mathcal{L}_E$.

We just showed that except with probability at most $q_G \cdot \delta + q_D \cdot \gamma$, Games 1 and 2 behave identically. Further note that in Game 1, for any epoch t, oracle queries to $\mathcal{O}_{dec}$ can be simulated without the knowledge of the secret key sk_t.

Indistinguishability of Games 1 and 2. Let us call FIND the event that an adversary $\mathcal{A}$ makes a query $G(pk, m^*)$ or $H(m^*, c)$ with m^* being the (uniformly random) message used to compute the challenge encapsulation c^*, where pk and c are arbitrary. As already detailed, adversary $\mathcal{A}$ has advantage at most $\mathbb{P}[\mathsf{FIND}]$ in distinguishing between Games 1 and 2. We now bound the probability $\mathbb{P}[\mathsf{FIND}]$ by constructing an adversary $\mathcal{B}$ for the IND-CR-CPA game such that

$$\mathbb{P}[\mathsf{FIND}] \leqslant 2\left(\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{B}) + \frac{q_G + q_H}{|\mathcal{M}|}\right) . \tag{17}$$

Adversary $\mathcal{B}$ first receives pk_0 from its IND-CR-CPA challenger and forwards pk_0 to $\mathcal{A}$. Whenever $\mathcal{A}$ makes an $\mathcal{O}_{up}$ oracle query, adversary $\mathcal{B}$ makes the same $\mathcal{O}_{up}$ query to its challenger. Whenever $\mathcal{A}$ makes a G, H or $\mathcal{O}_{dec}$ query, adversary $\mathcal{B}$ runs them as in Game 1, which is possible as it does not need to know the secret key, as observed above. When $\mathcal{A}$ requests a challenge, $\mathcal{B}$ samples two random messages $m_0, m_1 \hookleftarrow \mathcal{U}(\mathcal{M})$ and sends them to its challenger.

The challenger answers with the IND-CR-CPA challenge c^*. Adversary $\mathcal{B}$ samples $K^* \hookleftarrow \mathcal{U}(\mathcal{K})$ and sends the challenge (K^*, c^*) to $\mathcal{A}$.

From now, adversary $\mathcal{B}$ continues to simulate $\mathcal{A}$'s challenger. If $\mathcal{A}$ makes a query $G(pk, m_{b'})$ or $H(m_{b'}, c)$ for any $b' \in \{0,1\}$, adversary $\mathcal{B}$ stops running $\mathcal{A}$ and returns b' to its challenger. If $\mathcal{A}$ makes no such request, then $\mathcal{B}$ samples $b' \hookleftarrow \mathcal{U}(\{0,1\})$ and returns b'.

Call WRG the event that $\mathcal{A}$ makes an oracle query to G or H containing m_{1-b}, where b is the challenge bit. Since $\mathcal{A}$ has absolutely no information about m_{1-b}, this happens with probability at most $\mathbb{P}[\mathsf{WRG}] \leqslant (q_G + q_H)/|\mathcal{M}|$. Then:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{B}) &= \left|\mathbb{P}\left[b = b'\right] - \frac{1}{2}\right| \\
&= \left|\mathbb{P}[\mathsf{FIND} \wedge \neg\mathsf{WRG}] + \frac{1}{2}\mathbb{P}[\neg\mathsf{FIND}] - \frac{1}{2}\right| \\
&= \left|\mathbb{P}[\mathsf{FIND}] - \mathbb{P}[\mathsf{FIND} \wedge \mathsf{WRG}] + \frac{1}{2}\mathbb{P}[\neg\mathsf{FIND}] - \frac{1}{2}\right| \\
&\geqslant \frac{1}{2}\mathbb{P}[\mathsf{FIND}] - \mathbb{P}[\mathsf{FIND} \wedge \mathsf{WRG}] \\
&\geqslant \frac{1}{2}\mathbb{P}[\mathsf{FIND}] - \mathbb{P}[\mathsf{WRG}] .
\end{aligned}$$

The second equality holds as $\mathcal{B}$ finds b' if and only if $\mathcal{A}$ makes an oracle query containing m_b (i.e., both FIND and $\neg$WRG occur) or if no such query occurs, by guessing randomly. For the third equality, we use that for any two events A, B, we have

$\mathbb{P}[A \wedge B] = \mathbb{P}[A] - \mathbb{P}[A \wedge \neg B]$. Equation (17) then follows, which completes the proof of Theorem 9.

□

5.5 OBTAINING IND-CU-CCA SECURITY

In this section, we further boost security in order to get IND-CU-CCA-security. As in [49], we use a NIZK argument that two keys encrypt the same message in order to make a reduction from IND-CU-CCA to IND-CR-CCA. This technique allows to extract the randomness used by the adversary for the oracle queries to $\mathcal{O}_{up}(\cdot)$, to forward it to the update oracle of the IND-CR-CCA challenger.

