The Direction of Updatable Encryption does not Matter Much

Yao Jiang

Norwegian University of Science and Technology, NTNU, Norway.
yao.jiang@ntnu.no

June 20, 2021

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.

† © IACR 2020: An extended abstract of this paper appears in the proceedings of Asiacrypt 2020, with DOI: 10.1007/978-3-030-64840-4_18. This is the full and updated version. A summary of the changes made in the update is given in Section 1.5.

Contents

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 ciphertext 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) [BLMR13, EPRS17, LT18, KLR19, BDGJ20, BEKS20] provides a better solution for key rotation. A client generates an *update token* and send 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 Section 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 [BLMR15, LT18, KLR19, BDGJ20] focus on ciphertext-independent schemes. In this paper, we will focus on such schemes.

Consider the following four variants of update setting 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 [BLMR15, LT18, KLR19, BDGJ20] 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 [LT18], 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 confidentiality notion is strictly stronger than the uni- and bi-directional update variant of the same confidentiality 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 [LT18], Klooß et al. [KLR19] and Boyd et al. [BDGJ20] base their security on the DDH problem, and thus are only secure in the classical setting. Boneh et al. [BLMR13] 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. [BLMR13] cannot achieve IND-UPD security (introduced in [LT18]).

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 [BDGJ20], 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. [BLMR13] introduced a security definition for UE, however, this notion is less adaptive than the later works [LT18, KLR19, BDGJ20] 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. [EPRS17] provided two security notions, a weak form of ciphertext integrity and re-encryption indistinguishability, that strengthen the security notion in [BLMR13]. Recently, Boneh et al. [BEKS20] introduced new definitions for updatable encryption in the ciphertext-dependent setting to further strengthen the confidentiality property and the integrity definition in [EPRS17]. Boneh et al. [BEKS20] 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 [BDGJ20].

Lehmann and Tackmann [LT18] 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. [KLR19] attempted to provide stronger security notions for ciphertext-independent UE than LT18, specifically, CCA security and integrity protection.

Boyd et al. [BDGJ20] 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 [KLR19] and showed the following generic composition result: CPA + CTXT $\implies$ CCA.

Constructing Ciphertext-Independent Updatable Encryption Schemes. The UE scheme BLMR in [BLMR13] 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 [LT18] 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. [KLR19] 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. [BDGJ20] 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 [ABD$^+$b], FrodoKEM [ABD$^+$a], LAC [LLJ$^+$], NewHope [ADPS16], NTRU [CDH$^+$, BCLvV17], Round5 [OZS$^+$], SABER [DKRV18] and Three Bears [Ham]. 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 Section 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 implies that the search for uni-directional updatable encryption scheme seems less important.

Furthermore, we show that confidentiality 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.

[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 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.

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.

1.4 Organization

We provide the background of updatable encryption in Section 2. In Section 3 we define the six variants of security notions for UE schemes and prove the relations among the six variants of security notions.

In Section 4 we construct an LWE-based PKE scheme LWEPKE and prove that it is secure, this PKE scheme will be used to construct a UE scheme. We then construct an LWE-based UE scheme LWEUE in Section 5, and include the restrictions we encountered when constructing a secure UE scheme from a PKE scheme.

1.5 Changes w.r.t. the Asiacrypt2020 Version.

In this updated version we correct two misconceptions present in our original paper. First, one of the trivial win conditions for integrity notions (see page 14) is insufficient when we consider no-directional key update variant of integrity notions. Trivial forgeries by keys (the trivial win condition named in the old version) should be extended to trivial forgeries by keys and tokens (the trivial win condition named in this updated version). Second, the relation between the no-directional and the directional variants of integrity notion (see Section 3.2.2) is influenced by the above mistake.

2 Preliminaries

In this section we describe the notation used in this paper and present the necessary background material of updatable encryption. In Appendix A, we provide 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. The notation $X \overset{s}{\approx} Y$ ($X \overset{c}{\approx} Y$, resp.) means X is statistically indistinguishable (computationally indistinguishable, resp.) from Y. Let $\mathcal{U}(S)$ denote the uniform distribution over set S.

2.2 Security Notions for Encryption Schemes

We describe the real or random variant of indistinguishability under chosen-plaintext attack ($\mathsf{IND\$\text{-}CPA}$) for public key encryption (PKE) and symmetric key encryption (SKE). Note that $\mathsf{IND\$\text{-}CPA}$ implies $\mathsf{IND\text{-}CPA}$.

Definition 1. [The $\mathsf{IND\$\text{-}CPA}$ notion for PKE] Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption scheme. The $\mathsf{IND\$\text{-}CPA}$ advantage of any adversary $\mathcal{A}$ against PKE is

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{PKE},\ \mathcal{A}}(1^\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}1}}_{\mathsf{PKE},\ \mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{PKE},\ \mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{PKE},\ \mathcal{A}}$ is given in Figure 1.

Definition 2. [The $\mathsf{IND\$\text{-}CPA}$ notion for SKE] Let $\mathsf{SKE} = (\mathsf{SKE.KG}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a symmetric key encryption scheme. The $\mathsf{IND\$\text{-}CPA}$ advantage of any adversary $\mathcal{A}$ against SKE is

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SKE},\ \mathcal{A}}(1^\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}1}}_{\mathsf{SKE},\ \mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{SKE},\ \mathcal{A}} = 1]\right|,$$

where the experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{SKE},\ \mathcal{A}}$ is given in Figure 2.

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

```
$\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{PKE},\,\mathcal{A}}$ :
  $(\mathbf{s}, \mathbf{p}) \leftarrow \mathsf{PKE.KG}$
  $(\mathbf{m}, \mathrm{state}) \leftarrow \mathcal{A}(\mathbf{p})$
  if $\mathrm{b} = 0$ then
    $\mathbf{c} \leftarrow \mathsf{PKE.Enc}(\mathbf{p}, \mathbf{m})$
  else
    $\mathbf{c} \xleftarrow{\$} \mathcal{CS}$
  $\mathrm{b}' \leftarrow \mathcal{A}(\mathrm{state}, \mathbf{c})$
  return $\mathrm{b}'$
```

Figure 1: The experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{PKE},\,\mathcal{A}}$ for a PKE scheme PKE.

```
$\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{SKE},\,\mathcal{A}}$ :
  $\mathbf{s} \leftarrow \mathsf{SKE.KG}$
  $(\mathbf{m}, \mathrm{state}) \leftarrow \mathcal{A}(1^\lambda)$
  if $\mathrm{b} = 0$ then
    $\mathbf{c} \leftarrow \mathsf{SKE.Enc}(\mathbf{s}, \mathbf{m})$
  else
    $\mathbf{c} \xleftarrow{\$} \mathcal{CS}$
  $\mathrm{b}' \leftarrow \mathcal{A}(\mathrm{state}, \mathbf{c})$
  return $\mathrm{b}'$
```

Figure 2: The experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}b}}_{\mathsf{SKE},\,\mathcal{A}}$ for a SKE scheme SKE.

Definition 3. [Correctness of a PKE] Let $\mathsf{PKE} = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption scheme. We say PKE has *$(1-\epsilon)$-correctness* if: for any message $\mathbf{m}$, any key $(\mathbf{s}, \mathbf{p}) \leftarrow \mathsf{PKE.KG}$

$$\mathbf{Pr}[\mathsf{PKE.Dec}(\mathbf{s}, \mathsf{PKE.Enc}(\mathbf{p}, \mathbf{m})) = \mathbf{m}] \geq 1 - \epsilon.$$

2.3 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.

In the updatable encryption setting, the total number of epoch will be a comparatively small integer in practice, we consider the total number of epoch to be bounded in this paper. In particular, we denote l as an upper bound on the last epoch.

Definition 4 (Correctness of an UE)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. We say UE has *$(1-\epsilon)$-correctness* if: for any message $\mathbf{m}$ and any epochs $\mathsf{e}_1 \leq \mathsf{e}_2 \leq l$, we have

$$\mathbf{Pr}[\mathsf{UE.Dec}(\mathbf{k}_{\mathsf{e}_2}, \mathbf{c}_{\mathsf{e}_2}) = \mathbf{m}] \geq 1 - \epsilon,$$

where $\mathbf{k}_{\mathsf{e}_1}, \ldots, \mathbf{k}_{\mathsf{e}_2} \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$, $\mathbf{c}_{\mathsf{e}_1} \xleftarrow{\$} \mathsf{UE.Enc}(\mathbf{k}_{\mathsf{e}_1}, \mathbf{m})$, and

$$\Delta_j \leftarrow \mathsf{UE.TG}(\mathbf{k}_{j-1}, \mathbf{k}_j), \mathbf{c}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathbf{c}_{j-1}), \text{ for } j \in \{\mathsf{e}_1 + 1, \ldots, \mathsf{e}_2\}.$$

2.4 Existing Security Notions for Updatable Encryption

Klooß et al. [KLR19] and Boyd et al. [BDGJ20] 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. [BDGJ20]. An overview of the oracles the adversary has access to in each security game is given in Fig. 3. A generic description of all confidentiality experiments and integrity experiments described in this paper is detailed in Fig. 4 and Fig. 5, resp.. Our oracle algorithms, see Fig. 6, are stated differently than in [BDGJ20] and [KLR19], however, conceptually they are the same. The oracles we use in our security games are as follows, encrypt $\mathcal{O}.\mathsf{Enc}$, decrypt $\mathcal{O}.\mathsf{Dec}$, move to the next epoch $\mathcal{O}.\mathsf{Next}$, update ciphertext $\mathcal{O}.\mathsf{Upd}$, corrupt key or token $\mathcal{O}.\mathsf{Corr}$, ask for the challenge ciphertext $\mathcal{O}.\mathsf{Chall}$, get an updated version of the challenge ciphertext $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$, or test if a ciphertext is a valid forgery $\mathcal{O}.\mathsf{Try}$. The detailed discussion of trivial win conditions are discussed in Section 2.7.

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

Figure 3: 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.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ :
  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 $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried$)\big)\Big)$ then
    $\mathsf{twf} \leftarrow 1$
  if $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
  return $\mathrm{b}'$
```

Figure 4: 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. 3 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 Section 2.6.

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

Figure 5: 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. 3 shows the oracles the adversary have access to in a specific security game.

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 $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$ oracles or corrupt tokens to know some (updated) versions of ciphertexts, we call them *non-challenge ciphertexts*. In addition, the adversary may query $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ oracles or corrupt tokens to infer some (updated) versions of the challenge ciphertext, we call them *challenge-equal ciphertexts*.

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

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

$\mathcal{O}.\mathsf{Dec}(\mathbf{c})$ :
  $\mathbf{m}'$ 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 $\perp$

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

$\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}-1})$ :
  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 $\mathbf{c}_\mathsf{e}$
```

```
$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  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 $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Chall}(\bar{\mathbf{m}}, \bar{\mathbf{c}})$ :
  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 $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ :
  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 $\tilde{\mathbf{c}}_\mathsf{e}$

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

Figure 6: Oracles in security games for updatable encryption. How to compute the leakage sets $\mathcal{K}^*, \mathcal{T}^*, \mathcal{C}^*, \tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \hat{\mathcal{K}}^*, \mathcal{L}^*, \mathcal{Q}^*$ are discussed in Section 2.6 and Section 2.7.

Definition 5. 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 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

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

where the experiment $\mathbf{Exp}^{\mathsf{notion}\text{-}b}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 4 and Fig. 6.

Definition 6. 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 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. 5 and Fig. 6.

2.5 Notations of the Leakage Sets

In this section, we describe the definition of leakage sets given by [LT18] and [KLR19], 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 Section 3.1.

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 $\mathcal{K}^*$, $(\hat{\mathcal{K}}^*)$, $\mathcal{T}^*$ 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 Section 2.6. We show how to compute $\hat{\mathcal{K}}^*$ in Equation (6).

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 $(\tilde{\mathsf{c}}_{\mathsf{e}}, \mathsf{e})$. The adversary learned these ciphertexts from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$.

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 2.1. 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 Section 2.7.

2.6 Epoch Leakage Sets of Keys, Tokens and Ciphertexts

We follow the bookkeeping techniques and base our notations of the work of Lehmann and Tackmann [LT18], 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 $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, l\}$, 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 $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, 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:

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

Bi-directional key updates:

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

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.

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

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:

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

Bi-directional ciphertext updates:

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

2.7 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 [LT18, KLR19, BDGJ20] 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. 7.

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

Figure 7: 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.

2.7.1 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 challenge 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.

2.7.2 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 Section 2.5.

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

General Idea. At the moment when the adversary queries a decryption query $\mathcal{O}$.Dec or a try query $\mathcal{O}$.Try, the challenger computes the knowledge the adversary $\underline{\text{currently}}$ 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 $\tilde{\mathcal{L}}^*$, $\tilde{\mathcal{Q}}^*$, $\hat{\mathcal{K}}^*$, $\mathcal{L}^*$ 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 detIND-UE-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 2.1, 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. 8 we show how the set $\tilde{\mathcal{L}}^*$ is computed, where the set $\mathcal{C}^*$ is computed by the algorithms discussed in Section 2.6.

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

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

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

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

Trivial Wins via Decryptions in the Randomized Update Setting. The following is used for analyzing the randIND-UE-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 2.1, 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. 9 we show how the set $\tilde{\mathcal{Q}}^*$ is computed.

Remark 2.2. Our definition of this trivial win restriction is more generous than that of [KLR19], 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 (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 and Tokens. The following is used for analyzing all integrity games. If the adversary corrupts an epoch key at some previous epoch, say $\hat{\mathsf{e}}$, and corrupts tokens $\Delta_{\hat{\mathsf{e}}+1}, \ldots, \Delta_{\mathsf{e}}$. Then the adversary can create arbitrary number of valid forgeries by creating ciphertexts in epoch $\hat{\mathsf{e}}$ and updating these ciphertexts from epoch $\hat{\mathsf{e}}$ to epoch e.

Note that even key $\mathbf{k}_{\mathsf{e}}$ is not known to the adversary in the no-directional update settings, the adversary can create forgeries in epoch e.

We use the set $\hat{\mathcal{K}}^*$ to check this trivial win condition, and show how the set is computed in Equation (6). Suppose the adversary queries a try oracle $\mathcal{O}.\mathsf{Try}(\mathbf{c})$ in epoch e, if $\mathsf{e} \in \hat{\mathcal{K}}^*$ then the challenger will set the trivial win flag to be 1.

$$\begin{aligned} \hat{\mathcal{K}}^* &\leftarrow \{i \in \{0, \ldots, \mathsf{e}\} | \mathsf{ForgK}(i) = \mathsf{true}\} \\ \mathsf{true} \leftarrow \mathsf{ForgK}(i) &\iff (i \in \mathcal{K}) \vee (\mathsf{ForgK}(i\text{-}1) \wedge i \in \mathcal{T}). \end{aligned} \tag{6}$$

Trivial Ciphertext Forgeries by Tokens. The following is used for analyzing the INT-CTXT security notion. From [KLR19] 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. 10 we show how the set $\mathcal{L}^*$ is computed.

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

Figure 10: Algorithm for computing the set $\mathcal{L}^*_{\mathsf{kk,cc}}$, 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**
    $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i)\}$
  **if** $i \in \mathcal{T}^*_{\mathsf{kk}}$ **then**
    **for** $(\mathbf{m}, i-1) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ **do**
      $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i)\}$
    **if** $\mathsf{cc} = \mathsf{bi}$ **then**
      **for** $(\mathbf{m}, i) \in \mathcal{Q}^*_{\mathsf{kk,cc}}$ **do**
        $\mathcal{Q}^*_{\mathsf{kk,cc}} \leftarrow \mathcal{Q}^*_{\mathsf{kk,cc}} \cup \{(\mathbf{m}, i-1)\}$
```

Figure 11: Algorithm for computing the set $\mathcal{Q}^*_{\mathsf{kk,cc}}$, where $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Trivial Plaintext Forgeries by Tokens. The following is used for analyzing the $\mathsf{INT\text{-}PTXT}$ security notion. In the randomized update setting, if the adversary knows a ciphertext $(\mathsf{c}, \mathbf{c}, \mathsf{e}_0; \mathbf{m}) \in \mathcal{L}$ 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. 11 we show how the set $\mathcal{Q}^*$ is computed.

