Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption

Yao Jiang Galteland* and Jiaxin Pan**

NTNU – Norwegian University of Science and Technology, Norway
{yao.jiang, jiaxin.pan}@ntnu.no

Abstract. The understanding of directionality for updatable encryption (UE) schemes is important, but not yet completed in the literature. We show that security in the backward-leak uni-directional key updates setting is equivalent to the no-directional one. Combining with the work of Jiang (ASIACRYPT 2020) and Nishimaki (PKC 2022), it is showed that the backward-leak notion is the strongest one among all known key update notions and more relevant in practice. We propose two novel generic constructions of UE schemes that are secure in the backward-leak uni-directional key update setting from public key encryption (PKE) schemes: the first one requires a key and message homomorphic PKE scheme and the second one requires a bootstrappable PKE scheme. These PKE can be constructed based on standard assumptions (such as the Decisional Diffie-Hellman and Learning With Errors assumptions).

Keywords: Updatable Encryption · Public Key Encryption · Backward-leak Uni-Directional Key Update · No-Directional Key Update · Standard Assumption

1 Introduction

To mitigate key compromise over time, a data subject wishes to periodically update her outsourced data on a data host. The outsourced data is expected to be refreshed from the old key to the new key without changing the underlying message. During this process, it is also reasonable to expect that no information of plaintexts are leaked while updating.

Updatable encryption (UE) schemes [2, 3, 4, 6, 11, 12, 14, 16] are a special kind of encryption schemes that allow the data host to update ciphertexts with the help of a data subject generated updating material, namely update token.

* Her work has been co-funded by the IKTPLUSS program of the Research Council of Norway under the scope of and as part of the outcome from the research project Reinforcing the Health Data Infrastructure in Mobility and Assurance through Data Democratization (Health Democratization, 2019 - 2024, Project No. 288856.

** His work is supported by the Research Council of Norway under Project No. 324235.

Update tokens can rotate ciphertexts or keys, which makes UE schemes particularly interesting for outsourced data storage. However, leaked tokens together with key corruption an adversary may break confidentiality of the future or past epoch by upgrading or downgrading keys or ciphertexts, which is captured by the directionality of UE schemes. The study of UE mainly focuses on the security notions and efficient constructions. Directionality for UE schemes is important to study since it plays a central role in influencing the security result. The challenge is that there are two types of ciphertext update settings and four types of key update settings in the literature, and a combination of these settings results in eight different types of update settings for UE schemes to analyze.

Directionality of Ciphertext Updates. If an update token can only update a ciphertext under a key in the past to a ciphertext under a new key without changing the encrypted message, it is in the forward direction, then we call that such a UE scheme has uni-directional ciphertext updates; and if an update token can additionally update a ciphertext to another ciphertext under a key in the past, we call such a UE scheme with bi-directional ciphertext updates.

Directionality of Key Updates. Secret key leakage is a serious security threat to encryption. For instance, there is no security guarantee for standard encryption schemes if their secret keys are leaked. However, for UE schemes, the update token offers a potential to preserve confidentiality, since the update token allows us to update a secret key and the corresponding ciphertexts. Realizing this fully is very challenging and requires careful treatments, since the update token may also leak information about the key.

Directionality of key update is used to capture which information adversaries can learn about the secret keys given the update token. Roughly speaking, there are four key update settings given by the literature [9, 12, 14]. For a precise description, let e be an epoch, namely, the index of a time period. In the *forward-leak* uni-directional key update setting[1] [12], given a key k_{e} and an update token $\Delta_{\mathsf{e}+1}$ adversaries can only learn a key $k_{\mathsf{e}+1}$ in the forward direction. Similarly, in the *backward-leak* uni-directional key update setting [14], adversaries can only learn a key k_{e} in the backward direction, given $k_{\mathsf{e}+1}$ and $\Delta_{\mathsf{e}+1}$. If both forward-leak and backward-leak are satisfied in a setting, then it is called *bi-directional key update*. In contrary, if an update token leaks nothing about any secret key, then this setting is called *no-directional key update* [9].

