The Direction of Updatable Encryption does not Matter Much

Yao Jiang

Norwegian University of Science and Technology, NTNU, Trondheim, Norway

Abstract. Updatable encryption schemes allow for key rotation on ciphertexts. A client outsourcing storage of encrypted data to a cloud server can change its encryption key. The cloud server can update the stored ciphertexts to the new key using only a token provided by the client.
This paper solves two open problems in updatable encryption, that of uni-directional vs. bi-directional updates, and post-quantum security.
The main result in this paper is to analyze the security notions based on uni- and bi-directional updates. Surprisingly, we prove that uni- and bi-directional variants of each security notion are equivalent.
The second result in this paper is to provide a new and efficient updatable encryption scheme based on the Decisional Learning with Error assumption. This gives us post-quantum security. Our scheme is bi-directional, but because of our main result, this is sufficient.

Keywords: Updatable Encryption · Cloud Storage · Key Rotation · Lattice-based cryptography · Post-quantum cryptography

1 Introduction

Consider the following scenario: a client wishes to outsource data to a cloud storage provider with a cryptoperiod (client key lifetime). The cryptoperiod is decided by the client or the cloud storage provider or both. If the key lifetime is expired, the old key is no longer available for either encryption or decryption, a new key must be used in the new cryptoperiod. However, the client might still want to keep the data in the cloud storage in the new cryptoperiod and needs to update the data. The above requirement implies a need to update ciphertexts from the old key to the new key. During this process, it is also reasonable to expect that no information of plaintexts are leaked while updating. Another benefit to consider in such a scenario is that it can be used to protect the data and reduce the risk of key compromise over time.

Key rotation is the process of generating a new key and altering ciphertexts from the old key to the new key without changing the underlying massage.

Key rotation can be done by downloading the old ciphertext, decrypting with the old key, re-encrypting with a new key and reuploading the new ciphertext. However, this is expensive. *Updatable encryption* (UE) [5, 6, 8, 11, 14, 15] provides a better solution for key rotation. A client generates an *update token* and

sends it to the cloud server, the cloud server can use this update token to update the ciphertexts from the old key to the new key. In recent years there has been considerable interest in understanding UE, including defining the security notions for UE and constructing UE schemes (we make a detailed comparison of related work in Sect. 1.1).

Consider the following two variants of UE schemes: *ciphertext-dependent* schemes and *ciphertext-independent* schemes. If the generation of update token depends on the ciphertext to be updated then the UE scheme is ciphertext-dependent. In ciphertext-dependent schemes, the updating process of a ciphertext requires a specific token which forces the client to download the old ciphertext before this token can be generated. Therefore, ciphertext-dependent schemes are less practical. If the token is independent of the old ciphertext then the UE scheme is ciphertext-independent. Hence, a single token can be used to update all ciphertexts a client owns. As ciphertext-independent schemes are considerably more efficient than ciphertext-dependent schemes, in terms of bandwidth, most recent works [7, 8, 14, 15] focus on ciphertext-independent schemes. In this paper, we will focus on such schemes.

Consider the following four variants of updates for ciphertext-independent UE schemes: *uni-directional ciphertext* updates, *bi-directional ciphertext* updates, *uni-directional key* updates and *bi-directional key* updates. If the update token can only move ciphertexts from the old key to the new key then ciphertext updates in such UE schemes are uni-directional. If the update token can additionally downgrade ciphertexts from the new key to the old key then ciphertext updates in such UE schemes are bi-directional. On the other hand, the update token can potentially be used to derive keys from other keys. In the uni-directional key update setting, the update token can only infer the new key from the old key. While in the bi-directional key update setting, the update token can both upgrade and downgrade keys. Prior works [7, 8, 14, 15] focus on UE schemes with bi-directional updates, and no security notion was introduced in uni-directional update setting. We close this gap. Intuitively, UE schemes with uni-directional updates are desirable, such schemes leak less ciphertext/key information to an adversary compared to schemes with bi-directional updates. In this paper, we analyze the relationship between security notions with uni- and bi-directional updates. We show that the (confidentiality and integrity) security of UE schemes are not influenced by uni- or bi-directional updates.

No-directional key updates is another key update setting to consider, where the update token cannot be used to derive keys. A UE scheme with optimal leakage, discussed in [15], is a scheme where no token inference (no token can be inferred via keys), keys cannot be updated via a token, and ciphertext updates are only uni-directional. We do not consider no token inference, instead in this work an update token can be computed via two consecutive epoch keys. We show that the no-directional key update variant of a security notion is strictly stronger than the uni- and bi-directional update variant of the same security notion.

While the study of security notions appears promising, existing ciphertext-independent UE schemes are either vulnerable to quantum computers or only achieve weak security. The schemes of Lehmann and Tackmann [15], Klooß et al. [14] and Boyd et al. [8] base their security on the DDH problem, and thus are only secure in the classical setting. Boneh et al. [6] constructed key homomorphic PRFs, based on the learning with errors (LWE) problem, and it can be used to construct UE schemes. However, all of these schemes of Boneh et al. [6] cannot achieve IND-UPD security (introduced in [15]).

In this work, we construct a post-quantum secure UE scheme and the security of our construction is based on hard lattice problems. In particular, our scheme provides the randIND-UE-CPA security (introduced in [8], stronger than IND-UPD and IND-ENC security).

Efficiency. All of the previous known ciphertext-independent UE schemes with security proofs (RISE, E&M, NYUE (NYUAE), SHINE) have computation cost that are comparable to PKE schemes that rely on the DDH problem, while our scheme has a computation cost that is comparable to PKE schemes that rely on lattice problems.

1.1 Related Work

Security Notions. Boneh et al. [6] introduced a security definition for UE, however, this notion is less adaptive than the later works [8, 14, 15] which allows the adversary to adaptively corrupt epoch keys and update tokens at any point in the game.

In the ciphertext-dependent setting, Everspaugh et al. [11] provided two security notions, a weak form of ciphertext integrity and re-encryption indistinguishability, that strengthen the security notion in [6]. Recently, Boneh et al. [5] introduced new definitions for updatable encryption in the ciphertext-dependent setting to further strengthen the confidentiality property and the integrity definition in [11]. Boneh et al. [5] stated that for authenticated updatable encryption schemes it is necessary to expect that ciphertexts will not reveal how many times they have been updated, which was a desired property independently presented in [8].

Lehmann and Tackmann [15] introduced two notions to achieve CPA security for ciphertext-independent UE schemes. Their IND-ENC notion requires that ciphertexts output by the encryption algorithm are indistinguishable from each other. Their IND-UPD notion ensures ciphertexts output by the update algorithm are indistinguishable from each other.

Klooß et al. [14] attempted to provide stronger security notions for ciphertext-independent UE than LT18, specifically, CCA security and integrity protection.

Boyd et al. [8] provided a new notion IND-UE which states that a ciphertext output by the encryption algorithm is indistinguishable from a ciphertext output by the update algorithm. They showed that the new notion is strictly stronger than any combinations of prior notions, both under CPA and CCA. They also

tweaked the CTXT and CCA notions in [14] and showed the following generic composition result: CPA + CTXT $\implies$ CCA.

Constructing Ciphertext-Independent Updatable Encryption Schemes. The UE scheme BLMR in [6] is an application of key homomorphic PRFs, however, the encrypted nonce in the ciphertext can be decrypted by an update token which makes it impossible for BLMR to achieve IND-UPD security.

In the classical setting, RISE in [15] is built from (public-key) ElGamal encryption, which only uses the public key in the update token. The security of RISE is based on the DDH assumption. Klooß et al. [14] provided two generic constructions, based on encrypt-and-MAC (E&M) and the Naor-Yung paradigm (NYUE and NYUAE). The security of E&M is based on the DDH assumption, and the security of NYUE and NYUAE are based on the SXDH assumption. Boyd et al. [8] constructed three permutation-based UE schemes, SHINE, which achieves strong security notions based on DDH.

Post-Quantum Secure Schemes. In the past decade, much work has been done on constructing lattice-based post-quantum secure PKE schemes, specifically the NIST Post-Quantum Standardization Project, round 2, submissions: CRYSTALS-KYBER [3], FrodoKEM [1], LAC [16], NewHope [2], NTRU [4, 9], Round5 [18], SABER [10] and Three Bears [12]. A natural question is if we can turn a PKE scheme into a UE scheme, where the security of the UE follows from the PKE. We provide a specific UE scheme that is built form an LWE-based PKE scheme, and prove the security. The LWE-based scheme we use is in some sense very similar to RISE (which is based on ElGamal), however, as with most lattice-based constructions, there are significant technical problems in turning it into a UE scheme (see Sect. 5.2). Our LWE-based UE construction suggests that there is a limit to how generic any efficient construction can be, a generic construction that abstracts both our construction and RISE remains to be done.

1.2 Our Contributions

Our first contribution is defining six variants of security notions (a combination of three versions of key updates and two versions of ciphertext updates) for updatable encryption and analyzing the relations among these six variants of the same notion.

Our main result is that we demonstrate that our security notions with uni- and bi-directional updates are equivalent. When we analyze the security, we can treat UE schemes with uni-directional updates as with bi-directional updates, the security will not be influenced by the update direction. This means that UE schemes with uni-directional updates will not provide more security than UE schemes with bi-directional updates. This is a surprising result.[1] This result

[1] It is possible to construct a scenario where this result will not be true. Let's assume there exists a UE scheme with a leakage function that helps the adversary win the security game. This leakage function could, for example, give the adversary

implies that the search for uni-directional updatable encryption scheme seems less important.

Furthermore, we show that security notions with no-directional key updates are strictly stronger than uni- and bi- directional update variants of the corresponding notions. Finding UE schemes with no-directional key updates would be good, but it is much more challenge than finding UE schemes with uni-directional key updates (which is already believed to be difficult). We leave this as an open problem.

Our second major contribution is constructing an efficient post-quantum secure UE scheme. We analyze how to construct LWE-based updatable encryption schemes and provide one construction. Our construction follows the re-randomization idea of RISE, using public key in the update token to update ciphertexts. We build a suitable post-quantum secure PKE scheme to construct our UE scheme so that the encryption and update algorithms can use a public key as input instead of the secret key. We also show the difficulties of turning a PKE scheme into a UE scheme.

We show that our LWE-based UE scheme is randIND-UE-CPA secure under the DLWE assumption. In the randomized update setting, we show the difference between previous work (RISE, NYUE, NYUAE) and our scheme, and state that the method used in proving the security of LWE-based updatable encryption scheme is different from the previous approach.

1.3 Open Problems

Ideally we want UE schemes with no-directional key updates, no such UE schemes have been constructed so far. Whether such UE schemes exist and how to construct such UE schemes are still open problems.

Furthermore, not that many efficient UE schemes with strong security exist so far. It remains an open challenge to construct UE schemes with chosen ciphertext[2] post-quantum security.

---

information about plaintexts when it knows enough keys. In this scenario, a UE scheme with uni-directional updates has better security than a UE scheme with bi-directional updates. Because the scheme with uni-directional updates has less key leakage and the leakage function provides less data to the adversary. However, this and similar constructions cannot capture the security we wish to have for UE schemes. In terms of the security expectation of key rotation, the keys used in the past should not reveal any data.

For constructions that do follow the security model and update mechanism for UE schemes, we have this surprising result.

[2] It is ideal to achieve detIND-UE-CCA security for UE schemes with deterministic updates and to achieve INT-PTXT and randIND-UE-CCA security for UE schemes with randomized updates.

2 Preliminaries

In this section we describe the notation used in this paper and present the necessary background material of updatable encryption. In the full version [13], we provide the real or random variant of indistinguishability under chosen-plaintext attack (IND$-CPA) for encryption schemes and the background of hard lattice problems.

2.1 Notations

Let λ be the security parameter throughout the paper. Let negl denote as a negligible function. Let $\mathcal{U}(S)$ denote the uniform distribution over set S.