Let us consider $\mathsf{UPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ to be a k-IND-CR-CPA UPKE, for some $k > 0$. Define $\mathsf{UKEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ as the k-IND-CR-CCA UKEM obtained by applying our FO transform from Section 5.4 to UPKE, using G, H modeled as random oracles. Let F be a third function, also modeled as a random oracle. We assume that UpdatePk proceeds in two parts (this is the case for all known constructions, including the one from Section 5.3): $\mathsf{UpdatePk}(pk) = (\mathsf{Enc}(pk, r), \mathsf{NewPk}(pk, r))$, i.e., a first part which encrypts the randomness of the update using the UKEM encryption algorithm, and a second one which returns the updated public key. Let us define the language

$$\begin{aligned}\mathcal{L}_{up}^{\mathsf{UKEM}} = \{&(pk_0, pk_1, pk', ct_0, ct_1) \mid \exists r_0, r_1, r, \\ &ct_0 = \mathsf{Enc}(pk_0, r; r_0) \wedge ct_1 = \mathsf{Enc}(pk_1, r; r_1) \\ &\wedge (pk', ct_0) = \mathsf{UpdatePk}(pk_0; r)\}.\end{aligned}$$

Let $\Pi = (\mathsf{Prove}^F, \mathsf{Verify}^F)$ a NIZK argument in the random oracle for $\mathcal{L}_{up}^{\mathsf{UKEM}}$. We construct an k-IND-CU-CCA UKEM as described in Figure 9.

Theorem 10. *Let* UPKE, UKEM, Π *be defined as above. Then, the construction* $\overline{\mathsf{UKEM}}$ *described in Figure 9 is an* k-IND-CU-CCA *UKEM. Specifically, for any adversary* $\mathcal{A}$ *against the* k-IND-CU-CCA *security of* $\overline{\mathsf{UKEM}}$*, there exist adversaries* $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ *with running times similar to* $\mathcal{A}$*'s such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CU\text{-}CCA}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}_{\mathsf{UKEM}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UPKE}}(\mathcal{C}) \\ + \mathsf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{D}) + \mathsf{Adv}^{\mathsf{sound}}_{\Pi}(\mathcal{E}) \ .\end{aligned}$$

The proof closely follows the one of IND-CU-CCA security of the construction from [49].

Proof. We proceed by a sequence of hybrid games.

Game 0: This is the original IND-CU-CCA game where the challenger's bit is set to $b = 0$.

Game 1: We replace the proof π^* in the final update $up^* = (ct_0^*, ct_1^*, \pi^*)$ by a simulated NIZK proof. As the adversary only sees this simulated proof at the very end of the game and cannot submit any additional update or decryption queries, the two games are indistinguishable thanks to the computational zero-knowledge property of the underlying proof system.

Game 2: We now change the plaintext underlying ct_1^* to an encryption of 0 rather than r. This change remains undetected thanks to the IND-CPA security of the underlying encryption scheme. As an important remark, note that IND-CPA security (which is implied by IND-CR-CCA security) suffices here as no information about sk_1 is provided to the adversary, since neither the decapsulation oracle nor the final secret contain information about sk_1.

Game 3: In this game, when the adversary makes an update query which passes VerifyUpdate, the challenger does the following. Let the tuple $((ct_0, ct_1, \pi), (pk_0', pk_1))$ denote such a query. Then, the challenger uses both secret keys sk_0 and sk_1 to decrypt ct_0 and ct_1 and verify that the underlying plaintexts are indeed equal and that the new public key pk_0' is computed honestly. It halts if it is not the case. Unless Game 3 aborts, the two games are identical. The computational soundness of Π guarantees that any PPT adversary cannot trigger an abort, except with negligible probability. Here, we insist that standard (computational) soundness suffices as the adversary does not receive any proof until it can no longer make queries.

Game 4: This final game is identical to the previous game except that the challenger's bit is 1. We show that these two games are indistinguishable under the IND-CR-CCA security of UKEM.

Assume there exists a PPT adversary $\mathcal{A}$ that can distinguish Game 3 and Game 4. We construct a PPT adversary $\mathcal{B}$ against the IND-CR-CCA security of UKEM as follows. Adversary $\mathcal{B}$ gets pk_0 from its challenger and samples an additional keypair $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$. It also implements a random oracle F by storing a table. It forwards (pk_0, pk_1) to $\mathcal{A}$ as the public key.

