Efficient Updatable Public-Key Encryption from Lattices

Calvin Abou Haidar[1,2], Alain Passelègue[1,2,3], and Damien Stehlé[1,3]

[1] ENS de Lyon
calvin.abou-haidar@ens-lyon.fr
[2] INRIA
alain.passelegue@cryptolab.co.kr
[3] CryptoLab Inc., Lyon, France
damien.stehle@cryptolab.co.kr

Abstract. Updatable public key encryption has recently been introduced as a solution to achieve forward-security in the context of secure group messaging without hurting efficiency, but so far, no efficient lattice-based instantiation of this primitive is known.

In this work, we construct the first LWE-based UPKE scheme with polynomial modulus-to-noise rate, which is CPA-secure in the standard model. At the core of our security analysis is a generalized reduction from the standard LWE problem to (a stronger version of) the Extended LWE problem. We further extend our construction to achieve stronger security notions by proposing two generic transforms. Our first transform allows to obtain CCA security in the random oracle model and adapts the Fujisaki-Okamoto transform to the UPKE setting. Our second transform allows to achieve security against malicious updates by adding a NIZK argument in the update mechanism. In the process, we also introduce the notion of Updatable Key Encapsulation Mechanism (UKEM), as the updatable variant of KEMs. Overall, we obtain a CCA-secure UKEM in the random oracle model whose ciphertext sizes are of the same order of magnitude as that of CRYSTALS-Kyber.

1 Introduction

Secure group messaging aims to allow secure, long-lasting, communication for a large group of users. The larger the group and the longer the communication, the likelier one of the group member gets compromised. When the latter happens, ideally, one would like to guarantee that messages sent before the attack occurred remain hidden to the attacker. This corresponds to the notion of forward security [5,9,30,35,19,18,13] and can be achieved by relying on forward-secure public key encryption (FS-PKE), but it vastly hurts efficiency compared to relying on standard PKE. FS-PKE generates an initial key pair (pk_0, sk_0) which allows to derive a chain of key pairs $(pk_1, sk_1), (pk_2, sk_2), \ldots$ where each pk_{t+1} can be derived publicly from pk_t (and sk_{t+1} from sk_t). Hence, the first epoch key pair of an FS-PKE scheme implicitly defines all the subsequent epoch key pairs.

Forward security further requires that it should be hard to go back in the secret key chain, as compromising sk_j should not hurt the confidentiality of messages encrypted under pk_t for $t < j$. Therefore, FS-PKE can be seen as a simple form of hierarchical identity-based encryption (HIBE) [28,18], and tight connections between the notions have been observed [26]. As of today, FS-PKE schemes have similar performances as HIBE constructions, and therefore relying on FS-PKE for building secure group messaging seems inherently inefficient. The extreme alternative is to rely on standard PKE and to require every user to refresh their key pair on a regular basis. This assumes users to be active and online, which is an undesirable assumption in the context of group messaging. Moreover, a user refreshing its own key only guarantees confidentiality of messages it receives (and therefore messages sent by other users) but does not provide any guarantee about messages it sent. For the latter, users have to rely on the willingness of receivers to update their keys.

Updatable public-key encryption (UPKE), recently introduced in [31,4], offers a compromise between the above two approaches by relaxing the update mechanism of FS-PKE: in a UPKE scheme, any user can update any other user's key pair by running an update algorithm with (high-entropy) private coins. As a consequence, a key pair does not need to contain any information about the next epoch key pair as this information can be provided by the external user who proceeds in the update. This change allows to hope for UPKE constructions with similar efficiency as standard PKE, but a sender can now protect the messages it sent by updating the receiver's key.

To be formal, a UPKE scheme consists in a standard PKE scheme (KeyGen, $\mathsf{Enc}, \mathsf{Dec}$) augmented with two additional algorithms ($\mathsf{UpdatePk}, \mathsf{UpdateSk}$). The $\mathsf{UpdatePk}$ algorithm can be run by any user on inputs a target public key pk_t^U of a user U used at epoch t and fresh private coins r. It produces a public key pk_{t+1}^U for user U for epoch $t+1$ as well as an update ciphertext up (encrypted under pk_t^U). The $\mathsf{UpdateSk}$ algorithm then allows user U, given an update ciphertext up and its secret key sk_t^U to update the latter to obtain secret key sk_{t+1}^U corresponding to pk_{t+1}^U. Security of UPKE guarantees that ciphertexts encrypted under U's public key pk_t^U at any epoch t remain secret to an attacker which compromises sk_j^U for $j > t$, as long as any of the updates which occurred between epoch t and epoch j was performed by an honest user (i.e., using private coins unknown to the attacker). This is formalized by the notion of IND-CR-CPA security, in which the adversary can impose updates of the target user's public key with Chosen Randomness (CR) (i.e., providing the private coins used by the update mechanism to the challenger). This has been the main security notion studied so far [4,31,22]. For practical applications, stronger notions are desirable, and were introduced in [22]: first, the adversary could have access to a decryption oracle (using the secret key of the current epoch), which corresponds to CCA security. Second, the adversary could generate malicious updates. The latter notion corresponds to IND-CU-CPA/CCA security, where the adversary provides Chosen Updates (CU) by providing (possibly malicious) updates to the challenger rather than providing private coins (used to honestly generate updates in the chosen randomness setting).

UPKE has been constructed from various assumptions over the past years. An efficient IND-CR-CPA construction based on the Computational Diffie-Hellman (CDH) assumption and in the random oracle model (ROM) was proposed in [31,4]. Constructions in the standard model were first proposed in [22] from the Learning with Errors (LWE) assumption and from the Decisional Diffie-Hellman (DDH) assumption, but the latter two constructions are mainly of theoretical interest as they rely on bit-by-bit encryption, and circular-secure and leakage-resilient PKE. The LWE-based construction notably relies on super-polynomial modulus-to-noise rate due to the use of the noise flooding technique. Generic transforms from IND-CR-CPA security to IND-CU-CCA security are described in [22] but rely on heavy tools, namely one-time, strong, true-simulation f-extractable Non-Interactive Zero-Knowledge (NIZK) arguments [21]. In [1], an efficient construction based on the Decisional Composite Residuosity (DCR) assumption was proposed. The authors show that a variant of the ElGamal Paillier encryption scheme [17] can be turned into a (standard model) IND-CR-CPA UPKE scheme, and achieve IND-CR-CCA and IND-CU-CCA UPKE by further adding NIZK proofs using the Naor-Yung paradigm [38]. Concrete instantiations of the latter NIZKs are proposed in the random oracle model, resulting in the first efficient IND-CR-CCA and IND-CU-CCA instantiations from the DCR assumption and the strong RSA assumption [8], in the ROM.

1.1 Contributions

We provide the first efficient UPKE instantiation based on the LWE assumption with polynomial modulus-to-noise rate.

First, we construct a UPKE encryption scheme which follows the lines of the PKE scheme from [34], which underlies CRYSTALS-Kyber [11]. The main technicalities lie in its security analysis: (i) we prove our construction to achieve IND-CR-CPA security in the standard model, based on a new assumption which generalizes the extended-LWE assumption defined in [39], and (ii) we show that the latter assumption reduces to the standard LWE assumption by extending the results from [16,14].

Second, we provide two simple generic transforms which allow to convert any IND-CR-CPA UPKE construction into IND-CR-CCA and IND-CU-CCA UPKE schemes in the ROM. As we aim for practical constructions, we focus on constructing updatable key encapsulation mechanism (UKEM), which we introduce as the updatable variant of KEM. Our first transformation is an adaption of the Fujisaki-Okamoto transform [24] to the context of UPKE and allows to generically transform an IND-CR-CPA UPKE into an IND-CR-CCA UKEM with a minimal cost, in the ROM. The second transformation relies on the existence of a NIZK argument for a specific language. As an important remark, the underlying NIZK only plays a role in the update mechanism and should only satisfy basic properties (namely, a single-theorem NIZK with computational soundness and computational zero-knowledge is sufficient) while prior constructions [22,1] relied on strong NIZK notions (e.g., statistical-simulation-sound NIZKs for instantiating Naor-Yung). The latter NIZK argument can be efficiently instantiated from [33].

1.2 Technical Overview

We now present our contributions in more details, starting with our IND-CR-CPA UPKE construction.

IND-CR-CPA UPKE from lattices. Our IND-CR-CPA construction follows the LWE-variant of [34]: a public key is an LWE instance $(\mathbf{A}, \mathbf{b})$ with $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ for $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ and $\mathbf{s}, \mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$, the LWE secret $\mathbf{s}$ being the corresponding secret key. An encryption of a message $\boldsymbol{\mu} \in \mathbb{Z}_p^n$ is a pair $(\mathsf{ct}_0, \mathsf{ct}_1)$ of the form $(\mathbf{XA} + \mathbf{E}, \mathbf{Xb} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} \bmod q)$ with $\mathbf{X}, \mathbf{E} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma}, \mathbf{f} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$. Such a ciphertext can be decrypted by rounding $\mathsf{ct}_1 - \mathsf{ct}_0\mathbf{s}$ since:

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

where the term $\mathbf{Xe} + \mathbf{f} - \mathbf{Es}$ is small. Updating a key pair is done by sampling small vectors $\mathbf{r}, \boldsymbol{\eta} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$. The public key is updated to $(\mathbf{A}, \mathbf{b} + \mathbf{Ar} + \boldsymbol{\eta}) = (\mathbf{A}, \mathbf{A}(\mathbf{s} + \mathbf{r}) + \mathbf{e} + \boldsymbol{\eta})$. The corresponding update ciphertext *up* is an encryption of $\mathbf{r}$ (which fits in the plaintext space) under the original public key $(\mathbf{A}, \mathbf{b})$. The updated secret key is then $\mathbf{s} + \mathbf{r}$. Correctness follows from the correctness of the PKE scheme. We emphasize that the secret key and noise term might have increased in norm, which can hurt correctness of decryption. We provide more details about how we handle this issue later, when we mention concrete instantiations.

Let us now focus on the security analysis. An IND-CR-CPA attacker first sees a public key $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ and can make a first sequence of updates with private coins of its choice $(\mathbf{r}_1, \boldsymbol{\eta}_1), \ldots, (\mathbf{r}_{chall}, \boldsymbol{\eta}_{chall})$ before asking for a challenge ciphertext for a pair of plaintexts $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1)$ at epoch *chall*. The challenge ciphertext is then encrypted under public key $pk_{chall} = (\mathbf{A}, \mathbf{b}+\mathbf{A}\Delta^{\mathbf{r}}_{chall}+\Delta^{\boldsymbol{\eta}}_{chall})$, where $\Delta^{\mathbf{r}}_{chall} = \sum_{i=1}^{chall} \mathbf{r}_i$ and $\Delta^{\boldsymbol{\eta}}_{chall} = \sum_{i=1}^{chall} \boldsymbol{\eta}_i$. It can then ask for an additional sequence of updates $(\mathbf{r}_{chall+1}, \boldsymbol{\eta}_{chall+1}), \ldots, (\mathbf{r}_{last}, \boldsymbol{\eta}_{last})$ until it decides to compromise the secret key. When the latter happens, an honest update is performed by the challenger using randomness $\mathbf{r}^*, \boldsymbol{\eta}^*$. Let $\Delta^{\mathbf{r}}_{last}$ and $\Delta^{\boldsymbol{\eta}}_{last}$ denote respectively $\sum_{i=1}^{last} \mathbf{r}_i$ and $\sum_{i=1}^{last} \boldsymbol{\eta}_i$. Then, the adversary's goal is to guess which plaintext was encrypted, given the compromised secret key $\mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$ and the last update ciphertext which encrypts $\mathbf{r}^*$ under public key $(\mathbf{A}, \mathbf{b} + \mathbf{A}\Delta^{\mathbf{r}}_{last} + \Delta^{\boldsymbol{\eta}}_{last})$.