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 7 (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 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 $\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\,\mathcal{A}}$ is the same as the experiment $\mathbf{Exp}^{\mathsf{notion}\text{-}\mathsf{b}}_{\mathsf{UE},\,\mathcal{A}}$ (see Fig. 4 and Fig. 6) except for all leakage sets are both in the kk-directional key update setting and cc-directional ciphertext update setting.

Remark 3.1. Recall that we compute all leakage sets with kk-directional key updates and cc-directional ciphertext updates in Section 2.6 and Section 2.7.

Remark 3.2. The security notion RCCA, which we denote as $\mathsf{randIND\text{-}UE\text{-}CCA}$, is from [KLR19]. In our definition of this notion is stronger - the adversary has fewer trivial win restrictions - we discuss this difference in Remark 2.2.

Definition 8 (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 $(\mathsf{kk}, \mathsf{cc})$-notion advantage, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ and 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. 5 and Fig. 6) 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 Section 3.2.

3.1.1 Properties of Key Updates

In this section, 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 [LT18, KLR19, BDGJ20]) is used for doing cryptographic seperation. We follow the firewall definition in [BDGJ20] and use firewall set $\mathcal{FW}$ (defined in [BDGJ20]) to track each insulated region and its firewalls.

Definition 9. An *insulated region* with *firewalls* fwl and fwr is a consecutive sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ for which:

- no key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ is corrupted, i.e. $\{\mathsf{fwl}, \ldots, \mathsf{fwr}\} \cap \mathcal{K} = \emptyset$;
- the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are not corrupted (if they exist), i.e.$\mathsf{fwl}, \mathsf{fwr} + 1 \notin \mathcal{T}$;
- all tokens $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ are corrupted, i.e.$\{\mathsf{fwl} + 1, \ldots, \mathsf{fwr}\} \subseteq \mathcal{T}$.

Remark 3.3. Based on Definition 9, 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 3.1 states that $\mathcal{IR}$ is the complementary set of $\mathcal{K}^*_{\mathsf{bi}}$. Furthermore, Lemma 3.3 shows that the complementary set of $\mathcal{IR}$ is the union of two types of epoch sets (see Definition 10 and Definition 11).

Lemma 3.1. 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 $t \geq 0$.

In the bi-directional key update setting, if the adversary has corrupted a key in an epoch e, where $\mathsf{e} \in \{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$, then the adversary can infer all keys from epoch $\bar{\mathsf{e}}_{i-1}$ to epoch $\bar{\mathsf{e}}_i - 1$, that is $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, 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 $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is corrupted, then $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is an insulated region . Therefore, for any i, $\{\bar{\mathsf{e}}_{i-1}, \ldots, \bar{\mathsf{e}}_i - 1\}$ is either an insulated region or a subset of $\mathcal{K}^*_{\mathsf{bi}}$. $\square$

Epoch		$\{0\}$		1		2		3		$\{4$		$5\}$		6		7		8	
$\mathcal{K}$		×		×		✓		×		×		×		✓		✓		×	
$\mathcal{T}$	×		×		✓		✓		×		✓		×		×		✓		×
$\mathcal{K}^*_{\mathsf{bi}}$		×		✓		✓		✓		×		×		✓		✓		✓	
$\mathcal{T}^*_{\mathsf{bi}}$	×		×		✓		✓		×		✓		×		✓		✓		×

Figure 12: An example of Lemma 3.1, where $\mathcal{K} = \{2, 6, 7\}$ and $\mathcal{T} = \{2, 3, 5, 8\}$. So $\{0, \ldots, l+1\} \setminus \mathcal{T} = \{0, 1, 4, 6, 7, 9\}$, insulated regions are $\{0\}$ and $\{4, 5\}$. $\mathcal{IR} = \{0, 4, 5\}$ and $\mathcal{K}^*_{\mathsf{bi}} = \{1, 2, 3, 6, 7, 8\}$. $\times$ indicates the keys/tokens are not revealed to the adversary, $\checkmark$ indicates the keys/tokens are revealed to the adversary.

We define two types of epoch sets in Definition 10 and Definition 11, 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. 13.

Definition 10. A set of *type1* epochs is a consecutive sequence of epochs $(\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}})$ for which:

- no key in the sequence of epochs $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}} - 1\}$ is corrupted, i.e.$\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}} - 1\} \cap \mathcal{K} = \emptyset$;

- the key in epoch $\mathsf{e}_{\mathsf{end}}$ is corrupted, i.e.$\mathsf{e}_{\mathsf{end}} \in \mathcal{K}$;
- all tokens $\{\Delta_{\mathsf{e}_{\mathsf{start}}+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}\}$ are corrupted, i.e. $\{\mathsf{e}_{\mathsf{start}} + 1, \ldots, \mathsf{e}_{\mathsf{end}}\} \subseteq \mathcal{T}$.

Definition 11. A set of *type2* epochs is a consecutive sequence of epochs $(\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}})$ for which:

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

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

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

Figure 13: 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.

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 3.3.

Lemma 3.2. 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 $\{\mathsf{e} + 1, \ldots, \mathsf{e}_u\} \subseteq \mathcal{T}$.

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 $\mathsf{e}+1, \ldots, \mathsf{e}_u \in \mathcal{T}$. 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 3.3. For any sets $\mathcal{K}, \mathcal{T} \subseteq \{0, \ldots, l\}$, we have $\{0, \ldots, l\} \setminus \mathcal{IR} = (\cup_{\mathsf{type}\ 1}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\mathsf{type}\ 2}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\})$, where the two types of epoch sets are defined in Definition 10 and Definition 11.

Proof. Suppose $\mathsf{e} \in \{0, \ldots, l\} \setminus \mathcal{IR}$, by Lemma 3.1, we have $\mathsf{e} \in \mathcal{K}^*_{\mathsf{bi}}$. If $\mathsf{e} \notin \mathcal{K}^*_{\mathsf{uni}}$, we can apply Lemma 3.2 and have a set of type 1 epochs, assume $\{\mathsf{e}, \ldots, \mathsf{e}_u\}$. 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 3.4. As a conclusion of Lemma 3.1 and Lemma 3.3, 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. $\{0, \ldots, l\} = (\cup_{(\mathsf{fwl},\mathsf{fwr}) \in \mathcal{FW}}\{\mathsf{fwl}, \ldots, \mathsf{fwr}\}) \cup (\cup_{\mathsf{type}\ 1}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}) \cup (\cup_{\mathsf{type}\ 2}\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\})$.

3.1.2 Trivial Win Equivalences

In this section we 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 bi-directional update setting are triggered then the same trivial win conditions in the uni-directional update setting would be triggered. We will use these trivial win equivalences to prove the relation between uni- and bi-directional variants of security notions in Theorem 3.1. Additionally, we use these trivial win equivalences to prove the relation between the no-directional and the directional variants of integrity notions in Theorem 3.3.

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

Lemma 3.4. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{kk} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{bi}}$, $\mathcal{L}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{L}^*_{\mathsf{kk},\mathsf{bi}}$, and $\mathcal{Q}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk},\mathsf{bi}}$.

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 $\mathcal{K}^*_{\mathsf{kk}}$, $\mathcal{T}^*_{\mathsf{kk}}$, $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}}$ and $\mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$ using Equations (1, 2, 3, 4, 5). Then we have $\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{kk},\mathsf{bi}}$. Furthermore, we use algorithms discussed in Section 2.7.2 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{kk},\mathsf{bi}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{kk},\mathsf{bi}}$, $\mathcal{L}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{L}^*_{\mathsf{kk},\mathsf{bi}}$, and $\mathcal{Q}^*_{\mathsf{kk},\mathsf{uni}} \subseteq \mathcal{Q}^*_{\mathsf{kk},\mathsf{bi}}$. □

Lemma 3.5. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ and any $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}} \subseteq \mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{cc}}$, $\mathcal{L}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{bi},\mathsf{cc}}$ and $\mathcal{Q}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi},\mathsf{cc}}$.

Proof. The proof is similar to the proof of Lemma 3.4. 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 $\mathcal{K}^*_{\mathsf{uni}}$, $\mathcal{K}^*_{\mathsf{bi}}$, $\mathcal{T}^*_{\mathsf{uni}}$, $\mathcal{T}^*_{\mathsf{bi}}$, $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}}$ and $\mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$ 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 $\mathcal{C}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{cc}}$. Furthermore, we use algorithms discussed in Section 2.7.2 to compute ciphertext/message leakage sets $\tilde{\mathcal{L}}^*, \tilde{\mathcal{Q}}^*, \mathcal{L}^*, \mathcal{Q}^*$. Similarly we get $\tilde{\mathcal{L}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{L}}^*_{\mathsf{bi},\mathsf{cc}}$, $\tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{cc}}$, $\mathcal{L}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{L}^*_{\mathsf{bi},\mathsf{cc}}$ and $\mathcal{Q}^*_{\mathsf{uni},\mathsf{cc}} \subseteq \mathcal{Q}^*_{\mathsf{bi},\mathsf{cc}}$. □

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

Lemma 3.6. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset$.

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

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

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

Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset$. 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 $\{0, \ldots, l\} \setminus \mathcal{IR}$, then the adversary cannot infer any challenge-equal ciphertext in this set even in the bi-directional update setting. That is, $\mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \cap (\{0, \ldots, l\} \setminus \mathcal{IR}) = \emptyset$. However, $\{0, \ldots, l\} \setminus \mathcal{IR} \overset{\text{Lemma 3.1}}{=} \mathcal{K}^*_{\mathsf{bi}}$, then $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$, which contradicts with the assumption. Therefore, there exists an epoch $\mathsf{e}' \in \{0, \ldots, l\} \setminus \mathcal{IR}$ such that the adversary has asked for a challenge-equal ciphertext in this epoch, that is $\mathsf{e}' \in \mathcal{C}$.

By Lemma 3.3, we know that e' is located in an epoch set which is either type 1 or type 2. Suppose $\mathsf{e}' \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$, 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 $\Delta_{\mathsf{e}'+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}$ 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 $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \neq \emptyset$. □

As a corollary of Lemma 3.4 to 3.6, we have the following equivalence. We only provide Corollary 3.1 with a fully detailed proof, since we will use similar proof techniques for Corollary 3.2 to 3.3.

Corollary 3.1. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset$.

Proof. By Lemma 3.4, we have $\mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \subseteq \mathcal{C}^*_{\mathsf{uni},\mathsf{bi}}$. By Lemma 3.5, we have $\mathcal{C}^*_{\mathsf{uni},\mathsf{bi}} \subseteq \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}}$. Hence, $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \subseteq \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{bi}} \subseteq \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}}$. By Lemma 3.6, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{bi}} \neq \emptyset$.

Similarly, we have $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \overset{\text{Lemma 3.5}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{uni}} \overset{\text{Lemma 3.4}}{\subseteq} \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}}$ and therefore $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{uni}} \neq \emptyset$. □

Remark 3.5. 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 3.1 we have $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$. By Lemma 3.1, we have $\{0, \ldots, l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. Therefore, $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \{0, \ldots, l\} \setminus \mathcal{K}^*_{\mathsf{bi}} = \mathcal{IR}$. 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. 14.

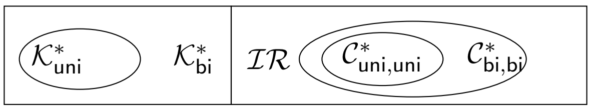
$\mathcal{K}^*_{\mathsf{uni}}$
$\mathcal{K}^*_{\mathsf{bi}}$
$\mathcal{IR}$
$\mathcal{C}^*_{\mathsf{uni,uni}}$
$\mathcal{C}^*_{\mathsf{bi,bi}}$

Figure 14: 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 3.7. For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$. Suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk,cc}} = \emptyset$, 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 $\underline{\text{epoch } \mathsf{e}}$. 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 3.8. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

Proof. By Remark 3.5 we have $\mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \mathcal{IR}$. By Remark 2.1 we have $(\tilde{\mathbf{c}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}^* \iff \mathsf{e} \in \mathcal{C}^*$. Therefore, if $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}}$ we have $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni,uni}} \subseteq \mathcal{C}^*_{\mathsf{bi,bi}}$ and $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

If $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$, then $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}} \subseteq \mathcal{IR}$. 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 $\{\mathsf{fwl}, \ldots, \mathsf{e}\} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, which contradicts with $\mathsf{e} \in \mathcal{C}^*_{\mathsf{bi,bi}} \cap \{\mathsf{fwl}, \ldots, \mathsf{e}\}$. 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. $\mathsf{e} \in \mathcal{C}^*_{\mathsf{uni,uni}}$. Therefore, $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}}$ as well. □

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.8, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.2. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,uni}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{bi,bi}}$.