When $\mathcal{A}$ makes a decapsulation query, adversary $\mathcal{B}$ simply submits the same query to its decapsulation oracle and returns the result to $\mathcal{A}$. When $\mathcal{A}$ makes an update query $((ct_0, ct_1, \pi), (pk_0', pk_1))$, adversary $\mathcal{B}$ verifies the validity of π and if it passes verification, uses sk_1 to decrypt ct_1 in order to recover the randomness r used by $\mathcal{A}$ to generate its update. Adversary $\mathcal{B}$ can then submit r to its own update oracle to produce the same update.

When $\mathcal{A}$ asks for a challenge, so does $\mathcal{B}$, and the latter forwards its challenge encapsulation c^* to the former.

Finally, when $\mathcal{A}$ stops making updates, so does $\mathcal{B}$. Its challenger then replies by (pk^*, sk^*, up^*), where up^* is simply an encryption ct_0^* of the last (unknown) update under the last epoch public key pk_{last}^ℓ. It generates an encryption ct_1^* of 0 under pk_1, as well as a simulated proof π^* that $(pk_{last}^\ell, pk_1, pk^*, ct_0^*, ct_1^*)$ is a valid update. It

finally sends the tuple $((pk^*, pk_1), sk^*, ct_0^*, ct_1^*, \pi^*)$ to $\mathcal{A}$. When $\mathcal{A}$ halts with output b', so does $\mathcal{B}$.

This completes the proof of Theorem 10.

□

5.6 CONCRETE PARAMETERS

In this section, we give some concrete parameters for the scheme presented in Section 5.3, which can directly be transformed into an IND-CR-CCA UKEM by applying the FO transform from Section 5.4. We focus on the latter. We conjecture that security holds in the module setting and use the lattice-estimator SAGE module (commit fd4a460) from [3] to estimate the security of the given parameter sets. For our UPKE/UKEM, we consider the module variant of the scheme presented in Section 5.3, i.e., we define $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ and we consider the base ring to be $\mathcal{R}$ instead of $\mathbb{Z}$.

Note that, for $p > 0$ a prime, the message space of Enc for the module variant is $\mathcal{M} = \mathcal{R}_p^n$ which is of size p^{dn}. For optimization purposes, we drop the last $n-1$ rows of the whole ciphertext computed by Enc in our encapsulation mechanism, so that an encapsulation is just:

$$c = (\mathbf{x}^T\mathbf{A} + \mathbf{e}^T, \mathbf{x}\mathbf{b} + f + \lfloor q/p \rfloor m)$$

for $\mathbf{x}, \mathbf{e} \in \mathcal{R}_q^n$, $f \in \mathcal{R}_q$ and $m \in \mathcal{R}_p$. The message space is now $\mathcal{M} = \mathcal{R}_p$, of size p^d. This optimization is made possible by considering the UKEM, which only require a message space with at least λ bits of entropy, which is the case when setting $d = 256$. The whole message space $\mathcal{R}_p^n$ is only used to encrypt updates, as an update changes all components of the secret key.

Also, as done in [22], we replace Gaussian distributions by the centered binomial distributions B_η, which for $\eta > 0$, samples elements $(a_i, b_i)_{i \leqslant \eta} \hookleftarrow \mathcal{U}(\{0,1\}^2)$ and returns $\sum_{i=1}^{\eta}(a_i - b_i)$. Samples from B_η are contained in $[-\eta, \eta]$, and we choose the modulus q such that perfect correctness ($\delta = 0$) is guaranteed up to a bounded number of (possibly malicious) updates. We let k denote this bound, and provide parameters for $k \in \{2^5, 2^{10}, 2^{15}, 2^{20}\}$. We are assuming worst-case updates and then make q scale linearly with k. It could be tempting to make it scale with $\sqrt{k}$ as updates are symmetric and centered in 0 though we should not, as they are chosen by the attacker. Due to this requirement, our UPKE/UKEM suffers from a loss compared to Kyber, which can take q as small as 3329 and then have ciphertexts of size 0.8KB.

As we are working in the UPKE setting, we consider that the adversary gets a leakage $\mathbf{s} + \mathbf{r}$ on the initial secret key $\mathbf{s}$, which roughly halves the variance of the distribution of $\mathbf{s}$ in the adversary's view (as shown in the proof of Theorem 8). We use a script to compute the average variance left on $\mathbf{s}$ conditioned on the value of $\mathbf{s} + \mathbf{r}$. We obtain that for $\mathbf{s} \hookleftarrow B_{2\eta}^n$, we are left on average as if $\mathbf{s}$ was sampled from B_η^n. This is taken into account for the security estimates.