The prior lattice-based construction from [22] has a similar structure (though it is based on the Dual-Regev PKE scheme [25]) and the authors argue about security by using the following observation, which we adapt to our construction for the exposition. First, notice that the final update ciphertext, which encrypts $\mathbf{r}^*$, can be transformed into an encryption of $-\mathbf{s}$ as we are given $\mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$ and $\Delta^{\mathbf{r}}_{last}$ is known. It then suffices to argue that the scheme is circular-secure, given the compromised secret key (which is additional leakage about $\mathbf{s}$). To do so, observe that, for a ciphertext $(\mathsf{ct}_0, \mathsf{ct}_1) = (\mathbf{XA} + \mathbf{E}, \mathbf{Xb} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu})$, the second term can be re-written as $(\mathbf{XA} + \mathbf{E})\mathbf{s} + \mathbf{Xe} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} - \mathbf{Es}$, where $\mathbf{XA} + \mathbf{E}$ is the first part ct_0 of the ciphertext. That is, we have:

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

Therefore, assuming $\mathbf{f}$ is much larger than $\mathbf{Xe} - \mathbf{Es}$, the ciphertext distribution is statistically close to $(\mathsf{ct}_0, \mathsf{ct}_0\mathbf{s}+\mathbf{f}+\lfloor q/p \rfloor \cdot \boldsymbol{\mu})$. Under the LWE assumption, ct_0 is pseudorandom, and then any (linear) information about the secret $\mathbf{s}$ contained in $\boldsymbol{\mu}$ can be absorbed by the term $\mathsf{ct}_0\mathbf{s}$. The Leftover Hash Lemma allows to complete the security analysis by proving that the latter term is statistically close to uniform, as long as $\mathbf{s}$ retains enough min-entropy (in this case, conditioned on the leaked key $\mathbf{s} + \Delta^{\mathbf{r}}_{last} + \mathbf{r}^*$). Hence, using noise-flooding and assuming LWE, the scheme is proven secure. The proof additionally relies on the (key)-homomorphism of Dual-Regev to incorporate updates required by the adversary in the challenge/update ciphertext and keys.

Our analysis deviates from the above and avoids the noise-flooding step. Instead, we directly prove pseudorandomness of the above $\mathbf{XA} + \mathbf{E}$ term. It seems that the LWE assumption for the instance $(\mathbf{A}, \mathbf{XA} + \mathbf{E})$ would suffice, but the issue is that the second term $\mathsf{ct}_1 = (\mathbf{XA} + \mathbf{E})\mathbf{s} + \mathbf{Xe} + \mathbf{f} + \lfloor q/p \rfloor \cdot \boldsymbol{\mu} - \mathbf{Es}$ of the above tuple contains information about $\mathbf{X}$ and $\mathbf{E}$, namely the terms $\mathbf{Xe}$ and $-\mathbf{Es}$. This is similar to the Extended-LWE assumption [39], which 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 $\langle \mathbf{z}, \mathbf{e} \rangle \bmod q$ for a small $\mathbf{z}$ chosen by the adversary independently of $\mathbf{A}$. However, the latter assumption is not sufficient: in our case, the sample contains a hint about both the error and the secret, and additionally, as we are interested in updatable encryption, the adversary can make update queries before asking for the challenge. To answer such queries, one needs to know $\mathbf{A}$, which is part of the public key, in advance. We introduce the Hermite Normal Form Adaptive Extended LWE assumption $\mathsf{HNF\text{-}AextLWE}$, which precisely states that pseudorandomness of $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ still holds, provided an additional hint of the form $\langle \mathbf{z}_0, \mathbf{s} \rangle + \langle \mathbf{z}_1, \mathbf{e} \rangle \bmod q$, with $\mathbf{z}_0, \mathbf{z}_1$ arbitrarily chosen by the adversary after it sees $\mathbf{A}$. Equipped with this assumption, the rest of the proof can be adapted and we are able to prove the IND-CR-CPA security of our UPKE scheme under the $\mathsf{HNF\text{-}AextLWE}$ assumption. It remains to show that the latter assumption is implied by the standard LWE assumption.

Reduction from LWE. We first make a reduction from the adaptive extended-LWE ($\mathsf{AextLWE}$) problem to the $\mathsf{HNF\text{-}AextLWE}$ problem. $\mathsf{AextLWE}$ generalizes the Extended-LWE problem by allowing the adversary to choose $\mathbf{z}$ arbitrarily (and not necessarily small) given $\mathbf{A}$. The reduction adapts the one from LWE to $\mathsf{HNF\text{-}LWE}$ given in [6] to our setting. It relies on the observation made in [16] that, if $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ for $m \geq 16n + 4 \log\log q$, one can extract an invertible matrix $\mathbf{A}_0$ from $\mathbf{A}$ together with another matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{m' \times n}$ with $m' = m - 16n - 4\log\log q$ such that the matrix $\mathbf{A}_1 \cdot \mathbf{A}_0^{-1}$ is uniformly distributed. Importantly, a hint $\langle \mathbf{z}_0, \mathbf{s}^* \rangle + \langle \mathbf{z}_1, \mathbf{e}^* \rangle \bmod q$ for an $\mathsf{HNF\text{-}AextLWE}$ instance using $\mathbf{A}^*$ can be computed as a hint $\langle \mathbf{z}, \mathbf{e} \rangle \bmod q$ for an $\mathsf{AextLWE}$ instance using $\mathbf{A}$.

We then show that LWE reduces to this new adaptive version by seeing it as an instance of (an adaptive version of) entropic-LWE. Introduced in [14], Entropic-LWE is a version of LWE where the secret $\mathbf{s}$ is sampled from any predetermined distribution instead of being uniform. However, the distribution being fixed (it is a problem parameter), this version of entropic-LWE does not suit our adaptive framework. We generalize the proof of [14, Theorem 4.1] to the adaptive setting. The reduction goes as follows. First observe that the hint $h = \langle \mathbf{z}, \mathbf{e} \rangle$ can also be computed as $\langle \mathbf{z}, \mathbf{b} \rangle - \langle \mathbf{z}, \mathbf{A}\mathbf{s} \rangle$, with $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$ being the LWE instance. One can then replace h by $\langle \mathbf{z}, \mathbf{A}\mathbf{s} \rangle$ (since $\mathbf{z}$ and $\mathbf{b}$ are known to the adversary). This makes the hint h only depend on $\mathbf{s}$ and $\mathbf{A}$. Then, the main idea is to replace the matrix $\mathbf{A}$ by an LWE instance $\mathbf{BC} + \mathbf{F}$ and to replace the error $\mathbf{e}$ by $\mathbf{F}\mathbf{e}_1 + \mathbf{e}_2$ with $\mathbf{e}_1, \mathbf{e}_2$ chosen from appropriate Gaussian distributions, which results in the LWE instance $(\mathbf{A}, \mathbf{b})$ to be of the form $(\mathbf{BC} + \mathbf{F}, \mathbf{BCs} + \mathbf{F}(\mathbf{s} + \mathbf{e}_1) + \mathbf{e}_2)$. The Leftover Hash Lemma allows to argue that $\mathbf{Cs}$ is statistically close to uniform if $\mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s} + \mathbf{e}_1, h)$ is large enough. This allows to transform the sample to a form $(\mathbf{BC} + \mathbf{F}, \mathbf{B}\mathbf{s}^* + \mathbf{F}\mathbf{s} + \mathbf{e})$ by further reverting the error modification. Note that $\mathbf{s}^*$ and $\mathbf{e}$ are now independent from h, which only depends on $\mathbf{A} = \mathbf{BC} + \mathbf{F}$ and $\mathbf{s}$ via our first remark, and we can rely on the LWE assumption for instance $(\mathbf{B}, \mathbf{B}\mathbf{s}^* + \mathbf{e})$ to prove pseudorandomness, completing the reduction.

Combining the above two results, we then obtain an IND-CR-CPA UPKE construction based on the standard LWE assumption, leading to the first lattice-based UPKE with polynomial modulus-to-noise ratio. We now explain how we transform this construction in order to achieve IND-CU-CCA security.

A Fujisaki-Okamoto transform for UPKE. Prior works [22,1] have relied on the Naor-Yung paradigm [38] to achieve CCA-security, which requires simulation-sound NIZK proofs. While this allows to remain in the standard model, efficient instantiations of NIZKs rely on random oracles, which motivates us to consider a ROM-based transform following the Fujisaki-Okamoto transform [24]. As we aim for practical efficiency, we focus on constructing IND-CR-CCA updatable key encapsulation mechanism (UKEM), a notion we introduce in this work. Our transform allows to construct IND-CR-CCA UKEM in the ROM with similar efficiency as that of the underlying IND-CR-CPA UPKE. To encapsulate a key for a target user with public key pk_t (at epoch t), one produces a ciphertext ct as an encryption of a uniform message m with randomness extracted from applying a hash function G (modeled as a random oracle) to the public key pk_t and the message m. The encapsulated key is defined as $\mathsf{H}(\mathsf{ct}, m)$ for another hash function (also modeled as a random oracle). Decapsulation recovers m by decrypting ct and re-encrypts it to check that ct was properly generated, in which case one computes the key $\mathsf{H}(\mathsf{ct}, m)$. The update mechanism $\mathsf{UpdatePk}, \mathsf{UpdateSk}$ are exactly the same as that of the underlying IND-CR-CPA UPKE scheme. Overall, this is the same transform as for PKE [27] except that pk_t is fed as input to G. The security analysis follows the standard route for FO analyses: we modify oracles to allow the challenger to simulate the decapsulation oracle without knowledge of the secret key sk. The main change is that we rely on the additional pk_t which is fed as an additional input to the hash function G in order to keep track of possibly valid ciphertexts known by the adversary for each epoch t.

In a concurrent work, Asano et al. [7] define a similar FO transform to build IND-CR-CCA secure UPKEs. The authors point out a weakness in the generic CCA transform from [22]: the latter work does not consider the possibility of updates of the public key that would allow the adversary to come back to the challenge public key and then trivially break security by querying the CCA decryption oracle on the ciphertext. This is allowed as in [22], this query is forbidden only at the challenge epoch. This is solved in [7] by generalizing the technique of [1], which adjoins a counter to the public key that is incremented at each update. The construction of [7] relies on using this counter in the derandomization step of their FO transform, which then makes any ciphertext generated in a previous epoch invalid for decryption queries. Our security model for IND-CR-CCA UKEM deals with this problem by adding another sanity check in the decapsulation oracle: we require that the adversary is not allowed to make a decapsulation query of the challenge ciphertext only if it current public key is the same as the challenge one.

Adding security against malicious updates. Next, we extend our IND-CR-CCA construction to achieve IND-CU-CCA security. This is achieved via the standard Naor-Yung "double-encrypt + NIZK" paradigm [38] applied (only) to the update mechanism: a user's public key is now a pair of public keys (pk_0^L, pk^R). The first one is an evolving key, for which the user keeps the corresponding secret key sk_0^L, while the second one is never updated and its corresponding secret key is discarded after generation. To update a target public key (pk_t^L, pk^R) used at epoch t, one updates the first key as before by revealing the next epoch public key pk_{t+1}^L and encrypting the private coins r used for the update. However, rather than encrypting r under pk_t^L only, one also encrypts it under pk^R. Additionally, one produces a NIZK argument that the private coins underlying each ciphertext and used for updating the public key match. The encapsulation and decapsulation mechanisms are unchanged (and only use pk_t^L).