2.2 Updatable Encryption.

Updatable encryption (UE) scheme is parameterized by a tuple of algorithms $\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ that operate in epochs, the epoch starts at 0. The key generation algorithm $\mathsf{UE.KG}$ outputs an epoch key $\mathbf{k}_\mathsf{e}$. The token generation algorithm $\mathsf{UE.TG}$ takes as input two epoch keys $\mathbf{k}_\mathsf{e}$ and $\mathbf{k}_{\mathsf{e}+1}$ and outputs an update token $\Delta_{\mathsf{e}+1}$, the update token can be used to move ciphertexts from epoch e to $\mathsf{e}+1$. The encryption algorithm $\mathsf{UE.Enc}$ takes as input an epoch key $\mathbf{k}_\mathsf{e}$ and a message $\mathbf{m}$ and outputs a ciphertext $\mathbf{c}_\mathsf{e}$. The decryption algorithm $\mathsf{UE.Dec}$ takes as input an epoch key $\mathbf{k}_\mathsf{e}$ and a ciphertext $\mathbf{c}_\mathsf{e}$ and outputs a message $\mathbf{m}'$. The update algorithm $\mathsf{UE.Upd}$ takes as input an update token $\Delta_{\mathsf{e}+1}$ and a ciphertext $\mathbf{c}_\mathsf{e}$ from epoch e and outputs an updated ciphertext $\mathbf{c}_{\mathsf{e}+1}$.

We stress that an update token can be computed via two consecutive epoch keys by token generation algorithm in this paper.

2.3 Existing Security Notions for Updatable Encryption

Klooß et al. [14] and Boyd et al. [8] defined the confidentiality and the integrity notions for updatable encryption schemes using experiments that are running between an adversary and a challenger. In each experiment, the adversary may send a number of oracle queries. The main differences between an experiment running the confidentiality game and one running the integrity game are the challenge and win condition. In the confidentiality game, the adversary tries to distinguish a fresh encryption from an updated ciphertext. In the integrity game, the adversary attempts to provide a valid forgery. At the end of an experiment the challenger evaluates whether or not the adversary wins, if a trivial win condition was triggered the adversary will always lose.

We follow the notation of security notions from Boyd et al. [8]. An overview of the oracles the adversary has access to in each security game is given in Fig. 1. A generic description of all confidentiality experiments and integrity experiments described in this paper is detailed in Fig. 2 and Fig. 3, resp.. Our oracle algorithms, see Fig. 4, are stated differently than in [8] and [14], however, conceptually they are the same. The oracles we use in our security games

Notions	$\mathcal{O}$.Enc	$\mathcal{O}$.Dec	$\mathcal{O}$.Next	$\mathcal{O}$.Upd	$\mathcal{O}$.Corr	$\mathcal{O}$.Chall	$\mathcal{O}$.Upd$\tilde{\mathsf{C}}$	$\mathcal{O}$.Try
detIND-UE-CPA	✓	×	✓	✓	✓	✓	✓	×
randIND-UE-CPA	✓	×	✓	✓	✓	✓	✓	×
detIND-UE-CCA	✓	✓	✓	✓	✓	✓	✓	×
randIND-UE-CCA	✓	✓	✓	✓	✓	✓	✓	×
INT-CTXT	✓	×	✓	✓	✓	×	×	✓
INT-PTXT	✓	×	✓	✓	✓	×	×	✓

Fig. 1: Oracles given to the adversary in different security games for updatable encryption schemes. × indicates the adversary does not have access to the corresponding oracle, ✓ indicates the adversary has access to the corresponding oracle.

are as follows, encrypt $\mathcal{O}$.Enc, decrypt $\mathcal{O}$.Dec, move to the next epoch $\mathcal{O}$.Next, update ciphertext $\mathcal{O}$.Upd, corrupt key or token $\mathcal{O}$.Corr, ask for the challenge ciphertext $\mathcal{O}$.Chall, get an updated version of the challenge ciphertext $\mathcal{O}$.Upd$\tilde{\mathsf{C}}$, or test if a ciphertext is a valid forgery $\mathcal{O}$.Try. The detailed discussion of trivial win conditions are discussed in Sect. 2.6.

For the confidentiality game we have the following additional definitions that we will frequently use. While the security game is running, the adversary may query $\mathcal{O}$.Enc or $\mathcal{O}$.Upd oracles or corrupt tokens to know some (updated) versions of ciphertexts, we call them *non-challenge ciphertexts*. In addition, the adversary may query $\mathcal{O}$.Chall or $\mathcal{O}$.Upd$\tilde{\mathsf{C}}$ oracles or corrupt tokens to infer some (updated) versions of the challenge ciphertext, we call them *challenge-equal ciphertexts*.

Definition 1. *Let* $\mathsf{UE}=\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an updatable encryption scheme. Then the* notion *advantage, for* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$*, of an adversary* $\mathcal{A}$ *against* UE *is defined as*

$$\mathbf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \Big|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion}\text{-}1}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion}\text{-}0}_{\mathsf{UE},\,\mathcal{A}} = 1]\Big|,$$

where the experiment $\mathbf{Exp}^{\mathsf{notion}\text{-}\mathrm{b}}_{\mathsf{UE},\,\mathcal{A}}$ *is given in Fig. 2 and Fig. 4.*

Definition 2. *Let* $\mathsf{UE}=\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an updatable encryption scheme. Then the* notion *advantage, for* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$*, of an adversary* $\mathcal{A}$ *against* UE *is defined as*

$$\mathbf{Adv}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}} = 1],$$

where the experiment $\mathbf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}$ *is given in Fig. 3 and Fig. 4.*

2.4 Notations of the Leakage Sets

In this section, we describe the definition of leakage sets given by [15] and [14], these sets will later be used to check whether the leaked information will allow the adversary trivially win the security game. We analyze some properties of leakage sets and trivial win conditions in Sect. 3.1.

$\underline{\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}}$:
do Setup; $\mathsf{phase} \leftarrow 0$
$\mathrm{b}' \leftarrow \mathcal{A}^{oracles}(1^\lambda)$
if $\Big((\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset)$ **or** $\big(\mathsf{xx}{=}\mathsf{det}$ **and**
$(\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathsf{c}})$ is queried$)\big)\Big)$ **then**
$\mathsf{twf} \leftarrow 1$
if $\mathsf{twf} = 1$ **then**
$\mathrm{b}' \xleftarrow{\$} \{0,1\}$
return b'

Fig. 2: Generic description of the confidentiality experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ for updatable encryption scheme UE and adversary $\mathcal{A}$, for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. The flag phase tracks whether or not $\mathcal{A}$ has queried the $\mathcal{O}.\mathsf{Chall}$ oracle, $\tilde{\mathsf{e}}$ denotes the epoch in which the $\mathcal{O}.\mathsf{Chall}$ oracle happens, and twf tracks if the trivial win conditions are triggered. Fig. 1 shows the oracles the adversary have access to in a specific security game. How to compute the leakage sets $\mathcal{K}^*, \mathcal{T}^*, \mathcal{C}^*$ are discussed in Sect. 2.5.

$\underline{\mathbf{Exp}^{\mathsf{INT\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}}$
do Setup; $\mathsf{win} \leftarrow 0$
$\mathcal{A}^{oracles}(1^\lambda)$
if $\mathsf{twf} = 1$ **then**
$\mathsf{win} \leftarrow 0$
return win

Fig. 3: Generic description of the integrity experiment $\mathbf{Exp}^{\mathsf{INT\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}$ for updatable encryption scheme UE and adversary $\mathcal{A}$, for $\mathsf{atk} \in \{\mathsf{CTXT}, \mathsf{PTXT}\}$. The flag win tracks whether or not the adversary provided a valid forgery and twf tracks if the trivial win conditions are triggered. Fig. 1 shows the oracles the adversary have access to in a specific security game.

Epoch Leakage Sets. We use the following sets that track epochs in which the adversary corrupted a key or a token, or learned a version of challenge-ciphertext.

- $\mathcal{K}$: Set of epochs in which the adversary corrupted the epoch key (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{T}$: Set of epochs in which the adversary corrupted the update token (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{C}$: Set of epochs in which the adversary learned a challenge-equal ciphertext (from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$).

We use $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ as the extended sets of $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{C}$ in which the adversary has learned or inferred information via its known tokens. We show how to compute $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ in Sect. 2.5.

Information Leakage Sets. We use the following sets to track ciphertexts and their updates that can be known to the adversary.

- $\mathcal{L}$: Set of non-challenge ciphertexts $(\mathrm{c}, \mathbf{c}, \mathsf{e}; \mathbf{m})$, where query identifier c is a counter incremented with each new $\mathcal{O}.\mathsf{Enc}$ query. The adversary learned these ciphertexts from $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$.
- $\tilde{\mathcal{L}}$: Set of challenge-equal ciphertexts $(\tilde{\mathbf{c}}_\mathsf{e}, \mathsf{e})$. The adversary learned these ciphertexts from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$.

$\underline{\mathbf{Setup}(1^\lambda)}$
$\mathbf{k}_0 \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
$\Delta_0 \leftarrow \perp; \mathsf{e}, \mathrm{c}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\underline{\mathcal{O}.\mathsf{Enc}(\mathbf{m}) :}$
$\mathrm{c} \leftarrow \mathrm{c} + 1$
$\mathbf{c} \xleftarrow{\$} \mathsf{UE.Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}, \mathsf{e}; \mathbf{m})\}$
return $\mathbf{c}$

$\underline{\mathcal{O}.\mathsf{Dec}(\mathbf{c}) :}$
$\mathbf{m}'$ **or** $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c})$
if $\Big((\mathsf{xx} = \mathsf{det}$ **and** $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*)$ **or**
$(\mathsf{xx} = \mathsf{rand}$ **and** $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*)\Big)$ **then**
$\mathsf{twf} \leftarrow 1$
return $\mathbf{m}'$ **or** $\perp$

$\underline{\mathcal{O}.\mathsf{Next}() :}$
$\mathsf{e} \leftarrow \mathsf{e} + 1$
$\mathbf{k}_\mathsf{e} \xleftarrow{\$} \mathsf{UE.KG}(1^n)$
$\Delta_\mathsf{e} \leftarrow \mathsf{UE.TG}(\mathbf{k}_{\mathsf{e}-1}, \mathbf{k}_\mathsf{e})$
if $\mathsf{phase} = 1$ **then**
$\tilde{\mathbf{c}}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \tilde{\mathbf{c}}_{\mathsf{e}-1})$

$\underline{\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}-1}) :}$
if $(j, \mathbf{c}_{\mathsf{e}-1}, \mathsf{e}-1; \mathbf{m}) \notin \mathcal{L}$ **then**
return $\perp$
$\mathbf{c}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathbf{c}_{\mathsf{e}-1})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \mathbf{c}_\mathsf{e}, \mathsf{e}; \mathbf{m})\}$
return $\mathbf{c}_\mathsf{e}$

$\underline{\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}}) :}$
if $\hat{\mathsf{e}} > \mathsf{e}$ **then**
return $\perp$
if $\mathsf{inp} = \mathsf{key}$ **then**
$\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
return $\mathbf{k}_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$ **then**
$\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
return $\Delta_{\hat{\mathsf{e}}}$

$\underline{\mathcal{O}.\mathsf{Chall}(\bar{\mathbf{m}}, \bar{\mathbf{c}}) :}$
if $\mathsf{phase} = 1$ **then**
return $\perp$
$\mathsf{phase} \leftarrow 1; \tilde{\mathsf{e}} \leftarrow \mathsf{e}$
if $(\cdot, \bar{\mathbf{c}}, \tilde{\mathsf{e}} - 1; \bar{\mathbf{m}}_1) \notin \mathcal{L}$ **then**
return $\perp$
if $\mathrm{b} = 0$ **then**
$\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathbf{k}_{\tilde{\mathsf{e}}}, \bar{\mathbf{m}})$
else
$\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathbf{c}})$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{\mathsf{e}}\}$
$\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}, \tilde{\mathsf{e}})\}$
return $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$

$\underline{\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}} :}$
if $\mathsf{phase} \neq 1$ **then**
return $\perp$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
$\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_\mathsf{e}, \mathsf{e})\}$
return $\tilde{\mathbf{c}}_\mathsf{e}$

$\underline{\mathcal{O}.\mathsf{Try}(\tilde{\mathbf{c}}) :}$
$\mathbf{m}'$ **or** $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \tilde{\mathbf{c}})$
if $\Big((\mathsf{atk} = \mathsf{CTXT}$ **and** $(\tilde{\mathbf{c}}, \mathsf{e}) \in \mathcal{L}^*)$ **or**
$(\mathsf{atk} = \mathsf{PTXT}$ **and** $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*)$ **or**
$\mathsf{e} \in \mathcal{K}^*\Big)$ **then**
$\mathsf{twf} \leftarrow 1$
if $\mathbf{m}' \neq \perp$ **then**
$\mathsf{win} \leftarrow 1$

Fig. 4: Oracles in security games for updatable encryption. How to compute the leakage sets $\mathcal{K}^*, \mathcal{T}^*, \mathcal{C}^*, \tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$ are discussed in Sect. 2.5 and Sect. 2.6.