	λ	q	n	d	p	η	δ	k	$\lvert ct\rvert$	$\lvert up\rvert$
Estimate for [39]	120	$\approx 2^{85}$	11	256	21	10	0	2^5	33KB	360KB
This work	128	$\approx 2^{21}$	3	256	5	2	2^{-136}	2^5	1.8KB	5.4KB
	128	$\approx 2^{26}$	4	256	5	2	0	2^{10}	3.0KB	12KB
	116	$\approx 2^{31}$	2	512	5	2	0	2^{15}	5.8KB	12KB
	128	$\approx 2^{36}$	3	512	5	2	0	2^{20}	9.1KB	27KB

Table 6: Parameter sets for the module variant of our IND-CR-CCA UKEM.

Our parameters are given in Table 6. Note that as done in Kyber, in order to have fast multiplication using the Number Theoretic Transform in the ring, we take modulus $q = 1 \bmod 2d$. This is the first practical lattice-based construction of UPKE/UKEM, hence there are no equivalent constructions to compare our results to. We achieve similar efficiency as the IND-CR-CPA construction of [49], which is based on the DCR assumption achieves a ciphertext and update size of 1.5KB (for the CPA case only, although our FO transform applies to their scheme). Note that by increasing d, the matrices involved become smaller. Hence, a tradeoff can be made to reduce the sizes of the updates at the cost of increasing ciphertext size. For small number of updates, we also apply the bit-dropping technique from Kyber to improve parameters. This optimization drops parts of the least significant bits of the ciphertexts to reduce their size. We use the script provided at `https://github.com/pq-crystals/security-estimates` to estimate the correctness loss implied by using this technique.

IND-CU-CCA **instantiation**. In order to add security against chosen updates via our transform from Section 5.5, we can further add a computationally sound NIZK argument for $\mathcal{L}_{up}^{\mathsf{UKEM}}$ in the updates. In the module setting, the language $\mathcal{L}_{up}^{\mathsf{UKEM}}$ can be defined as:

$$
\begin{aligned}
\mathcal{L}_{up}^{\mathsf{UKEM}} = \{&(pk_0, pk_1, pk', ct_0, ct_1) \mid \\
&\exists \mathbf{X}_0, \mathbf{X}_1, \mathbf{E}_0, \mathbf{E}_1 \in \mathcal{R}^{n\times n}, \mathbf{f}_0, \mathbf{f}_1, \mathbf{r} \in \mathcal{R}^n, \\
&\quad ct_0 = (\mathbf{X}_0\mathbf{A} + \mathbf{E}_0, \mathbf{X}_0\mathbf{b} + \mathbf{f}_0 + \lfloor q/p \rfloor \cdot \mathbf{r}) \bmod q \\
&\wedge \|\mathbf{X}_0\|_2, \|\mathbf{E}_0\|_2, \|\mathbf{f}_0\|_2 < B_0 \\
&\wedge\ ct_1 = (\mathbf{X}_1\tilde{\mathbf{A}} + \mathbf{E}_1, \mathbf{X}_1\tilde{\mathbf{b}} + \mathbf{f}_1 + \lfloor q/p \rfloor \cdot \mathbf{r}) \bmod q \\
&\wedge \|\mathbf{X}_1\|_2, \|\mathbf{E}_1\|_2, \|\mathbf{f}_1\|_2 < B_0 \\
&\wedge \|\mathbf{b}' - (\mathbf{b} + \mathbf{A}\mathbf{r})\|_2 \leqslant B_1 \wedge \|\mathbf{r}\|_2 < B_1\ \}.
\end{aligned}
$$

where $\mathsf{pk}_0 = (\mathbf{A}, \mathbf{b})$, $\mathsf{pk}_1 = (\tilde{\mathbf{A}}, \tilde{\mathbf{b}})$, $\mathsf{pk}' = (\mathbf{A}, \mathbf{b}')$ and B_0, B_1 are bounds for correctness.

Proving membership in $\mathcal{L}_{up}^{\mathsf{UKEM}}$ then corresponds to proving 4 norm bounds for matrices, 4 norm bounds for vectors and $2n^2 + 2n$ linear equations over $\mathcal{R}_q$. This can be achieved by applying [64], which allows to prove exact norm bounds and linear relations using a commit-and-prove protocol. This only affects the size of the updates, since the ciphertext remains the same as in the IND-CR-CCA setting.

$\overline{\mathsf{KeyGen}}(1^\lambda)$:
  $(\mathrm{pk}_0, \mathrm{sk}_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
  $(\mathrm{pk}_1, \mathrm{sk}_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
  return $\overline{\mathrm{pk}} = (\mathrm{pk}_0, \mathrm{pk}_1)$, $\overline{\mathrm{sk}} = \mathrm{sk}_0$.

$\overline{\mathsf{Encaps}}(\overline{\mathrm{pk}})$:
  parse $\overline{\mathrm{pk}}$ as $(\mathrm{pk}_0, \mathrm{pk}_1)$;
  $(\mathrm{c}, \mathrm{K}) \leftarrow \mathsf{Encaps}(\mathrm{pk}_0)$;
  return (c, K).