These changes allow us to argue about IND-CU-CCA security using standard techniques. Let $up^* = (\mathsf{ct}_L^*, \mathsf{ct}_R^*, \pi^*)$ denote the honest update generated by the challenger before leaking the secret key, and r^* denote the underlying private coins. In the IND-CU-CCA security reduction, one can then replace π^* by a simulated proof and ct_R^* by an encryption of 0 using the zero-knowledge property and the IND-CPA security of the underlying PKE, since no information about sk^R is revealed to the adversary. The soundness of the NIZK argument guarantees that the adversary cannot produce an accepting argument for invalid updates. Hence, security can be reduced to that of the underlying IND-CR-CCA UKEM: the IND-CR-CCA attacker can use the additional key sk_R to decrypt the private coins r used by the IND-CU-CCA adversary in its valid updates queries, and forward r to its IND-CR-CCA challenger for producing the same update. A crucial remark is that the adversary gets to see an update (and then a NIZK argument) generated by the challenger only at the very end of the game, when it compromises the key. In particular, it can no longer query oracles from this point and therefore cannot use this proof as part of oracle queries. This allows us to rely on a NIZK argument which is only computational zero-knowledge.

Concrete parameters. We provide concrete parameters for our (IND-CR-CPA / IND-CR-CCA) scheme, following design choices of CRYSTALS-Kyber [11]: we instantiate our construction in the module lattices setting, using binomial distributions. In particular, we assume that our scheme is secure in the module setting though our security analysis does not immediately carries over to the Module Learning With Errors (MLWE) setting [15,32]. To extend it, one would need a similar reduction from decision entropic-MLWE to MLWE, which is currently lacking though a recent work from [12] shows a reduction for the search variants, providing a first step in this direction.

Notice that, as our modulus is small and the key can keep growing with (adversarially generated) updates, we can only guarantee correctness for a bounded number of updates as the decryption error might become too large at some point. We introduce a parameter k which is the maximal number of updates for which correctness is guaranteed with probability extremely close to 1. This parameter affects the size of the modulus q and forces us to use a larger modulus compared to Kyber (which uses $q = 3329$ and achieves a ciphertext size of 0.8KB for 128 bit CCA security). Note that in practice, if randomness is honestly sampled from centered distribution (e.g., $\mathbf{r} \leftarrow U(\{-1, 0, 1\}^n)$), the expected number of supported updates is $O(k^2)$. In Table 1, we provide parameters for our IND-CR-CPA/CCA UKEM schemes, for $k \in \{2^5, 2^{10}, 2^{15}, 2^{20}\}$, and for a security of λ close to 128 bits.

	λ	q	k	$\|ct\|$	$\|up\|$
DCR-based construction [1]	128	_	∞	8.3KB	1.5KB
Estimate for [22]	120	$\approx 2^{85}$	2^5	33KB	360KB
This work	128	$\approx 2^{21}$	2^5	1.8KB	5.4KB
	128	$\approx 2^{26}$	2^{10}	3.0KB	12KB
	116	$\approx 2^{31}$	2^{15}	5.8KB	12KB
	128	$\approx 2^{36}$	2^{20}	9.1KB	27KB

Table 1. Concrete parameters for our IND-CR-CCA UKEM.

We provide a brief comparison with the DCR-based (IND-CR-CPA) construction of [1], whose ciphertext/update size is about 1.5KB. Note that in the latter work, the authors achieve CCA-security by adding NIZKs, which hurts their ciphertext size for the CCA setting (about 8.3KB for 128 bits of security), while using our FO transform leaves us with the same numbers for our IND-CR-CCA construction. In order to give an insight on the efficiency gain compared to the construction of [22] (which was not meant to be efficient), we provide estimates of practical parameters for their scheme. As it requires flooding, we first make the assumption that flooding by 64 bits suffices (see [37]). In order to give optimistic parameters, we relax their statistical leftover hash lemma to a computational one, i.e., we use an adaptation of the scheme from [34] rather than dual Regev encryption. This leads to considering parameters for our scheme but with flooding. Also, to achieve IND-CR-CCA security, we apply our efficient FO transform and not their generic one.

2 Preliminaries

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

Throughout this paper, 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.

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, polynomial-time. We use log to denote the logarithm in base 2 and ln to denote the logarithm in base e.

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 1 (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 $X + Y \sim \mathcal{S}_1 * \mathcal{S}_2$.

We recall the definition of min-entropy.

Definition 2 (Min-entropy). *Let X, Y be random variables. We define the min-entropy*

$$\mathsf{H}_\infty(X) = -\log\left(\max_x \mathbb{P}\left[X = x\right]\right)$$

and the average conditional min-entropy:

$$\mathsf{H}_\infty(X \mid Y) = -\log\left(\mathbb{E}_y[\max_x \mathbb{P}\left[X = x \mid Y = y\right]]\right).$$

Definition 3 (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:*

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

In our security analysis, we will rely on the leftover hash lemma, which we recall below.

Lemma 1 (Leftover Hash Lemma). *Fix $\varepsilon > 0$. Let X be a random variable on $\{0,1\}^m$ and E be a random variable possibly correlated to X. Assume that the conditional min-entropy satisfies $\mathsf{H}_\infty(X|E) \geq k$. Let $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$ where $\mathcal{H}_n = \{h_s\}_{s \in \{0,1\}^d}$ for all n, be a universal hash family with output length $m \leq k - 2\log(1/\varepsilon)$. Then, we have*

$$\Delta\left((h_{U_d}(X), U_d, E), (U_m, U_d, E)\right) < \varepsilon,$$

where $U_d \sim \mathcal{U}(\{0,1\}^d)$, $U_m \sim \mathcal{U}(\{0,1\}^m)$.

2.1 Gaussian distributions

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

Definition 4 (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 2 (Gaussian tail-bound, [20, Lemma 2.13]). *Let $\mathbf{x} \sim \mathcal{D}_{\mathbb{Z}^m, \sigma}$, then for all $t > 1$, we have*

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

Lemma 3 (Gaussian convolution, [10, 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 4 (Gaussian decomposition, instantiated from [36, Lemma 1]). *For $m \geq n$, 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 4, one needs to control the ratio $s_1(\mathbf{F})$. This is the purpose of the following result.

Lemma 5 (Adapted from [2, Lemma 8]). *There exists a constant $K > 1$ such that the following holds. For $m \geq 2n$, $\sigma > K\sqrt{n}$ 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 $\mathbf{F}$

In Section 3, we adapt one of the main results from [14] to reduce LWE to an adaptive version of extended-LWE. The authors originally reduce LWE to entropic-LWE. Considering a secret $\mathbf{s} \in \mathbb{Z}_q^n$ and a (continuous or discrete) noise $\mathbf{e}$, their result relies on a bound for the quantity $\mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s}+\mathbf{e})$ which they call *noise-lossiness*. We give the following lemma, which is the discrete version of [14, Lemma 5.2]. A proof is given in Section B for completeness.

Lemma 6. *Let $q, n, m, \sigma > 0$ with $m > n$. Let $\mathbf{s}$ be a random variable on $\mathbb{Z}_q^n$ and $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$. If $q/\sigma > \sqrt{\ln(4n)/\pi}$, then we have:*

$$\mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s}+\mathbf{e}) \geq \mathsf{H}_\infty(\mathbf{s}) - n\log\left(\frac{q}{\sigma}\right) - 1 \ .$$

2.2 Updatable Public Key Encryption

We recall the syntax of Updatable Public Key Encryption (UPKE) and adapt the underlying $\mathsf{IND\text{-}CR\text{-}CPA}$ security notion defined in [22], 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 our LWE-based scheme, updates make the key slightly larger and then after a (large but polynomial) number of updates, correctness of decryption is no longer guaranteed. This results from the fact that we are able to work over a (small) polynomial modulus.

Definition 5. *(Updatable Public Key Encryption) An updatable public key encryption scheme is 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* (pk, sk).
- $\mathsf{Enc}(pk, m)$ *takes as input a public key* pk *and a message* m *and outputs a ciphertext* ct.
- $\mathsf{Dec}(sk, ct)$ *takes as input a secret key* sk *and a ciphertext* ct *and outputs a message* m'.
- $\mathsf{UpdatePk}(pk)$ *takes as input a public key* pk *and outputs an update* up *and a new public key* pk'.
- $\mathsf{UpdateSk}(sk, up)$ *takes as input a secret key* sk *and an update* up *and outputs a new secret key* 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* $t \leq k$

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

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

We give the definition from [22] which we adapt to the bounded number of updates setting by adding a parameter k for the number of updates.

Definition 6 **(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*

$\mathsf{UpdatePk}$. *We give the* k-IND-CR-CPA *security game in Figure 1.*

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

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

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

We also 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 for our FO transform.

Definition 7 **(γ-spreadness, adapted from [23, 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*

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

Parameters: λ, k.

GAME($\mathcal{A}$):

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

$\mathcal{O}_{up}(r)$:

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

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

2.3 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 [22] for UPKEs.

Definition 8 (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* (pk, sk)*.*
- $\mathsf{Encaps}(pk)$ *takes as input a public key* pk *and outputs an encapsulation* c *and a key* K*.*
- $\mathsf{Decaps}(sk, c)$ *takes as input a secret key* sk *and an encapsulation* c *and outputs a key* K'*.*
- $\mathsf{UpdatePk}(pk)$ *takes as input a public key* pk *and outputs an update* up *and a new public key* pk'*.*
- $\mathsf{UpdateSk}(sk, up)$ *takes as input a secret key* sk *and an update* up *and outputs a new secret key* 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 UKEM scheme is said to be (k, δ)*-correct, for* $\delta > 0$*, if for all* $t \leq k$

$$\mathbb{P}\left[\mathsf{Decaps}(sk_t, c_t) \neq K_t \mid (c_t, K_t) \leftarrow \mathsf{Encaps}(pk_t)\right] < \delta \ ,$$

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

The 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 in order to build our IND-CR-CCA UKEM.

Definition 9 (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*. We give the game for k-IND-CR-CCA security for an adversary that has access to a random oracle* H *in Figure 2.*

Parameters: λ, k.

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

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

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

Fig. 2: k-IND-CR-CCA security game in the ROM. Note that if $\beta = 0$, then the value of the key $K^\star$ is the output of Encaps.

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

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

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

Notice that compared to the IND-CR-CCA definition for UPKE given in [22], 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 [7], which generalizes the one considered in [1].

In order to define the stronger k-$\mathsf{IND\text{-}CU\text{-}CCA}$ security notions for UKEM, we add an algorithm $\mathsf{VerifyUpdate}$ to the UKEM syntax that allows a user to check the validity of an update. Specifically, $\mathsf{VerifyUpdate}(pk, (pk', up))$ takes as input the current epoch public key pk and a proposed update (pk', up) and returns a Boolean value. k-$\mathsf{IND\text{-}CU\text{-}CCA}$ security aims to guarantee security against adversaries who makes malicious updates.