Security Implications among Different Key Update Settings. Security of a UE scheme is defined with respect to the aforementioned key update settings. Roughly speaking, UE security guarantees confidentiality if the trivial win conditions are not triggered. The trivial win conditions are defined differently in each key update setting, and more information leaked about keys leads to more trivial win

[1] This was called uni-directional key updates in [12], but here we follow the more precise terminology of Nishimaki [14] and call it forward-leak uni-directional key updates.

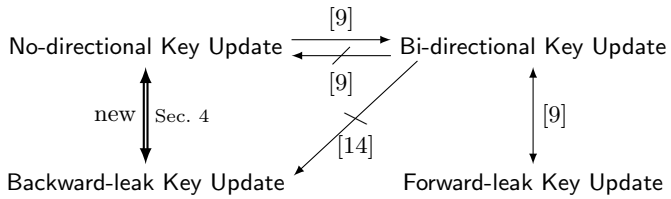

Fig. 1: Security implications among different key update settings assuming uni-directional ciphertext updates. $\mathsf{X} \rightarrow \mathsf{Y}$ means that security in the X setting implies that in the Y setting, and $\mathsf{X} \not\rightarrow \mathsf{Y}$ means that security in the X setting does not imply that in the Y setting, and $\mathsf{X} \leftrightarrow \mathsf{Y} = (\mathsf{X} \rightarrow \mathsf{Y}) \wedge (\mathsf{Y} \rightarrow \mathsf{X})$. Contribution in this paper is marked with a double arrow '$\Leftrightarrow$'.

conditions in the confidentiality game for UE schemes. With more trivial win conditions, it seems harder for an adversary to win the confidentiality, since it is easier for it to trigger the trivial win conditions. Thus, intuitively, a setting with less key leakage seems to give stronger security.

This intuition partially holds true, according to the work of Jiang [9] and Nishimaki [14]. More precisely, in [9] it has been showed that security in the no-directional setting is strictly stronger than that in the bi-directional setting, and security in the bi-directional key update setting is equivalent to that in the forward-leak setting. To further complete the work of Jiang, Nishimaki [14] proposed the backward-leak uni-directional setting and showed that UE schemes in prior works [4, 9, 12] are secure in the bi-directional setting but insecure in the backward-leak uni-directional setting. Here we consider that an update token can only update a ciphertext in the forward direction, since if the ciphertext updates are bi-directional then all four settings are equivalent as shown in [9]. The implications among these four key update settings are shown figuratively in Figure 1.

To sum up the discussions above, it is currently unclear that the relation between the no-directional and backward-leak uni-directional key update settings, although they both are stronger than the bi-directional and forward-leak setting.

Our Goal: UE schemes with Strong Security from Weak Assumptions. We aim at constructing UE schemes with strong security from weak assumptions. In achieving our goal, we first need to decide which notion is the strongest among the above four settings. Jumping ahead, our first contribution is proving the no-directional and backward-leak settings are equivalent. Given our equivalence result, we claim it is more desirable to construct a UE scheme that is secure in the backward-leak uni-directional key update setting for the following reasons.

Firstly, UE schemes secure in the backward-leak setting are technically more promising to construct based on weak assumptions, since the existing UE scheme [14] with no-directional key updates are based on strong assumptions. Namely, the scheme in [14] requires a rather strong and impractical primitive, indistinguishability obfuscation.

Secondly, although there is a backward-leak UE scheme based on the Learning With Errors (LWE) assumption proposed by Nishimaki [14], it is unknown whether backward-leak UE schemes can be constructed from a wider class of weak assumptions, for instance, the Diffie-Hellman assumption without pairings. We are particularly interested in constructing UE schemes generically from public-key encryption (PKE), since this not only is theoretically interesting, but also can give us UE schemes from rather weak assumptions. Recently, Alamati, Montgomery, and Patranabis [1] have proved that ciphertext-independent UE implies PKE, but the implication in the other direction is unknown.