$\overline{\mathsf{Decaps}}(\overline{\mathrm{sk}}, \mathrm{c}) = \mathsf{Decaps}(\overline{\mathrm{sk}}, \mathrm{c})$.

$\overline{\mathsf{UpdatePk}}(\overline{\mathrm{pk}})$:
  parse $\overline{\mathrm{pk}}$ as $(\mathrm{pk}_0, \mathrm{pk}_1)$;
  sample $\mathrm{r} \hookleftarrow \mathsf{R}$;
  $\mathrm{pk}_0' \leftarrow \mathsf{NewPk}(\mathrm{pk}_0, \mathrm{r})$;
  $\mathrm{ct}_0 \leftarrow \mathsf{Enc}(\mathrm{pk}_0, \mathrm{r})$;
  $\mathrm{ct}_1 \leftarrow \mathsf{Enc}(\mathrm{pk}_1, \mathrm{r})$;
  $\pi \leftarrow \mathsf{Prove}^{\mathsf{F}}(\mathrm{pk}_0, \mathrm{pk}_1, \mathrm{pk}_0', \mathrm{ct}_0, \mathrm{ct}_1, \mathrm{r})$;
  return $\overline{\mathrm{up}} = (\mathrm{ct}_0, \mathrm{ct}_1, \pi)$, $\overline{\mathrm{pk}}' = (\mathrm{pk}_0', \mathrm{pk}_1)$.

$\overline{\mathsf{VerifyUpdate}}(\overline{\mathrm{up}}, \overline{\mathrm{pk}}')$:
  parse $\overline{\mathrm{up}}$ as $(\mathrm{ct}_0, \mathrm{ct}_1, \pi)$ and $\overline{\mathrm{pk}}'$ as $(\mathrm{pk}_0', \mathrm{pk}_1)$;
  return $\mathsf{Verify}^{\mathsf{F}}((\mathrm{pk}_0, \mathrm{pk}_1, \mathrm{pk}_0', \mathrm{ct}_0, \mathrm{ct}_1), \pi)$;

$\overline{\mathsf{UpdateSk}}(\overline{\mathrm{up}}, \overline{\mathrm{pk}}')$:
  if $\mathsf{VerifyUpdate}(\overline{\mathrm{up}}, \overline{\mathrm{pk}}') = 0$
    return $\perp$;
  parse $\overline{\mathrm{up}}$ as $(\mathrm{ct}_0, \mathrm{ct}_1, \pi)$;
  run $\mathsf{UpdateSk}(\overline{\mathrm{sk}}, \mathrm{ct}_0)$.

Figure 9: Construction of a IND-CU-CCA UKEM.

6

CONCLUSION AND PERSPECTIVES

In this thesis we presented two efficient constructions of Updatable Public Key Encryption, whose security are based on well-established assumptions. Both follow the same structure as previous constructions by relying on homomorphic properties and leakage resilience and circular security of their underlying PKE. Even if the proofs of IND-CR-CPA security of our schemes - and all previous schemes - depend in a non black-box way on the underlying cryptographic assumptions, it is important to notice that all known constructions follow the same pattern: the PKE is used to encrypt the randomness of the UpdatePk algorithm by the updater and is sent to the owner of the key. Of course, this is a quite intuitive solution and we proved that in two instantiations, given an efficient PKE, it provides an efficient UPKE. We underline this to point out that this similarity leads to the possible existence of a generic transform that would allow to build a secure UPKE out of carefully chosen PKE. This would be the most favorable scenario, confirming that updatability is just a natural consequence of strong security properties and homomorphisms of the underlying PKE. It would of course also allow existing systems to upgrade their PKE to UPKE if needed, without worrying too much about security losses.

To analyze our post-quantum construction, we also introduced a new assumption: Adaptive Extended LWE. We believe this contribution to be of independent interest. As another instance of LWE with additional information type assumption, it provides a weaker integer variant of Hint MLWE of [58]. However, we believe that the most important point about our assumption resides in the introduction of this adaptiveness. While we show that here, adaptiveness does not introduce any major challenge in the reduction to LWE, it is in general non-trivial to reduce adaptive versions of assumptions to the non-adaptive ones. Our proof crucially relies on the presence of gaussian noise in the hint, and breaks down if one is to use extended LWE with exact hints. A previous version of the assumption did not include this additional gaussian noise g in the hint and did not have the restriction on the size of the adversarially chosen vector $\mathbf{z}$. It turned out to be insecure, as in the HNF variant, if $\mathbf{z}_0, \mathbf{z}_1$ can be arbitrary, one can take $\mathbf{z}_0 = \mathbf{a}_0$, $\mathbf{z}_1 = [1 \| 0 \cdots 0]^\mathsf{T}$ and $\mathbf{a}_0$ the first column of the matrix $\mathbf{A}$. This makes the hint equal to the first coefficient of the LWE sample in the case were it is of the form $\mathbf{As} + \mathbf{e}$, which allows to distinguish from such a sample and a uniform one. This attack can be lifted to the non-HNF version using our reduction. As a side note, we notice that the reduction of Hint MLWE of [58] also lifts to an adaptive version of Hint MLWE, that is defined analogously to AextLWE. Adaptive variants of LWE with hints assumptions capture scenarios where the adversary is allowed to make choices