In the deterministic update setting, we use $\mathcal{L}^*$ and $\tilde{\mathcal{L}}^*$ as the extended (ciphertext) sets of $\mathcal{L}$ and $\tilde{\mathcal{L}}$ in which the adversary has learned or inferred ciphertexts via its known tokens. In particular, we only use partial information of $\mathcal{L}^*$: the ciphertext and the epoch. Hence, we only track the set $\mathcal{L}^* = \{(\mathbf{c}, \mathsf{e})\}$.

In the randomized update setting, we use $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ as the extended (plaintext) sets of $\mathcal{L}$ and $\tilde{\mathcal{L}}$, that contain messages that the adversary can provide a ciphertext of - i.e. a forgery. Similarly, only partial information is needed: the plaintext and the epoch. Hence, we track sets $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ as follows.

- $\mathcal{Q}^*$: Set of plaintexts $(\mathbf{m}, \mathsf{e})$. The adversary learned or was able to create a ciphertext in epoch e with the underlying message $\mathbf{m}$.
- $\tilde{\mathcal{Q}}^*$: Set of challenge plaintexts $\{(\bar{\mathbf{m}}, \mathsf{e}), (\bar{\mathbf{m}}_1, \mathsf{e})\}$, where $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ is the input of challenge query $\mathcal{O}.\mathsf{Chall}$ and $\bar{\mathbf{m}}_1$ is the underlying message of $\bar{\mathbf{c}}$. The adversary learned or was able to create a challenge-equal ciphertext in epoch e with the underlying message $\bar{\mathbf{m}}$ or $\bar{\mathbf{m}}_1$.

Remark 1. Based on the definition of these sets, we observe that

a. $(\tilde{\mathbf{c}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}} \iff \mathsf{e} \in \mathcal{C}$,
b. $(\tilde{\mathbf{c}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^* \iff (\bar{\mathbf{m}}, \mathsf{e}), (\bar{\mathbf{m}}_1, \mathsf{e}) \in \tilde{\mathcal{Q}}^*$.

We will use this remark to discuss how to compute $\mathcal{L}^*$, $\tilde{\mathcal{L}}^*$, $\mathcal{Q}^*$ and $\tilde{\mathcal{Q}}^*$ in Sect. 2.6.

2.5 Epoch Leakage Sets of Keys, Tokens and Ciphertexts

We follow the bookkeeping techniques and base our notations of the work of Lehmann and Tackmann [15], where we further analyze the epoch leakage sets. Specifically, we add a no-directional key update setting. Suppose a security game ends at epoch l, then, for any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., l\}$, the following algorithms show how to compute the extended sets $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ in different update settings.

Key Leakage. The adversary learned all keys in epochs in $\mathcal{K}$. In the no-directional key update setting, the adversary does not have more information about keys except for this set. In the uni-directional key update setting, if the adversary knows a key $\mathbf{k}_\mathsf{e}$ and an update token $\Delta_{\mathsf{e}+1}$ then it can infer the next key $\mathbf{k}_{\mathsf{e}+1}$. In the bi-directional key update setting, the adversary can additionally downgrade a key by a known token. In the kk-directional key update setting, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, we denote the set $\mathcal{K}^*_\mathsf{kk}$ as the extended set of corrupted key epochs. We compute these sets as follows.

No-directional key updates: $\mathcal{K}^*_\mathsf{no} = \mathcal{K}$.

Uni-directional key updates:

$$\begin{aligned} &\mathcal{K}^*_\mathsf{uni} \leftarrow \{\mathsf{e} \in \{0, ..., l\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\} \\ &\quad \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \iff (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}). \end{aligned} \tag{1}$$

Bi-directional key updates:

$$\begin{aligned}&\mathcal{K}^*_{\mathsf{bi}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}\\ &\quad \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \iff\\ &\qquad (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}{+}1) \wedge \mathsf{e}{+}1 \in \mathcal{T}).\end{aligned} \tag{2}$$

Token Leakage. A token is known to the adversary is either a corrupted token or a token inferred from two consecutive epoch keys, so the extended set of corrupted token epochs is computed by information in set $\mathcal{T}$ and set $\mathcal{K}^*_{\mathsf{kk}}$. The set $\mathcal{K}^*_{\mathsf{kk}}$ is computed as above depending on the key updates is no- or uni- or bi-directional. Hence, we denote $\mathcal{T}^*_{\mathsf{kk}}$ as the extended set of corrupted token epochs.

$$\mathcal{T}^*_{\mathsf{kk}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{kk}} \wedge \mathsf{e}\text{-}1 \in \mathcal{K}^*_{\mathsf{kk}})\}. \tag{3}$$

Challenge-Equal Ciphertext Leakage. The adversary learned all challenge-equal ciphertexts in epochs in $\mathcal{C}$. Additionally, the adversary can infer challenge-equal ciphertexts via tokens. In the uni-directional ciphertext update setting, the adversary can upgrade ciphertexts. In the bi-directional ciphertext update setting, the adversary can additionally downgrade ciphertexts.

We compute the extended set of challenge-equal epochs using the information contained in $\mathcal{C}$ and $\mathcal{T}^*_{\mathsf{kk}}$. The set $\mathcal{T}^*_{\mathsf{kk}}$ is computed as above depending on the key updates is no- or uni- or bi-directional. In the cc-directional ciphertext update setting, for $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, denote the set $\mathcal{C}^*_{\mathsf{kk},\mathsf{cc}}$ as the extended set of challenge-equal epochs. We compute these sets as follows.

Uni-directional ciphertext updates:

$$\begin{aligned}&\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}\\ &\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}).\end{aligned} \tag{4}$$

Bi-directional ciphertext updates:

$$\begin{aligned}&\mathcal{C}^*_{\mathsf{kk},\mathsf{bi}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}\\ &\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff\\ &\qquad (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}) \vee (\mathsf{ChallEq}(\mathsf{e}{+}1) \wedge \mathsf{e}{+}1 \in \mathcal{T}^*_{\mathsf{kk}}).\end{aligned} \tag{5}$$

2.6 Trivial Win Conditions

The main benefit of using ciphertext-independent updatable encryption scheme is that it offers an efficient way for key rotation, where a single token can be used to update all ciphertexts. However, this property provides the adversary more power, the tokens can be used to gain more information, and gives the adversary more chances to win the security games. We again follow the trivial win analysis in [8, 14, 15] and exclude these trivial win conditions in the security games for UE. An overview of the trivial win conditions the challenger will check in each security game is given in Fig. 5.

Notions	"$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$"	"$\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried"	"$(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$"	"$(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$"	"$\mathsf{e} \in \mathcal{K}^*$"	"$(\bar{\mathbf{c}}, \mathsf{e}) \in \mathcal{L}^*$"	"$(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$"
detIND-UE-CPA	✓	✓	×	×	×	×	×
randIND-UE-CPA	✓	×	×	×	×	×	×
detIND-UE-CCA	✓	✓	✓	×	×	×	×
randIND-UE-CCA	✓	×	×	✓	×	×	×
INT-CTXT	×	×	×	×	✓	✓	×
INT-PTXT	×	×	×	×	✓	×	✓

Fig. 5: Trivial win conditions considered in different security games for updatable encryption schemes. × indicates the security notion does not consider the corresponding trivial win condition, ✓ indicates the security notion considers the corresponding trivial win condition.

Checking Trivial Win Conditions at the End of a Game.

Trivial Wins via Keys and Ciphertexts. The following is used for analyzing all confidentiality games. If there exists an epoch $\mathsf{e} \in \mathcal{K}^* \cap \mathcal{C}^*$ in which the adversary knows the epoch key $\mathbf{k}_\mathsf{e}$ and a valid update of the challenge ciphertext $\tilde{\mathbf{c}}_\mathsf{e}$, then the adversary can use this epoch key to decrypt the challenge-equal ciphertext and know the underlying plaintext to win the confidentiality game. The trivial win condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$" is checked in the end of a confidentiality game.

Trivial Wins via Direct Updates. The following is used for analyzing all confidentiality games with deterministic updates. If the adversary knows the update token $\Delta_{\tilde{\mathsf{e}}}$ in the challenge epoch $\tilde{\mathsf{e}}$ or the adversary queried an update oracle on the challenge input ciphertext $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ in epoch $\tilde{\mathsf{e}}$, then it knows the updated ciphertext of $\bar{\mathbf{c}}$ in epoch $\tilde{\mathsf{e}}$ and it can compare the updated ciphertext with the challenge ciphertext to win the confidentiality game. The trivial win condition "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried" is checked in the end of a confidentiality game.

Checking Trivial Win Conditions while Running a Game. The following overview of trivial win conditions are checked by an oracle. The sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{K}^*, \mathcal{L}^*$ and $\mathcal{Q}^*$ are defined in Sect. 2.4.

- "$(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$" are checked by $\mathcal{O}.\mathsf{Dec}$ oracles in the detIND-UE-CCA game,
- "$(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$" are checked by $\mathcal{O}.\mathsf{Dec}$ oracles in the randIND-UE-CCA game,
- "$\mathsf{e} \in \mathcal{K}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the INT-CTXT game or the INT-PTXT game,
- "$(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the INT-CTXT game
- "$(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$" are checked by $\mathcal{O}.\mathsf{Try}$ oracles in the INT-PTXT game.

General Idea. At the moment when the adversary queries a decryption query $\mathcal{O}.\mathsf{Dec}$ or a try query $\mathcal{O}.\mathsf{Try}$, the challenger computes the knowledge the adversary currently has, which is used to check if the adversary can trivially win a security game. More precisely, the challenger uses information in the sets $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}$ to compute the leakage sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{K}^*, \mathcal{L}^*$ and $\mathcal{Q}^*$. Note that the sets $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}$ contains information the adversary learns at such a moment.

Trivial Wins via Decryptions in the Deterministic Update Setting. The following is used for analyzing the $\mathsf{detIND\text{-}UE\text{-}CCA}$ security notion. In the deterministic update setting, if the adversary knows a challenge-equal ciphertext $(\tilde{\mathbf{c}}_{\mathsf{e}_0}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can compute the updated challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}}$ and send it to the decryption oracle to get the underlying message. Eventually, the adversary compares the received message with the challenge plaintexts to trivially win the security game.

We use the set $\tilde{\mathcal{L}}^*$ to check this trivial win condition, recall that $\tilde{\mathcal{L}}^*$ includes all challenge-equal ciphertexts the adversary has learned or inferred. Suppose the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ in epoch e, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$ then the response of the decryption oracle leads to a trivial win to the adversary, hence, the challenger will set the trivial win flag to be 1.