Finally, we stress that the backward-leak setting is relevant for practice, as discussed in [14]. In practice, the purpose of updating our keys is mostly because the current key and those in the past may be leaked. In such a scenario, UE schemes in the backward-leak uni-directional key update setting are required, since they can provide confidentiality in an epoch, even though all previous keys and tokens are corrupted. Moreover, backward-leak UE schemes remain secure even if the data host forgets to delete older keys and tokens, while this is not the case for forward-leak UE schemes, since with the older keys an adversary can learn the keys in the future.

1.1 Our Contributions

Intuition behind security definitions. Our first contribution is providing an intuitive understanding of trivial win conditions, firewalls, directionality and security notions. Explanations of all these topics exists in [3, 4, 9, 11, 12, 14], however, we aim to provide a simple description to show the relations among these definitions. We consider two classes of UE schemes (discussed in Section 3.2 and 3.3): the first class of UE schemes have update settings such that keys cannot upgrade and ciphertexts cannot downgrade, the second class of UE schemes have update settings where keys and ciphertexts both can leak information in the forward direction. We observe that the first class of UE schemes (including UE schemes with backward-leak uni-directional key updates and no-directional key updates) can achieve the strongest confidentiality notion (with post-compromise security). Thus, it is only necessary to analyze two classes of UE schemes, and these two classes of UE schemes matches with the equivalence result in the literature [9, 14] and our work. That is, the eight variants of confidentiality notions can be seen as only two classes of confidentiality notions. We will show that notions in the same class are equivalent and one class is strictly stronger that the other.

Equivalence result. Our second contribution is proving that security in the backward-leak uni-directional key update and uni-directional ciphertext update setting is equivalent to that in the no-directional key update and uni-directional ciphertext update setting. All our UE schemes have uni-directional ciphertext updates, and for simplicity, we do not mention it explicitly in the remaining of this section. This means that UE schemes with no-directional key updates do not provide stronger security than UE schemes with backward-leak uni-directional

key update. Our result suggests that constructing UE schemes with backward-leak uni-directional key update is equivalent to constructing UE schemes with no-directional key update.

Generic Constructions of UE from PKE. Our third contribution is constructing two generic constructions of UE schemes with backward-leak uni-directional key update from PKE schemes. Our constructions require additional properties of the underlying PKE schemes. Our first construction requires key and message homomorphism for PKE schemes. Such PKE schemes can be instantiated under the Decisional Diffie-Hellman (using the ElGamal encryption) and LWE (using the Regev encryption [15]) assumptions. Combining with our equivalence result, the aforementioned two schemes provide us with the *first* no-directional secure UE schemes without pairings in the Diffie-Hellman setting and based on a post-quantum assumption, respectively. We note that the uni-directional schemes from FHE or IO or lattice trapdoors [14] usually do not have this increased key-size or ciphertext-size. But without these assumptions and technique, uni-directional schemes relying on standard assumptions (namely, ours and the work in [13]) are constructed with growing key and cipehrtextext. It remains an open problem to construct uni-directional schemes relying on standard assumptions where the key and cipehrtextext size keeps the same.

Our second generic construction uses a bootstrappable PKE [8] that can be implemented using the LWE assumption, which again gives us a post-quantum UE scheme with security in the no-directional key update setting.

Of independent interest, we propose a generic construction of bi-directional UE scheme from a key homomorphic PKE scheme. Our generic construction abstracts the constructions of RISE [12] and LWEUE [9]. We stress that our notion of key homomorphic PKE is inspired by the key homomorphic PRF of Boneh et al. [3] but different to theirs. More precisely, our key homomorphic property is defined with respect to a public key and a secret key, while theirs is with respect to two secret keys.

Technical Overview. Here we provide a brief technical overview of our generic backward-leak UE constructions from PKEs. The full descriptions of our schemes can be found in Sections 5.3 and 6. The update token plays an important role in a UE scheme, and therefore we mostly focus on it in the following.

In our first construction, its key contains a pair of secret and public keys from the PKE scheme. The update token contains the difference between the old and new secret keys. In addition, the token includes an independently generated public key, which will play a central role for the confidentiality in the next epoch. To update a ciphertext, we have two steps: Firstly, by the key homomorphic property of the PKE, the difference between the old and new secret keys can be used to modify the ciphertext under the old key to one under the new key. Secondly, we randomize this ciphertext so that it is indistinguishable to a freshly generated ciphertext under the new key. In doing this, we use the aforementioned independent public key to encrypt a randomness to homomorphically randomize the ciphertext, since our PKE is further message homomorphic. In the security

proof, we can show that message is hidden by the randomness and confidentiality in the new epoch is preserved.