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

Lemma 3.9. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. 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 3.8. We use the property that $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^* \iff \mathsf{e} \in \mathcal{C}^*$. □

As a corollary of Lemma 3.4, Lemma 3.5 and Lemma 3.9, we have the following result. The proof is similar to the proof of Corollary 3.1.

Corollary 3.3. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \emptyset$, then $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni},\mathsf{bi}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{uni}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{bi},\mathsf{bi}}$.

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

Lemma 3.10. The trivial win condition " $\mathsf{e} \in \hat{\mathcal{K}}^*$" is either triggered or not triggered in all six variants of an integrity notion.

Proof. By Equation (6), set $\hat{\mathcal{K}}^*$ is independent from the key updates, so we have the desired result. □

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

Lemma 3.11. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \hat{\mathcal{K}}^*$, then $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk}',\mathsf{cc}'}$ for any $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}, \mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Proof. If $\mathsf{e} \notin \mathcal{T}$, the adversary cannot update ciphertexts from previous epoch to epoch e. Then we have the desired result.

Suppose $\mathsf{e} \in \mathcal{T}$. Since $\mathsf{e} \notin \hat{\mathcal{K}}^*$, by Equation (6), there exists an epoch $\mathsf{e}_1 (< \mathsf{e})$ such that tokens $\Delta_{\mathsf{e}_1+1}, \dots, \Delta_{\mathsf{e}}$ are corrupted $(\mathsf{e}_1 + 1, \dots, \mathsf{e} \in \mathcal{T})$, but token Δ_{e_1} and keys $\mathbf{k}_{\mathsf{e}_1}, \dots, \mathbf{k}_{\mathsf{e}}$ are unknown to the adversary ($\mathsf{e}_1 \notin \mathcal{T}$ and $\mathsf{e}_1, \dots, \mathsf{e} \notin \mathcal{K}$). If the adversary never asks for any ciphertext in epochs $\{\mathsf{e}_1, \dots, \mathsf{e}\}$, then there is no ciphertext in epoch e located in the set $\mathcal{L}^*_{\mathsf{kk},\mathsf{cc}}$ for any $(\mathsf{kk}, \mathsf{cc})$ (because token Δ_{e_1} is unknown to the adversary, it is impossible to update ciphertexts from an epoch earlier than epoch e_1 to epoch e). For all ciphertexts the adversary learns in an epoch i with $i \in \{\mathsf{e}_1, \dots, \mathsf{e}\}$, the adversary can update them to epoch e using tokens. Hence, we have $(\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk},\mathsf{cc}} \iff (\mathbf{c}, \mathsf{e}) \in \mathcal{L}^*_{\mathsf{kk}',\mathsf{cc}'}$ for any $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{no}, \mathsf{uni}, , \mathsf{bi}\}, \mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$. □

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

Lemma 3.12. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \dots, \mathsf{e}\}$. Suppose $\mathsf{e} \notin \hat{\mathcal{K}}^*$, then $(\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{kk},\mathsf{cc}} \iff (\mathbf{m}', \mathsf{e}) \in \mathcal{Q}^*_{\mathsf{kk}',\mathsf{cc}'}$ for any $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}, \mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$.

Proof. The proof is similar to the proof of Lemma 3.11. □

3.2 Relations among Security Notions

In Fig. 15, Fig. 16 and Fig. 17, we show the relationship among six variants of the same security notion for UE schemes.

Fig. 15 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. 16, 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.

Fig. 17 shows that the six variants of integrity notions are equivalent.

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.

$$\begin{tikzcd}
(\mathsf{bi}, \mathsf{bi})\text{-}\mathsf{notion} \arrow[r, leftrightarrow, "\text{Thm. 3.1}"] & (\mathsf{bi}, \mathsf{uni})\text{-}\mathsf{notion} \arrow[r, leftrightarrow, "\text{Thm. 3.1}"] & (\mathsf{uni}, \mathsf{bi})\text{-}\mathsf{notion} \arrow[r, leftrightarrow, "\text{Thm. 3.1}"] & (\mathsf{uni}, \mathsf{uni})\text{-}\mathsf{notion}
\end{tikzcd}$$

Figure 15: Relations among the uni- and bi-directional update variants of the same security notion, where notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-PTXT}.

$$\begin{tikzcd}
(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion} \arrow[r, Rightarrow, shift left, "\text{Lem. 3.14}"] & (\mathsf{no}, \mathsf{bi})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "/" marking, "\text{Thm. 3.3}"] \arrow[r, Rightarrow, shift left, "\text{Lem. 3.13}"] & (\mathsf{kk}, \mathsf{cc})\text{-}\mathsf{notion} \arrow[l, Rightarrow, shift left, "\text{Thm. 3.2}"]
\end{tikzcd}$$

Figure 16: Relations among the six variants of the same confidentiality notion, where kk, cc $\in$ {uni, bi} and notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA}.

$$\begin{tikzcd}
(\mathsf{no}, \mathsf{cc})\text{-}\mathsf{notion} \arrow[r, Leftrightarrow, "\text{Thm. 3.4}"] & (\mathsf{kk}, \mathsf{cc})\text{-}\mathsf{notion}
\end{tikzcd}$$

Figure 17: Relations among the six variants of the same integrity notion, where kk, cc $\in$ {uni, bi} and notion $\in$ {INT-CTXT, INT-PTXT}.

Remark 3.6 (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 any key update settings, the adversary can create ciphertexts in epoch 1 and use token Δ_2 to update ciphertexts from epoch 1 to epoch 2, these updated ciphertexts are valid forgeries in epoch 2. Hence, the adversary can trivially win the integrity game.

Remark 3.7. We can define six variants of the IND-ENC and the IND-UPD notions[3] and can extend the result in Fig. 15 and Fig. 16 for notion $\in$ {IND-ENC, IND-UPD}.

3.2.1 Relations between the Uni- and Bi-Directional Update Variants of Security Notions

We will only provide Theorem 3.1 with a fully detailed proof, since we will use similar proof techniques for Lemmas 3.13, 3.14 and Theorems 3.2, 3.4.

[3] Note that IND-ENC and IND-UPD are introduced in [LT18]

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

Theorem 3.1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-PTXT}. For any $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$ and any $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$ against UE, there exists a $(\mathsf{kk}', \mathsf{cc}')$-notion adversary $\mathcal{B}_{3.1}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-notion}}_{\mathsf{UE},\, \mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-notion}}_{\mathsf{UE},\, \mathcal{B}_{3.1}}(1^\lambda).$$

Proof. We construct a reduction $\mathcal{B}_{3.1}$ 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 Section 3.1.2 (Corollary 3.1 to 3.3, Lemma 3.7 and Lemma 3.10 to 3.12), 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 $\mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-notion}}_{\mathsf{UE},\, \mathcal{B}_{3.1}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-notion}}_{\mathsf{UE},\, \mathcal{A}}(1^\lambda)$. □

Remark 3.8. For any security notion notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-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 scheme. For simplicity, we will denote the notion $(\mathsf{bi}, \mathsf{bi})$-notion as notion.

3.2.2 Relations between the No-Directional and the Directional Variants of Security Notions

General Relations. We prove $(\mathsf{no}, \mathsf{bi})$-notion implies $(\mathsf{bi}, \mathsf{bi})$-notion in Lemma 3.13, combining this result with Theorem 3.1 we have that $(\mathsf{no}, \mathsf{bi})$-notion implies any $(\mathsf{kk}, \mathsf{cc})$-notion, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Lemma 3.13. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-PTXT}. For any $(\mathsf{bi}, \mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists an $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{B}_{3.13}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{bi},\mathsf{bi})\text{-notion}}_{\mathsf{UE},\, \mathcal{A}}(1^\lambda) \leq \mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-notion}}_{\mathsf{UE},\, \mathcal{B}_{3.13}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.13}$ running the $(\mathsf{no}, \mathsf{bi})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{bi}, \mathsf{bi})$-notion adversary $\mathcal{A}$. Consider the leakage sets. For bi-directional ciphertext updates, no-directional key updates has less leakage than bi-directional key updates.

If $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{bi}, \mathsf{bi})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{no}, \mathsf{bi})$-notion game either. Hence the reduction has as least $\mathbf{Adv}^{(\mathsf{bi},\mathsf{bi})\text{-notion}}_{\mathsf{UE},\, \mathcal{A}}(1^\lambda)$ advantage to win its $(\mathsf{no}, \mathsf{bi})$-notion game. □

The following lemma shows that $(\mathsf{no}, \mathsf{uni})$-notion implies $(\mathsf{no}, \mathsf{bi})$-notion.

Lemma 3.14. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in$ {detIND-UE-CPA, randIND-UE-CPA, detIND-UE-CCA, randIND-UE-CCA, INT-CTXT, INT-PTXT}. For any $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists an $(\mathsf{no}, \mathsf{uni})$-notion adversary $\mathcal{B}_{3.14}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) \leq \mathbf{Adv}^{(\mathsf{no},\mathsf{uni})\text{-notion}}_{\mathsf{UE},\,\mathcal{B}_{3.14}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.14}$ running the $(\mathsf{no}, \mathsf{uni})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$. Consider the leakage sets. For no-directional key updates, uni-directional ciphertext updates has less leakage than bi-directional ciphertext updates.

If $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{no}, \mathsf{bi})$-notion game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{no}, \mathsf{uni})$-notion game either. Hence the reduction has as least $\mathbf{Adv}^{(\mathsf{no},\mathsf{bi})\text{-notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda)$ advantage to win its $(\mathsf{no}, \mathsf{uni})$-notion game. □

Relations among Confidentiality Notions. Before we start to prove any relation in this section, we consider the equivalences for the following trivial win conditions "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$", "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{\mathbf{c}})$ is queried" (already discussed on page 19), "$(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*$" and "$(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$". These results will be used to prove Theorem 3.2.

Lemma 3.15. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, l\}$, we have $\mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$.

Proof. Similar to the proof of Lemma 3.6, it suffices to prove

$$\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}} \neq \emptyset.$$

Suppose $\mathcal{K}^*_{\mathsf{uni}} \cap \mathcal{C}^*_{\mathsf{uni,bi}} \neq \emptyset$, as the analysis of Lemma 3.6, there exists an epoch $\mathsf{e}' \in \{0, \ldots, l\} \setminus \mathcal{IR}$ such that the adversary has asked for a challenge-equal ciphertext in this epoch, that is $\mathsf{e}' \in \mathcal{C}$. By Lemma 3.3, we know that e' is located in an epoch set which is either type 1 or type 2. Suppose $\mathsf{e}' \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is the biggest such set around epoch e'.

If e' is located in a type 1 epoch set, then $\mathbf{k}_{\mathsf{e}_{\mathsf{end}}}$ and $\Delta_{\mathsf{e}'+1}, \ldots, \Delta_{\mathsf{e}_{\mathsf{end}}}$ are corrupted. 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 $\mathsf{e}_{\mathsf{end}} \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$.

If e' is located in a type 2 epoch set, we claim that any epoch $\mathsf{e} \in \{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is either in $\mathcal{K}$ or $\mathcal{T}$ or both, because revealed epoch keys in the uni-directional update setting is either a corrupted key or a key inferred from prior epoch via a corrupted update token. Furthermore, we claim that $\mathsf{e}_{\mathsf{start}} \in \mathcal{K}$. Otherwise, we have $\mathsf{e}_{\mathsf{start}} \in \mathcal{T}$ and $\mathsf{e}_{\mathsf{start}} - 1 \in \mathcal{K}^*_{\mathsf{uni}}$ (in this situation, $\mathbf{k}_{\mathsf{e}_{\mathsf{start}}}$ is inferred from $\mathbf{k}_{\mathsf{e}_{\mathsf{start}}-1}$) and find a bigger type 2 set, which is contradict with the assumption that $\{\mathsf{e}_{\mathsf{start}}, \ldots, \mathsf{e}_{\mathsf{end}}\}$ is the biggest type 2 set around epoch e'. Then we discuss the joint set $\mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. If $\mathsf{e}' \in \mathcal{K}$, then we have $\mathsf{e}' \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. If $\mathsf{e}' \notin \mathcal{K}$, then $\mathsf{e}' \in \mathcal{T}$. Iteratively, we can find an epoch $\mathsf{e}_s < \mathsf{e}'$ such that $\mathsf{e}_s \in \mathcal{K}$ and $\mathsf{e}_s + 1, \ldots, \mathsf{e}' \in \mathcal{T}$. Hence, the adversary can reversely update the challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}'}$ from epoch e' to epoch e_s to know $\tilde{\mathbf{c}}_{\mathsf{e}_s}$. Then we have $\mathsf{e}_s \in \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}}$. □

The proof of the following lemma is similar to the proof of Lemma 3.8.

Lemma 3.16. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{uni,bi}} \iff (\mathbf{c}, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{no,bi}}$.

The proof of the following lemma is similar to the proof of Lemma 3.9.

Lemma 3.17. For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, \ldots, \mathsf{e}\}$. Suppose $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \emptyset$, then $(\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{uni,bi}} \iff (\mathbf{m}', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{no,bi}}$.

The following theorem shows that the $(\mathsf{uni}, \mathsf{bi})$- variant of confidentiality notions implies the $(\mathsf{no}, \mathsf{bi})$- variant of the corresponding confidentiality notions. Combining this result with Theorem 3.1 we have that any $(\mathsf{kk}, \mathsf{cc})$- variant of confidentiality notions implies the $(\mathsf{no}, \mathsf{bi})$- variant of the corresponding confidentiality notions, where $\mathsf{kk}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$.

Theorem 3.2. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and notion $\in \{\mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. For any $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$ against UE, there exists a $(\mathsf{uni}, \mathsf{bi})$-notion adversary $\mathcal{B}_{3.2}$ against UE such that

$$\mathbf{Adv}_{\mathsf{UE},\,\mathcal{A}}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}(1^\lambda) = \mathbf{Adv}_{\mathsf{UE},\,\mathcal{B}_{3.2}}^{(\mathsf{uni},\mathsf{bi})\text{-}\mathsf{notion}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.2}$ running the $(\mathsf{uni}, \mathsf{bi})$-notion experiment which will simulate the responses of queries made by the $(\mathsf{no}, \mathsf{bi})$-notion adversary $\mathcal{A}$.