By Remark 1, we have $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^*$, using this method we can easily compute the set $\tilde{\mathcal{L}}^*$. In Fig. 6 we show how the set $\tilde{\mathcal{L}}^*$ is computed, where the set $\mathcal{C}^*$ is computed by the algorithms discussed in Sect. 2.5.

Trivial Wins via Decryptions in the Randomized Update Setting. The following is used for analyzing the $\mathsf{randIND\text{-}UE\text{-}CCA}$ security notion. In the randomized update setting, if the adversary knows a challenge-equal ciphertext $(\tilde{\mathbf{c}}_{\mathsf{e}_0}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create arbitrary number of ciphertexts by updating $\tilde{\mathbf{c}}_{\mathsf{e}_0}$ from epoch e_0 to epoch e. Let $\mathbf{c}_{\mathsf{e}}$ denote a ciphertext generated in such a way. Notice that the ciphertext $\mathbf{c}_{\mathsf{e}}$ has the same underlying message as the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}_0}$. The adversary can send the computed ciphertext $\mathbf{c}_{\mathsf{e}}$ to the decryption oracle to get the underlying message and trivially win the security game.

We use the set $\tilde{\mathcal{Q}}^*$ to check this trivial win condition, recall that $\tilde{\mathcal{Q}}^*$ includes information about challenge plaintexts that the adversary has learned or can create challenge-equal ciphertexts of. Suppose the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ in epoch e, if $\mathsf{UE.Dec}(\mathbf{k}_{\mathsf{e}}, \mathbf{c}) = \mathbf{m}'$ and $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$ then the response of the decryption oracle leads to a trivial win to the adversary, hence, the challenger will set the trivial win flag to be 1.

By Remark 1, we have $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^* \iff \mathsf{e} \in \mathcal{C}^*$, using this method we can easily compute the set $\tilde{\mathcal{Q}}^*$. Suppose the challenge input is $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ and the underlying message of $\bar{\mathbf{c}}$ is $\bar{\mathbf{m}}_1$. In Fig. 7 we show how the set $\tilde{\mathcal{Q}}^*$ is computed.

Remark 2. Our definition of this trivial win restriction is more generous than that of [14], they disallow the decryption of any ciphertext that decrypts to either of the two challenge plaintexts. We allow the decryption of a ciphertext that decrypts to a challenge plaintext as long as the adversary cannot learn

for $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  if $i \in \mathcal{C}^*_{\mathsf{kk},\mathsf{cc}}$ **then**
    $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\tilde{\mathbf{c}}_i, i)\}$

Fig. 6: Algorithm for computing the set $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

for $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  if $i \in \mathcal{C}^*_{\mathsf{kk},\mathsf{cc}}$ **then**
    $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\bar{\mathbf{m}}, i)\} \cup \{(\bar{\mathbf{m}}_1, i)\}$

Fig. 7: Algorithm for computing the set $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

(from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$) or infer (from tokens) a valid ciphertext of challenge plaintext in that epoch.

Trivial Forgeries by Keys. The following is used for analyzing all integrity games. If the adversary knows an epoch key $\mathbf{k}_\mathsf{e}$, then the adversary can create arbitrary number of valid forgeries of arbitrary messages under this epoch key $\mathbf{k}_\mathsf{e}$.

We use the set $\mathcal{K}^*$ to check this trivial win condition, recall that $\mathcal{K}^*$ includes all epochs the adversary learned or inferred an epoch key. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $\mathsf{e} \in \mathcal{K}^*$ then the challenger will set the trivial win flag to be 1. We use algorithms discussed in Sect. 2.5 to compute the set $\mathcal{K}^*$.

Trivial Ciphertext Forgeries by Tokens. The following is used for analyzing the INT-CTXT security notion. From [14] we know that only UE schemes with deterministic updates can possibly achieve INT-CTXT security. In the deterministic update setting, if the adversary knows a ciphertext $(\mathrm{c}, \mathbf{c}, \mathsf{e}_0; \mathbf{m}) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create a valid updated ciphertext by updating $\mathbf{c}$ from epoch e_0 to epoch e.

We use the set $\mathcal{L}^*$ to check this trivial win condition, recall that $\mathcal{L}^*$ includes all ciphertexts that can be known or inferred to the adversary. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$ then the challenger will set the trivial win flag to be 1. In Fig. 8 we show how the set $\mathcal{L}^*$ is computed.

Trivial Plaintext Forgeries by Tokens. The following is used for analyzing the INT-PTXT security notion. In the randomized update setting, if the adversary knows a ciphertext $(\mathrm{c}, \mathbf{c}, \mathsf{e}_0; \mathbf{m}) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can create arbitrary number of valid forgeries of message $\mathbf{m}$ by updating $\mathbf{c}$ from epoch e_0 to epoch e.

We use the set $\mathcal{Q}^*$ to check this trivial win condition, recall that $\mathcal{Q}^*$ includes information about plaintexts that the adversary has learned or can create ciphertexts of. Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $\mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}) = \mathbf{m}'$ and $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$ then the challenger will set the trivial win flag to be 1. In Fig. 9 we show how the set $\mathcal{Q}^*$ is computed.

for $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  for $(\cdot, \mathbf{c}, i; \cdot) \in \mathcal{L}$ **do**
    $\mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{c}, i)\}$
  if $i \in \mathcal{T}^*_{\mathsf{kk}}$ **then**
    for $(\mathbf{c}_{i-1}, i-1) \in \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}}$ **do**
      $\mathbf{c}_i \leftarrow \mathsf{UE.Upd}(\Delta_i, \mathbf{c}_{i-1})$
      $\mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{c}_i, i)\}$
    if $\mathsf{cc} = \mathsf{bi}$ **then**
      for $(\mathbf{c}_i, i) \in \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}}$ **do**
        $\mathbf{c}_{i-1} \leftarrow \mathsf{UE.Upd}^{-1}(\Delta_i, \mathbf{c}_i)$
        $\mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{c}_{i-1}, i-1)\}$

Fig. 8: Algorithm for computing the set $\mathcal{L}^*_{\mathsf{kk},\mathsf{cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

for $i \in \{0, \ldots, \mathsf{e}\}$ **do**
  for $(\cdot, \cdot, i; \mathbf{m}) \in \mathcal{L}$ **do**
    $\mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{m}, i)\}$
  if $i \in \mathcal{T}^*_{\mathsf{kk}}$ **then**
    for $(\mathbf{m}, i-1) \in \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}}$ **do**
      $\mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{m}, i)\}$
    if $\mathsf{cc} = \mathsf{bi}$ **then**
      for $(\mathbf{m}, i) \in \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}}$ **do**
        $\mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \cup \{(\mathbf{m}, i-1)\}$

Fig. 9: Algorithm for computing the set $\mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

3 Six Variants of Security Notions

In this section we first define six variants of security notions for updatable encryption schemes. In the end of this section, we compare the relationship among all these variants of each security notion.

For $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, we define $(\mathsf{kk}, \mathsf{cc})$- variants of security notions, where kk refers to UE schemes with kk-directional key updates and cc to cc-directional ciphertext updates.

Definition 3 (The $(\mathsf{kk}, \mathsf{cc})$- variant of confidentiality notions). *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an updatable encryption scheme. Then the* $(\mathsf{kk}, \mathsf{cc})$-notion *advantage, for* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ *and* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$, *of an adversary* $\mathcal{A}$ *against* UE *is defined as*

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}1}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}0}_{\mathsf{UE},\,\mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}\mathrm{b}}_{\mathsf{UE},\,\mathcal{A}}$ *is the same as the experiment* $\mathbf{Exp}^{\mathsf{notion}\text{-}\mathrm{b}}_{\mathsf{UE},\,\mathcal{A}}$ *(see Fig. 2 and Fig. 4) except for all leakage sets are both in the* kk*-directional key update setting and* cc*-directional ciphertext update setting.*

Remark 3. Recall that we compute all leakage sets with kk-directional key updates and cc-directional ciphertext updates in Sect. 2.5 and Sect. 2.6.

Remark 4. The security notion RCCA, which we denote as randIND-UE-CCA, is from [14]. In our definition of this notion is stronger - the adversary has fewer trivial win restrictions - we discuss this difference in Remark 2.

Definition 4 (The $(\mathsf{kk}, \mathsf{cc})$- variant of integrity notions). *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an updatable encryption scheme. Then the*

$(\mathsf{kk},\mathsf{cc})$-$\mathsf{notion}$ *advantage, for* $\mathsf{kk} \in \{\mathsf{no},\mathsf{uni},\mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni},\mathsf{bi}\}$ *and* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT},\mathsf{INT\text{-}PTXT}\}$*, of an adversary* $\mathcal{A}$ *against* UE *is defined as*

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Pr}[\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}} = 1],$$

where the experiment $\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}$ *is the same as the experiment* $\mathbf{Exp}^{\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}$ *(see Fig. 3 and Fig. 4) except for all leakage sets are both in the* kk*-directional key update setting and* cc*-directional ciphertext update setting.*

3.1 Properties of Leakage Sets and Trivial Win Conditions

In this section, we prove some essential properties of key leakage, which will be used to analyze the trivial win conditions. We will use these trivial win properties to prove the relations among six variants of the same security notion in Sect. 3.2.

Properties of Key Updates. Here we look at some properties of sets $\mathcal{K}, \mathcal{T}, \mathcal{K}^*$ and $\mathcal{T}^*$ in terms of uni- and bi-directional key updates.

Firewall and Insulated Region. We first describe the definition of firewall and insulated region, which will be widely used in this paper. Firewall technique (see [8, 14, 15]) is used for doing cryptographic seperation. We follow the firewall definition in [8] and use firewall set $\mathcal{FW}$ (defined in [8]) to track each insulated region and its firewalls.

Definition 5. *An* insulated region *with* firewalls fwl *and* fwr *is a consecutive sequence of epochs* $(\mathsf{fwl},\ldots,\mathsf{fwr})$ *for which:*

- $\{\mathsf{fwl},\ldots,\mathsf{fwr}\} \cap \mathcal{K} = \emptyset$*;*
- $\mathsf{fwl}, \mathsf{fwr}+1 \notin \mathcal{T}$*;*
- $\{\mathsf{fwl}+1,\ldots,\mathsf{fwr}\} \subseteq \mathcal{T}$*.*

Remark 5. Based on Definition 5, we notice that all firewalls or all insulated regions (in other words, set $\mathcal{FW}$) are uniquely determined by $\mathcal{K}$ and $\mathcal{T}$. In particular, we denote the union of all insulated regions as set $\mathcal{IR}$, i.e. $\mathcal{IR} = \cup_{(\mathsf{fwl},\mathsf{fwr})\in\mathcal{FW}}\{\mathsf{fwl},\ldots,\mathsf{fwr}\}$.

Then we look at the structure of the set $\mathcal{IR}$. Lemma 1 states that $\mathcal{IR}$ is the complementary set of $\mathcal{K}^*_{\mathsf{bi}}$. Furthermore, Lemma 3 shows that the complementary set of $\mathcal{IR}$ is the union of two types of epoch sets (see Definition 6 and Definition 7).

Lemma 1. *For any sets* $\mathcal{K}, \mathcal{T} \subseteq \{0,\ldots,l\}$*, we have* $\mathcal{K}^*_{\mathsf{bi}} = \{0,\ldots,l\} \setminus \mathcal{IR}$*.*

Proof. Note that Δ_0 and Δ_{l+1} do not exist, however, 0 and l can possibly be firewalls. For convenience, we just assume Δ_0 and Δ_{l+1} exist and the adversary is not allowed to corrupt these two tokens. Thus the set of epochs in which the adversary never corrupted the update token is: $\{0,\ldots,l+1\} \setminus \mathcal{T} = \{\bar{\mathsf{e}}_0 := 0, \bar{\mathsf{e}}_1, \ldots, \bar{\mathsf{e}}_t, \bar{\mathsf{e}}_{t+1} := l+1\}$, where $t \geq 0$.