In our second construction, the update token is an encryption of the old secret key under the new public key. This token will not reveal any information about the new secret key even with the knowledge of the old secret key. To update a ciphertext, we evaluate the decryption circuit on the encryption of the old key (that is the token) and the encryption of the old ciphertext. The bootstrapping property states that this output is statistically close to a fresh ciphertext under the new key.

Concurrent Work. We note a recent work from Miao, Patranabis, and Watson [13] which has a construction of backward-leak uni-directional UE similar to ours, while our work contains a formal proof about the equivalence between backward-leak and no-directional UE.

1.2 More Discussion

Ciphertext-Dependent v.s. Ciphertext-Independent. We call an updatable encryption scheme is ciphertext-dependent [2, 3, 5, 6] if the token generation process depends on the old ciphertext. If the token generation process is independent of the ciphertext to be updated, then the UE scheme is called ciphertext-independent [4, 9, 11, 12, 14, 16]. A ciphertext-independent UE scheme is usually more efficient in terms of bandwidth cost, and, thus, we focus on such schemes in this paper.

Deterministic Update. The update algorithm in UE schemes can be deterministic or randomized. UE schemes with randomized update algorithm provide stronger security, in which the updated ciphertext can be in the same distribution of a fresh encryption. However, the work of Klooß et al. [11] shows that such UE schemes with randomized update cannot achieve ciphertext integrity and security against chosen-cipertext attacks (CCA), but replayable CCA security. For instance, SHINE [4] and E&M [11] are UE schemes with deterministic update that have ciphertext integrity and CCA security. Klooß et al.'s result also means that our constructions cannot be CCA secure, but it is promising to make it RCCA secure. We leave constructing this as an open problem.

Security Notions. Boneh et al. [3] presented the first security notion for UE schemes. After that, the works in [4, 9, 11, 12] proposed more realistic security notion by providing more capability to an adversary. For instance, it can adaptively corrupt epoch keys or update tokens at any point within the security game. Jiang [9] first discussed the directionality of security notions and showed that security notions with forward-leak uni- and bi-directional updates are equivalent in the current state-of-the-art UE security notion of Boyd et al. [4]. In addition, Jiang [9] proved that confidentiality notions with no-directional key updates are strictly stronger than uni- and bi-directional update variants of the corresponding notions. Nishimaki [14] defined a new type of key update, backward-leak uni-directional key update, which is not covered in the work of [9]. We will prove the

relation between the backward-leak uni-directional key update variant of a confidentiality notion with other update variants of the same confidentiality notion. We will show that confidentiality notions with backward-leak uni-directional key update is equivalent to no-directional key updates variants of the corresponding notions. Which means the backward-leak uni-directional key update variant of notions are the strongest notions.

Slamanig and Striecks [16] introduced a stronger model for UE schemes, where they consider an "expiry epoch", $\mathsf{e}_{\mathsf{exp}}$. If a ciphertext is updated to an epoch e, where $\mathsf{e} \geq \mathsf{e}_{\mathsf{exp}}$ and $\mathsf{e}_{\mathsf{exp}}$ is this ciphertext's expiry epoch, then this ciphertext can no longer be decryptable. Their model allows the adversary knows "all"[2] tokens. Forward security is guaranteed if the key updates are at most in the forward-direction and leaked keys are in epochs after the expiry epoch of a ciphertext. Post compromise security is guaranteed if the key updates are at most in the backward-direction. Realizing such strong security strictly requires no-directional UE schemes where keys must not be updatable in any direction. Note that Jiang's equivalence theorem [9] holds in the setting where there is no expiry date for ciphertexts, i.e., $\mathsf{e}_{\mathsf{exp}} = \infty$. We consider the case for $\mathsf{e}_{\mathsf{exp}} = \infty$ in this paper. We do not compare the efficiency of our construction with the UE construction in [16], due to schemes are in different UE models.