Definition 10 (k-$\mathsf{IND\text{-}CU\text{-}CCA}$ KEM security in the ROM). *Let* $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk}, \mathsf{VerifyUpdate})$ *be a UKEM. The security game for* $\mathsf{IND\text{-}CU\text{-}CCA}$ *is identical to the* $\mathsf{IND\text{-}CR\text{-}CCA}$ *game, except for the modified* $\mathcal{O}_{up}(\cdot)$ *oracle. We present the modified* $\mathcal{O}_{up}$ *oracle in Figure 3.*

$\mathcal{O}_{up}(pk', up)$:
  if $\mathsf{VerifyUpdate}(pk_t, (pk', up)) = \perp$ **then**
    return $\perp$;
  end
  $pk_{t+1} = pk'$;
  $sk_{t+1} \leftarrow \mathsf{UpdateSk}(sk_t, up_{t+1})$;
  $t = t + 1$.

Fig. 3: k-$\mathsf{IND\text{-}CU\text{-}CCA}$ security game in the ROM.

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

In the rest of the paper, we omit the k in k-$\mathsf{IND\text{-}CR\text{-}CPA}$/$k$-$\mathsf{IND\text{-}CR\text{-}CCA}$/$k$-$\mathsf{IND\text{-}CU\text{-}CCA}$ when it is implicit.

3 Extended LWE

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

Definition 11. *(Learning With Errors -* LWE*) Let* $\lambda \geq 0$ *be a security parameter. Let* $q = q(\lambda), n = n(\lambda), m = m(\lambda) \geq 0$*,* $\mathcal{S}$ *be a distribution on* $\mathbb{Z}_q^n$ *and* χ *be an error distribution on* $\mathbb{Z}^m$*. The goal of* $\mathsf{LWE}_{q,n,m,\chi}(\mathcal{S})$ *for an adversary* $\mathcal{A}$ *is to distinguish between* $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$ *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*

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

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

The extended-LWE assumption claims that pseudorandomness of an LWE instance $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ still holds when the adversary is given an additional hint h

computed as $\langle \mathbf{z}, \mathbf{e} \rangle \bmod q$ 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 [39]. Furthermore, we remove the constraint that the hint vector $\mathbf{z}$ be short relative to the modulus q. In Theorem 1, we prove that LWE reduces to this adaptive version by seeing it as an instance of (adaptive) entropic-LWE, which generalizes the LWE assumption by allowing the secret $\mathbf{s}$ to be sampled in an arbitrary (fixed) set.

Definition 12 (Adaptive extended-LWE - AextLWE). *Let $\lambda \geq 0$ be a security parameter. Let $q = q(\lambda), n = n(\lambda), m = m(\lambda) \in \mathbb{N}$ and χ be an error distribution on $\mathbb{Z}^m$. The goal of $\mathsf{AextLWE}_{q,n,m,\chi}$ for an adversary $\mathcal{A}$ is to distinguish between the case where $\beta = 0$ and $\beta = 1$ in the interactive game depicted in Figure 4. We define the advantage of $\mathcal{A}$ in the AextLWE game as*

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

where the elements are distributed as shown in Figure 4.

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

$\mathcal{C}^{\mathsf{AextLWE}}$		$\mathcal{A}$
$\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$ $\beta \hookleftarrow \mathcal{U}(\{0,1\})$	$\xrightarrow{\mathbf{A}}$	
		$\mathbf{z}, st \leftarrow \mathcal{A}_1(\mathbf{A})$
	$\xleftarrow{\mathbf{z}}$	
$\mathbf{s} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n), \mathbf{e} \hookleftarrow \chi^m$ $h = \langle \mathbf{z}, \mathbf{e} \rangle \bmod q$ $\mathbf{b} = \begin{cases} \mathbf{As} + \mathbf{e} & \text{if } \beta = 0 \\ \mathbf{u} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m) & \text{if } \beta = 1 \end{cases}$		
	$\xrightarrow{\mathbf{b}, h}$	
		$\beta' \leftarrow \mathcal{A}_2(\mathbf{A}, \mathbf{b}, \mathbf{z}, h, st)$
	$\xleftarrow{\beta' \in \{0,1\}}$	

Fig. 4: 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 [6, Lemma 2]. Lemma 7 shows that the HNF variant reduces to the standard Adaptive extended-LWE.

Definition 13 (HNF Adaptive extended-LWE - HNF-AextLWE). *Let $\lambda \in \mathbb{N}$ be a security parameter. Let $q = q(\lambda), n = n(\lambda), m = m(\lambda) \in \mathbb{N}$ and χ be an error distribution on $\mathbb{R}^m$. The goal of $\mathsf{HNF\text{-}AextLWE}_{q,n,m,\chi}$ for an adversary $\mathcal{A}$*

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 $\mathsf{HNF\text{-}AextLWE}$ *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 5.

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

$\mathcal{C}^{\mathsf{HNF\text{-}AextLWE}}$		$\mathcal{A}$
$\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$ $\beta \hookleftarrow \mathcal{U}(\{0,1\})$	$\xrightarrow{\mathbf{A}}$	
		$\mathbf{z}_0, \mathbf{z}_1, st \leftarrow \mathcal{A}_1(\mathbf{A})$
	$\xleftarrow{\mathbf{z}_0, \mathbf{z}_1}$	
$\mathbf{s} \hookleftarrow \chi^n, \mathbf{e} \hookleftarrow \chi^m$ $h = \langle \mathbf{z}_0, \mathbf{s} \rangle + \langle \mathbf{z}_1, \mathbf{e} \rangle \bmod q$ $\mathbf{b} = \begin{cases} \mathbf{As} + \mathbf{e} & \text{if } \beta = 0 \\ \mathbf{u} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m) & \text{if } \beta = 1 \end{cases}$		
	$\xrightarrow{\mathbf{b}, h}$	
		$\beta' \leftarrow \mathcal{A}_2(\mathbf{A}, \mathbf{b}, \mathbf{z}, h, st)$
	$\xleftarrow{\beta' \in \{0,1\}}$	

Fig. 5: 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 $\mathsf{AextLWE}$ and $\mathsf{HNF\text{-}AextLWE}$, the hint $h \in \mathbb{Z}_q$ also becomes a vector $\mathbf{h} \in \mathbb{Z}_q^k$. Also, the multiple-secret variants for $\mathsf{AextLWE}$ and $\mathsf{HNF\text{-}AextLWE}$ 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 run-time against the single-secret problem Asp such that $\mathcal{A}$'s advantage is bounded by $k \cdot \mathsf{Adv}^{\mathsf{Asp}}(\mathcal{B})$.

Lemma 7. *Let $q \geq 25, n \geq 1, m \geq 16n + 4 \log\log q$, then any adversary $\mathcal{A}$ for* $\mathsf{HNF\text{-}AextLWE}_{q,n,m',\sigma}$ *, where $m' = m - 16n - 4 \log\log q$, running in time T can be used to build an adversary $\mathcal{B}$ for* $\mathsf{AextLWE}_{q,n,m,\sigma}$ *running in time $\approx T$, with advantage*

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

Proof. Assume $\mathcal{A}$ is an adversary against $\mathsf{HNF\text{-}AextLWE}$. We construct an adversary $\mathcal{B}$ against $\mathsf{AextLWE}$ 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 $\mathsf{AextLWE}$ 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 [16, Claim 2.13], with probability at least $1 - 2e^{-1} \geq 1/4$, 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 $\tilde{\mathbf{A}}_1 \in \mathbb{Z}_q^{m' \times n}$.

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 $h = \langle \mathbf{z}, \mathbf{e} \rangle \bmod q$ from the $\mathsf{AextLWE}$ challenger, with $\mathbf{b}_0 \in \mathbb{Z}_q^n$, $\mathbf{b}_1 \in \mathbb{Z}_q^{m'}$ and $\mathbf{d} \in \mathbb{Z}_q^{m-m'}$. 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

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

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 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 1. *Let q be a prime, $\varepsilon > 0$, $n, m, k, \gamma, \sigma \geq 0$ and $K > 1$ be the constant from Lemma 5. Assume that*

1. $n \log(\sqrt{2n}) > (k+1)\log(q) + 2\log(1/\varepsilon)$,
2. $m > 2n$,

3. $\gamma > K\sqrt{n}$,
4. $\sigma \ > 2K\gamma\sqrt{mn}$,
5. $q > \sqrt{2n\ln(4n)/\pi}$.

Then for any adversary $\mathcal{A}$ for $\mathsf{AextLWE}_{q,n,m,\sigma}$ running in time T, there exists an adversary $\mathcal{B}$ for $\mathsf{LWE}_{q,k,m,\gamma}$ running in time $\mathsf{poly}(m, \log q) \cdot T$ such that:

$$\mathsf{Adv}^{\mathsf{AextLWE}}(\mathcal{A}) \leq 2me^{-\frac{m}{2}} + 17e^{-n/K} + \varepsilon + (2n+1) \cdot \mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B}) \ .$$

The first condition of the theorem allows us to argue that there will be enough entropy on the secret key, even after revealing a leakage on it. Conditions 2 and 4 allow us to apply Lemma 5. Conditions 3 allows us to analyze a discrete Gaussian convolution (using Lemma 3). Condition 5 allows us to apply Lemma 6. As an example of parameter instantiation, we can take $\varepsilon = e^{-n}$, $n = \mathcal{O}(k)$, $m = \mathcal{O}(n)$, $q = n^{O(1)}$, $\gamma = \mathcal{O}(n^{1/2})$ and $\sigma = \mathcal{O}(n^{3/2})$.

The proof of Theorem 1 is adapted from that of hardness of entropic LWE from [14, Theorem 4.1]. We show that a hint on the error $\mathbf{e}$ of the LWE sample can be viewed as a hint on the secret $\mathbf{s}$. As the hint lives in $\mathbb{Z}_q$, it leaks at most $\log q$ bits of entropy, making it an adaptive instance of entropic-LWE, where the secret retains high entropy. The leftover hash lemma allows to handle the adaptive part of the proof.

Proof. We define a sequence of games and show that they are computationally or statistically indistinguishable.

Game G_0: This is the original $\mathsf{AextLWE}$ game with $\beta = 0$. The challenger samples $\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m\times n})$ and sends $\mathbf{A}$ to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ chooses $\mathbf{z} \in \mathbb{Z}_q^m$ and sends it to the challenger. Then the challenger samples $\mathbf{s} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, computes $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ and $h = \langle \mathbf{z}, \mathbf{e}\rangle \bmod q$ to finally send $(\mathbf{b}, h)$ to $\mathcal{A}$.

Game G_1: This is the same game as the last one, except that instead of computing the hint h as $h = \langle \mathbf{z}, \mathbf{e}\rangle \bmod q$, the challenger sets $h = \langle \mathbf{z}, \mathbf{A}\mathbf{s}\rangle = \left\langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\right\rangle \bmod q$.

Note that $\langle \mathbf{z}, \mathbf{A}\mathbf{s}\rangle = \langle \mathbf{z}, \mathbf{b}\rangle - \langle \mathbf{z}, \mathbf{e}\rangle$, where $\langle \mathbf{z}, \mathbf{b}\rangle$ is computable by the adversary. Hence this game is computationally equivalent to the last one.

Game G_2: Here we change the way $\mathbf{A}$ is computed by the challenger. It now samples $\mathbf{B} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m\times k})$, $\mathbf{C} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{k\times n})$, $\mathbf{F} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m\times n},\gamma}$ and sets $\mathbf{A} = \mathbf{B}\mathbf{C} + \mathbf{F}$.