Epoch	$\mathsf{e}_{\mathsf{start}}$	$\mathsf{e}_{\mathsf{start}}+1$...		$\mathsf{e}_{\mathsf{end}-1}$	$\mathsf{e}_{\mathsf{end}}$
$\mathcal{K}$	×	×	...	×	✓
$\mathcal{T}$	✓	✓	... ✓	✓	

Epoch	$\mathsf{e}_{\mathsf{start}}$	$\mathsf{e}_{\mathsf{start}}+1$...		$\mathsf{e}_{\mathsf{end}}$
$\mathcal{K}^*_{\mathsf{uni}}$	✓	✓	...	✓
$\mathcal{T}^*_{\mathsf{uni}}$	✓	✓	... ✓	

Fig. 10: Type 1 set of epochs (left), type 2 set of epochs (right). × indicates the keys/tokens are not revealed to the adversary, ✓ indicates the keys/tokens are revealed to the adversary.

In the bi-directional key update setting, if the adversary has corrupted a key in an epoch e, where $\mathsf{e} \in \{\bar{\mathsf{e}}_{i-1}, ..., \bar{\mathsf{e}}_i - 1\}$, then the adversary can infer all keys from epoch $\bar{\mathsf{e}}_{i-1}$ to epoch $\bar{\mathsf{e}}_i - 1$, that is $\{\bar{\mathsf{e}}_{i-1}, ..., \bar{\mathsf{e}}_i - 1\} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, because all tokens from epoch $\bar{\mathsf{e}}_{i-1} + 1$ to epoch $\bar{\mathsf{e}}_i - 1$ are corrupted. Otherwise, when no key in the sequence of epochs $\{\bar{\mathsf{e}}_{i-1}, ..., \bar{\mathsf{e}}_i - 1\}$ is corrupted, then $\{\bar{\mathsf{e}}_{i-1}, ..., \bar{\mathsf{e}}_i - 1\}$ is an insulated region . Therefore, for any i, $\{\bar{\mathsf{e}}_{i-1}, ..., \bar{\mathsf{e}}_i - 1\}$ is either an insulated region or a subset of $\mathcal{K}^*_{\mathsf{bi}}$.

We define two types of epoch sets in Definition 6 and Definition 7, which will later be used to analyze the structure of $\mathcal{IR}$. An overview of the corruption model of these two epoch sets are shown in Fig. 10.

Definition 6. *A set of* type1 *epochs is a consecutive sequence of epochs* ($\mathsf{e}_{\mathsf{start}}$, ..., $\mathsf{e}_{\mathsf{end}}$) *for which:*

- $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}} - 1\} \cap \mathcal{K} = \emptyset$;
- $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}$;
- $\{\mathsf{e}_{\mathsf{start}} + 1, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{T}$.

Definition 7. *A set of* type2 *epochs is a consecutive sequence of epochs* ($\mathsf{e}_{\mathsf{start}}$, ..., $\mathsf{e}_{\mathsf{end}}$) *for which:*

- $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{K}^*_{\mathsf{uni}}$;
- $\{\mathsf{e}_{\mathsf{start}} + 1, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{T}^*_{\mathsf{uni}}$.

The following Lemma explains that if a key is revealed in the bi-directional key update setting but not in the uni-directional key update setting then the revealed key epoch can stretch to a type 1 epoch set. We use this property to prove Lemma 3.

Lemma 2. *If* $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}} \setminus \mathcal{K}^*_{\mathsf{uni}}$, *then there exists an epoch (say* e_u*) after* e *such that* $\mathsf{e}_u \in \mathcal{K}$, $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K} = \emptyset$ *and* $\{\mathsf{e}+1, ..., \mathsf{e}_u\} \subseteq \mathcal{T}$.

Proof. As the assumption and Equations (1, 2), we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$ is inferred from the next epoch key $\mathbf{k}_{\mathsf{e}+1}$ via token $\Delta_{\mathsf{e}+1}$. That is $\mathsf{e}+1 \in \mathcal{K}^*_{\mathsf{bi}}$ and $\mathsf{e}+1 \in \mathcal{T}$. If $\mathsf{e}+1 \notin \mathcal{K}^*_{\mathsf{uni}}$, then $\mathsf{e}+2 \in \mathcal{K}^*_{\mathsf{bi}}$ and $\mathsf{e}+2 \in \mathcal{T}$. Iteratively, we know that there exists an epoch after e, say e_u, such that $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K}^*_{\mathsf{uni}} = \emptyset$, $\mathsf{e}_u \in \mathcal{K}^*_{\mathsf{uni}}$ and $\mathsf{e}+1, ..., \mathsf{e}_u \in \mathcal{T}$. Hence, $\{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K} \subseteq \{\mathsf{e}, \ldots, \mathsf{e}_u - 1\} \cap \mathcal{K}^*_{\mathsf{uni}} = \emptyset$. In particular, we know that $\mathsf{e}_u \in \mathcal{K}$ since $\mathsf{e}_u - 1 \notin \mathcal{K}^*_{\mathsf{uni}}$.

Lemma 3. *For any sets* $\mathcal{K}, \mathcal{T} \subseteq \{0, ..., l\}$, *we have* $\{0, ..., l\} \backslash \mathcal{IR} = (\cup_{\mathsf{type}\ 1}\{\mathsf{e}_{\mathsf{start}}, ..., \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\mathsf{type}\ 2}\{\mathsf{e}_{\mathsf{start}}, ..., \mathsf{e}_{\mathsf{end}}\})$, *where the two types of epoch sets are defined in Definition 6 and Definition 7.*

Proof. Suppose $\mathsf{e} \in \{0, ..., l\} \setminus \mathcal{IR}$, by Lemma 1, we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$. If $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{uni}}$, we can apply Lemma 2 and have a set of type 1 epochs, assume $\{\mathsf{e}, ..., \mathsf{e}_u\}$. For all $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}} \setminus \mathcal{K}^*_{\mathsf{uni}}$, we can find a set of type 1 epochs. Hence, the rest epochs are in the type 2 epoch sets.

Remark 6. As a conclusion of Lemma 1 and Lemma 3, we have the sequence of all epochs are a union of three types of epoch sets, that are insulated regions, type 1 epochs and type 2 epochs. $\{0, ..., l\} = (\cup_{(\mathsf{fwl},\mathsf{fwr})\in\mathcal{FW}}\{\mathsf{fwl}, ..., \mathsf{fwr}\}) \cup (\cup_{\text{type 1}}\{\mathsf{e}_{\mathsf{start}}, ..., \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\text{type 2}}\{\mathsf{e}_{\mathsf{start}}, ..., \mathsf{e}_{\mathsf{end}}\})$.

Trivial Win Equivalences in the Uni- and Bi-Directional Update Setting. We now prove seven equivalences of the trivial win conditions. As a result, we have that in any security game if the trivial win conditions in the uni-directional update setting are triggered then the same trivial win conditions in the bi-directional update setting would be triggered as well. We will use these trivial win equivalences to prove the relation between uni- and bi-directional variants of security notions in Theorem 2.

The following two lemmas show that UE schemes with uni-directional updates has less leakage than UE schemes with bi-directional updates.

Lemma 4. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{C}^*_{\mathsf{kk,uni}} \subseteq \mathcal{C}^*_{\mathsf{kk,bi}}$, $\tilde{\mathcal{L}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk,bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk,bi}}$, $\mathcal{L}^*_{\mathsf{kk,uni}} \subseteq \mathcal{L}^*_{\mathsf{kk,bi}}$, and $\mathcal{Q}^*_{\mathsf{kk,uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk,bi}}$.*

Proof. For any fixed kk-directional key updates, uni-directional ciphertext updates has less leakage than bi-directional ciphertext updates. More precisely, for any $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and a fixed kk, we compute $\mathcal{K}^*_{\mathsf{kk}}$, $\mathcal{T}^*_{\mathsf{kk}}$, $\mathcal{C}^*_{\mathsf{kk,uni}}$ and $\mathcal{C}^*_{\mathsf{kk,bi}}$ using Equations (1, 2, 3, 4, 5). Then we have $\mathcal{C}^*_{\mathsf{kk,uni}} \subseteq \mathcal{C}^*_{\mathsf{kk,bi}}$. Furthermore, we use algorithms discussed in Sect. 2.6 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk,bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk,uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk,bi}}$, $\mathcal{L}^*_{\mathsf{kk,uni}} \subseteq \mathcal{L}^*_{\mathsf{kk,bi}}$, and $\mathcal{Q}^*_{\mathsf{kk,uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk,bi}}$.

Lemma 5. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni,cc}} \subseteq \mathcal{C}^*_{\mathsf{bi,cc}}$, $\tilde{\mathcal{L}}^*_{\mathsf{uni,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi,cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi,cc}}$, $\mathcal{L}^*_{\mathsf{uni,cc}} \subseteq \mathcal{L}^*_{\mathsf{bi,cc}}$ and $\mathcal{Q}^*_{\mathsf{uni,cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi,cc}}$.*

Proof. The proof is similar to the proof of Lemma 4. For any fixed cc-directional ciphertext updates, uni-directional key updates has less leakage than bi-directional key updates. More precisely, for any $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and a fixed cc, we compute $\mathcal{K}^*_{\mathsf{uni}}$, $\mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}}$, $\mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni,cc}}$ and $\mathcal{C}^*_{\mathsf{bi,cc}}$ using Equations (1, 2, 3, 4, 5). Then we have $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, and therefore $\mathcal{C}^*_{\mathsf{uni,cc}} \subseteq \mathcal{C}^*_{\mathsf{bi,cc}}$. Furthermore, we use algorithms discussed in Sect. 2.6 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{uni,cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi,cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni,cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi,cc}}$, $\mathcal{L}^*_{\mathsf{uni,cc}} \subseteq \mathcal{L}^*_{\mathsf{bi,cc}}$ and $\mathcal{Q}^*_{\mathsf{uni,cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi,cc}}$.

Equivalence for Trivial Win Condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$".

Lemma 6. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$.*

Proof. For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$, we compute $\mathcal{K}^*_{\mathsf{uni}}, \mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{K}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{bi,bi}}$ using Equations (1, 2, 4, 5).

Note that $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$ and $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$, so $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$. It suffices to prove

$$\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset.$$

Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$. We know that firewalls provide cryptographic separation, which make sure insulated regions are isolated from other insulated regions and the complementary set of all insulated regions. If the adversary never asks for any challenge-equal ciphertext in an epoch in the set $\{0, ..., l\} \setminus \mathcal{IR}$, then the adversary cannot infer any challenge-equal ciphertext in this set even in the bi-directional update setting. That is, $\mathcal{C}^*_{\mathsf{bi,bi}} \cap (\{0, ..., l\} \setminus \mathcal{IR}) = \emptyset$. However, $\{0, ..., l\} \setminus \mathcal{IR} \overset{\text{Lemma 1}}{=} \mathcal{K}^*_{\mathsf{bi}}$, then $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, which contradicts with the assumption. Therefore, there exists an epoch $\mathsf{e}' \in \{0, ..., l\} \setminus \mathcal{IR}$ such that the adversary has asked for a challenge-equal ciphertext in this epoch, that is $\mathsf{e}' \in \mathcal{C}$.

By Lemma 3, we know that e' is located in an epoch set which is either type 1 or type 2. Suppose $\mathsf{e}' \in \{\mathsf{e}_{\mathsf{start}}, ..., \mathsf{e}_{\mathsf{end}}\}$, we know that the epoch key $\mathbf{k}_{\mathsf{e}_{\mathsf{end}}}$ is known to the adversary even in the uni-directional key update setting, i.e. $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{uni}}$. Furthermore, all tokens $\Delta_{\mathsf{e}'+1}, ..., \Delta_{\mathsf{e}_{\mathsf{end}}}$ are known to the adversary even in the uni-directional key update setting. Hence, the adversary can update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch $\mathsf{e}_{\mathsf{end}}$ to know $\tilde{\mathbf{c}}_{\mathsf{e}_{\mathsf{end}}}$. Which means $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset$.