By the analysis of trivial win equivalences (Lemmas 3.15, 3.7, 3.16 and 3.17), we have that the trivial win results are the same in both the $(\mathsf{no}, \mathsf{bi})$-notion game and the $(\mathsf{uni}, \mathsf{bi})$-notion game. Hence, the reduction perfectly simulates the $(\mathsf{no}, \mathsf{bi})$-notion game to adversary $\mathcal{A}$. And we have $\mathbf{Adv}_{\mathsf{UE},\,\mathcal{A}}^{(\mathsf{no},\mathsf{bi})\text{-}\mathsf{notion}}(1^\lambda) = \mathbf{Adv}_{\mathsf{UE},\,\mathcal{B}_{3.2}}^{(\mathsf{uni},\mathsf{bi})\text{-}\mathsf{notion}}(1^\lambda)$. □

The following Theorem states that the $(\mathsf{no}, \mathsf{bi})$- variant of confidentiality notions do not imply the $(\mathsf{no}, \mathsf{uni})$- variant of the corresponding confidentiality notions.

Theorem 3.3. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Let α_{notion} be the $(\mathsf{no}, \mathsf{bi})$-notion advantage of an adversary $\mathcal{A}$ against UE, where notion $\in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. Then there exists a modified scheme $\mathsf{UE}_1^{\mathsf{new}}$, detailed in Fig. 18, such that the $(\mathsf{no}, \mathsf{bi})$-notion advantage of $\mathcal{A}$ against $\mathsf{UE}_1^{\mathsf{new}}$ is $\alpha_{\mathsf{notion}} + l\mathbf{Adv}_{\mathsf{SKE}}^{\mathsf{IND\$\text{-}CPA}}$, and there exists an $(\mathsf{no}, \mathsf{uni})$-notion adversary $\mathcal{B}_{3.3}$ against $\mathsf{UE}_1^{\mathsf{new}}$ that wins with probability 1.

Proof technique of Theorem 3.3. For a confidentiality notion notion, we use a $(\mathsf{no}, \mathsf{bi})$-notion secure UE scheme UE to construct a new UE scheme $\mathsf{UE}_1^{\mathsf{new}}$, which is still $(\mathsf{no}, \mathsf{bi})$-notion secure but not $(\mathsf{no}, \mathsf{uni})$-notion secure. As a result, $(\mathsf{no}, \mathsf{uni})$-notion is strictly stronger than $(\mathsf{no}, \mathsf{bi})$-notion.

```
$\mathsf{UE}_1^{\mathsf{new}}.\mathsf{KG}(1^\lambda)$ :
  $\mathbf{k} \xleftarrow{\$} \mathsf{UE.KG}(1^\lambda)$
  return $\mathbf{k}$

$\mathsf{UE}_1^{\mathsf{new}}.\mathsf{TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1})$ :
  $\Delta^1_{\mathsf{e}+1} \leftarrow \mathsf{UE.TG}(\mathbf{k}_\mathsf{e}, \mathbf{k}_{\mathsf{e}+1})$
  $sk_{\mathsf{e}+1} \xleftarrow{\$} \mathsf{SKE.KG}$
  return $(\Delta^1_{\mathsf{e}+1}, sk_{\mathsf{e}+1})$

$\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$ :
  $\mathbf{c}^1_\mathsf{e} \xleftarrow{\$} \mathsf{UE.Enc}(\mathbf{k}_\mathsf{e}, \mathbf{m})$
  $\mathbf{c}^2_\mathsf{e} \xleftarrow{\$} \mathcal{CS}$
  return $(\mathbf{c}^1_\mathsf{e}, \mathbf{c}^2_\mathsf{e})$

$\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}_\mathsf{e})$ :
  parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}^1_\mathsf{e}, \mathbf{c}^2_\mathsf{e})$
  $\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathbf{k}_\mathsf{e}, \mathbf{c}^1_\mathsf{e})$
  return $\mathbf{m}'$

$\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Upd}((\Delta^1_{\mathsf{e}+1}, sk_{\mathsf{e}+1}), \mathbf{c}_\mathsf{e})$ :
  parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}^1_\mathsf{e}, \mathbf{c}^2_\mathsf{e})$
  $\mathbf{c}^1_{\mathsf{e}+1} \leftarrow \mathsf{UE.Upd}(\Delta^1_{\mathsf{e}+1}, \mathbf{c}^1_\mathsf{e})$
  $\mathbf{c}^2_{\mathsf{e}+1} \leftarrow \mathsf{SKE.Enc}(sk_{\mathsf{e}+1}, \mathbf{c}^1_\mathsf{e})$
  return $(\mathbf{c}^1_{\mathsf{e}+1}, \mathbf{c}^2_{\mathsf{e}+1})$
```

Figure 18: Updatable encryption scheme $\mathsf{UE}_1^{\mathsf{new}}$ for proof of Theorem 3.3, built from IND\$-CPA-secure SKE SKE and updatable encryption scheme UE.

Proof. $\underline{\mathsf{UE}_1^{\mathsf{new}} \text{ is not } (\mathsf{no}, \mathsf{uni})\text{-notion secure.}}$ If a challenge-equal ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}+1}$ and the corresponding epoch update token $(\Delta^1_{\mathsf{e}+1}, sk_{\mathsf{e}+1})$ are corrupted then the adversary can compute the prior challenge-equal ciphertext $\tilde{\mathbf{c}}^1_{\mathsf{e}}$ by using $sk_{\mathsf{e}+1}$ to decrypt $\tilde{\mathbf{c}}^2_{\mathsf{e}+1}$. Then the adversary can corrupt a key $\mathbf{k}_{\mathsf{e}}$ to win the $(\mathsf{no}, \mathsf{uni})$-notion game without trigger the trivial win conditions.

$\underline{\mathsf{UE}_1^{\mathsf{new}} \text{ is } (\mathsf{no}, \mathsf{bi})\text{-notion secure.}}$ All algorithms for $\mathsf{UE}_1^{\mathsf{new}}$ are the same as for UE, except for the encryption algorithm $\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Enc}$ and the update algorithm $\mathsf{UE}_1^{\mathsf{new}}.\mathsf{Upd}$, where we use a SKE scheme SKE to make the $\mathsf{UE}_1^{\mathsf{new}}$ scheme has bi-directional ciphertext updates. This does not affect an adversary's ability to win the $(\mathsf{no}, \mathsf{bi})$-notion game. The additional advantage term $l\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{SKE}}$ is because if the adversary knows a challenge-equal ciphertext, but not a token, then the second challenge-equal ciphertext term will not leak the prior challenge-equal ciphertext. □

Relations among Integrity Notions. The following Theorem states that for any $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$ and $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$, $(\mathsf{kk}', \mathsf{cc}')$-notion implies $(\mathsf{kk}, \mathsf{cc})$-notion. Consequently, all six variants of the same integrity notion are equivalent.

Theorem 3.4. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$ against UE, where notion $\in \{\mathsf{INT\text{-}CTXT}, \mathsf{INT\text{-}PTXT}\}$, there exists an $(\mathsf{kk}', \mathsf{cc}')$-notion adversary $\mathcal{B}_{3.4}$ against UE such that

$$\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{B}_{3.4}}(1^\lambda).$$

Proof. The proof is similar to the proof of Theorem 3.1, we construct a reduction $\mathcal{B}_{3.4}$ running the $(\mathsf{kk}', \mathsf{cc}')$-notion experiment which will simulate the responses of queries made by the $(\mathsf{kk}, \mathsf{cc})$-notion adversary $\mathcal{A}$. Suppose $\mathcal{A}$ queries a $\mathcal{O}.\mathsf{Try}$ oracle in epoch e.

By the analysis of trivial win equivalences in Section 3.1.2 (Lemma 3.10 to 3.12), 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 $\mathbf{Adv}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{A}}(1^\lambda) = \mathbf{Adv}^{(\mathsf{kk}',\mathsf{cc}')\text{-}\mathsf{notion}}_{\mathsf{UE},\,\mathcal{B}_{3.4}}(1^\lambda)$. □

4 LWE-based PKE Scheme

In this section, we look at an LWE-based PKE scheme LWEPKE, which is detailed in Fig. 19. We prove that LWEPKE is $\mathsf{IND\$\text{-}CPA}$-secure, if the underlying LWE problem is hard. We will later use this PKE scheme to construct an updatable encryption scheme in Section 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 $\frac{q}{2} \mod q$.

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

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

Figure 19: 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 $\{0,1\}^{1\times t}$.

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,
- $m = \mathsf{poly}(n)$ 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 $n \leq (k - 2\log(1/\epsilon) - O(1))/\log(q)$ 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 $\beta \leq \frac{q}{8}$ and $\alpha B/\beta = \mathsf{negl}(n)$.
- $D_{\mathbb{Z},\alpha}$ and $D_{\mathbb{Z},\beta}$ be two discrete Gaussian distributions.

Remark 4.1. 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 $\mathsf{LWEPKE.Dec}$ 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 $\mathbf{0} \mod q$.

Let $\mathbf{R} = [\mathbf{r}_1,\ldots,\mathbf{r}_t]$. By Lemma A.1, we have that each entry of $\mathbf{e}$ have size at most α with overwhelming probability. As each entry of $\mathbf{R}$ has size at most B, then each entry of $\mathbf{e}^\intercal \cdot \mathbf{R}$ has size at most $\alpha \cdot B$ with overwhelming probability. That is, for all $1 \leq j \leq t$, $|\mathbf{e}^\intercal \cdot \mathbf{r}_j| \leq \alpha \cdot B$. Hence, $|\mathbf{e}^\intercal \cdot \mathbf{r}_j|/\beta \leq \alpha \cdot B/\beta = \mathsf{negl}(n)$ is negligible, and by Lemma A.2 we have that $D_{\mathbb{Z},\beta}$ and $D_{\mathbb{Z},\beta} + \mathbf{e}^\intercal \cdot \mathbf{r}_j$ are statistically close. Therefore, $D_{\mathbb{Z},\beta}^{1\times t}$ is statistically close to $D_{\mathbb{Z},\beta}^{1\times t} + \mathbf{e}^\intercal \cdot \mathbf{R}$, so $\mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' \overset{s}{\approx} \mathbf{e}'$. By Lemma A.1, we have that each entry of $\mathbf{e}'$ has size at most β with overwhelming probability. Since $\beta \leq \frac{q}{8}$, we have the desired result.

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

Theorem 4.1. Let LWEPKE be the public key encryption described in Fig. 19, using the parameter setting described in Section 4.1. Then for any adversary $\mathsf{IND\$\text{-}CPA}$ $\mathcal{A}$ 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. The proof of theorem 4.1 consists of a sequence of games. Denote E_i be the event that the adversary's guess $\mathrm{b}' = 1$ in game i.

Game 0

The first game is the experiment $\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{LWEPKE},\,\mathcal{A}}$, given in Fig. 1. Then we have

$$\mathbf{Pr}[E_0] = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}0}}_{\mathsf{LWEPKE},\,\mathcal{A}} = 1].$$

Game 1

We consider a modified game, which is the same as Game 0 except for the second component of the challenge ciphertext ($\mathbf{c}_2$) is generated as $\mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}' + \frac{q}{2}\mathbf{m}$ instead of $\mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' + \frac{q}{2}\mathbf{m}$.

In game 0, $\mathbf{c}_2 - \frac{q}{2}\mathbf{m} = \mathbf{p}^\intercal \cdot \mathbf{R} + \mathbf{e}' = (\mathbf{A}\cdot\mathbf{s}+\mathbf{e})^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{A}^\intercal \cdot \mathbf{R} + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{A}^\intercal \cdot \mathbf{R} + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' = \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}'$. Similar to the correctness analysis, we know by Lemma A.1 and Lemma A.2 that $\mathbf{e}^\intercal \cdot \mathbf{R} + \mathbf{e}' \overset{s}{\approx} \mathbf{e}'$. Thus,

$$|\mathbf{Pr}[E_0] - \mathbf{Pr}[E_1]| \le \mathsf{negl}(n).$$

Game 2

We consider a modified game that is the same as Game 1, except that the first component of the challenge ciphertext ($\mathbf{C}_1$) is sampled from the uniform distribution over $\mathbb{Z}_q^{n\times t}$.

From the leftover hash lemma (Lemma. A.6) we know that the joint distribution of $(\mathbf{A}, \mathbf{A}^\intercal \cdot \mathbf{R})$ is $t\epsilon$-close to the uniform distribution over $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{n\times t}$, since the min-entropy of $\mathbf{r}_i$ is at least $n\log(q) + 2\log(1/\epsilon) + O(1)$. That is, $(\mathbf{A}, \mathbf{C}_1) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}_1)$, for a uniformly random matrix $\mathbf{U}_1$, and we have

$$|\mathbf{Pr}[E_1] - \mathbf{Pr}[E_2]| \le t\epsilon.$$

Game 3

We consider a modified game that is the same as Game 2, except that the second component of the ciphertext ($\mathbf{c}_2$) is sampled from the uniform distribution over $\mathbb{Z}_q^{1\times t}$.

Since the $\mathsf{DLWE}_{n,q,\alpha}$ problem is hard, the $\mathsf{DLWE}_{n,q,\beta}$ problem is also hard by Lemma A.4. Therefore, the joint distribution of $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}')$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n\times t} \times \mathbb{Z}_q^{1\times t}$, under the $\mathsf{DLWE}_{n,q,\beta}$ problem is hard. That is, $(\mathbf{C}_1, \mathbf{s}^\intercal \cdot \mathbf{C}_1 + \mathbf{e}') \overset{c}{\approx} (\mathbf{C}_1, \mathbf{u}_2)$, for a uniformly random vector $\mathbf{u}_2$, and we have

$$|\mathbf{Pr}[E_2] - \mathbf{Pr}[E_3]| \le \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

We know that $\mathbf{Pr}[E_3] = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\$\text{-}CPA\text{-}1}}_{\mathsf{LWEPKE},\,\mathcal{A}} = 1]$. By Definition 1, we have $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\,\mathcal{A}}(1^\lambda) = |\Pr[E_3] - \Pr[E_0]|$, which concludes the proof. $\square$

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. 19). 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 [LT18]. 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. 20.