2 Preliminaries

In this section we describe the notation used in this paper and present the necessary background material of updatable encryption. Due to space limitations, we provide the preliminaries for public key encryption in the full version [7]. y parameter and negl denotes a negligible function. For distributions X and Y, $X \overset{s}{\approx} Y$ means X is statistically indistinguishable from Y.

2.1 Updatable Encryption and Confidentiality Notions

Updatable encryption (UE) schemes [3, 4, 11, 12] are a special kind of encryption schemes with an additional functionality where ciphertexts under one key can be transferred to ciphertexts under another key by an update token.

Definition 1 (UE). *An updatable encryption scheme* UE *is parameterized by a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ *that operates over epochs such that*

- *The setup algorithm* $\mathsf{Setup}(\lambda)$ *takes a security parameter* λ *as input, and outputs an initial epoch key* k_1*.*
- *The next algorithm* $\mathsf{Next}(k_{\mathsf{e}})$ *takes an epoch key* k_{e} *as input, and outputs a new key* $k_{\mathsf{e}+1}$ *and an update token* $\Delta_{\mathsf{e}+1}$*, the update token can be used to update ciphertexts from epoch* e *to* $\mathsf{e}+1$*.*

[2] except for some end epoch $\mathsf{e}_{\mathsf{end}}$, if $\mathsf{e}_{\mathsf{exp}} \leq \mathsf{e}_{\mathsf{end}}$.

– *The encryption algorithm* $\mathsf{Enc}(k_\mathsf{e}, m)$ *takes an epoch key* k_e *and a message* m *as input, and outputs a ciphertext* c_e.
– *The decryption algorithm* $\mathsf{Dec}(k_\mathsf{e}, c_\mathsf{e})$ *takes an epoch key* k_e *and a ciphertext* c_e *as input, and outputs a message* m.
– *The update algorithm* $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, c_\mathsf{e})$ *takes an update token* $\Delta_{\mathsf{e}+1}$ *and a ciphertext* c_e *as input, and outputs an updated ciphertext* $c_{\mathsf{e}+1}$.

UE *is* correct *if for any message* m, *any* $k_1 \leftarrow \mathsf{Setup}(\lambda)$, *any* $(k_j, \Delta_j) \leftarrow \mathsf{Next}(k_{j-1})$ *for* $j = 2, ..., \mathsf{e}$, *and any* $c_i \leftarrow \mathsf{Enc}(k_i, m)$ *with* $i \in \{1, ..., \mathsf{e}\}$, *we have* $m = \mathsf{Dec}(k_\mathsf{e}, c_\mathsf{e})$ *where* $c_j \leftarrow \mathsf{Upd}(\Delta_j, c_{j-1})$ *for* $j = i + 1, ..., \mathsf{e}$.

Security notions for UE schemes include confidentiality and integrity. We do not consider integrity notions in this paper, see the paper by Jiang [9] for details. We review the confidentiality notion for UE schemes in Definition 2 and extend the six variants of security notions for UE schemes given by [9], in which the backward-leak uni-directional key update [14] variants are not included.

The confidentiality game is played between a challenger and an adversary, the adversary aims to distinguish a fresh encryption from an updated ciphertext. It is allowed for the adversary to adaptively corrupt keys and tokens, if any trivial win condition is triggered during the game the adversary will always lose. We will provide technical and high level understanding of trivial win conditions in Section 2.2 and 3.

Definition 2 ([4, 9]). *Let* $\mathsf{UE} = (\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ *be an updatable encryption scheme. Then the* $(\mathsf{kk}, \mathsf{cc})$-$\mathsf{xx}$IND-UE-$\mathsf{atk}$ *advantage, for* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}, \mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ *and* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, *of an adversary* $\mathcal{A}$ *against* UE *is defined as*

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

where the experiment $\mathbf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{xx}\mathsf{IND}\text{-}\mathsf{UE}\text{-}\mathsf{atk}\text{-}\mathsf{b}}_{\mathsf{UE},\,\mathcal{A}}$ *is given in Fig. 2.*

2.2 Leakage Sets and Trivial Win Conditions

In this section, we review the definition of leakage sets and trivial win discussions [4, 9, 11, 12], the leaked information can be used to help an adversary trivially win the confidentiality game.