As a corollary of Lemma 4 to 6, we have the following equivalence. We only provide Corollary 1 with a fully detailed proof, since we will use similar proof techniques for Corollary 2 to 5.

Corollary 1. *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$.*

Proof. By Lemma 4, we have $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{uni,bi}}$. By Lemma 5, we have $\mathcal{C}^*_{\mathsf{uni,bi}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$. Hence, $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$. By Lemma 6, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$.

Similarly, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \overset{\text{Lemma 5}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \overset{\text{Lemma 4}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}}$ and therefore $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,uni}} \neq \emptyset$.

Remark 7. If the trivial win condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$" is never triggered in the uni- or bi-directional update setting, then by Corollary 1 we have $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$. By Lemma 1, we have $\{0, ..., l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. Therefore, $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \{0, ..., l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. The relationship among the sets $\mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{C}^*_{\mathsf{bi,bi}}, \mathcal{IR}, \mathcal{K}^*_{\mathsf{uni}}, \mathcal{K}^*_{\mathsf{bi}}$ is shown in Fig. 11.

Fig. 11: The relationship among the sets $\mathcal{C}^*_{\mathsf{uni,uni}}, \mathcal{C}^*_{\mathsf{bi,bi}}, \mathcal{IR}, \mathcal{K}^*_{\mathsf{uni}}, \mathcal{K}^*_{\mathsf{bi}}$ if the trivial win condition "$\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,cc}} \neq \emptyset$" is never triggered for any $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Equivalence for Trivial Win Condition " $\tilde{\mathsf{e}} \in \mathcal{T}^*$ ***or*** $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ *is queried".* The event "$\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried" is independent of the key and ciphertext updates, so this trivial win condition is either triggered or not triggered in all variants of a security notion. The following Lemma shows that if the challenge token is known to the adversary in the bi-directional key update setting, then it is also known to the adversary in the uni-directional key update setting.

Lemma 7. *For any* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*. Suppose* $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk},\mathsf{cc}} = \emptyset$*, where* $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$*, then* $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{no}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{uni}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$

Proof. We know that the challenge epoch $\tilde{\mathsf{e}} \in \mathcal{C}$, so $\tilde{\mathsf{e}} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any kk-key updates, where $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$. Since the adversary does not know the key $\mathbf{k}_{\tilde{\mathsf{e}}}$, which is needed to infer the update token $\Delta_{\tilde{\mathsf{e}}}$, so token $\Delta_{\tilde{\mathsf{e}}}$ cannot be inferred by the adversary. Therefore, $\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{kk}}$ if and only if $\tilde{\mathsf{e}} \in \mathcal{T}$. Hence $\tilde{\mathsf{e}} \in \mathcal{T} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{no}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{uni}} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{bi}}$.

From now on until the end of this section, we assume the adversary queries a decryption oracle $\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ or a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e. We consider trivial win conditions which are checked in these oracles.

Equivalence for Trivial Win Condition " $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$ *".*

Lemma 8. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$*. Suppose* $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$*, then* $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{bi}}$.

Proof. By Remark 7 we have $\mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \subseteq \mathcal{IR}$. By Remark 1 we have $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^*$. Therefore, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{uni}}$ we have $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}}$ and $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{bi}}$.

If $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{bi}}$, then $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \subseteq \mathcal{IR}$. Suppose $\{\mathsf{fwl}, \ldots, \mathsf{e}\}$ is the last insulated region. If the adversary never asks for any challenge-equal ciphertext in this region, then $\{\mathsf{fwl}, \ldots, \mathsf{e}\} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$, which contradicts with $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \cap \{\mathsf{fwl}, \ldots, \mathsf{e}\}$. Hence, $\{\mathsf{fwl}, \ldots, \mathsf{e}\} \cap \mathcal{C} \neq \emptyset$, and we can assume $\mathsf{e}' \in \{\mathsf{fwl}, \ldots, \mathsf{e}\} \cap \mathcal{C}$. By the definition of insulated region we have $\{\mathsf{fwl}+1, \ldots, \mathsf{e}\} \subseteq \mathcal{T}$, and the adversary can update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch e to know $\tilde{\mathbf{c}}_{\mathsf{e}}$, i.e. $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}}$. Therefore, $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{uni}}$ as well.

As a corollary of Lemma 4, Lemma 5 and Lemma 8, we have the following result. The proof is similar to the proof of Corollary 1.

Corollary 2. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$*. Suppose* $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$*, then* $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{bi}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{bi}}$.

Equivalence for Trivial Win Condition " $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$ *".*

Lemma 9. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$*. Suppose* $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$*, then* $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{bi}}$.

Proof. The proof is similar to the proof of Lemma 8. We use the property that $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^* \iff \mathsf{e} \in \mathcal{C}^*$.

As a corollary of Lemma 4, Lemma 5 and Lemma 9, we have the following result. The proof is similar to the proof of Corollary 1.

Corollary 3. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. *Suppose* $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, *then* $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,bi}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi,bi}}$.

Equivalence for Trivial Win Condition " $\mathsf{e} \in \mathcal{K}^*$ ".

Lemma 10. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$, *we have* $\mathsf{e} \in \mathcal{K}^*_{\mathsf{uni}} \iff \mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$.

Proof. The adversary never knows any information in the future, that is, the adversary does not know a key in an epoch $\hat{\mathsf{e}} > \mathsf{e}$. If the adversary knows the current epoch key $\mathbf{k}_{\mathsf{e}}$, then it is either a corrupted key or a key inferred from prior epoch key, thus $\mathsf{e} \in \mathcal{K}^*_{\mathsf{uni}} \iff \mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$.

Equivalence for Trivial Win Condition " $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*$ ".

Lemma 11. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. *Suppose* $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, *then* $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

Proof. By assumption and Lemma 10 the current epoch $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$. We know that, by Remark 6, e is located in an insulated region, assume it is in $\{\mathsf{fwl}, ..., \mathsf{e}\}$. Thus tokens $\Delta_{\mathsf{fwl}+1}, ..., \Delta_{\mathsf{e}}$ are known to the adversary in any update setting, that is, $\{\mathsf{fwl}+1, ..., \mathsf{e}\} \subseteq \mathcal{T} \subseteq \mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$. If the adversary never asks for any ciphertext in this region, then there is no ciphertext in epoch e located in the set $\mathcal{L}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$. For all ciphertexts the adversary learns in an epoch i with $i \in \{\mathsf{fwl}, ..., \mathsf{e}\}$, the adversary can update them to epoch e using tokens. Hence, we have $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

As a corollary of Lemma 4, Lemma 5 and Lemma 11, we have the following result. The proof is similar to the proof of Corollary 1.

Corollary 4. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. *Suppose* $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, *then* $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{bi,bi}}$.

Equivalence for Trivial Win Condition " $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*$ ".

Lemma 12. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. *Suppose* $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, *then* $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni,uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi,bi}}$.

Proof. The proof is similar to the proof of Lemma 11. As $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{kk}}$ for any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$, we know that e is located in an insulated region. Assume it is in $\{\mathsf{fwl}, ..., \mathsf{e}\}$, then the adversary has corrupted the tokens $\Delta_{\mathsf{fwl}+1}, ..., \Delta_{\mathsf{e}}$. If the adversary never asks for any ciphertext with the underlying message $\mathbf{m}'$ in this region, then $(\mathbf{m}', \mathsf{e}) \notin \mathcal{Q}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$. Otherwise, suppose $(\cdot, \mathbf{c}_i, i; \mathbf{m}') \in \mathcal{L}$ with $i \in \{\mathsf{fwl}, ..., \mathsf{e}\}$, then the adversary can update $\mathbf{c}_i$, via tokens $\Delta_{i+1}, ..., \Delta_{\mathsf{e}}$, to a ciphertext in epoch e with the underlying message $\mathbf{m}'$ and we have $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$.

As a corollary of Lemma 4, Lemma 5 and Lemma 12, we have the following result. The proof is similar to the proof of Corollary 1.

Corollary 5. *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. *Suppose* $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{bi}}$, *then* $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{uni},\mathsf{bi}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{bi},\mathsf{bi}}$.

3.2 Relations among Security Notions

In Fig. 12, Fig. 13 and Fig. 14, we show the relationship among six variants of the same security notion for UE schemes.

Fig. 12 demonstrates that the uni- and bi-directional update variants of the same security notion are equivalent, which means that the security notions (confidentiality and integrity) in the uni-directional update setting are not strictly stronger than the corresponding security notions in the bi-directional update setting. Hence, the security of a UE scheme is not influenced if the update setting is uni- or bi-directional. In terms of confidentiality and integrity, when we analyze the security of a UE scheme we can analyze the security based on the UE scheme with bi-directional updates.

The six variants of confidentiality notions have the relationship shown in Fig. 13, where we present that the $(\mathsf{no}, \mathsf{uni})$- variant of any confidentiality notion is strictly stronger than the other five variants of the corresponding confidentiality notion.

The six variants of integrity notions have the relationship shown in Fig. 14. No-directional key update variants of the same integrity notion is strictly stronger than the uni- or bi-directional key update variants. However, the two variants of no-directional key update notions are equivalent, that is, for the integrity notions uni- or bi-directional ciphertext update setting (with no-directional key updates) does not matter much.

It is ideal to construct an efficient UE scheme with no-directional key updates and uni-directional ciphertext updates. However, whether such a scheme exists is an open problem.

Theorem 1 (Informal Theorem). *The relations among the six variants of the same security notion are as in Fig. 12, Fig. 13 and Fig. 14. The precise results are stated and proven in the full version [13] and due to space constraints we only show Theorem 2.*

$$(\mathsf{bi},\mathsf{bi})\text{-}\mathsf{notion} \overset{\text{Thm. 2}}{\Longleftrightarrow} (\mathsf{bi},\mathsf{uni})\text{-}\mathsf{notion} \overset{\text{Thm. 2}}{\Longleftrightarrow} (\mathsf{uni},\mathsf{bi})\text{-}\mathsf{notion} \overset{\text{Thm. 2}}{\Longleftrightarrow} (\mathsf{uni},\mathsf{uni})\text{-}\mathsf{notion}$$

Fig. 12: Relations among the uni- and bi-directional update variants of the same security notion, where $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$.

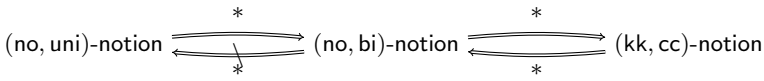

Fig. 13: Relations among the six variants of the same confidentiality notion, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. Results that are given only in the full version [13] are marked with $*$.

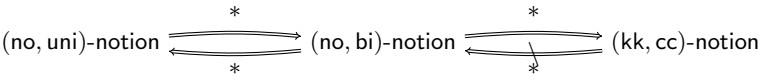

Fig. 14: Relations among the six variants of the same integrity notion, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$. Results that are given only in the full version [13] are marked with $*$.

Remark 8 (Informal intuition of these relations). Consider the following confidentiality game, where we have an adversary against some variant of the confidentiality game for a UE scheme. The adversary corrupts a key $\mathbf{k}_1$ and a token Δ_2, and asks for a challenge ciphertext in epoch 2. For both uni- and bi-directional key update settings, the adversary can move the key $\mathbf{k}_1$ to epoch 2 and decrypt the challenge ciphertext to trivially win the confidentiality game. If the UE scheme has no-directional key updates and bi-directional ciphertext updates, the adversary can move the challenge ciphertext back to epoch 1 and decrypt it to trivially win the confidentiality game. However, if the UE scheme has no-directional key updates and uni-directional ciphertext updates, the adversary cannot trivially win the confidentiality game in this action.

Similarly, we consider the following integrity game, where we have an adversary against some variant of the integrity game for a UE scheme. The adversary corrupts a key $\mathbf{k}_1$ and a token Δ_2, and queries a try oracle in epoch 2. For both uni- and bi-directional key update settings, the adversary can move the key $\mathbf{k}_1$ to epoch 2 and provide forgeries in epoch 2 to trivially win the integrity game. However, if the UE scheme has no-directional key updates the adversary does not know $\mathbf{k}_2$, and cannot trivially win the integrity game.