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

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

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

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

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

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

Figure 20: The algorithms of LWE-based updatable encryption scheme LWEUE, which is parameterized by an LWE-based PKE scheme LWEPKE.

Parameter Setting. We use the parameter setting of the scheme LWEPKE, described in Section 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 $\Delta^{\mathsf{s}}_{\mathsf{e}+1} (= \mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}})$ 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.

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 $\mathbf{p}_i - \mathbf{p}_{i-1} - \mathbf{A} \cdot \Delta^{\mathsf{s}}_i = \mathbf{e}_i - \mathbf{e}_{i-1}$, 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 knows $\{\mathbf{e}_{i+1} - \mathbf{e}_i, \mathbf{e}_{i+2} - \mathbf{e}_i, ..., \mathbf{e}_{i+j} - \mathbf{e}_i\}$, 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 $(\Delta^{\mathsf{s}}_{\mathsf{e}+1}, \Delta^{\mathbf{p}}_{\mathsf{e}+1})$ to compute the new epoch key and the update token, which makes sure the update token $\Delta_{\mathsf{e}+1} = (\Delta^{\mathsf{s}}_{\mathsf{e}+1}, \mathbf{p}_{\mathsf{e}+1})$ 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 epoch 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.

Following the correctness analysis of the underlying PKE scheme, encrypted ciphertexts decrypt to the correct message with overwhelming probability. So, we only need to consider if updated ciphertexts will decrypt to the correct message.

First, assume a ciphertext is encrypted in epoch e and that it will be updated from epoch e to epoch $\mathsf{e}+1$. For any $\mathbf{m} \in \mathcal{M}$, suppose $\mathbf{p}_{\mathsf{e}} = \mathbf{A} \cdot \mathbf{s}_{\mathsf{e}} + \sum_{i=0}^{\mathsf{e}} \mathbf{e}_i$, $\mathbf{p}_{\mathsf{e}+1} = \mathbf{A} \cdot \mathbf{s}_{\mathsf{e}+1} + \sum_{i=0}^{\mathsf{e}+1} \mathbf{e}_i$, $(\mathbf{C}^1_{\mathsf{e}}, \mathbf{c}^2_{\mathsf{e}}) = (\mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}}, \mathbf{p}^\intercal_{\mathsf{e}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m})$ and $(\mathbf{C}^1, \mathbf{c}^2) = (\mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}+1}, \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+1} + \mathbf{e}'_{\mathsf{e}+1})$. Then

$$\begin{aligned} \mathbf{C}^1_{\mathsf{e}+1} &= \mathbf{C}^1_{\mathsf{e}} + \mathbf{C}^1 \\ &= \mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}+1} \\ &= \mathbf{A}^\intercal \cdot (\mathbf{R}_{\mathsf{e}} + \mathbf{R}_{\mathsf{e}+1}), \end{aligned} \tag{7}$$

and

$$\begin{aligned} \mathbf{c}^2_{\mathsf{e}+1} &= \mathbf{c}^2_{\mathsf{e}} + (\Delta^{\mathsf{s}}_{\mathsf{e}+1})^\intercal \cdot \mathbf{C}^1_{\mathsf{e}} + \mathbf{c}^2 \\ &= \mathbf{p}^\intercal_{\mathsf{e}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m} + (\mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}})^\intercal \cdot (\mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}}) + \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+1} + \mathbf{e}'_{\mathsf{e}+1} \\ &= \mathbf{p}^\intercal_{\mathsf{e}} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+1} + (\mathbf{s}_{\mathsf{e}+1} - \mathbf{s}_{\mathsf{e}})^\intercal \cdot \mathbf{A}^\intercal \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1} + \frac{q}{2}\mathbf{m} \\ &= \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot (\mathbf{R}_{\mathsf{e}+1} + \mathbf{R}_{\mathsf{e}}) - \mathbf{e}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1} + \frac{q}{2}\mathbf{m} \\ &\overset{s}{\approx} \mathbf{p}^\intercal_{\mathsf{e}+1} \cdot (\mathbf{R}_{\mathsf{e}+1} + \mathbf{R}_{\mathsf{e}}) + (\mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1}) + \frac{q}{2}\mathbf{m} && (8) \\ &= (\mathbf{p}^\intercal_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}}) + \mathbf{c}^2 + \frac{q}{2}\mathbf{m}. && (9) \end{aligned}$$

Note that by Lemma A.1 and Lemma A.2, Equation (8) holds. Notice that the updated ciphertext is of the same shape as the encrypted ciphertext with the new randomness $(\mathbf{R}_{\mathsf{e}} + \mathbf{R}_{\mathsf{e}+1})$ and new error $(\mathbf{e}'_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}+1})$. Which means when we update a ciphertext multiple times the updated ciphertext will keep the same shape as well, only with a bigger randomness and bigger error.

Iteratively, we consider ciphertext $\mathbf{c}_{\mathsf{e}'}$ to be an updated ciphertext updated from epoch e to epoch e', where $0 \leq \mathsf{e}' - \mathsf{e} < l$. Assume $\mathbf{c}_{\mathsf{e}}$ is the original encryption. For $0 \leq i \leq \mathsf{e}' - \mathsf{e}$, let the re-randomization performed in epoch $\mathsf{e} + i$ outputs $(\mathbf{A}^{\mathsf{T}} \cdot \mathbf{R}_{\mathsf{e}+i}, \mathbf{p}^{\mathsf{T}}_{\mathsf{e}+1} \cdot \mathbf{R}_{\mathsf{e}+i} + \mathbf{e}'_{\mathsf{e}+i})$. Then

$$\mathbf{C}^1_{\mathsf{e}'} = \mathbf{A}^{\mathsf{T}} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i},$$

and

$$\begin{aligned}\mathbf{c}^2_{\mathsf{e}'} &= \mathbf{p}^{\mathsf{T}}_{\mathsf{e}'} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m} - (\sum_{i=1}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}^{\mathsf{T}}_{\mathsf{e}+i} \cdot (\sum_{j=0}^{i-1} \mathbf{R}_{\mathsf{e}+j})) \\ &\overset{s}{\approx} \mathbf{p}^{\mathsf{T}}_{\mathsf{e}'} \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m}. \end{aligned} \tag{10}$$

By Lemma A.1 and Lemma A.2, Equation (10) holds. Then we have $\mathbf{c}^2_{\mathsf{e}'} - \mathbf{s}^{\mathsf{T}}_{\mathsf{e}'} \cdot \mathbf{C}^1_{\mathsf{e}'} \overset{s}{\approx} \sum_{i=0}^{\mathsf{e}'} \mathbf{e}^{\mathsf{T}}_i \cdot \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{R}_{\mathsf{e}+i} + \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m} \overset{s}{\approx} \sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i} + \frac{q}{2}\mathbf{m}$. By Lemma A.3, each entry of $\sum_{i=0}^{\mathsf{e}'-\mathsf{e}} \mathbf{e}'_{\mathsf{e}+i}$ is of the size at most $\sqrt{\mathsf{e}' - \mathsf{e}} \cdot \beta$ with overwhelming probability, which has distance at most $\frac{q}{8}$ from $0 \mod q$ because $\beta \leq \frac{q}{8\sqrt{l}}$ and $\mathsf{e}' - \mathsf{e} < l$. In summary, LWEUE.Dec decrypts correctly with overwhelming probability.

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 [LT18] and Klooß et al. [KLR19] 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 5). This provides the underlying intuition for the security proof.

5.5.1 Technical Simulations in the Proof

Simulate Public Keys. Suppose $\mathbf{p}_0 = \mathbf{A} \cdot \mathrm{s}_0 + \mathbf{e}_0$ and $(\Delta_i^{\mathbf{s}}, \Delta_i^{\mathbf{p}}) = (\Delta_i^{\mathbf{s}}, \mathbf{A} \cdot \Delta_i^{\mathbf{s}} + \mathbf{e}_i)$ for $i > 0$, where $\mathrm{s}_0, \Delta_i^{\mathbf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $\mathbf{e}_i \leftarrow D_{\mathbb{Z},\alpha}^m$. Then

$$\begin{aligned}
\mathbf{p}_{\mathsf{e}} &= \mathbf{p}_{\mathsf{e}-1} + \mathbf{A} \cdot \Delta_{\mathsf{e}}^{\mathbf{s}} + \mathbf{e}_{\mathsf{e}} \\
&= \mathbf{p}_j + \mathbf{A} \cdot \Big(\sum_{i=j+1}^{\mathsf{e}} \Delta_i^{\mathbf{s}}\Big) + \sum_{i=j+1}^{\mathsf{e}} \mathbf{e}_i \qquad (11) \\
&= \mathbf{A} \cdot \Big(\mathrm{s}_0 + \sum_{i=1}^{\mathsf{e}} \Delta_i^{\mathbf{s}}\Big) + \sum_{i=0}^{\mathsf{e}} \mathbf{e}_i \\
&= \mathbf{A} \cdot \mathrm{s}_{\mathsf{e}} + \sum_{i=0}^{\mathsf{e}} \mathbf{e}_i. \qquad (12)
\end{aligned}$$

Equation (11) shows that any public key can be simulated by any other public key and tokens. Equation (12) shows that any public key can be simulated by the corresponding secret key.

Simulate Updated Ciphertexts. Then we consider how to simulate updated ciphertexts by only using public keys. We will have a bookkeeping in set $\mathcal{L}$ to track the randomness $\mathbf{R}$ and the big error $\mathbf{e}'$. Specifically, we record $\mathcal{L}$ as $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$.

To simulate an updated ciphertext from epoch e to epoch $\mathsf{e}+1$, suppose $(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m}) \in \mathcal{L}$, and the new public key is $\mathbf{p}_{\mathsf{e}+1}$. We use the new public key to perform the re-randomization, to produce $(\mathbf{C}^1, \mathbf{c}^2)$. Suppose $(\mathbf{C}^1, \mathbf{c}^2) = (\mathbf{A}^\intercal \cdot \mathbf{R}, \mathbf{p}_{\mathsf{e}+1}^\intercal \cdot \mathbf{R} + \mathbf{e}')$, then we use Equations (7) and (9) to simulate the updated ciphertext $\mathbf{c}_{\mathsf{e}+1} = (\mathbf{C}_{\mathsf{e}+1}^1, \mathbf{c}_{\mathsf{e}+1}^2)$, where $\mathbf{C}_{\mathsf{e}+1}^1 = \mathbf{C}_{\mathsf{e}}^1 + \mathbf{C}^1$ and $\mathbf{c}_{\mathsf{e}+1}^2 = (\mathbf{p}_{\mathsf{e}+1}^\intercal \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m}) + \mathbf{c}^2$. After simulation, the set $\mathcal{L}$ will store the updated ciphertext with the updated randomness $\mathbf{R}_{\mathsf{e}} + \mathbf{R}$ and the updated error $\mathbf{e}'_{\mathsf{e}} + \mathbf{e}'$, i.e. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}+1}, \mathsf{e}+1; \mathbf{R}_{\mathsf{e}} + \mathbf{R}, \mathbf{e}'_{\mathsf{e}} + \mathbf{e}', \mathbf{m})\}$. We use the re-randomization algorithm reR described in Fig. 21 to simulate updated ciphertexts.

```
$\mathsf{reR}(\mathbf{p}_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e}} = (\mathbf{C}_{\mathsf{e}}^1, \mathbf{c}_{\mathsf{e}}^2); \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})$ :
  $\mathbf{R} \leftarrow \mathcal{D}_r^t$
  $\mathbf{e}' \leftarrow D_{\mathbb{Z},\beta}^{1\times t}$
  $\mathbf{C}^1 \leftarrow \mathbf{A}^\intercal \cdot \mathbf{R}$
  $\mathbf{c}^2 \leftarrow \mathbf{p}_{\mathsf{e}+1}^\intercal \cdot \mathbf{R} + \mathbf{e}'$
  $\mathbf{C}_{\mathsf{e}+1}^1 \leftarrow \mathbf{C}_{\mathsf{e}}^1 + \mathbf{C}^1$
  $\mathbf{c}_{\mathsf{e}+1}^2 \leftarrow (\mathbf{p}_{\mathsf{e}+1}^\intercal \cdot \mathbf{R}_{\mathsf{e}} + \mathbf{e}'_{\mathsf{e}} + \frac{q}{2}\mathbf{m}) + \mathbf{c}^2$
  $\mathbf{R}_{\mathsf{e}+1} \leftarrow \mathbf{R}_{\mathsf{e}} + \mathbf{R}$
  $\mathbf{e}'_{\mathsf{e}+1} \leftarrow \mathbf{e}'_{\mathsf{e}} + \mathbf{e}'$
  return $(\mathbf{c}_{\mathsf{e}+1}; \mathbf{R}_{\mathsf{e}+1}, \mathbf{e}'_{\mathsf{e}+1})$
```

Figure 21: Algorithm reR, used to simulate updated ciphertexts.

5.5.2 LWEUE is randIND-UE-CPA

Theorem 5.1 (LWEUE is randIND-UE-CPA)**.** Let LWEUE be the updatable encryption scheme described in Fig. 20, using parameter setting described in Section 5.1. For any randIND-UE-CPA adversary $\mathcal{A}$ against LWEUE, there exists an adversary $\mathcal{B}_{5.1}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$\mathbf{Adv}_{\mathsf{LWEUE},\,\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA}}(1^\lambda) \le 2(l+1)^3 \cdot \big(t\epsilon + 3\mathbf{Adv}_{n,q,\alpha}^{\mathsf{DLWE}}(\mathcal{B}_{5.1}) + \mathsf{negl}(n)\big).$$

Overview of the Security Proof. We now explain how we bound the advantage of any adversary playing the randIND-UE-CPA game for LWEUE by the DLWE advantage.

To prove the security, we play a hybrid game over epochs, where the reduction constructs one hybrid for each epoch. In hybrid i, to the left of epoch i the game returns real challenge-equal ciphertexts, to the right of epoch i the game returns random ciphertexts. This means we have one hybrid for each epoch, moving real-to-random across the epoch space.