Trivial Win via Key and Ciphertext Leakage. If an adversary knows both the key and the challenge-equal ciphertext in the same epoch period e, then the adversary can use this key to decrypt the challenge-equal ciphertext and obtain the underlying plaintext to win the confidentiality game. We use leakage sets to identify if this trivial win condition ("$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$") is triggered.

Leakage sets [9, 11, 12] are defined to track epochs in which the adversary knows a key, a token, or learned a version of challenge ciphertext. The direct

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

if $\mathsf{twf} = 1$ **then**
$\mathrm{b}' \xleftarrow{\$} \{0,1\}$
return b'

$\underline{\mathbf{Setup}(\lambda)}$
$k_1 \xleftarrow{\$} \mathsf{Setup}(\lambda)$
$\Delta_1 \leftarrow \perp; \mathsf{e}, \mathrm{c}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

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

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

$\underline{\mathcal{O}.\mathsf{Next}():}$
$(\Delta_{\mathsf{e}+1}, k_{\mathsf{e}+1}) \leftarrow \mathsf{Next}(k_{\mathsf{e}})$
if $\mathsf{phase} = 1$ **then**
$\tilde{c}_{\mathsf{e}+1} \leftarrow \mathsf{Upd}(\Delta_{\mathsf{e}+1}, \tilde{c}_{\mathsf{e}})$
$\mathsf{e} \leftarrow \mathsf{e} + 1$

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

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

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

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

Fig. 2: The confidentiality experiment $\mathbf{Exp}^{\mathsf{(kk,cc)\text{-}xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ for updatable encryption scheme UE and adversary $\mathcal{A}$, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ 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. Oracles an adversary can query are $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Upd}$, $\mathcal{O}.\mathsf{Corr}$, $\mathcal{O}.\mathsf{Chall}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ if $\mathsf{atk} = \mathsf{CPA}$. If $\mathsf{atk} = \mathsf{CCA}$, $\mathcal{O}.\mathsf{Dec}$ is included in the oracles. Leakage sets $\mathcal{C}, \mathcal{K}, \mathcal{T}$, $\mathcal{K}^*_{\mathsf{kk}}, \mathcal{T}^*_{\mathsf{kk}}, \mathcal{C}^*_{\mathsf{kk,cc}}, \tilde{\mathcal{L}}^*_{\mathsf{kk,cc}}, \tilde{\mathcal{Q}}^*_{\mathsf{kk,cc}}$ are discussed in Section 2.2.

leakage sets $\mathcal{K}, \mathcal{T}, \mathcal{C}$ are describe as follows. Furthermore, $\mathcal{K}^*, \mathcal{T}^*$ and $\mathcal{C}^*$ are defined as the extended sets of $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{C}$ to track the indirect leakage.

- $\mathcal{K}$: Set of epochs in which the adversary corrupted the key (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{T}$: Set of epochs in which the adversary corrupted the token (from $\mathcal{O}.\mathsf{Corr}$).
- $\mathcal{C}$: Set of epochs in which the adversary learned a challenge-equal ciphertext[3] (from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$).

Key Leakage. The size of the key leakage set $\mathcal{K}^*$ can be influenced by the key update direction of UE schemes. In the no-directional key update setting [9], the adversary does not have more information about keys except for set $\mathcal{K}$. In the forward-leak uni-directional key update setting, if the adversary knows a key k_{e} and an update token $\Delta_{\mathsf{e}+1}$ then it can infer the next key $k_{\mathsf{e}+1}$. In the backward-leak [14] uni-directional key update setting, if the adversary knows a key $k_{\mathsf{e}+1}$ and an update token $\Delta_{\mathsf{e}+1}$ then it can infer the previous key k_{e}.