The following Theorem shows that for any $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$, $(\mathsf{kk}', \mathsf{cc}')$-$\mathsf{notion}$ implies $(\mathsf{kk}, \mathsf{cc})$-$\mathsf{notion}$. Consequently, all four uni- and bi-directional update variants of the same notion are equivalent.

Theorem 2. *Let* $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an updatable encryption scheme and* $\mathsf{notion} \in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}, \mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. *For any* $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$ *and any* $(\mathsf{kk}, \mathsf{cc})$-$\mathsf{notion}$ *adversary* $\mathcal{A}$ *against* UE*, there exists a* $(\mathsf{kk}', \mathsf{cc}')$-$\mathsf{notion}$ *adversary* $\mathcal{B}_2$ *against* UE *such that*

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_2}(1^\lambda).$$

Proof. We construct a reduction $\mathcal{B}_2$ running the $(\mathsf{kk}', \mathsf{cc}')$-notion experiment which will simulate the responses of queries made by the $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$. The reduction will send all queries received from $\mathcal{A}$ to its $(\mathsf{kk}', \mathsf{cc}')$-notion challenger, and forwarding the responses to $\mathcal{A}$. Eventually, the reduction receives a guess from $\mathcal{A}$ and forwards it to its own challenger. In the end, the $(\mathsf{kk}', \mathsf{cc}')$-notion challenger evaluates whether or not the reduction wins, if a trivial win condition was triggered the reduction is considered as losing the game. This final win evaluation will be passed to the adversary $\mathcal{A}$.

By the analysis of trivial win equivalences in Sect. 3.1 (Corollary 1 to 5, Lemma 7 and Lemma 10), we have that if $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{kk}, \mathsf{cc})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{kk}', \mathsf{cc}')$-notion game either. Similarly, if $\mathcal{A}$ does trigger the trivial win conditions in the $(\mathsf{kk}, \mathsf{cc})$-notion game, then the reduction will also trigger the trivial win conditions in the $(\mathsf{kk}', \mathsf{cc}')$-notion game. Hence, the reduction perfectly simulates the $(\mathsf{kk}, \mathsf{cc})$-notion game to adversary $\mathcal{A}$. And we have $\mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_2}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(1^\lambda)$.

Remark 9. For any $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}, \mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$, all four uni- and bi-directional update variants of the same notion are equivalent. We will use the $(\mathsf{bi}, \mathsf{bi})$-notion variant to prove notion security for a specific UE schemes. For simplicity, we will denote the notion $(\mathsf{bi}, \mathsf{bi})$-notion as notion.

4 LWE-based PKE Scheme

In this section, we look at an LWE-based PKE scheme LWEPKE, which is detailed in Fig. 15. We prove that LWEPKE is IND$-CPA-secure, if the underlying LWE problem is hard. We will later use this PKE scheme to construct an updatable encryption scheme in Sect. 5.

4.1 PKE Construction

In the setup phase, the scheme LWEPKE randomly chooses a matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$. The key generation algorithm samples a secret $\mathbf{s}$ from the uniform distribution $\mathcal{U}(\mathbb{Z}_q^n)$ and computes $\mathbf{p} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, where the error $\mathbf{e}$ is chosen from the discrete Gaussian distribution $D^m_{\mathbb{Z},\alpha}$. The matrix $\mathbf{A}$ and the vector $\mathbf{p}$ form the public key. Encryption takes a bit string $\mathbf{m} \in \{0,1\}^{1 \times t}$ as input, and outputs a ciphertext $(\mathbf{A}^\intercal \cdot \mathbf{R}, \mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m} \mod q)$. Decryption is performed by computing $\mathbf{d} = \mathbf{c}_2 - \mathbf{s}^\intercal \cdot \mathbf{C}_1$. For each entry d_i of $\mathbf{d}$, the decryption algorithm outputs 0 if d_i is close to $0 \mod q$, and outputs 1 if d_i is close to $\frac{q}{2} \mod q$.

Parameter Setting. The parameter setting of the scheme LWEPKE is as follows:

- $n = \lambda$ is the security parameter,
- $q = q(n) \geq 2$ be a prime,

$\underline{\mathsf{LWEPKE.Setup}(1^\lambda):}$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$

$\underline{\mathsf{LWEPKE.KG}(1^\lambda):}$
$\mathbf{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$
$\mathbf{e} \leftarrow D_{\mathbb{Z},\alpha}^m$
$\mathbf{p} \leftarrow \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \mod q$
return $(\mathbf{s},\mathbf{p})$

$\underline{\mathsf{LWEPKE.Enc}(\mathbf{p},\mathbf{m}):}$
$\mathbf{R} \leftarrow \mathcal{D}_r^t$
$\mathbf{e}' \leftarrow D_{\mathbb{Z},\beta}^{1\times t}$
$\mathbf{C}_1 \leftarrow \mathbf{A}^\intercal\cdot\mathbf{R}$
$\mathbf{c}_2 \leftarrow \mathbf{p}^\intercal\cdot\mathbf{R}+\mathbf{e}'+\frac{q}{2}\mathbf{m} \mod q$
return $(\mathbf{C}_1,\mathbf{c}_2)$

$\underline{\mathsf{LWEPKE.Dec}(\mathbf{s},\mathbf{c}):}$
parse $\mathbf{c}=(\mathbf{C}_1,\mathbf{c}_2)$
$\mathbf{d} \leftarrow \mathbf{c}_2-\mathbf{s}^\intercal\cdot\mathbf{C}_1$
parse $\mathbf{d}=(d_1,\ldots,d_t)$
for $i\in\{1,2,\ldots,t\}$ **do**
  if $d_i \in (\frac{3q}{8},\frac{5q}{8})$ **then**
    $m_i' \leftarrow 1$
  else if $d_i \in (-\frac{q}{8},\frac{q}{8})$ **then**
    $m_i' \leftarrow 0$
  else
    return $\perp$
$\mathbf{m}' \leftarrow (m_1',\ldots,m_t')$
return $\mathbf{m}'$

Fig. 15: The algorithms of the LWE-based LWEPKE scheme. The randomness distribution $\mathcal{D}_r$ is defined over $\mathbb{Z}_q^m$. $D_{\mathbb{Z},\alpha}, D_{\mathbb{Z},\beta}$ are discrete Gaussian distributions. The message $\mathbf{m}$ lies in $\{0,1\}^{1\times t}$.

- $m=\mathsf{poly}(n)$ and $t=\mathsf{poly}(n)$ be two integers,
- $\mathcal{D}_r$ be a distribution over $\mathbb{Z}_q^m$ with min-entropy k such that $n \le (k-2\log(1/\epsilon)-O(1))/\log(q)$ for negligible $\epsilon>0$, the infinite norm of the vector outputted by this distribution is at most $B=\mathsf{poly}(n)$ with overwhelming probability,
- $\alpha,\beta>0$ be two numbers such that $\beta\le\frac{q}{8}$ and $\alpha B/\beta=\mathsf{negl}(n)$.
- $D_{\mathbb{Z},\alpha}$ and $D_{\mathbb{Z},\beta}$ be two discrete Gaussian distributions.

Remark 10. We specify that all operations in this paper are done in field $\mathbb{Z}_q$, and stop writing $\mod q$ for the rest of this paper.

4.2 Correctness and Security

Correctness. We claim that LWEPKE.Dec decrypts correctly with overwhelming probability. The decryption algorithm computes $\mathbf{d}=\mathbf{c}_2-\mathbf{s}^\intercal\cdot\mathbf{C}_1=\mathbf{e}^\intercal\cdot\mathbf{R}+\mathbf{e}'+\frac{q}{2}\mathbf{m}$, and outputs $\mathbf{m}$ if $\mathbf{e}^\intercal\cdot\mathbf{R}+\mathbf{e}'$ has distance at most $\frac{q}{8}$ from $\mathbf{0} \mod q$. The detailed analysis of the correctness is provided in the full version [13].

Security. We now show that LWEPKE is IND$-CPA-secure under the assumption that the $\mathsf{DLWE}_{n,q,\alpha}$ problem is hard.

Theorem 3. *Let* LWEPKE *be the public key encryption described in Fig. 15, using the parameter setting described in Sect. 4.1. Then for any adversary* IND$-CPA *$\mathcal{A}$ against* LWEPKE*, there exists an adversary $\mathcal{B}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that*

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\mathcal{A}}(1^\lambda) \le t\epsilon + \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

Proof sketch. We sketch the main idea of the proof and provide the full details in the full version [13]. We claim that the real challenge ciphertext $(\mathbf{C}_1, \mathbf{c}_2)$ is statistically close to the ciphertext generated as $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}')$. Then first entry $\mathbf{C}_1$ is statistically close to a random element because of the leftover hash lemma, and therefore the whole ciphertext $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}')$ is computationally indistinguishable from a random ciphertext based on the hardness of the learning with error.

5 LWE-based Updatable Encryption Scheme

We construct an LWE-based updatable encryption scheme LWEUE and prove that it is randIND-UE-CPA secure if the underlying LWE problem is hard.

5.1 UE Construction

We now introduce our updatable encryption scheme LWEUE, which is parameterized by an LWE-based PKE scheme LWEPKE (see Fig. 15). LWEUE uses algorithms from LWEPKE to do key generation, encryption and decryption. To generate a new key from an old key in the next algorithm, our UE scheme uses the homomorphic property of the LWE pairs. In particular, suppose the old key is $(\mathbf{s}_\mathsf{e}, \mathbf{p}_\mathsf{e})$, LWEUE.KG samples a new pair of LWE pairs $(\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ and sets $(\mathbf{s}_\mathsf{e} + \Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_\mathsf{e} + \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ as the new epoch key, where $(\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_\mathsf{e} + \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ is the update token. To update ciphertexts, LWEUE uses the re-randomization idea that was similar to the idea from RISE in the work by Lehmann and Tackmann [15]. As the ciphertext can be re-randomized by the update token, the update algorithm uses the update token to update ciphertext from an old one to a new one. More precisely, the scheme LWEUE is described in Fig. 16.

Parameter Setting We use the parameter setting of the scheme LWEPKE, described in Sect. 4.1. Additionally, we require $\beta \leq \frac{q}{8\sqrt{l}}$, where $l = \mathsf{poly}(n)$ is an upper bound on the last epoch.

5.2 Construction Challenges in LWE-based UE Schemes

In this section, we discuss leakage from tokens due to bad UE construction and show how to solve this leakage problems.

Secret Key Distribution. We first state that a binary secret does not work in the UE scheme, as an update token might reveal the secret information. Suppose an entry of the update token $\Delta^{\mathbf{s}}_{\mathsf{e}+1} (= \mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_\mathsf{e})$ is -1 (1, resp.), then we can conclude the corresponding entry of the previous secret $\mathbf{s}_\mathsf{e}$ is 1 (0, resp.) and the corresponding entry of the new secret $\mathbf{s}_{\mathsf{e}+1}$ is 0 (1, resp.).

We choose that secret keys and update tokens are sampled from the uniform distribution over $\mathbb{Z}_q^n$, which ensures that any corrupted token will not reveal any information about the relevant secret keys.