In each hybrid, we play an intermediate sequence of games. In the first game, we apply the firewall technique to set up a modified hybrid game. This modification ensures that the reduction (we will construct this reduction later) can simulate the modified hybrid game because it can provide valid keys and tokens to the adversary and easily check the trivial win conditions. In the second game, we change how the update algorithm runs by only using the public key without the update token (see the technique in Section 5.5.1), which helps the reduction to simulate updated ciphertexts. It is impossible to simulate updated ciphertext across firewalls, if the token is needed when updating, as the reduction does not know tokens in both firewalls. Our scheme makes it possible to generate updated ciphertexts using only the public key. In the third game, we change how public keys are generated, which makes sure that the reduction can simulate valid public keys. Eventually, we construct a reduction playing the IND\$-CPA (the IND\$-CPA advantage is upper bounded by the DLWE advantage) game by simulating the third game to an adversary. The IND\$-CPA game moves the real challenge-equal ciphertext in epoch i to a random ciphertext, which means the fresh encryption is computationally indistinguishable from the updated ciphertext.

Proof. In this proof, we use three steps to reach our desired goal. In the first step, we use a sequence of hybrid games H_i to bound the randIND-UE-CPA advantage. In the second step, we bound the advantage of each hybrid game to the advantage of a modified hybrid game $\mathcal{G}_i$. In the third step, we bound the advantage of the modified hybrid game $\mathcal{G}_i$ to the DLWE advantage, using an intermediate sequence of games.

Step 1. For $\mathrm{b} \in \{0,1\}$, we construct a sequence of hybrid games $H_0^{\mathrm{b}}, \ldots, H_{l+1}^{\mathrm{b}}$. In game H_i^{b}, if the adversary asks for a challenge-equal ciphertext by the $\mathcal{O}.\mathsf{Chall}$ query or a $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ query, with challenge input $(\bar{\mathbf{m}}, \bar{\mathbf{c}})$, in epoch j:

- if $j < i$, then return a real challenge-equal ciphertext, which is (updated from) the encryption of $\bar{\mathbf{m}}$ if $\mathrm{b} = 0$ or an updated ciphertext of $\bar{\mathbf{c}}$ if $\mathrm{b} = 1$,
- if $j \geq i$, return a random ciphertext.

Thus H_{l+1}^{b}, $i = l+1$, is $\mathbf{Exp}_{\mathsf{LWEUE},\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}$, i.e. all challenge responses are real challenge-equal ciphertexts. Notice that $H_0^0 = H_0^1$, $i = 0$, because all challenge responses are random ciphertexts. We have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{LWEUE},\,\mathcal{A}}^{\mathsf{randIND\text{-}UE\text{-}CPA}}(1^\lambda) =& \left|\mathbf{Pr}[H_{l+1}^1 = 1] - \mathbf{Pr}[H_{l+1}^0 = 1]\right| \\ \leq & \sum_{i=1}^{l+1} |\mathbf{Pr}[H_i^1 = 1] - \mathbf{Pr}[H_{i-1}^1 = 1]| \\ & + \sum_{i=1}^{l+1} |\mathbf{Pr}[H_i^0 = 1] - \mathbf{Pr}[H_{i-1}^0 = 1]|.\end{aligned}$$

Step 2. In Hybrid i, for $\mathrm{b} \in \{0,1\}$, let $\mathcal{A}_i'$ be an adversary trying to distinguish game H_i^{b} from game H_{i-1}^{b}. For all queries concerning epochs other than i the responses will be equal in either game, so we assume that $\mathcal{A}_i'$ asks for a challenge-equal ciphertext in epoch i, therefore there exist two epochs[4]

[4]This observation was introduced in the work of Lehmann and Tackmann [LT18], Klooß et al. [KLR19] provided an extended description of this *key insulation* technique, and Boyd et al. [BDGJ20] formally defined it as *firewall technique*.

(denote $\mathsf{fwl}, \mathsf{fwr}$) around the epoch i such that no key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ and no token in epochs fwl and $\mathsf{fwr}+1$ are corrupted.

Furthermore, $H_j^{\mathrm{b}} = H_0^{\mathrm{b}}$, for all $j < \tilde{\mathsf{e}}$, since the adversary never asks for a challenge-equal ciphertext before the challenge epoch ($\tilde{\mathsf{e}}$). So we assume $i \geq \tilde{\mathsf{e}}$.

Define a new game $\mathcal{G}_i^{\mathrm{b}}$ that is the same as game H_i^{b}, except for the game randomly picks two numbers $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \ldots, l\}$. If the adversary corrupts a key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ or a token in epochs fwl and $\mathsf{fwr}+1$, the game aborts. This loss is upper bounded by $(l+1)^2$. Then we have

$$|\mathbf{Pr}[H_i^{\mathrm{b}} = 1] - \mathbf{Pr}[H_{i-1}^{\mathrm{b}} = 1]| \leq (l+1)^2 |\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|.$$

Step 3. In this step, we will prove that $|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|$ is upper bounded by the DLWE advantage.

Claim 5.1.1. For any $\mathrm{b} \in \{0,1\}$, there exists an adversary $\mathcal{B}_{5.1}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]| \leq t\epsilon + 3\mathbf{Adv}_{n,q,\alpha}^{\mathsf{DLWE}}(\mathcal{B}_{5.1}) + \mathsf{negl}(n).$$

Proof of Claim 5.1.1. Assume $\mathcal{A}_i$ is an adversary trying to distinguish $\mathcal{G}_i^{\mathrm{b}}$ from $\mathcal{G}_{i-1}^{\mathrm{b}}$, and $\mathcal{A}_i$ asks for a challenge-equal ciphertext in epoch i ($i \geq \tilde{\mathsf{e}}$). The proof of Claim 5.1.1 consists of the following sequence of games.

Game 0

The game flips a coin $d \xleftarrow{\$} \{0,1\}$, if $d = 0$ it plays game $\mathcal{G}_{i-1}^{\mathrm{b}}$ (responses a real challenge-equal ciphertext in epoch i) to $\mathcal{A}_i$, if $d = 1$ it plays game $\mathcal{G}_i^{\mathrm{b}}$ (responses a random ciphertext in epoch i) to $\mathcal{A}_i$. Denote E_i be the event that the adversary succeeds in guessing d in game i.

Then we have

$$\mathbf{Pr}[E_0] = |\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|.$$

Game 1

We consider a modified game that is the same as Game 0, except for the updated ciphertext is generated by using only public key without the secret token. More precisely, the game running the update algorithm as follows.

```
$\underline{\mathcal{O}.\mathsf{Upd}(\mathbf{c}_{\mathsf{e}-1})}:$
  **if** $(\cdot, \mathbf{c}_{\mathsf{e}-1}, \mathsf{e}-1; \mathbf{R}_{\mathsf{e}-1}, \mathbf{e}'_{\mathsf{e}-1}, \mathbf{m}) \notin \mathcal{L}$ **then**
    **return** $\perp$
  $(\mathbf{c}_{\mathsf{e}}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}-1}; \mathbf{R}_{\mathsf{e}-1}, \mathbf{e}'_{\mathsf{e}-1}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$
  **return** $\mathbf{c}_{\mathsf{e}}$
```

The re-randomization algorithm reR is described in Fig. 21. By the analysis in Section 5.5.1, we have

$$|\mathbf{Pr}[E_1] - \mathbf{Pr}[E_0]| \leq \mathsf{negl}(n).$$

Game 2

We consider a modified game that is the same as Game 1, except for the public key in epoch fwl is generated with one additional error and the public key in epoch $\mathsf{fwr}+1$ is generated with one error less. In particular, let $\Delta_{\mathsf{fwl}}^{\mathbf{s}}, \Delta_{\mathsf{fwr}+1}^{\mathbf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $\mathbf{e}_{\mathsf{fwl}}, \mathbf{e} \leftarrow D_{\mathbb{Z},\alpha}^m$, these public keys are generated as follows.

$$\mathbf{p}_{\mathsf{fwl}} = \mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta_{\mathsf{fwl}}^{\mathbf{s}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e},$$

$$\mathbf{p}_{\mathsf{fwr}+1} = \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}+1} - \mathbf{e}.$$

We claim that, in Game 2, all public key distributions except for $\mathbf{p}_{\mathsf{fwl}}$ are equal to the corresponding real public key distributions. Furthermore, the public key $\mathbf{p}_{\mathsf{fwl}}$ produced in Game 2 is computationally indistinguishable from the real public key $\mathbf{p}_{\mathsf{fwl}}$.

Any public key that is in an epoch smaller than fwl, or greater than fwr+1, are generated by algorithm LWEUE.KG in both games. So we only consider public keys from epoch fwl to epoch $\mathsf{fwr}+1$. In Game 2, the public keys from epoch fwl to epoch $\mathsf{fwr}+1$ are computed as follows.

$$\begin{aligned}
\mathbf{p}_{\mathsf{fwl}} &= \mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e}, && (13)\\
&\dots \\
\mathbf{p}_i &= \mathbf{p}_{i-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_i + \mathbf{e}_i, && (14)\\
&\dots \\
\mathbf{p}_{\mathsf{fwr}} &= \mathbf{p}_{\mathsf{fwr}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}} + \mathbf{e}_{\mathsf{fwr}}, && (15)\\
\mathbf{p}_{\mathsf{fwr}+1} &= \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwr}+1} - \mathbf{e}. && (16)
\end{aligned}$$

Based on Equation (14), (15), and (16), we know that public keys $\mathbf{p}_{\mathsf{fwl}+1}, \dots, \mathbf{p}_{\mathsf{fwr}}, \mathbf{p}_{\mathsf{fwr}+1}$ are generated as real public keys by running algorithm LWEUE.KG. Furthermore, note that $\mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}}$ is computationally indistinguishable from a uniform random element in $\mathbb{Z}_q^m$. Therefore, the public value $\mathbf{p}_{\mathsf{fwl}}$ in both games are computationally indistinguishable from a uniform random element in $\mathbb{Z}_q^m$, under the $\mathsf{DLWE}_{n,q,\alpha}$ problem. Then we have

$$|\mathbf{Pr}[E_2] - \mathbf{Pr}[E_1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}.$$

Now we are almost finished, all we need to prove now is that $\mathbf{Pr}[E_2] \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}(1^\lambda) + \mathsf{negl}(n)$, which we claim is true in Claim 5.1.2. Finally, by Theorem 4.1, we get $\mathbf{Pr}[E_2] \leq t\epsilon + \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha} + \mathsf{negl}(n)$, which concludes the proof of Claim 5.1.1.

Claim 5.1.2. $\mathbf{Pr}[E_2] \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}(1^\lambda) + \mathsf{negl}(n)$.

Proof of Claim 5.1.2. Before we start constructing a reduction to simulate Game 2 we need the following equations, describing the public key in an epoch j:

- For $\mathsf{fwl} < j \leq \mathsf{fwr}$, we have

$$\begin{aligned}
\mathbf{p}_j &= \mathbf{p}_{j-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_j + \mathbf{e}_j \\
&= \mathbf{p}_{\mathsf{fwl}} + \mathbf{A} \cdot (\sum_{k=\mathsf{fwl}+1}^{j} \Delta^{\mathbf{s}}_k) + \sum_{k=\mathsf{fwl}+1}^{j} \mathbf{e}_k \\
&= (\mathbf{p}_{\mathsf{fwl}-1} + \mathbf{A} \cdot \Delta^{\mathbf{s}}_{\mathsf{fwl}} + \mathbf{e}_{\mathsf{fwl}} + \mathbf{e}) + \mathbf{A} \cdot (\sum_{k=\mathsf{fwl}+1}^{j} \Delta^{\mathbf{s}}_k) + \sum_{k=\mathsf{fwl}+1}^{j} \mathbf{e}_k \\
&= \mathbf{e} + \mathbf{A} \cdot (\mathbf{s}_0 + \sum_{k=1}^{j} \Delta^{\mathbf{s}}_k) + \sum_{k=0}^{j} \mathbf{e}_k \\
&= (\mathbf{A} \cdot \mathbf{s}_j + \mathbf{e}) + \sum_{k=0}^{j} \mathbf{e}_k, && (17)
\end{aligned}$$

Equation (17) also holds for $j = \mathsf{fwl}$, it follows from Equation (12) and (13). This equation will be used to simulate the public key in epoch i.

- For $i < j \leq \mathsf{fwr}$, we have

$$\mathbf{p}_j = \mathbf{p}_i + \big(\mathbf{A} \cdot (\sum_{k=i+1}^{j} \Delta_k^{\mathbf{s}}) + \sum_{k=i+1}^{j} \mathbf{e}_k\big). \tag{18}$$

- Similarly, for $\mathsf{fwl} \leq j < i$, we have

$$\mathbf{p}_j = \mathbf{p}_i - \big(\mathbf{A} \cdot (\sum_{k=j+1}^{i} \Delta_k^{\mathbf{s}}) + \sum_{k=j+1}^{i} \mathbf{e}_k\big). \tag{19}$$

- For $j = \mathsf{fwr} + 1$, we have

$$\begin{aligned}\mathbf{p}_{\mathsf{fwr}+1} &= \mathbf{p}_{\mathsf{fwr}} + \mathbf{A} \cdot \Delta_{\mathsf{fwr}+1}^{\mathbf{s}} - \mathbf{e} \\ &= (\mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}} + \mathbf{e} + \sum_{k=0}^{\mathsf{fwr}} \mathbf{e}_k) + \mathbf{A} \cdot \Delta_{\mathsf{fwr}+1}^{\mathbf{s}} - \mathbf{e} \\ &= \mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}+1} + \sum_{j=0}^{\mathsf{fwr}} \mathbf{e}_j. \end{aligned} \tag{20}$$

We construct a reduction $\tilde{\mathcal{B}}_{5.1}$, detailed in Fig. 22, that is playing the IND\$-CPA game and will simulate the responses of queries made by adversary $\mathcal{A}_i$ in game 2. Initially, the reduction guesses two numbers fwl, fwr. If $\mathcal{A}_i$ corrupts $\mathbf{k}_{\mathsf{fwl}}, \ldots, \mathbf{k}_{\mathsf{fwr}}$, Δ_{fwl}, or $\Delta_{\mathsf{fwr}+1}$ the reduction aborts the game.

A summary of the technical simulations are as follows.