after receiving parts of the public information. These variants extend the range of applicability of lattice assumptions and seem to be necessary in the context of UPKEs, if one wants to handle cross terms without employing flooding techniques.

To achieve IND-CU-CCA security efficiently, we provided two efficient transforms that, composed together, allow to reach our goal in the ROM. The first is an FO-transform for UPKEs, that is well known for PKEs, which with this setup allows to achieve CCA security. The second is a generic transform, using the Naor-Yung paradigm, that allows to achieve CU security. While having an FO transform is satisfactory performance wise, the CU transform requires an additional ciphertext together with a NIZK proof. In the lattice setting, this is probably too expensive for real-world use. Some optimization techniques can be applied (shared randomness for the two ciphertexts, approximate range proofs...) to make this more affordable, but it remains to see whether or not these suffice. Ideally, one would hope for an FO-like transform that would allow to achieve CU-CCA security directly. The main challenge is that updates need to be publically verifiable. In the group messaging setting, a way to relax this constraint would be to rather consider a designated verifier scenario, where updates are only verifiable by members of the group.

For now, the main application of UPKE has been in Continuous Group Key Agreement (CGKA), introduced by [31]. This protocol allows members of a group to agree on a shared key and update it. UPKEs were used by [6] to solve forward secrecy issues in the TreeKEM CGKA from [13]. TreeKEM has been selected by the IETF working group Message Layer Security (MLS) to be part of a standard for group messaging, published in July 2023 [78]. However UPKE was not implemented in the standard for two reasons. The first was that the primitive was not well studied and lacked of efficient constructions. The second is related to a technique used to ensure that newly added group members receive the real group structure. When adding group members, a malicious user could send a fake group topology, allowing to trick new members. If those users were to remove the malicious user from the group at some point, the malicious user could still retain ability to decrypt group messages as newly added user might not be aware that it still has access to some information in the group. To prevent that from happening, it is possible to rely on merkle-tree like hashing and signatures to prove that the group structure is wellformed and has not been tempered with. This is still possible in the UPKE setting but requires additional signatures for each UPKE update without the possibility to aggregate them, which makes the protocols much less efficient. Our work provides solutions to the first problem. If CR-CCA suffices, the construction from [57] can be used together with our FO transform. If CU-CCA is required, our DCR construction provides a first practical candidate. It remains, however, to handle the second problem in order to allow real world use of UPKE in group messaging.

BIBLIOGRAPHY

[1] Michel Abdalla and Mihir Bellare. "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques." In: *Advances in Cryptology - ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Lecture Notes in Computer Science. Springer, 2000.

[2] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. "Sampling Discrete Gaussians Efficiently and Obliviously." In: *ASIACRYPT*. 2013.

[3] M. R. Albrecht, R. Player, and S. Scott. *On the concrete hardness of Learning with Errors*. ePrint 2015/046. 2015.

[4] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. "The double ratchet: security notions, proofs, and modularization for the signal protocol." In: (2019), pp. 129–158.

[5] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. "Security Analysis and Improvements for the IETF MLS Standard for Group Messaging." In: Lecture Notes in Computer Science (2020).

[6] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. "Modular Design of Secure Group Messaging Protocols and the Security of MLS." In: (2021).

[7] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems." In: *CRYPTO*. 2009.

[8] Kyoichi Asano and Yohei Watanabe. "Updatable Public Key Encryption with Strong CCA Security: Security Analysis and Efficient Generic Construction." In: *IACR Cryptol. ePrint Arch.* (2023). URL: https://eprint.iacr.org/2023/976.

[9] N. Barić and B. Pfitzmann. "Collision-free accumulators and fail-stop signature schemes without trees." In: *Eurocrypt*. 1997.

[10] Mihir Bellare and Sara K. Miner. "A Forward-Secure Digital Signature Scheme." In: Lecture Notes in Computer Science (1999). Ed. by Michael J. Wiener.

[11] Mihir Bellare and Phillip Rogaway. "Entity Authentication and Key Distribution." In: Lecture Notes in Computer Science (1993). Ed. by Douglas R. Stinson.