$\underline{\mathsf{Setup}(1^\lambda):}$
$\quad \mathbf{A} \leftarrow \mathsf{LWEPKE.Setup}(1^\lambda)$

$\underline{\mathsf{LWEUE.KG}(1^\lambda):}$
$\quad \textbf{if } \mathsf{e} = 0 \textbf{ then}$
$\quad\quad (\mathbf{s}_0, \mathbf{p}_0) \leftarrow \mathsf{LWEPKE.KG}(1^\lambda)$
$\quad \textbf{else}$
$\quad\quad \text{parse } \mathbf{k}_{\mathsf{e}-1} = (\mathbf{s}_{\mathsf{e}-1}, \mathbf{p}_{\mathsf{e}-1})$
$\quad\quad (\Delta^{\mathbf{s}}_{\mathsf{e}}, \Delta^{\mathbf{p}}_{\mathsf{e}}) \leftarrow \mathsf{LWEPKE.KG}(1^\lambda)$
$\quad\quad \mathbf{s}_{\mathsf{e}} \leftarrow \mathbf{s}_{\mathsf{e}-1} + \Delta^{\mathbf{s}}_{\mathsf{e}}$
$\quad\quad \mathbf{p}_{\mathsf{e}} \leftarrow \mathbf{p}_{\mathsf{e}-1} + \Delta^{\mathbf{p}}_{\mathsf{e}}$
$\quad\quad \mathbf{k}_{\mathsf{e}} \leftarrow (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
$\quad \textbf{return } \mathbf{k}_{\mathsf{e}}$

$\underline{\mathsf{LWEUE.TG}(\mathbf{k}_{\mathsf{e}}, \mathbf{k}_{\mathsf{e}+1}):}$
$\quad \text{parse } \mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
$\quad \text{parse } \mathbf{k}_{\mathsf{e}+1} = (\mathbf{s}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
$\quad \Delta^{\mathbf{s}}_{\mathsf{e}+1} \leftarrow \mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}}$
$\quad \Delta_{\mathsf{e}+1} \leftarrow (\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
$\quad \textbf{return } \Delta_{\mathsf{e}+1}$

$\underline{\mathsf{LWEUE.Enc}(\mathbf{k}_{\mathsf{e}}, \mathbf{m}):}$
$\quad \text{parse } \mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
$\quad \mathbf{c}_{\mathsf{e}} \leftarrow \mathsf{LWEPKE.Enc}(\mathbf{p}_{\mathsf{e}}, \mathbf{m})$
$\quad \textbf{return } \mathbf{c}_{\mathsf{e}}$

$\underline{\mathsf{LWEUE.Dec}(\mathbf{k}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}}):}$
$\quad \text{parse } \mathbf{k}_{\mathsf{e}} = (\mathbf{s}_{\mathsf{e}}, \mathbf{p}_{\mathsf{e}})$
$\quad \mathbf{m}' \leftarrow \mathsf{LWEPKE.Dec}(\mathbf{s}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}})$
$\quad \textbf{return } \mathbf{m}'$

$\underline{\mathsf{LWEUE.Upd}(\Delta_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e}}):}$
$\quad \text{parse } \Delta_{\mathsf{e}+1} = (\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$
$\quad \text{parse } \mathbf{c}_{\mathsf{e}} = (\mathbf{C}^1_{\mathsf{e}}, \mathbf{c}^2_{\mathsf{e}})$
$\quad (\mathbf{C}^1, \mathbf{c}^2) \xleftarrow{\$} \mathsf{LWEPKE.Enc}(\mathbf{p}_{\mathsf{e}+1}, \mathbf{0})$
$\quad \mathbf{C}^1_{\mathsf{e}+1} \leftarrow \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1$
$\quad \mathbf{c}^2_{\mathsf{e}+1} \leftarrow \mathbf{c}^2_{\mathsf{e}} + (\Delta^{\mathbf{s}}_{\mathsf{e}+1})^\intercal \cdot \mathbf{C}^1_{\mathsf{e}} + \mathbf{c}^2$
$\quad \mathbf{c}_{\mathsf{e}+1} \leftarrow (\mathbf{C}^1_{\mathsf{e}+1}, \mathbf{c}^2_{\mathsf{e}+1})$
$\quad \textbf{return } \mathbf{c}_{\mathsf{e}+1}$

Fig. 16: The algorithms of LWE-based updatable encryption scheme LWEUE, which is parameterized by an LWE-based PKE scheme LWEPKE.

Epoch Key Generation. Intuitively, it is natural to consider generating the epoch keys by sampling a secret $\mathbf{s}_i \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and setting the public key to be $\mathbf{p}_i = \mathbf{A} \cdot \mathbf{s}_i + \mathbf{e}_i$, where $\mathbf{e}_i \leftarrow D^m_{\mathbb{Z},\alpha}$. Then the update token is set as $\Delta_i = (\mathbf{s}_i - \mathbf{s}_{i-1}, \mathbf{p}_i)$.

In a confidentiality game for such UE schemes, suppose the adversary knows two consecutive tokens Δ_{i-1} and Δ_i. Using these tokens the adversary can compute $\mathbf{p}_i - \mathbf{p}_{i-1} - \mathbf{A} \cdot \Delta^{\mathbf{s}}_i = \mathbf{e}_i - \mathbf{e}_{i-1}$, and knows $\mathbf{e}_i - \mathbf{e}_{i-1}$. Which means if the adversary knows a set of consecutive tokens $\Delta_i, \Delta_{i+1}, ..., \Delta_{i+j}$ then it will also know $\{\mathbf{e}_{i+1} - \mathbf{e}_i, \mathbf{e}_{i+2} - \mathbf{e}_i, ..., \mathbf{e}_{i+j} - \mathbf{e}_i\}$, the values in this set are sampled from a discrete Gaussian distribution centered at $\mathbf{e}_i$. Through evaluating these errors the adversary can possibly find the error value $\mathbf{e}_i$ and therefore knows the secret value $\mathbf{s}_i$. Furthermore, the adversary is allowed to ask for a challenge-equal ciphertext in epoch i, which will not trigger the trivial win condition, and can therefore break this confidentiality game. The above attack shows that this epoch key generation approach is not safe, it might leak the secret epoch key information.

We choose to generate a fresh pair $(\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ to compute the new epoch key and the update token, which makes sure the update token $\Delta_{\mathsf{e}+1} = (\Delta^{\mathbf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$ is independent from the previous epoch key. Additionally, this pair is computationally indistinguishable from a uniformly random pair as long as the underlying LWE problem is hard.

5.3 Correctness

Errors in updated ciphertexts increase when they are updated. Since the total number of epochs is bounded with a comparatively small integer l, the UE scheme supports a limited number of ciphertext updates. As a result, errors in updated ciphertexts will not grow too big and the decryption will be correct with overwhelming probability for some parameter setting. The correctness analysis is discussed in the full version [13].

5.4 Challenges of the Security Proof in LWE-based UE Schemes

In this section we highlight the difficulties when proving that LWEUE is a secure UE scheme, specifically, our UE scheme has a randomized update algorithm. Lehmann and Tackmann [15] and Klooß et al. [14] both described a method, similar to each other, to prove that updatable encryption schemes with randomized update algorithms are secure. Their technique can be seen when they prove that RISE and NYUE (NYUAE) are secure, resp. However, this method can not be directly used to prove that LWEUE is secure. The method introduced requires that UE schemes have perfect re-encryption, which means the distribution of updated ciphertexts has the same distribution as fresh encryptions. In their proof, they replace updated ciphertexts by fresh encryptions of the underlying messages. However, in the LWEUE scheme, we cannot simply replace updated ciphertexts by a fresh encryption because the randomness terms and the error terms grow while updating and an updated ciphertext does not have the same distribution as a fresh encryption.

5.5 Security

If LWEPKE is IND$-CPA-secure then the output of the encryption algorithm is computationally indistinguishable from a pair of uniformly random elements. Hence, the fresh encryption in the LWEUE scheme is computationally indistinguishable from a pair of uniformly random elements as well. Furthermore, the update algorithm LWEUE.Upd runs the encryption algorithm of LWEPKE to re-randomize the old ciphertext to a new ciphertext, therefore, the updated ciphertext is also computationally indistinguishable from a pair of uniformly random elements. So, a fresh encryption is computationally indistinguishable from an updated ciphertext and LWEUE is randIND-UE-CPA secure (see Definition 1). This provides the underlying intuition for the security proof.

The full proof of Theorem 4 is given in the full version [13].

Theorem 4 (LWEUE is randIND-UE-CPA). *Let* LWEUE *be the updatable encryption scheme described in Fig. 16, using parameter setting described in Sect. 5.1. For any* randIND-UE-CPA *adversary* $\mathcal{A}$ *against* LWEUE*, there exists an adversary* $\mathcal{B}_4$ *against* $\mathsf{DLWE}_{n,q,\alpha}$ *such that*

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{LWEUE},\,\mathcal{A}}(1^\lambda) \leq 2(l+1)^3 \cdot \big(t\epsilon + 3\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}_4) + \mathsf{negl}(n)\big).$$

Remark 11. Klooß et al. [14] introduced a generic construction of transforming CPA-secure UE schemes to UE schemes with PTXT and RCCA security. The main idea is to use the extended Naor-Yung (NY) CCA-transform [17] (for public-key schemes). The NY approach is to encrypt a message under two (public) keys of a CPA-secure encryption scheme. The extended NY approach additionally includes a proof that shows the owner knows a valid signature that contains the NY ciphertext pair and the underlying message. A potential future work would be to incorporate LWEUE to their construction to create a UE scheme that achieves PTXT and RCCA security.

Acknowledgements. We would like to thank Gareth T. Davies, Herman Galteland and Kristian Gjøsteen for fruitful discussions, and the anonymous reviewers for a number of valuable suggestions.

References

1. Alkim, E., Bos, J.W., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM: Learning With Errors Key Encapsulation. `https://frodokem.org/files/FrodoKEM-specification-20190330.pdf`, Submission to the NIST Post-Quantum Standardization project, round 2
2. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum Key Exchange - A New Hope. In: USENIX Security Symposium. pp. 327–343. USENIX Association (2016)
3. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber (version 2.0). `https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf`, Submission to the NIST Post-Quantum Standardization project, round 2
4. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU Prime: Reducing attack surface at low cost. In: SAC. Lecture Notes in Computer Science, vol. 10719, pp. 235–260. Springer (2017)
5. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving Speed and Security in Updatable Encryption Schemes. IACR Cryptology ePrint Archive **2020**, 222 (2020), `https://eprint.iacr.org/2020/222`
6. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) Proceedings of CRYPTO 2013 I. LNCS, vol. 8042, pp. 410–428. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_23
7. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. IACR Cryptology ePrint Archive **2015**, 220 (2015), `http://eprint.iacr.org/2015/220`
8. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 464–493. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_16

9. Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z.: NTRU. https://ntru.org/f/ntru-20190330.pdf, Submission to the NIST Post-Quantum Standardization project, round 2
10. D'Anvers, J., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10831, pp. 282–305. Springer (2018). https://doi.org/10.1007/978-3-319-89339-6_16
11. Everspaugh, A., Paterson, K.G., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Proceedings of CRYPTO 2017 III. LNCS, vol. 10403, pp. 98–129. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_4
12. Hamburg, M.: Three Bears. https://sourceforge.net/projects/threebears/, Submission to the NIST Post-Quantum Standardization project, round 2
13. Jiang, Y.: The direction of updatable encryption does not matter much. Cryptology ePrint Archive, Report 2020/622 (2020), https://eprint.iacr.org/2020/622
14. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Proceedings of EUROCRYPT 2019 I. LNCS, vol. 11476, pp. 68–99. Springer (2019). https://doi.org/10.1007/978-3-030-17653-2_3
15. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 685–716. Springer (2018). https://doi.org/10.1007/978-3-319-78372-7_22
16. Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z., Liu, Z., Yang, H., Li, B., Wang, K.: LAC Lattice-based Cryptosystems, Submission to the NIST Post-Quantum Standardization project, round 2
17. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Ortiz, H. (ed.) Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA. pp. 427–437. ACM (1990). https://doi.org/10.1145/100216.100273
18. Oscar, G.M., Zhenfei, Z., Sauvik, B., Ronald, R., Ludo, T., Jose-Luis, T.A., Hayo, B., Markku-Juhani O., S., Scott, F., Thijs, L., Rachel, P., Jung, Hee, C., Yongha, S.: Round5. https://round5.org, Submission to the NIST Post-Quantum Standardization project, round 2