- In the setup phrase,
  - $\tilde{\mathcal{B}}_{5.1}$ samples errors and tokens as follows.
    * For $j \in \{0, \ldots, \mathsf{fwr}\} \cup \{\mathsf{fwr} + 2, \ldots, n\}$, $\tilde{\mathcal{B}}_{5.1}$ samples each error as $\mathbf{e}_j \leftarrow D_{\mathbb{Z},\alpha}^m$,
    * For $j \in \{\mathsf{fwl} + 1, \ldots, \mathsf{fwr}\}$, $\tilde{\mathcal{B}}_{5.1}$ samples each token as $\Delta_j^{\mathbf{s}} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$.
  - $\tilde{\mathcal{B}}_{5.1}$ generates all keys and tokens, except for $\mathbf{k}_{\mathsf{fwl}}, \ldots, \mathbf{k}_{\mathsf{fwr}}, \Delta_{\mathsf{fwl}}, \Delta_{\mathsf{fwr}+1}$, as follows.
    * $\tilde{\mathcal{B}}_{5.1}$ generates the initial key by running the key generation algorithm. In particular, $\mathbf{k}_0 \leftarrow \mathsf{LWEUE.KG}(1^\lambda)$ is computed as: randomly sample a secret key $\mathbf{s}_0 \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set $\mathbf{k}_0 \leftarrow (\mathbf{s}_0, \mathbf{A} \cdot \mathbf{s}_0 + \mathbf{e}_0)$.
    * For $j \in \{1, \ldots, \mathsf{fwl} - 1\} \cup \{\mathsf{fwr} + 2, \ldots, n\}$, $\tilde{\mathcal{B}}_{5.1}$ generates the epoch key and update token by running the key generation algorithm and the token generation algorithm. In particular, $\mathbf{k}_j \leftarrow \mathsf{LWEUE.KG}(1^\lambda)$ and $\Delta_j \leftarrow \mathsf{LWEUE.TG}(\mathbf{k}_{j-1}, \mathbf{k}_j)$ are computed as: randomly sample a secret key $\mathbf{s}_j \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set token $\Delta_j^{\mathbf{s}} \leftarrow \mathbf{s}_j - \mathbf{s}_{j-1}$ and public key $\mathbf{p}_j \leftarrow \mathbf{A} \cdot \mathbf{s}_j + \sum_{k=0}^{j} \mathbf{e}_k$[5].
    * $\tilde{\mathcal{B}}_{5.1}$ generates $\mathbf{k}_{\mathsf{fwr}+1}$ by using Equation (20): randomly sample a secret key $\mathbf{s}_{\mathsf{fwr}+1} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, set public key $\mathbf{p}_{\mathsf{fwr}+1} \leftarrow \mathbf{A} \cdot \mathbf{s}_{\mathsf{fwr}+1} + \sum_{k=0}^{\mathsf{fwr}} \mathbf{e}_k$.
  - $\tilde{\mathcal{B}}_{5.1}$ generates public keys $\mathbf{p}_{\mathsf{fwl}}, \ldots, \mathbf{p}_{\mathsf{fwr}}$ as follows.
    * $\tilde{\mathcal{B}}_{5.1}$ generates $\mathbf{p}_i$ by using Equation (17): embed the public key $\mathbf{p}$ received from its IND\$-CPA challenger to $\mathbf{p}_i$, set $\mathbf{p}_i \leftarrow \mathbf{p} + \sum_{k=0}^{i} \mathbf{e}_k$[6].
    * For $\mathsf{fwl} \leq j < i$, $\mathbf{p}_j$ is generated by using Equations (17,19): set public key $\mathbf{p}_j \leftarrow \mathbf{p} - \mathbf{A} \cdot \sum_{k=j+1}^{i} \Delta_j^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$.

[5]If $j > \mathsf{fwr} + 1$, skip $\mathbf{e}_{\mathsf{fwr}+1}$ in the computation. This computation is based on Equations (12,20)

[6]set $\mathbf{A} \cdot \mathbf{s}_i + \mathbf{e} \leftarrow \mathbf{p}$

```
For b $\in \{0,1\}$, $\tilde{\mathcal{B}}_{5.1}$ plays IND\$-CPA game
by running $\mathcal{A}_i$ :
  receive $(\mathbf{A}, \mathbf{p})$
  do Setup
  $\mathrm{b}' \leftarrow \mathcal{A}_i^{oracles}(1^\lambda)$
  if ABORT occurred or $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
   or $i \notin \{\mathsf{fwl}, \dots, \mathsf{fwr}\}$ or $i < \tilde{\mathsf{e}}$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

Setup$(1^\lambda)$
  $\Delta_0 \leftarrow \perp$;  $\mathsf{e} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$;
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
  $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \dots, l\}$
  for $j \in \{0, \dots, n\}^{\dagger\dagger}$ do
    $\mathbf{e}_j \leftarrow D^m_{\mathbb{Z},\alpha}$
  for $j \in \{\mathsf{fwl}+1, \dots, \mathsf{fwr}\}$ do
    $\Delta^{\mathbf{s}}_j \leftarrow \mathcal{U}(\mathbb{Z}^n_q)$
  for $j \in \{0, \dots, \mathsf{fwl}\text{-}1\} \cup \{\mathsf{fwr}+1, \dots, n\}$ do
    $\mathbf{s}_j \leftarrow \mathcal{U}(\mathbb{Z}^n_q)$;
    $\Delta^{\mathbf{s}}_j \leftarrow \mathbf{s}_j\text{-}\mathbf{s}^{\dagger}_{j\text{-}1}$;
    $\mathbf{p}_j \leftarrow \mathbf{A} \cdot \mathbf{s}_j + \sum_{k=0}^{j} \mathbf{e}_k^{\dagger\dagger}$
  $\mathbf{p}_i \leftarrow \mathbf{p} + \sum_{k=0}^{i} \mathbf{e}_k$
  for $j \in \{\mathsf{fwl}, \dots, i\text{-}1\}$ do
    $\mathbf{p}_j \leftarrow \mathbf{p}\text{-}\mathbf{A} \cdot \sum_{k=j+1}^{i} \Delta^{\mathbf{s}}_k + \sum_{k=0}^{j} \mathbf{e}_k$
  for $j \in \{i+1, \dots, \mathsf{fwr}\}$ do
    $\mathbf{p}_j \leftarrow \mathbf{p} + \mathbf{A} \cdot \sum_{k=i+1}^{j} \Delta^{\mathbf{s}}_k + \sum_{k=0}^{j} \mathbf{e}_k$

$\mathcal{O}$.Enc$(\mathbf{m})$ :
  $\mathrm{c} \leftarrow \mathrm{c}+1$
  $\mathbf{c}_{\mathsf{e}} \leftarrow \mathsf{LWEUE.Enc}(\mathbf{p}_{\mathsf{e}}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathrm{c}, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$
  return $\mathbf{c}_{\mathsf{e}}$

$\mathcal{O}$.Next :
  $\mathsf{e} \leftarrow \mathsf{e}+1$

$\mathcal{O}$.Upd$(\mathbf{c}_{\mathsf{e}\text{-}1})$ :
  if $(\cdot, \mathbf{c}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathbf{R}_{\mathsf{e}\text{-}1}, \mathbf{e}'_{\mathsf{e}\text{-}1}, \mathbf{m}) \notin \mathcal{L}$ then
    return $\perp$
  $(\mathbf{c}_{\mathsf{e}}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}\text{-}1}; \mathbf{R}_{\mathsf{e}\text{-}1}, \mathbf{e}'_{\mathsf{e}\text{-}1}, \mathbf{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_{\mathsf{e}}, \mathsf{e}; \mathbf{R}_{\mathsf{e}}, \mathbf{e}'_{\mathsf{e}}, \mathbf{m})\}$
  return $\mathbf{c}_{\mathsf{e}}$
```

```
$\mathcal{O}$.Corr$(\mathsf{inp}, \hat{\mathsf{e}})$ :
  if $\hat{\mathsf{e}} > \mathsf{e}$ then
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$ then
    if $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \dots, \mathsf{fwr}\}$ then
      ABORT
    else
      $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
      return $\mathbf{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    if $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \mathsf{fwr}+1\}$ then
      ABORT
    else
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
      return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}$.Chall$(\bar{\mathbf{m}}_0, \bar{\mathbf{c}})$ :
  if $\mathsf{phase} = 1$ then
    return $\perp$
  $\mathsf{phase} \leftarrow 1$; $\tilde{\mathsf{e}} \leftarrow \mathsf{e}$
  if $(\cdot, \bar{\mathbf{c}} = (\bar{\mathbf{C}}^1, \bar{\mathbf{c}}^2), \tilde{\mathsf{e}}\text{-}1; \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1}, \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1}, \bar{\mathbf{m}}_1) \notin \mathcal{L}$ then
    return $\perp$
  Send $\bar{\mathbf{m}}_{\mathrm{b}}$ to the IND\$-CPA challenger, get $(\mathbf{X}, \mathbf{y})$
  if $\tilde{\mathsf{e}} = i$ then
    if $\mathrm{b} = 0$ then
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow (\mathbf{X}, \mathbf{y})$
    else
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow (\mathbf{X} + \bar{\mathbf{C}}^1, \mathbf{y} + \mathbf{p}^{\mathsf{T}}_{\tilde{\mathsf{e}}} \cdot \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1} + \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1})$
  else
    if $\mathrm{b} = 0$ then
      $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{LWEUE.Enc}(\mathbf{p}_{\tilde{\mathsf{e}}}, \bar{\mathbf{m}}_0)$
    else
      $(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}; \mathbf{R}_{\tilde{\mathsf{e}}}, \mathbf{e}'_{\tilde{\mathsf{e}}}) \leftarrow \mathsf{reR}(\mathbf{p}_{\tilde{\mathsf{e}}}, \bar{\mathbf{c}}; \mathbf{R}_{\tilde{\mathsf{e}}\text{-}1}, \mathbf{e}'_{\tilde{\mathsf{e}}\text{-}1}, \bar{\mathbf{m}}_1)$
  for $j \in \{\tilde{\mathsf{e}}+1, \dots, i\text{-}1\}$ do
    $(\tilde{\mathbf{c}}_j; \mathbf{R}_j, \mathbf{e}'_j) \leftarrow \mathsf{reR}(\mathbf{p}_j, \tilde{\mathbf{c}}_{j\text{-}1}; \mathbf{R}_{j\text{-}1}, \mathbf{e}'_{j\text{-}1}, \bar{\mathbf{m}}_{\mathrm{b}})$
  $\tilde{\mathbf{c}}_i \leftarrow (\mathbf{X} + \tilde{\mathbf{C}}^1_{i\text{-}1}, \mathbf{y} + \mathbf{p}^{\mathsf{T}}_i \cdot \mathbf{R}_{i\text{-}1} + \mathbf{e}'_{i\text{-}1})$
  for $j \in \{i+1, \dots, n\}$ do
    $\tilde{\mathbf{c}}_j \xleftarrow{\$} \mathcal{CS}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{\mathsf{e}}\}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}, \tilde{\mathsf{e}})\}$
  return $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}$.Upd$\tilde{\mathbf{C}}$ :
  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 $\tilde{\mathbf{c}}_{\mathsf{e}}$
```

Figure 22: Reduction $\tilde{\mathcal{B}}_{5.1}$ for proof of Claim 5.1.2. $\dagger$ indicates Δ_0 and $\Delta_{\mathsf{fwr}+1}$ are skipped in the computation. $\dagger\dagger$ indicates $\mathbf{e}_{\mathsf{fwr}+1}$ is skipped in the computation.

    * For $i < j \leq \mathsf{fwr}$, $\mathbf{p}_j$ is generated by using Equations (17,18): set public key $\mathbf{p}_j \leftarrow \mathbf{p} + \mathbf{A} \cdot \sum_{k=i+1}^{j} \Delta_j^{\mathbf{s}} + \sum_{k=0}^{j} \mathbf{e}_k$.

- To simulate non-challenge ciphertexts: $\tilde{\mathcal{B}}_{5.1}$ uses public keys to simulate encrypted ciphertexts and updated ciphertexts. In particular, $\tilde{\mathcal{B}}_{5.1}$ runs the re-randomization algorithm reR to simulate updated ciphertexts.

- To simulate challenge-equal ciphertexts in an epoch that is:
  - to the left of epoch i: $\tilde{\mathcal{B}}_{5.1}$ uses public keys to simulate encryption and updating. When $\tilde{\mathcal{B}}_{5.1}$ runs update algorithm, it performs re-randomization using algorithm reR, which is described in Fig. 21.
  - epoch i: $\tilde{\mathcal{B}}_{5.1}$ embeds the challenge ciphertext $(\mathbf{X}, \mathbf{y})$ received from its IND\$-CPA challenger to the challenge-equal ciphertext in epoch i. Using $(\mathbf{X}, \mathbf{y})$ as the re-randomization component or the random encryption component. More precisely, suppose $\tilde{\mathcal{B}}_{5.1}$ receives a challenge query $\mathcal{O}.\mathsf{Chall}$ with input $(\bar{\mathbf{m}}_0, \bar{\mathbf{c}})$ in challenge epoch $\tilde{\mathsf{e}}$, where the underlying message of $\bar{\mathbf{c}}$ is $\bar{\mathbf{m}}_1$. $\tilde{\mathcal{B}}_{5.1}$ sends $\bar{\mathbf{m}}_{\mathrm{b}}$ to its IND\$-CPA challenger and obtains $(\mathbf{X}, \mathbf{y})$. Hence, $(\mathbf{X}, \mathbf{y})$ is either an output of $\mathsf{LWEPKE.Enc}(\mathbf{p}, \bar{\mathbf{m}}_{\mathrm{b}})$[7] or a random ciphertext.
    For $\mathrm{b} = 1$. In the former case, based on Equations (7, 9), we know that $(\tilde{\mathbf{C}}_i^1, \tilde{\mathbf{c}}_i^2) = (\tilde{\mathbf{C}}_{i-1}^1 + \mathbf{X}, (\mathbf{p}_i^\intercal \cdot \mathbf{R}_{i-1} + \mathbf{e}'_{i-1}) + \mathbf{y})$ is a valid simulation of the challenge-equal ciphertext in epoch i. In the latter case, $(\tilde{\mathbf{C}}_i^1, \tilde{\mathbf{c}}_i^2) = (\tilde{\mathbf{C}}_{i-1}^1 + \mathbf{X}, (\mathbf{p}_i^\intercal \cdot \mathbf{R}_{i-1} + \mathbf{e}'_{i-1}) + \mathbf{y})$ is a random ciphertext.
    For $\mathrm{b} = 0$. The analysis is similar to the discussion when $\mathrm{b} = 1$.
  - to the right of epoch i: $\tilde{\mathcal{B}}_{5.1}$ samples a random ciphertext.