[12] Mihir Bellare and Phillip Rogaway. "Provably secure session key distribution: the three party case." In: (1995). Ed. by Frank Thomson Leighton and Allan Borodin.

[13] Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. "TreeKEM: asynchronous decentralized key management for large dynamic groups a protocol proposal for Messaging Layer Security." PhD thesis. Inria Paris, 2018.

[14] Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. "A More Complete Analysis of the Signal Double Ratchet Algorithm." In: *Advances in Cryptology - CRYPTO 2022*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Lecture Notes in Computer Science. Springer, 2022.

[15] J. Black, P. Rogaway, and T. Shrimpton. "Encryption-Scheme Security in the Presence of Key-Dependent Messages." In: *SAC*. 2002.

[16] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. "Key Agreement Protocols and Their Security Analysis." In: Lecture Notes in Computer Science (1997). Ed. by Michael Darnell.

[17] Olivier Blazy, Ioana Boureanu, Pascal Lafourcade, Cristina Onete, and Léo Robert. "How fast do you heal? A taxonomy for post-compromise security in secure-channel establishment." In: (2023).

[18] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing." In: Lecture Notes in Computer Science (2001).

[19] Dan Boneh and David Mandell Freeman. "Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures." In: *PKC*. 2011.

[20] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. "Circular-Secure Encryption from Decision Diffie-Hellman." In: Lecture Notes in Computer Science (2008).

[21] Nikita Borisov, Ian Goldberg, and Eric A. Brewer. "Off-the-record communication, or, why not to use PGP." In: (2004).

[22] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM." In: (2018).

[23] Z. Brakerski and S. Goldwasser. "Circular and Leakage Resilient Public-Key Encryption Under Subgroup Indistinguishability (or: Quadratic Residuosity Strikes Back)." In: *Crypto*. 2010.

[24] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping." In: (2012).

[25] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors." In: (2013).

[26] 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*. Springer. 2021, pp. 404–430.

[27] Jan Camenisch and Victor Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms." In: Lecture Notes in Computer Science (2003). Ed. by Dan Boneh.

[28] Ran Canetti, Shai Halevi, and Jonathan Katz. "A Forward-Secure Public-Key Encryption Scheme." In: Lecture Notes in Computer Science (2003). Ed. by Eli Biham.

[29] Ran Canetti, Palak Jain, Marika Swanberg, and Mayank Varia. "Universally composable end-to-end secure messaging." In: *Annual International Cryptology Conference*. Springer. 2022, pp. 3–33.

[30] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. "A formal security analysis of the signal messaging protocol." In: *Journal of Cryptology* 33 (2020), pp. 1914–1983.

[31] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. "On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees." In: (2018).

[32] R. Cramer, I. Damgård, and B. Schoenmakers. "Proofs of partial knowledge and simplified design of witness hiding protocols." In: *Crypto*. 1994.

[33] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. "On the Closest Vector Problem with a Distance Guarantee." In: *CCC*. 2014.

[34] Ivan Damgård and Mads Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System." In: Lecture Notes in Computer Science (2001).

[35] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. "Bloom Filter Encryption and Applications to Efficient Forward-Secret 0-RTT Key Exchange." In: *Advances in Cryptology - EUROCRYPT 2018*. Lecture Notes in Computer Science. Springer, 2018.

[36] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography." In: *IEEE Trans. Inf. Theory* (1976).

[37] Samuel Dobson and Steven D Galbraith. "Post-Quantum Signal Key Agreement from SIDH." In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 422–450.

[38] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. "Efficient Public-Key Cryptography in the Presence of Key Leakage." In: Lecture Notes in Computer Science (2010).

[39] Yevgeniy Dodis, Harish Karthikeyan, and Daniel Wichs. "Updatable Public Key Encryption in the Standard Model." In: Lecture Notes in Computer Science (2021).

[40] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. "Efficient Identity-Based Encryption over NTRU Lattices." In: Lecture Notes in Computer Science (2014).

[41] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. "Faster Lattice-Based KEMs via a Generic Fujisaki-Okamoto Transform Using Prefix Hashing." In: *CCS*. 2021.

[42] Ksenia Ermoshina, Francesca Musiani, and Harry Halpin. "End-to-End Encrypted Messaging Protocols: An Overview." In: Lecture Notes in Computer Science (2016). Ed. by Franco Bagnoli, Anna Satsiou, Ioannis Stavrakakis, Paolo Nesi, Giovanna Pacini, Yanina Welp, Thanassis Tiropanis, and Dominic DiFranzo.

[43] S. Faust, M. Kohlweiss, G.-A. Marson, and D. Venturi. "On the non-malleability of the Fiat-Shamir transform." In: *Indocrypt*. Springer. 2012, pp. 60–79.

[44] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: Lecture Notes in Computer Science (1986).