Observe that $\mathbf{B}\mathbf{C}+\mathbf{F}$ corresponds to a sample from the multiple-secret variant of $\mathsf{LWE}_{q,k,m,\gamma}$ with n secrets. By a hybrid argument, for any efficient adversary $\mathcal{B}$ that distinguishes between G_1 and G_2, we can build an efficient adversary $\mathcal{B}'$ for $\mathsf{LWE}_{q,k,m,\gamma}$ such that $\mathsf{Adv}^{\mathsf{dist}}_{G_1,G_2}(\mathcal{B}) \ \leq \ n \cdot \mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B}')$.

Game G_3: In this game, the challenger aborts if $\|\mathbf{F}\|_2 > 2\gamma\sqrt{m}$ or if $s_1(\mathbf{F}) > K\gamma\sqrt{m}$.

Lemma 2 and Lemma 5 show that this can happen with probability at most $me^{-\frac{m}{2}} + 4e^{-n/K}$. This implies that any adversary $\mathcal{B}$ has advantage at most $\mathsf{Adv}^{\mathsf{dist}}_{G_2,G_3}(\mathcal{B}) \leq me^{-\frac{m}{2}} + 4e^{-n/K}$ in distinguishing between games G_2 and G_3.

Game G_4: This time, we change the way the error $\mathbf{e}$ is computed by the challenger. Let $\sigma_1 = \sqrt{2n}$. Instead of sampling $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, the challenger computes $\mathbf{\Sigma} = \sigma^2\mathbf{Id} - \sigma_1^2\mathbf{F}^T\mathbf{F}$ to sample $\mathbf{e}_2 \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\mathbf{\Sigma}}$, $\mathbf{e}_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma_1}$ and set $\mathbf{e} = \mathbf{F}\mathbf{e}_1 + \mathbf{e}_2$. Note that now

$$\mathbf{b} = (\mathbf{BC} + \mathbf{F})\mathbf{s} + \mathbf{e}_1\mathbf{F} + \mathbf{e}_2 = \mathbf{BCs} + \mathbf{F}(\mathbf{e}_1 + \mathbf{s}) + \mathbf{e}_2.$$

Let $\mathcal{S}$ be the distribution of the random variable $\mathbf{F}\mathbf{e}_1 + \mathbf{e}_2$. By Lemma 4 we have $\Delta(\mathcal{S}, \mathcal{D}_{\mathbb{Z}^m,\sigma}) < 4e^{-m}$. Hence any adversary $\mathcal{B}$ has advantage at most $\mathsf{Adv}^{\mathsf{dist}}_{G_3,G_4}(\mathcal{B}) \leq 4e^{-m}$ in distinguishing between games G_3 and G_4.

Game G_5: In this game, when computing $\mathbf{b}$, the challenger samples a value $\mathbf{s}^\star \hookleftarrow \mathcal{U}(\mathbb{Z}_q^k)$ and instead of computing $\mathbf{b} = \mathbf{BCs} + \mathbf{F}(\mathbf{s} + \mathbf{e}_1) + \mathbf{e}_2$, it sets $\mathbf{b} = \mathbf{B}\mathbf{s}^\star + \mathbf{F}(\mathbf{s} + \mathbf{e}_1) + \mathbf{e}_2$.

Here we replaced $\mathbf{Cs}$ by a uniform value $\mathbf{s}^\star \hookleftarrow \mathcal{U}(\mathbb{Z}_q^k)$. To show that this modification is statistically indistinguishable for the adversary, we show that $\mathbf{s}$ has enough entropy to apply the leftover hash lemma. Indeed, apart from $\mathbf{Cs}$, the information that the adversary has in game G_4 can be computed from $\mathbf{B}$, $\mathbf{F}$, $\mathbf{e}_2$, $\mathbf{e}_1 + \mathbf{s}$ and $h = \langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\rangle$. Only the last two values depend on $\mathbf{s}$. We have

$$\begin{aligned}
\mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s} + \mathbf{e}_1, h) &\geq \mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s} + \mathbf{e}_1) - \log(q) \\
&\geq \mathsf{H}_\infty(\mathbf{s}) - n\log(\frac{q}{\sigma_1}) - \log(q) \\
&= n\log(q) - n\log(\frac{q}{\sigma_1}) - \log(q) \\
&= n\log(\sigma_1) - \log(q) \\
&> k\log(q) + 2\log(1/\varepsilon).
\end{aligned}$$

The second inequality comes from applying Lemma 6 and the last one comes from the first condition of the theorem. Indeed, as $\sigma_1 = \sqrt{2n}$, Condition 1 can be rewritten as $n\log(\sigma_1) > (k+1)\log(q) + 2\log(1/\varepsilon)$. Lemma 1 gives that

$$\Delta\left((\mathbf{C}, \mathbf{Cs}, \mathbf{s} + \mathbf{e}_1, h), (\mathbf{C}, \mathbf{s}^\star, \mathbf{s} + \mathbf{e}_1, h)\right) < \varepsilon,$$

which implies that any adversary $\mathcal{B}$ distinguishing between games G_4 and G_5 has advantage at most $\mathsf{Adv}^{\mathsf{dist}}_{G_4,G_5}(\mathcal{B}) < \varepsilon$.

Game G_6: Here we revert the modification made on the error sampling. Instead of computing the error $\mathbf{e}$ as a decomposition $\mathbf{e} = \mathbf{F}\mathbf{e}_1 + \mathbf{e}_2$, we sample $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ again. As for the analysis of game G_4, this game is statistically close from the last via Lemma 4.

Game G_7: In this game, we still have $\mathbf{A} = \mathbf{BC} + \mathbf{F}$. However this time, the challenger samples $\mathbf{u} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m)$ and sets $\mathbf{b} = \mathbf{u}$, so that the adversary receives $(\mathbf{b} = \mathbf{u}, h = \langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\rangle \bmod q)$.

Assume we have a distinguisher $\mathcal{A}$ between D_6 and D_7. We build an adversary $\mathcal{B}$ against $\mathsf{LWE}_{q,k,m,\sigma}$ that has the same advantage. Adversary $\mathcal{B}$ receives $\mathbf{B} \in \mathbb{Z}_q^{m\times k}$ and $\mathbf{b} \in \mathbb{Z}_q^m$ from its $\mathsf{LWE}_{q,k,m,\sigma}$ challenger. It then samples

$\mathbf{C} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{k\times n})$ and $\mathbf{F} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m\times n},\gamma}$, sets $\mathbf{A} = \mathbf{BC} + \mathbf{F}$ and sends $\mathbf{A}$ to $\mathcal{A}$. From there, it receives a vector $\mathbf{z} \in \mathbb{Z}_q^m$ from $\mathcal{A}$, which it uses to compute $h = \langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\rangle \bmod q$, for $\mathbf{s} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$. Finally, adversary $\mathcal{B}$ sets $\mathbf{b}' = \mathbf{b} + \mathbf{Fs}$ and sends $(\mathbf{b}', h)$ to $\mathcal{A}$.

If $\mathbf{b} = \mathbf{Bs}^\star + \mathbf{e}$, then $\mathbf{b}' = \mathbf{Bs}^\star + \mathbf{e} + \mathbf{Fs}$, so $\mathcal{A}$ was given a tuple from game G_6. If $\mathbf{b} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m)$, then $\mathbf{b}' \sim \mathcal{U}(\mathbb{Z}_q)$ and $\mathcal{A}$ was given a tuple from game G_7. We thus have $\mathsf{Adv}^{\mathsf{dist}}_{G_6,G_7}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B})$. As we assumed $\gamma < \sigma$, $\mathsf{LWE}_{q,k,m,\sigma}$ is no easier than $\mathsf{LWE}_{q,k,m,\gamma}$. By using Lemma 3 and the condition on γ, one can build an adversary $\mathcal{B}'$ for $\mathsf{LWE}_{q,k,m,\gamma}$. It simply adds a Gaussian noise of standard deviation $\sqrt{\sigma^2-\gamma^2}$ to the LWE sample and calls $\mathcal{B}$ on this sample, so that $\mathsf{Adv}^{\mathsf{LWE}_{q,k,m,\sigma}}(\mathcal{B}) \leq \mathsf{Adv}^{\mathsf{LWE}_{q,k,m,\gamma}}(\mathcal{B}') + e^{-n}$.

From now on we only revert the modifications that were made at the beginning.

Game G_8**:** In this game, the challenger does not abort anymore if $\|\mathbf{F}\|_2 > 2\gamma\sqrt{m}$, $s_1(\mathbf{F})/s_n(\mathbf{F}) > C_1$ or if $s_1(\mathbf{F}) > C_2\sigma\sqrt{m}$. As for the analysis of game G_3, this induces an advantage discrepancy $\leq me^{-\frac{m}{2}} + 4e^{-n/K}$.

Game G_9**:** Here $\mathbf{A}$ is again sampled as $\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{m\times n})$. As for the analysis of game G_2, this game is computationally indistinguishable from the last by the hardness of the $\mathsf{LWE}_{q,k,m,\gamma}$ assumption.

Game G_{10}**:** This is the final game. The hint $h = \langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\rangle$ is now computed as $h = \langle \mathbf{z}, \mathbf{e}\rangle$ for $\mathbf{e} \hookleftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$. At the end, the adversary thus receives the tuple $(\mathbf{b} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^m), \langle \mathbf{z}, \mathbf{e}\rangle)$. This change results in a computationally equivalent game. Indeed, both $\langle \mathbf{A}^T\mathbf{z}, \mathbf{s}\rangle$ and $\langle \mathbf{z}, \mathbf{e}\rangle$ are publicly computable as $\mathbf{s}$ and $\mathbf{e}$ are values totally unrelated to $\mathbf{b}$. This is the $\beta = 1$ case in the $\mathsf{AextLWE}$ game. By collecting all the inequalities:

$$\mathsf{Adv}^{\mathsf{AextLWE}}(\mathcal{A}) \leq 2me^{-\frac{m}{2}} + 17e^{-n/K} + \varepsilon + (2n+1)\mathsf{Adv}^{\mathsf{LWE}}(\mathcal{B}),$$

which completes the proof of Theorem 1. □

4 IND-CR-CPA UPKE from LWE

We now describe a UPKE scheme with security based on the $\mathsf{HNF\text{-}AextLWE}$ assumption. As already shown, it is implied by the standard LWE assumption. Our scheme, detailed in Figure 6, avoid noise flooding by taking advantage of the $\mathsf{HNF\text{-}AextLWE}$ assumption defined in Section 3. We then provide the first efficient UPKE scheme based on lattices. Our construction follows the lines of [34] which underlies Kyber [11].

In contrast, the only prior lattice-based construction, proposed in [22] and based on the Dual-Regev PKE from [25], 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, σ)

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

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

$\mathsf{Dec}(sk = \mathbf{s}, ct = (\mathbf{ct}_0, ct_1))$:
 $\mathbf{v} = ct_1 - \mathbf{ct_0 s}$;
 return $\lfloor p/q \cdot \mathbf{v} \rceil_p$.

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

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

Fig. 6: LWE-based IND-CR-CPA UPKE construction.

Theorem 2. *Let $\varepsilon, \delta \in (0, 1), k > 0$. Let q, p be primes and $n, m > 0$ and $\sigma > 0$ such that $\sigma \geq \sqrt{2\ln(2n(1 + 1/\varepsilon))/\pi}$. Assuming the hardness of* HNF-AextLWE, *the scheme presented in Figure 6 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:*

$$\mathsf{Adv}_{\mathsf{UPKE}}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{A}) \leq \frac{14\varepsilon}{1-\varepsilon} + (2n+1) \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{B})\ .$$

Furthermore, assuming $q > 2p \cdot (2y^2\sigma^2 nk + y\sigma)$ and $p > 2y\sigma$ where $y = \sqrt{-2\log(\delta/(4n))}$, the scheme is (k, δ)-correct.