Eventually, $\tilde{\mathcal{B}}_{5.1}$ receives the output bit from $\mathcal{A}_i$. If $\mathcal{A}_i$ guesses that it receives a real challenge-equal ciphertext in epoch i, then $\tilde{\mathcal{B}}_{5.1}$ guesses that it has seen the real encryption (returns 0 to its IND\$-CPA challenger). Otherwise, $\tilde{\mathcal{B}}_{5.1}$ guess it has seen a random ciphertext (returns 1 to its IND\$-CPA challenger).

Note that $\tilde{\mathcal{B}}_{5.1}$ perfectly simulates the responses of queries made by adversary $\mathcal{A}_i$ in game 2 except for a negligible probability. Then we have that $\mathbf{Pr}[E_2] \leq \mathbf{Adv}_{\mathsf{LWEPKE},\, \tilde{\mathcal{B}}_{5.1}}^{\mathsf{IND\$\text{-}CPA}}(1^\lambda) + \mathsf{negl}(n)$. ☐

Remark 5.1. Klooß et al. [KLR19] 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 [NY90] (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

[ABD$^+$a] Erdem Alkim, Joppe W. Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM: Learning With Errors Key Encapsulation. `https://frodokem.org/files/FrodoKEM-specification-20190330.pdf`. Submission to the NIST Post-Quantum Standardization project, round 2.

[7] $\mathbf{y}$ is statistically close to the second component of $\mathsf{LWEPKE.Enc}(\mathbf{p}_i, \bar{\mathbf{m}}_{\mathrm{b}})$ in this case.

[ABD$^+$b] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 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.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum Key Exchange - A New Hope. In *USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[BCLvV17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime: Reducing attack surface at low cost. In *SAC*, volume 10719 of *Lecture Notes in Computer Science*, pages 235–260. Springer, 2017.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 464–493. Springer, 2020.

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving Speed and Security in Updatable Encryption Schemes. *IACR Cryptology ePrint Archive*, 2020:222, 2020.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

[BLMR15] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. *IACR Cryptology ePrint Archive*, 2015:220, 2015.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[CDH$^+$] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU. https://ntru.org/f/ntru-20190330.pdf. Submission to the NIST Post-Quantum Standardization project, round 2.

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, volume 10831 of *Lecture Notes in Computer Science*, pages 282–305. Springer, 2018.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In *Proceedings of CRYPTO 2017 III*, volume 10403 of *LNCS*, pages 98–129. Springer, 2017.

[Gen09] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010.

[Ham] Mike Hamburg. Three Bears. https://sourceforge.net/projects/threebears/. Submission to the NIST Post-Quantum Standardization project, round 2.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In *Proceedings of EUROCRYPT 2019 I*, volume 11476 of *LNCS*, pages 68–99. Springer, 2019.

[LLJ$^+$] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC Lattice-based Cryptosystems. Submission to the NIST Post-Quantum Standardization project, round 2.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *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*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

[OZS$^+$] Garcia-Morchon Oscar, Zhang Zhenfei, Bhattacharya Sauvik, Rietman Ronald, Tolhuizen Ludo, Torre-Arce Jose-Luis, Baan Hayo, Saarinen Markku-Juhani O., Fluhrer Scott, Laarhoven Thijs, Player Rachel, Cheon Jung, Hee, and Son Yongha. Round5. https://round5.org. Submission to the NIST Post-Quantum Standardization project, round 2.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[SD19] Vipin Singh Sehrawat and Yvo Desmedt. Bi-homomorphic lattice-based prfs and unidirectional updatable encryption. In *Cryptology and Network Security - 18th International Conference, CANS 2019, Fuzhou, China, October 25-27, 2019, Proceedings*, pages 3–23, 2019.

A Lattice Background

A.1 Hardness Assumptions – Learning With Errors

Regev [Reg05] introduced the learning with error (LWE) problem, where is as follows. For a secret element s and an error distribution $\mathcal{X}$, the problem is to distinguish m pairs $(\mathbf{a}_i, \mathbf{a}_i \cdot \mathbf{s} + e_i)$ from uniformly random pairs, where $\mathbf{a}_i$ are uniformly random elements and the e_is are sampled from the error distribution $\mathcal{X}$.

Definition 12. [Learning With Errors] Let n, $q = q(n) \geq 2$, $m = \text{poly}(n)$ be positive integers, $\mathcal{X} = \mathcal{X}(n)$ be an error distribution over $\mathbb{Z}$. The *decision learning with errors problem* $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is to distinguish between the following pairs of distributions: $\{\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q\}$ and $\{\mathbf{A}, \mathbf{u}\}$, where $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{X}^m$ and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^m$. The advantage of an adversary $\mathcal{A}$ against $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is defined as

$$\begin{aligned}\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\mathcal{X}}(\mathcal{A}) = &\big|\mathbf{Pr}[\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n, \mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times n}, \mathbf{e} \leftarrow \mathcal{X}^m : \mathcal{A}(\mathbf{A}, \mathbf{A}\cdot\mathbf{s}+\mathbf{e} \mod q) = 1]\\ &- \mathbf{Pr}[\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times n}, \mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^m : \mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]\big|.\end{aligned}$$

We say that the $\mathsf{DLWE}_{n,q,\mathcal{X}}$ problem is hard if $\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\mathcal{X}}(\mathcal{A})$ is negligible for any PPT adversary $\mathcal{A}$.

Regev [Reg05] showed that, for some parameter setting, LWE problem is as hard as some hard lattice problem, namely the worst-case SIVP and GapSVP, under a quantum reduction.

Another variant of the LWE problem is described in the work of Peikert et al. [PVW08], where secret key $\mathbf{s}$ is not a vector but a randomly chosen matrix. Specifically, $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times n}$, $\mathbf{S} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times l}$ and $\mathbf{E} \leftarrow \mathcal{X}^{m\times l}$, the problem is to distinguish $\{\mathbf{A}, \mathbf{A}\cdot\mathbf{S}+\mathbf{E}\}$ from $\{\mathbf{A}, \mathbf{U}\}$, where $\mathbf{U} \overset{\$}{\leftarrow} \mathbb{Z}_q^{m\times l}$. We denote this problem as $\mathsf{DLWE}_{n,l,q,\mathcal{X}}$. Peikert et al. [PVW08, Lemma 7.3] proved that $\mathsf{DLWE}_{n,l,q,\mathcal{X}}$ is hard if $\mathsf{DLWE}_{n,q,\mathcal{X}}$ is hard.

Gaussian Distribution. Let $D_{\mathbb{Z}^n,\alpha}$ denote the n-dimensional discrete Gaussian distribution with standard deviation α. We let $\mathsf{DLWE}_{n,q,\alpha}$ denote the LWE problem $\mathsf{DLWE}_{n,q,D_{\mathbb{Z}^n,\alpha}}$ using a (discrete) Gaussian distribution as the error distribution.

The following three lemmas regarding discrete Gaussian distribution we will frequently use in this paper. The first lemma states that the discrete Gaussian distribution is spherical. The second lemma shows the discrete Gaussian distribution is statistical close to its (short distance) translated discrete Gaussian distribution. The third lemma describes the sum of two discrete Gaussian distributions is statistical close to another discrete Gaussian distribution.

Lemma A.1. [Brakerski and Vaikuntanathan [BV11]] Let $n \in \mathbb{N}$. For any real number $\alpha = \omega(\sqrt{\log n})$, we have $\mathbf{Pr}[\|\mathbf{x}\| > \alpha \cdot \sqrt{n} \mid \mathbf{x} \leftarrow D_{\mathbb{Z}^n,\alpha}] \leq 2^{-n+1}$.

Lemma A.2. [Brakerski and Vaikuntanathan [BV11]] Let $n \in \mathbb{N}$. For any real number $\alpha = \omega(\sqrt{\log n})$, any $\mathbf{c} \in \mathbb{Z}^n$, then the statistical distance between $D_{\mathbb{Z}^n,\alpha}$ and $D_{\mathbb{Z}^n,\alpha,\mathbf{c}}$ is at most $\|\mathbf{c}\|/\alpha$.

Lemma A.3. [Gentry [Gen09]] Let $n \in \mathbb{N}$. For any real number $\alpha, \beta > 0$ satisfies $(\alpha\beta)/\sqrt{\alpha^2+\beta^2} = \omega(\sqrt{\log n})$, then $D_{\mathbb{Z}^n,\alpha} + D_{\mathbb{Z}^n,\beta}$ (and $D_{\mathbb{Z}^n,\alpha} - D_{\mathbb{Z}^n,\beta}$) is statistical to $D_{\mathbb{Z}^n,\sqrt{\alpha^2+\beta^2}}$.

Next we show a lemma that states that if an LWE problem with small standard deviation is hard, then the corresponding LWE with a big standard deviation is hard.

Lemma A.4. Let $n \in \mathbb{N}$. For any real number $\alpha, \beta > 0$ satisfies $\alpha/\beta = \mathsf{negl}(n)$ then for any adversary $\mathcal{A}$ against $\mathsf{DLWE}_{n,q,\beta}$, there exists an adversary $\mathcal{B}$ against $\mathsf{DLWE}_{n,q,\alpha}$ such that

$$\mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\beta}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DLWE}}_{n,q,\alpha}(\mathcal{B}) + \mathsf{negl}(n).$$

Proof. We construct a reduction $\mathcal{B}$ plays $\mathsf{DLWE}_{n,q,\alpha}$ game by running adversary $\mathcal{A}$. When the reduction receives a $\mathsf{DLWE}_{n,q,\alpha}$ pair $(\mathbf{A}, \mathbf{p})$, it samples a big error $\mathbf{e}' \leftarrow D^m_{\mathbb{Z},\beta}$ and sends $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ to $\mathcal{A}$. If $(\mathbf{A}, \mathbf{p})$ is a real $\mathsf{DLWE}_{n,q,\alpha}$ sample, by Lemma A.2, we have $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ is statistical close to a real $\mathsf{DLWE}_{n,q,\beta}$ sample. If $(\mathbf{A}, \mathbf{p})$ is a random $\mathsf{DLWE}_{n,q,\alpha}$ sample, then $(\mathbf{A}, \mathbf{p}+\mathbf{e}' \mod q)$ is a random $\mathsf{DLWE}_{n,q,\beta}$ sample as well. So the reduction $\mathcal{B}$ perfectly simulate $\mathsf{DLWE}_{n,q,\beta}$ game to $\mathcal{A}$ except for a negligible probability. □

A.2 Leftover Hash Lemma

We use a variant of Leftover Hash Lemma, showed by Goldwasser et al. [GKPV10]. We use this lemma to prove our LWE-based PKE scheme is IND\$-CPA-secure in Section 4.

Lemma A.5. [Leftover Hash Lemma] Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \leq (k - 2\log(1/\epsilon) - O(1))/\log(q)$, the joint distribution of $(\mathbf{C}, \mathbf{C}\mathbf{s})$ is ϵ-close to the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^l$, where $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{l\times n}$ and $\mathbf{s} \leftarrow \mathcal{D}$.

We prove that when we reuse $\mathbf{C}$ to generate t samples, the above result is still hold.

Lemma A.6. [Matrix Variant of Leftover Hash Lemma] Let $\mathcal{D}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \leq (k - 2\log(1/\epsilon) - O(1))/\log(q)$, the joint distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{S})$ is $t\epsilon$-close to the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^{l\times t}$, where $\mathbf{C} \xleftarrow{\$} \mathbb{Z}_q^{l\times n}$ and $\mathbf{S} \leftarrow \mathcal{D}^t$.

Proof. The proof method is the same as the proof of Lemma 7.3 stated by Peikert et al. [PVW08]. Consider hybrid distributions $H_0, H_1, \dots, H_t$ for the pair $(\mathbf{C}, \mathbf{B})$: in H_i the entire matrix $\mathbf{C}$ and the first i columns of $\mathbf{B}$ are all uniformly random. Then H_0 is the joint distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{S})$, H_t is the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^{l\times t}$.

Hence,

$$\begin{aligned}&\left|\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_0 : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] - \mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_t : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1]\right| \\ &\leq \sum_{1\leq i\leq t} \left|\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_{i-1} : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] - \mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_i : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1]\right|.\end{aligned}$$

We construct a reduction $\mathcal{B}$ trying to simulate distribution H_{i-1} or H_i by contacting with an oracle $\mathcal{O}$. The oracle $\mathcal{O}$ returns a sample either from the distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{s})$ or the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^l$, where $\mathbf{s}$ is chosen from $\mathcal{D}$. $\mathcal{B}$ simulates a distribution in the following way:

1. $\mathcal{B}$ queries oracle $\mathcal{O}$ to obtain a sample $(\mathbf{C}, \mathbf{p})$, sets $\mathbf{p}_i \leftarrow \mathbf{p}$.
2. For $j > i$, $\mathcal{B}$ samples $\mathbf{s}_j \leftarrow \mathcal{D}$ and computes $\mathbf{p}_j \leftarrow \mathbf{C}\cdot\mathbf{s}_j$.
3. For $j < i$, $\mathcal{B}$ samples $\mathbf{p}_j \xleftarrow{\$} \mathbb{Z}_q^l$.
4. $\mathcal{B}$ sets $\mathbf{p}_j$ as the j-th column of $\mathbf{B}$.
5. $\mathcal{B}$ outputs $(\mathbf{C}, \mathbf{B})$.

Note that $\mathcal{B}$ can perfectly simulate the distribution H_{i-1} if it receives a sample from the distribution of $(\mathbf{C}, \mathbf{C}\cdot\mathbf{s})$, that is $\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_{i-1} : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] = \mathbf{Pr}[(\mathbf{C},\mathbf{C}\cdot\mathbf{s}) \leftarrow \mathcal{O} : \mathcal{B}(\mathbf{C},\mathbf{C}\cdot\mathbf{s}) = 1]$. Otherwise, $\mathcal{B}$ can perfectly simulate the distribution H_i, that is $\mathbf{Pr}[(\mathbf{C},\mathbf{B}) \leftarrow H_i : \mathcal{A}(\mathbf{C},\mathbf{B}) = 1] = \mathbf{Pr}[(\mathbf{C},\mathbf{u}) \leftarrow \mathcal{O} : \mathcal{B}(\mathbf{C},\mathbf{u}) = 1]$. □