[45] P.-A. Fouque and D. Pointcheval. "Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks." In: *Asiacrypt*. 2001.

[46] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations." In: Lecture Notes in Computer Science (1997).

[47] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." In: (2008).

[48] S. Goldwasser, S. Micali, and C. Rackoff. "The knowledge complexity of interactive proof systems." In: *SIAM Journal on Computing* (1989).

[49] Calvin Abou Haidar, Benoît Libert, and Alain Passelègue. "Updatable Public Key Encryption from DCR: Efficient Constructions With Stronger Security." In: (2022).

[50] Calvin Abou Haidar, Alain Passelègue, and Damien Stehlé. "Efficient Updatable Public-Key Encryption from Lattices." In: (2023).

[51] 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: Lecture Notes in Computer Science (2021). Ed. by Juan A. Garay.

[52] D. Hofheinz. "Circular Chosen-Ciphertext Security with Compact Ciphertexts." In: *Eurocrypt*. 2013.

[53] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation." In: *TCC*. 2017.

[54] Jeremy Horwitz and Ben Lynn. "Toward Hierarchical Identity-Based Encryption." In: Lecture Notes in Computer Science (2002). Ed. by Lars R. Knudsen.

[55] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. "Failing Gracefully: Decryption Failures and the Fujisaki-Okamoto Transform." In: *ASIACRYPT*. 2022.

[56] D. Jost, U. Maurer, and M. Mularczyk. "Efficient ratcheting: Almost-optimal guarantees for secure messaging." In: *Eurocrypt*. 2019.

[57] Daniel Jost, Ueli Maurer, and Marta Mularczyk. "Efficient Ratcheting: Almost-Optimal Guarantees for Secure Messaging." In: Lecture Notes in Computer Science (2019).

[58] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. "Toward Practical Lattice-Based Proof of Knowledge from Hint-MLWE." In: Lecture Notes in Computer Science (2023).

[59] F. Kitagawa, T. Matsuda, and K. Tanaka. "Simple and efficient KDM-CCA secure public key encryption." In: *Asiacrypt*. 2019.

[60] Ehren Kret and Rolfe Schmidt. "The PQXDH Key Agreement Protocol." In: (). URL: https://signal.org/docs/specifications/pqxdh/pqxdh.pdf.

[61] Caroline Kudla and Kenneth G. Paterson. "Modular Security Proofs for Key Agreement Protocols." In: Lecture Notes in Computer Science (2005). Ed. by Bimal K. Roy.

[62] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices." In: *Des. Codes Cryptogr.* (2015).

[63] V. Lyubashevsky, A. Palacio, and G. Segev. "Public-Key Cryptographic Primitives Provably as Secure as Subset Sum." In: *TCC*. 2010.

[64] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General." In: Lecture Notes in Computer Science (2022).

[65] T. Malkin, I. Teranishi, and M. Yung. "Efficient circuit-size independent public key encryption with KDM security." In: *Eurocrypt*. 2011.

[66] Tal Malkin, Isamu Teranishi, and Moti Yung. "Efficient Circuit-Size Independent Public Key Encryption with KDM Security." In: Lecture Notes in Computer Science (2011).

[67] Moxie Marlinspike and Trevor Perrin. "The x3dh key agreement protocol." In: *Open Whisper Systems* 283 (2016), p. 10.

[68] Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. "Efficient Lattice-Based Inner-Product Functional Encryption." In: *PKC*. 2022.

[69] G. Miller. "Riemann's hypothesis and tests for primality." In: *Journal of computer and system sciences* 13.3 (1976), pp. 300–317.

[70] Moni Naor and Moti Yung. "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks." In: (1990). Ed. by Harriet Ortiz.

[71] Adam O'Neill, Chris Peikert, and Brent Waters. "Bi-Deniable Public-Key Encryption." In: Lecture Notes in Computer Science (2011).

[72] P. Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes." In: *Eurocrypt*. 1999.

[73] DongGook Park, Colin Boyd, and Sang-Jae Moon. "Forward Secrecy and Its Application to Future Mobile Communications Security." In: Lecture Notes in Computer Science (2000). Ed. by Hideki Imai and Yuliang Zheng.

[74] D. Pointcheval and J. Stern. "Security Proofs for Signature Schemes." In: *Eurocrypt*. 1996.

[75] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography." In: (2005). Ed. by Harold N. Gabow and Ronald Fagin.

[76] A. Sahai. "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security." In: *FOCS*. 1999.

[77] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM J. Comput.* (1997).

[78] MLS group. *RFC 9420 The Messaging Layer Security (MLS) Protocol*. 2023. URL: https://www.rfc-editor.org/rfc/rfc9420.html.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of January 18, 2024 (`classicthesis` version 4.2).