The notations $\mathsf{f\text{-}uni}$ and $\mathsf{b\text{-}uni}$ denote forward-leak uni and backward-leak uni, resp.. In the kk-directional key update setting, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$, denote the set $\mathcal{K}^*_{\mathsf{kk}}$ as the extended set of corrupted key epochs. We compute these sets as follows, where the boxed part is only computed for $\mathsf{kk} \in \{\mathsf{b\text{-}uni}, \mathsf{bi}\}$, the gray boxed part is only computed for $\mathsf{kk} \in \{\mathsf{f\text{-}uni}, \mathsf{bi}\}$

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

Token Leakage. The adversary directly learns all corrupted tokens, it can also compute a token from two consecutive epoch keys. We follow the assumption (an update token can be computed via two consecutive epoch keys) in the page 7 of [10], this assumption is essential to formulate the known knowledge to the adversary. Hence, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$, denote $\mathcal{T}^*_{\mathsf{kk}}$ as the extended set of corrupted token epochs.

$$\mathcal{T}^*_{\mathsf{kk}} \leftarrow \{\mathsf{e} \in \{0, ..., l\} | (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{kk}} \wedge \mathsf{e}\text{-}1 \in \mathcal{K}^*_{\mathsf{kk}})\}. \tag{2}$$

Challenge-Equal Ciphertext Leakage. The adversary learned all versions of challenge ciphertexts in epochs in $\mathcal{C}$. Additionally, the adversary can compute 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.

For $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, denote the set $\mathcal{C}^*_{\mathsf{kk},\mathsf{cc}}$ as the extended set of challenge-equal epochs. We compute these sets as follows, where

[3] A challenge-equal ciphertext is either a challenge ciphertext or an updated ciphertext of the challenge ciphertext.

the boxed part is only computed for $\mathsf{cc} = \mathsf{bi}$.

$$\begin{aligned}
&\mathcal{C}^*_{\mathsf{kk},\mathsf{cc}} \leftarrow \{\mathsf{e} \in \{0,\ldots,l\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\} \\
&\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff \\
&\quad (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}) \vee \boxed{\mathsf{ChallEq}(\mathsf{e}{+}1) \wedge \mathsf{e}{+}1 \in \mathcal{T}^*_{\mathsf{kk}}}.
\end{aligned} \tag{3}$$

Trivial Win due to Deterministic Update. In a confidentiality game with deterministic update setting. If the adversary knows the updated version (by either knowing the update token $\Delta_{\tilde{\mathsf{e}}}$ or asking for an update oracle $\mathcal{O}.\mathsf{Upd}$ on $\bar{c}$) of the challenge input ciphertext $\bar{c}$, it can compare the updated ciphertext with the challenge ciphertext to win the confidentiality game. This trivial win condition is " $\tilde{\mathsf{e}} \in \mathcal{T}^*$ **or** $\mathcal{O}.\mathsf{Upd}(\bar{c})$ is queried".

Trivial Wins via Decryption. If the adversary submits a challenge-equal ciphertext to the decryption oracle, it can trivially win the confidentiality game by comparing the challenge plaintexts with the decryption output. Hence, the adversary is not allowed to ask for a decryption oracle on such ciphertexts. We use the following sets to track challenge ciphertexts, challenge plaintexts and their updated versions that can be known to the adversary. These sets can be used to identify the above trivial win condition. More precisely, " $(c, \mathsf{e}) \in \tilde{\mathcal{L}}^*$" is a trivial win condition that is checked by the decryption oracle in the $\mathsf{detIND\text{-}UE\text{-}CCA}$ game, " $(m', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$" is a trivial win condition that is checked by the decryption oracle in the $\mathsf{randIND\text{-}UE\text{-}CCA}$ game.

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

3 Intuitions behind Security Definitions.

In this section, we propose some high level intuitions behind the security notions for UE.

We aim to clarify the relations among trivial win conditions, directionality and security results. If a UE scheme potentially leaks more information (which can be influenced by directionality), then there exists more vulnerabilities (trivial win conditions) for such scheme. Forward security in prior work of UE are achieved under the limitation that no trivial win condition is triggered, namely, there may exist some token after the challenge epoch cannot be corrupted. In our work, we define a relaxed version of forward security that, after the challenge epoch, any keys and tokens can be corrupted and the confidentiality of the challenge epoch remains. Similarly, we discussed post-compromise security.