Proof. The proof of correctness is detailed in the appendix (Appendix B). We also provide a proof of γ-spreadness there, which is relevant for the next section.

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 $(\mathbf{r}_1, \boldsymbol{\eta}_1), \ldots, (\mathbf{r}_{chall}, \boldsymbol{\eta}_{chall})$ 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

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

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

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

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

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

together with $pk^* = (\mathbf{A}, \mathbf{b}_{last} + \mathbf{A}\mathbf{r}^* + \boldsymbol{\eta}^*)$ 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

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

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 substraction 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 8 below states that games G_2 and G_3 are computationally indistinguishable. The proof relies on the hardness of $\mathsf{HNF\text{-}AextLWE}$. In particular, any adversary $\mathcal{B}$ has advantage at most $\mathsf{Adv}(\mathcal{B}) \le n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}$ 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 $pk^* = (\mathbf{A}, \mathbf{A}\mathbf{t}+\tilde{\mathbf{e}}+\mathbf{A}\Delta^{\mathbf{r}}_{last}+\Delta^{\boldsymbol{\eta}}_{last})$.

Let $\delta = 2\varepsilon/(1-\varepsilon)$. Lemma 3 shows that this change only induces a statistically negligible bias. Specifically, assuming $\sigma \geq \sqrt{2\ln(2n(1+1/\varepsilon))/\pi}$, $\mathbf{t}$ 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 3 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 9 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 $\mathbf{c}^* = (\mathbf{X}_{chall}\mathbf{A}+\mathbf{E}_{chall}, \mathbf{X}_{chall}\mathbf{b}_{chall}+\mathbf{f}_{chall}+\lfloor q/p\rceil\cdot\boldsymbol{\mu}_\beta)$. We can rewrite $\mathbf{c}^*$ in a matrix form as:

$$\mathbf{X}_{chall}\left(\mathbf{A}\|\mathbf{b}_{chall}\right)+\left(\mathbf{E}_{chall}\|\mathbf{f}_{chall}\right)+\lfloor q/p\rceil\cdot\left(\mathbf{0}\|\boldsymbol{\mu}_\beta\right) \tag{1}$$

with $\mathbf{A} \hookleftarrow \mathcal{U}(\mathbb{Z}_q^{n\times n})$ and $\mathbf{b}_{chall} = \mathbf{b}_0+\mathbf{A}\Delta^{\mathbf{r}}_{chall}+\Delta^{\boldsymbol{\eta}}_{chall}$. Recall that we have $\mathbf{b}_0 \hookleftarrow \mathcal{U}(\mathbb{Z}_q^n)$ since game G_5. The last column of Equation (1) is

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

The first term is a multiple-secret LWE sample that is independent of any adverserially chosen value and the second one can be viewed as an HNF-AextLWE hint on the secret $\mathbf{X}_{chall}$ with the vector $\mathbf{v} = \mathbf{A}\Delta^{\mathbf{r}}_{chall} + \Delta^{\boldsymbol{\eta}}_{chall}$. Note that Equation (1) does not involve a hint on the error of the multi-sercet LWE sample, hence taking the $\mathbf{0}$ vector for the second part of the hint vector suffices.

The above indicates that the modification between this game and game G_5 can be analyzed by using the multiple-secret variant of $\mathsf{HNF\text{-}AextLWE}_{q,n,n+1,\sigma}$ with n secrets and hint vector $\mathbf{z} = [\mathbf{v}^T \| \mathbf{0}^T]^T$. Consequently, any adversary $\mathcal{A}$ has advantage at most $n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}$ in distinguishing between games G_5 and G_6.

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

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}_{\mathsf{UPKE}}(\mathcal{A}) \leq \frac{10\varepsilon}{1-\varepsilon} + (2n+1) \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}.$$

This completes the proof, up to Lemmas 8 and 9 below. □

Lemma 8. *For any adversary $\mathcal{A}$ that distinguishes between games G_2 and G_3, there exists an efficient algorithm $\mathcal{B}$ for $\mathsf{HNF\text{-}AextLWE}_{q,n,n,\sigma}$, calling $\mathcal{A}$ once, such that $\mathsf{Adv}^{\mathsf{dist}}_{G_2,G_3}(\mathcal{A}) \leq n \cdot \mathsf{Adv}^{\mathsf{HNF\text{-}AextLWE}}(\mathcal{B})$.*

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 $\Delta^{\mathbf{r}}_{last}$ and the sum of all the noises used for each updates $\Delta^{\boldsymbol{\eta}}_{last}$ as $\mathcal{A}$ has finished querying the $\mathcal{O}_{up}$ oracle.

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

It then receives from the challenger a matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times n}$ and a hint $\mathbf{h} = \mathbf{X}_{last}\mathbf{z}_0 + \mathbf{E}_{last}\mathbf{z}_1 = (\mathbf{X}_{last} \| \mathbf{E}_{last})\mathbf{z}$, where $\mathbf{z} = \left(\mathbf{z}_0^T \| \mathbf{z}_1^T\right)^T$. The matrix $\mathbf{B}$ is either uniform or of the form $\mathbf{X}_{last}\mathbf{A} + \mathbf{E}_{last}$.

Adversary $\mathcal{B}$ sets

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

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 $\mathbf{f}_{last}, \mathbf{r}^*, \boldsymbol{\eta}^* \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma}$.

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 $\mathsf{HNF\text{-}AextLWE}_{q,n,n,\sigma}$ 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. $\square$

Lemma 9. *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:*

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

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 $\mathbf{E}_{last}, \mathbf{X}_{last} \hookleftarrow \mathcal{D}_{\mathbb{Z}^{n\times n},\sigma}$. It then computes

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

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

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

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

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 $\mathbf{c} = \mathbf{B}\mathbf{s} + (\mathbf{e}^T \| \mathbf{f}^T)^T$, 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

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

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:

$$\begin{aligned} up^* &= (\mathbf{T}_{last},\, \mathbf{u}_1 + (\mathbf{T}_{last} - \mathbf{E}_{last} + \lfloor q/p \rfloor \cdot \mathbf{Id})\Delta^{\mathbf{r}}_{last} + \mathbf{X}_{last}\Delta^{\boldsymbol{\eta}}_{last}) \\ &= (\mathbf{T}_{last},\, (\mathbf{T}_{last} - \mathbf{E}_{last})\bar{\mathbf{s}} + \mathbf{X}_{last}(\bar{\mathbf{e}} + \Delta^{\boldsymbol{\eta}}_{last}) + \bar{\mathbf{f}} \\ &\quad + (\mathbf{T}_{last} - \mathbf{E}_{last} + \lfloor q/p \rfloor \cdot \mathbf{Id})\Delta^{\mathbf{r}}_{last}) \\ &= (\mathbf{T}_{last},\, (\mathbf{T}_{last} - \mathbf{E}_{last})(\bar{\mathbf{s}} + \Delta^{\mathbf{r}}_{last}) + \mathbf{X}_{last}(\bar{\mathbf{e}} + \Delta^{\boldsymbol{\eta}}_{last}) + \bar{\mathbf{f}} + \lfloor q/p \rfloor \cdot \Delta^{\mathbf{r}}_{last}). \end{aligned} \tag{2}$$

Let $\delta = 2\varepsilon/(1-\varepsilon)$, for $\varepsilon \in (0,1)$. As $\mathbf{s} \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2}$ and $\mathbf{s}' \hookleftarrow \mathcal{D}_{\mathbb{Z}^n,\sigma/2,\mathbf{t}/2}$, Lemma 3 gives that the distribution of $\bar{\mathbf{s}}$ has statistical distance at most δ from $\mathcal{D}_{\mathbb{Z}^n,\sigma/\sqrt{2},\mathbf{t}/2}$. 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 $\mathcal{D}_{\mathbb{Z}^n,\sigma/\sqrt{2},\tilde{\mathbf{e}}}$. 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 $\mathsf{Adv}^{\mathsf{dist}}_{G_4,G_5}(\mathcal{A}) - 3\delta$, completing the proof. □

5 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 [24] technique.

Definition 14 (FO-transform for UPKEs). *Let* UPKE *be a UPKE, and* G *and* H *be two functions modeled as random oracles. We define the transform* $\mathsf{FO}(\mathsf{UPKE},\mathsf{G},\mathsf{H})$ *in Figure 7.*

$\mathsf{KeyGen} = \mathsf{UPKE.KeyGen}$.

$\mathsf{Encaps}(pk)$:
- $m \hookleftarrow \mathcal{U}(\mathcal{M})$;
- $c \leftarrow \mathsf{UPKE.Enc}(pk, m; \mathsf{G}(pk, m))$;
- $K = \mathsf{H}(m, c)$;
- **return** (c, K).

$\mathsf{Decaps}(sk, c)$:
- $m' \leftarrow \mathsf{UPKE.Dec}(sk, c)$;
- **if** $c \neq \mathsf{UPKE.Enc}(pk, m'; \mathsf{G}(pk, m'))$
  - **then return** $\perp$;
- **return** $K' = \mathsf{H}(m', c)$.

$\mathsf{UpdatePk} = \mathsf{UPKE.UpdatePk}$.

$\mathsf{UpdateSk} = \mathsf{UPKE.UpdateSk}$.

Fig. 7: Transform $\mathsf{FO}(\mathsf{UPKE},\mathsf{G},\mathsf{H})$ for a UPKE using random oracles G,H.

Our FO transform is essentially the $\mathsf{KEM}^{\perp}$ construction from [27]. 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 3 (FO **transform for UPKEs).** *Let* $\gamma, \delta \in (0,1), k > 0$. *Let* $\mathsf{UPKE} = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ *denote a* γ*-spread and* (k,δ)*-correct* k-$\mathsf{IND\text{-}CR\text{-}CPA}$ *UPKE scheme. Then the UPKE* $\mathsf{FO}(\mathsf{UPKE}, \mathsf{G}, \mathsf{H})$ *is a* (k,δ)*-correct* k-$\mathsf{IND\text{-}CR\text{-}CCA}$ *UKEM in the ROM.*

More precisely, for any adversary $\mathcal{A}$ *for the* k-$\mathsf{IND\text{-}CR\text{-}CCA}$ *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* $\mathcal{O}_{dec}$, *there exists an adversary* $\mathcal{B}$ *for the* k-$\mathsf{IND\text{-}CR\text{-}CPA}$ *game of* UPKE *with a similar running time such that:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CCA}}(\mathcal{A}) \leq q_{\mathsf{G}} \cdot \delta + q_{\mathsf{D}} \cdot \gamma + 2\left(\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{B}) + \frac{q_{\mathsf{G}} + q_{\mathsf{H}}}{|\mathcal{M}|}\right) .$$

The proof of the above theorem follows standard techniques for FO analysis (e.g., [27]), and we postpone it to the appendix (Appendix C).

Note that we rely on the γ-spreadness of the underlying UPKE scheme. We prove this property for the scheme from Section 4 in the appendix (Appendix B.3).

6 Obtaining IND-CU-CCA Security

In this section, we further boost security in order to achieve $\mathsf{IND\text{-}CU\text{-}CCA}$-security. As in [1], we use a NIZK argument that two keys encrypt the same message in order to make a reduction from $\mathsf{IND\text{-}CU\text{-}CCA}$ to $\mathsf{IND\text{-}CR\text{-}CCA}$. 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 $\mathsf{IND\text{-}CR\text{-}CCA}$ challenger. We give the definitions about Non Interactive Zero Knowledge (NIZK) argument in the ROM in the appendix (Appendix A).

Let $\mathsf{UPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ be a k-$\mathsf{IND\text{-}CR\text{-}CPA}$ UPKE, for some $k > 0$. Define $\mathsf{UKEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps}, \mathsf{UpdatePk}, \mathsf{UpdateSk})$ as the k-$\mathsf{IND\text{-}CR\text{-}CCA}$ UKEM scheme obtained by applying our FO transform from Section 5 to UPKE, using G, H modeled as random oracles. Let F be a third function, also modeled as a random oracle. We assume that $\mathsf{UpdatePk}$ proceeds in two parts (this is the case for all known constructions, including the one from Section 4): $\mathsf{UpdatePk}(pk) = (\mathsf{Enc}(pk, r), \mathsf{NewPk}(pk, r))$, 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

$$\mathcal{L}_{up}^{\mathsf{UKEM}} = \{(pk_0, pk_1, pk', \mathsf{ct}_0, \mathsf{ct}_1) \mid \exists r_0, r_1, r,$$
$$\mathsf{ct}_0 = \mathsf{Enc}(pk_0, r; r_0) \wedge \mathsf{ct}_1 = \mathsf{Enc}(pk_1, r; r_1) \wedge (pk', ct_0) = \mathsf{UpdatePk}(pk_0; r)\}.$$

Let $\Pi = (\mathsf{Prove}^{\mathsf{F}}, \mathsf{Verify}^{\mathsf{F}})$ a NIZK argument in the random oracle for $\mathcal{L}_{up}^{\mathsf{UKEM}}$. We construct an k-$\mathsf{IND\text{-}CU\text{-}CCA}$ UKEM as described in Figure 8.

Theorem 4. *Let* $\mathsf{UPKE}, \mathsf{UKEM}, \Pi$ *be defined as above. Then, the construction* $\overline{\mathsf{UKEM}}$ *described in Figure 8 is an* k-$\mathsf{IND\text{-}CU\text{-}CCA}$ *UKEM. Specifically, for any*

$\overline{\mathsf{KeyGen}}(1^\lambda)$:
  $(pk_0, sk_0) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
  $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
  return $\overline{pk} = (pk_0, pk_1)$, $\overline{sk} = sk_0$.

$\overline{\mathsf{Encaps}}(\overline{pk})$:
  parse $\overline{pk}$ as (pk_0, pk_1);
  $(c, K) \leftarrow \mathsf{Encaps}(pk_0)$;
  return (c, K).

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

$\overline{\mathsf{UpdatePk}}(\overline{pk})$:
  parse $\overline{pk}$ as (pk_0, pk_1);
  sample $r \hookleftarrow R$;
  $pk'_0 \leftarrow \mathsf{NewPk}(pk_0, r)$;
  $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(pk_0, r)$;
  $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(pk_1, r)$;
  $\pi \leftarrow \mathsf{Prove}^{\mathsf{F}}(pk_0, pk_1, pk'_0, \mathsf{ct}_0, \mathsf{ct}_1, r)$;
  return $\overline{up} = (\mathsf{ct}_0, \mathsf{ct}_1, \pi)$, $\overline{pk}' = (pk'_0, pk_1)$.

$\overline{\mathsf{VerifyUpdate}}(\overline{up}, \overline{pk}')$:
  parse $\overline{up}$ as $(\mathsf{ct}_0, \mathsf{ct}_1, \pi)$ and $\overline{pk}'$ as (pk'_0, pk_1);
  return $\mathsf{Verify}^{\mathsf{F}}((pk_0, pk_1, pk'_0, \mathsf{ct}_0, \mathsf{ct}_1), \pi)$;

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

Fig. 8: Construction of a IND-CU-CCA UKEM.

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:

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

The proof closely follows the one of IND-CU-CCA security of the construction from [1] and is detailed in the appendix (Appendix D).

7 Concrete Parameters

In this section, we give some concrete parameters for the scheme presented in Section 4, which can directly be transformed into an IND-CR-CCA UKEM by applying the FO transform from Section 5. 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 4, 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 [11], we replace Gaussian distributions by the centered binomial distributions B_η, which for $\eta > 0$, samples elements $(a_i, b_i)_{i \le \eta} \hookleftarrow \mathcal{U}(\{0,1\}^2)$ 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 2). 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.

Our parameters are given in Table 2. 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 $\mathsf{IND\text{-}CR\text{-}CPA}$ construction of [1], 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.

	λ	q	n	d	p	η	δ	k	$\lvert ct\rvert$	$\lvert up\rvert$
DCR-based construction [1]	128	_	_	_	_	_	0	∞	8.3KB	1.5KB
Estimate for [22]	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 2. Parameter sets for the module variant of our IND-CR-CCA UKEM.

$\mathsf{IND\text{-}CU\text{-}CCA}$ **instantiation.** In order to add security against chosen updates via our transform from Section 6, 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:

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

where $pk_0 = (\mathbf{A}, \mathbf{b})$, $pk_1 = (\tilde{\mathbf{A}}, \tilde{\mathbf{b}})$, $pk' = (\mathbf{A}, \mathbf{b}')$ 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 [33], 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.

Acknowledgments. The first author is funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER). This work was supported by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute and the France 2030 ANR Project ANR-22-PETQ-0008 PQ-TLS.

References

1. Abou Haidar, C., Libert, B., Passelègue, A.: Updatable public key encryption from DCR: efficient constructions with stronger security. In: CCS (2022)
2. Agrawal, S., Gentry, C., Halevi, S., Sahai, A.: Sampling discrete gaussians efficiently and obliviously. In: ASIACRYPT (2013)

3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. ePrint 2015/046 (2015)
4. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: CRYPTO (2020)
5. Anderson, R.: Two remarks on public-key cryptology. In: CCS (1997), invited talk
6. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: CRYPTO (2009)
7. Asano, K., Watanabe, Y.: Updatable public key encryption with strong CCA security: Security analysis and efficient generic construction. Cryptology ePrint Archive 2023/976 (2023)
8. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: EUROCRYPT (1997)
9. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: CRYPTO (1999)
10. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: PKC (2011)
11. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: EuroS&P (2018)
12. Boudgoust, K., Jeudy, C., Roux-Langlois, A., Wen, W.: Entropic hardness of Module-LWE from Module-NTRU. In: Indocrypt (2022)
13. Boyen, X., Shacham, H., Shen, E., Waters, B.: Forward-secure signatures with untrusted update. In: CCS (2006)
14. Brakerski, Z., Döttling, N.: Hardness of LWE on general entropic distributions. In: EUROCRYPT (2020)
15. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012)
16. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehle, D.: Classical hardness of learning with errors. In: STOC (2013)
17. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: CRYPTO (2003)
18. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: EUROCRYPT (2003)
19. Cronin, E., Jamin, S., Malkin, T., McDaniel, P.: On the performance, feasibility, and use of forward-secure signatures. In: CCS (2003)
20. Dadush, D., Regev, O., Stephens-Davidowitz, N.: On the closest vector problem with a distance guarantee. In: CCC (2014)
21. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: ASIACRYPT (2010)
22. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: TCC (2021)
23. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G.: Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing. In: CCS (2021)
24. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: CRYPTO (1997)
25. Gentry, C., Peikert, C., Vaikunthanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC (2008)
26. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: ASIACRYPT (2002)

27. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: TCC (2017)
28. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: EUROCRYPT (2002)
29. Hövelmanns, K., Hülsing, A., Majenz, C.: Failing gracefully: Decryption failures and the fujisaki-okamoto transform. In: ASIACRYPT (2022)
30. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: CRYPTO (2001)
31. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: EUROCRYPT (2019)
32. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr. (2015)
33. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: CRYPTO (2022)
34. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: TCC (2010)
35. Malkin, T., Micciancio, D., Miner, S.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: EUROCRYPT (2002)
36. Mera, J.M.B., Karmakar, A., Marc, T., Soleimanian, A.: Efficient lattice-based inner-product functional encryption. In: PKC (2022)
37. Micciancio, D., Walter, M.: On the bit security of cryptographic primitives. In: EUROCRYPT (2018)
38. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC (1990)
39. O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption. In: CRYPTO (2011)

Appendix

A Non-Interactive Zero-Knowledge Argument

In this section, we recall the definition of NIZK arguments. We focus on definitions in the ROM, without a trusted setup, as these are the ones we aim to use for our instantiations.

Definition 15 (NIZK in the ROM). *Let $\mathcal{L} \subseteq \{0,1\}^*$ an NP language, defined by an efficient relation $\mathcal{R}$, such that $x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{q(|x|)} : R(x,w) = 1$, for some polynomial q. Let H denote a hash function, modeled as a random oracle. A non-interactive zero-knowledge argument in the ROM for $\mathcal{L}$ is a pair of PPT algorithms $(\mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$, such that:*

- $\mathsf{Prove}^{\mathsf{H}}(x, w)$: *on input a statement x and a witness w, returns a proof π;*
- $\mathsf{Verify}^{\mathsf{H}}(x, \pi)$: *on input a statement x, and a proof π, returns 0 or 1.*

We further require the following properties.

Completeness. *For all x, w such that $R(x, w) = 1$, we have:*

$$\mathsf{Verify}^{\mathsf{H}}(x, \mathsf{Prove}^{\mathsf{H}}(x, w)) = 1 \ .$$

Computational soundness. *For all $x \notin \mathcal{L}$, for all PPT adversary $\mathcal{A}$:*

$$\Pr[\mathsf{Verify}^{\mathsf{H}}(x, \pi^\star) = 1 \mid \pi^\star \leftarrow \mathcal{A}^{\mathsf{H}}(x)] \leq \mathsf{negl}(\lambda) \ ,$$

where the probability is over the choice of the random oracle. We let the above probability be denoted by $\mathsf{Adv}_{\Pi}^{\mathsf{sound}}(\mathcal{A})$.

Computational zero-knowledge. *There exists a PPT simulator Sim which can program the random oracle H values such that, for all x, w such that $R(x, w) = 1$, for all PPT adversary $\mathcal{A}$, we have:*

$$\mathsf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{A}) := \left| \Pr[\mathcal{A}^{\mathsf{H}}(x, \mathsf{Prove}^{\mathsf{H}}(x, w)) = 1] - \Pr[\mathcal{A}^{\mathsf{H}}(x, \mathsf{Sim}(x)) = 1] \right| \leq \mathsf{negl}(\lambda) \ .$$

B Complements on the IND-CR-CPA UPKE Scheme

B.1 Proof of Lemma 6

Proof. By [14, Lemma 5.1], we have

$$\mathsf{H}_\infty(\mathbf{s} \mid \mathbf{s} + \mathbf{e}) \geq \mathsf{H}_\infty(\mathbf{s}) - \log \left(\sum_{\mathbf{y} \in \mathbb{Z}_q^n} \max_{\mathbf{x} \in \mathbb{Z}_q^n} \mathbb{P}\left[\mathbf{e} = \mathbf{y} - \mathbf{x}\right] \right) .$$

Hence it suffices to show $\sum_{\mathbf{y}\in\mathbb{Z}_q^n} \max_{\mathbf{x}\in\mathbb{Z}_q^n} \mathbb{P}[\mathbf{e}=\mathbf{y}-\mathbf{x}] \leq 2q^n/\sigma^n$. It holds that,

$$\begin{aligned}\sum_{\mathbf{y}\in\mathbb{Z}_q^n} \max_{\mathbf{x}\in\mathbb{Z}_q^n} \mathbb{P}[\mathbf{e}=\mathbf{y}-\mathbf{x}] &= \frac{1}{\rho_\sigma(\mathbb{Z}^n)} \sum_{\mathbf{y}\in\mathbb{Z}_q^n} \max_{\mathbf{x}\in\mathbb{Z}_q^n} \rho_\sigma(\mathbf{y}-\mathbf{x}+q\mathbb{Z}^n) \\ &\leq \frac{1}{\rho_\sigma(\mathbb{Z}^n)} \sum_{\mathbf{y}\in\mathbb{Z}_q^n} 2 \\ &= 2\frac{q^n}{\rho_\sigma(\mathbb{Z}^n)} \\ &\leq 2\frac{q^n}{\sigma^n}.\end{aligned}$$

The second inequality comes from applying [14, Lemma 2.4], which requires $q/\sigma \geq \sqrt{\ln(4n)/\pi}$. The last equality comes from the Poisson summation formula which gives that $\rho_\sigma(\mathbb{Z}^n) = \sigma^n \rho_{1/\sigma}(\mathbb{Z}^n) \geq \sigma^n$. □

B.2 Correctness of the scheme

Proof. Let us prove the correctness of our UPKE. 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 $\|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leq ky\sigma$, for y a parameter that we set afterwards.

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

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

Then, we have

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

We obtain that $\mathsf{Dec}(\mathbf{s}, ct) = \mu$ 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 $\|\mathbf{Mv}\|_\infty \leq \|\mathbf{M}^T\|_1 \|\mathbf{v}\|_\infty \leq \sqrt{n}\|\mathbf{M}^T\|_2 \|\mathbf{v}\|_\infty$ 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:

$$\sqrt{n}\|\mathbf{X}^T\|_2\|\mathbf{e}\|_\infty + \sqrt{n}\|\mathbf{E}^T\|_2\|\mathbf{s}\|_\infty + \|\mathbf{f}\|_\infty < q/(2p). \tag{3}$$

If we assume that $\|\mathbf{X}^T\|_2, \|\mathbf{E}^T\|_2 < y\sqrt{n}\sigma$ and $\|\mathbf{f}\|_\infty < y\sigma$, for some $y > 0$, then (3) is verified if

$$q > 2p \cdot (2y^2\sigma^2 nk + y\sigma).$$

We bound the ℓ_2-norms using Lemma 2 and a union-bound, and the ℓ_∞-norms with Lemma 2 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

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

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 $p > 2y\sigma$. □

B.3 γ-Spreadness of the Scheme

We adapt the proof of [29, Lemma 6] for FrodoKEM to prove the following result.

Lemma 10. *The UPKE construction given in Section 4 is γ-spread.*

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

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

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

C Analysis of the FO Transform (Theorem 3)

The (k, δ)-correctness of $\mathsf{FO}(\mathsf{UPKE}, \mathsf{G}, \mathsf{H})$ 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, $\mathsf{UpdatePk}$ algorithms are unchanged.

In Figure 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.

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

$\mathsf{G}_0(pk, m)$:
- **if** $\exists r : (pk, m, r) \in \mathcal{L}_G$ **then**
  - **return** r
- **end**
- $r \hookleftarrow \mathcal{U}(\mathcal{R})$;
- $\mathcal{L}_G = \mathcal{L}_G \cup \{(pk, m, r)\}$;
- **return** r

$\mathsf{H}_0(m, c)$:
- **if** $\exists K : (m, c, K) \in \mathcal{L}_H$ **then**
  - **return** K
- **end**
- $K \hookleftarrow \mathcal{U}(\mathcal{K})$;
- $\mathcal{L}_H = \mathcal{L}_H \cup \{(m, c, K)\}$;
- **return** K

$\mathcal{O}_{dec,0}(c)$:
- **if** $c = c^\star \wedge pk_t = pk_{chall}$ **then**
  - **Abort**
- **end**
- **return** $\mathsf{Decaps}(sk_t, c)$

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 Figure 1.
- Game 1: In this game, we modify both the random oracles and the decapsulation oracle. We replace the oracles of Figure 1 by those in Figure 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 $\mathsf{G}(pk, m)$. 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.
- Game 2: In this game, the challenger additionally aborts if the adversary makes a query $\mathsf{G}(pk, m^\star)$ or $\mathsf{H}(m^\star, c)$ with $m^\star$ being the (uniformly random) message used to compute the challenge encapsulation $c^\star$, where pk and c are arbitrary. As the adversary $\mathcal{A}$ cannot learn $\mathsf{H}(m^\star, c^\star)$, 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 $m^\star$.

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

$\mathsf{G}_1(pk, m)$**:**
 if $\exists r : (pk, m, r) \in \mathcal{L}_G$ **then**
  return r
 end
 $r \hookleftarrow \mathcal{U}(\mathcal{R})$;
 $c = \mathsf{Enc}(pk, m; r)$;
 $\mathcal{L}_E = \mathcal{L}_E \cup \{(pk, m, r, c)\}$;
 $\mathcal{L}_G = \mathcal{L}_G \cup \{(pk, m, r)\}$;
 return r

$\mathsf{H}_1(m, c)$**:**
 if $\exists K : (m, c, K) \in \mathcal{L}_H$ **then**
  return K
 end
 $K \hookleftarrow \mathcal{U}(\mathcal{K})$;
 if $\exists pk, r : (pk, m, c, r) \in \mathcal{L}_E$ **then**
  $\mathcal{L}_D = \mathcal{L}_D \cup \{(pk, m, c, K)\}$;
 end
 $\mathcal{L}_H = \mathcal{L}_H \cup \{(m, c, K)\}$;
 return K

$\mathcal{O}_{dec,1}(c)$**:**
 if $c = c^\star \wedge pk_t = pk_{chall}$ **then**
  abort
 end
 if $\exists m, K : (pk_t, m, c, K) \in \mathcal{L}_D$ **then**
  return K
 end
 if $\exists m, r : (pk_t, m, c, r) \in \mathcal{L}_E$ **then**
  $K \hookleftarrow \mathcal{U}(\mathcal{K})$;
  $\mathcal{L}_H = \mathcal{L}_H \cup \{(m, c, K)\}$;
  $\mathcal{L}_D = \mathcal{L}_D \cup \{(pk_t, m, c, K)\}$;
  return K
 end
 return $\perp$

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[4] $(pk_t, m, r, c) \in \mathcal{L}_E$ such that $c = \mathsf{Enc}(pk_t, m; r)$, where $r = \mathsf{G}(pk_t, m)$. 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 $\mathsf{G}(pk_t, m)$ 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 $(pk_t, m, r, c) \in \mathcal{L}_E$ such that

[4] 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 $(pk_t, m, r, c) \in \mathcal{L}_E$.

$c = \mathsf{Enc}(pk_t, m; r)$, and as $\mathcal{O}_{dec,0}(c) \neq \perp$, this is a valid encryption. Hence $K = \mathsf{H}_0(m, c)$. 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 $\mathsf{H}_1(m, c) = K'$. Thus K' has the same distribution as K and H_1 has the same behaviour as H_0.

(b) Adversary $\mathcal{A}$ already knows $\mathsf{H}(m, c)$ 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 $\mathsf{H}_1(m, c)$ guarantees that $\mathcal{O}_{dec,1}(c)$ returns K', which has the same distribution as K and H_1 behaves identically to H_0.

We just showed that except with probability at most $q_{\mathsf{G}} \cdot \delta + q_{\mathsf{D}} \cdot \gamma$, 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 $\mathsf{G}(pk, m^\star)$ or $\mathsf{H}(m^\star, c)$ with $m^\star$ being the (uniformly random) message used to compute the challenge encapsulation $c^\star$, 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 $\mathsf{IND\text{-}CR\text{-}CPA}$ game such that

$$\mathbb{P}[\mathsf{FIND}] \leq 2 \left(\mathsf{Adv}^{\mathsf{IND\text{-}CR\text{-}CPA}}(\mathcal{B}) + \frac{q_{\mathsf{G}} + q_{\mathsf{H}}}{|\mathcal{M}|} \right) . \tag{4}$$

Adversary $\mathcal{B}$ first receives pk_0 from its $\mathsf{IND\text{-}CR\text{-}CPA}$ 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 $\mathsf{IND\text{-}CR\text{-}CPA}$ challenge $c^\star$. Adversary $\mathcal{B}$ samples $K^\star \hookleftarrow \mathcal{U}(\mathcal{K})$ and sends the challenge $(K^\star, c^\star)$ to $\mathcal{A}$.

From now, adversary $\mathcal{B}$ continues to simulate $\mathcal{A}$'s challenger. If $\mathcal{A}$ makes a query $\mathsf{G}(pk, m_{b'})$ or $\mathsf{H}(m_{b'}, c)$ 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 $\mathbb{P}[\mathsf{WRG}] \leq (q_\mathsf{G} + q_\mathsf{H})/|\mathcal{M}|$. Then:

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

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 $\neg\mathsf{WRG}$ 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 $\mathbb{P}[\mathsf{A} \wedge \mathsf{B}] = \mathbb{P}[\mathsf{A}] - \mathbb{P}[\mathsf{A} \wedge \neg\mathsf{B}]$. Equation (4) then follows, which completes the proof of Theorem 3. □

D Analysis of the IND-CU-CCA Transform (Theorem 4)

We proceed by a sequence of hybrid games.

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

Game 1: We replace the proof $\pi^\star$ in the final update $up^\star = (\mathsf{ct}_0^\star, \mathsf{ct}_1^\star, \pi^\star)$ 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 $\mathsf{ct}_1^\star$ 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 $\mathsf{IND\text{-}CR\text{-}CCA}$ 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 $\mathsf{VerifyUpdate}$, the challenger does the following. Let $((\mathsf{ct}_0, \mathsf{ct}_1, \pi), (pk_0', pk_1))$ 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 $\mathsf{IND\text{-}CR\text{-}CCA}$ 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 $\mathsf{IND\text{-}CR\text{-}CCA}$ security of UKEM as follows. Adversary $\mathcal{B}$ gets pk_0 from its challenger and further samples an additional key pair $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$. 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 $((\mathsf{ct}_0, \mathsf{ct}_1, \pi), (pk'_0, pk_1))$, 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 $c^\star$ to the former.

Finally, when $\mathcal{A}$ stops making updates, so does $\mathcal{B}$. Its challenger then replies by $(pk^\star, sk^\star, up^\star)$, where $up^\star$ is simply an encryption $\mathsf{ct}_0^\star$ of the last (unknown) update under the last epoch public key pk_{last}^ℓ. It generates an encryption $\mathsf{ct}_1^\star$ of 0 under pk_1, as well as a simulated proof $\pi^\star$ that $(pk_{last}^\ell, pk_1, pk^\star, \mathsf{ct}_0^\star, \mathsf{ct}_1^\star)$ is a valid update. It finally sends the tuple $((pk^\star, pk_1), sk^\star, \mathsf{ct}_0^\star, \mathsf{ct}_1^\star, \pi^\star)$ to $\mathcal{A}$. When $\mathcal{A}$ halts with output b', so does $\mathcal{B}$.

This completes the proof of Theorem 4. $\square$