In the end, we provide some observations about what types of UE schemes can achieve forward or post-compromise security.

3.1 Intuition of Trivial Wins Conditions

The more information that get leaked in a confidentiality game the more chances the adversary gains to win that game, and trivial win conditions are defined to exclude such winning conditions. Generally speaking, the more update directions a UE scheme has the more information the adversary can infer. That is, the directionality of the update setting influences how much information gets leaked.

Trivial win conditions can be seen as a way of limiting the adversary's attacking power and this can be used to compare two notions, where two notions for UE schemes are equivalent if they have the same winning probability. Consider a modified confidentiality game where the adversary is not allowed to perform certain actions that will trigger trivial win conditions. If such action happens, the game aborts. The winning probability of the original confidentiality game is the same as this modified confidentiality game. In other words, trivial win conditions are equivalent to an attacking model. Such attacking model defines the restriction to the adversary, for example, the adaptive corruption ability is restricted by not triggering the trivial win conditions. Only when the adversary's attack actions does not trigger the trivial win conditions it is possible to win the game. Informally, a security notion for a system can be seen as stronger if the system remains secure even if the adversary has more attacking ability.

Combining this with the discussion above, we see that the more update direction a UE scheme has the more key and ciphertext leakage there will be and more trivial win conditions to evaluate. This means that the adversary will be limited more in such a confidentiality game. From a security point of view, the less attacking ability the adversary has the weaker we can regard a security notion.

3.2 Intuition of Firewalls

The observation of firewalls was introduced in the work of Lehmann and Tackmann [12], Klooß et al. [11] provided an extended description of this *key insulation* technique, and Boyd et al. [4] formally defined it as *firewall technique*. Furthermore, Nishimaki [14] proposed a *relaxed firewall*, where the token on the left side of the insulated region (Δ_{fwl}) can be corrupted. In any security game, if the adversary never triggers the trivial win conditions, a cryptographic separation (firewalls) exists, for a detailed discussion of the existence of firewalls see [4, 11, 12, 14].

Definition 3 (Firewalls [4, 11, 12]). *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;*
- *the tokens* Δ_{fwl} *and* $\Delta_{\mathsf{fwr}+1}$ *are not corrupted (if they exist);*

– *all tokens* $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ *are corrupted (if any exist).*

Definition 4 (Relaxed Firewalls [14]). *A* relaxed insulated region *with* relaxed 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;*
– *the token* $\Delta_{\mathsf{fwr}+1}$ *is not corrupted (if it exists);*
– *all tokens* $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ *are corrupted (if any exist).*

The firewall technique is used when proving the security for UE schemes, it provides a method of describing a cryptographic separation, which is required in the epoch based model to simulate where keys and ciphertexts are known or unknown to the adversary. Firewalls, and relaxed firewalls, define a "safe" or insulated region for keys, where no key within the region can be inferred from keys outside of this region. We can regard the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ as the left and right firewalls, the cryptographic separation is created when these two tokens are unknown to the adversary. In some UE settings, the token Δ_{fwl} can be corrupted and the cryptographic separation still holds, which means that the insulated region can be relaxed even without the left firewall.

The relaxed insulated region is suitable for analyzing security for UE schemes with update settings such that keys cannot upgrade and ciphertexts cannot downgrade, namely ciphertext and key will not leak information in the same direction. In UE schemes with uni-directional ciphertext update setting and key update settings without forward-leak direction, we have that the token Δ_{fwl} cannot upgrade early epoch keys to learn keys inside the insulated region, it cannot downgrade challenge-equal ciphertexts to learn early challenge-equal ciphertexts outside of the insulated region as well. Hence, keys and ciphertexts inside and outside of the insulated region are separated even when Δ_{fwl} is known to the adversary, the insulated region can be relaxed to allow the token Δ_{fwl} to be corrupted.

The (original) insulated region is suitable for analyze UE schemes with update settings such that keys and ciphertexts can both leak information in the forward direction, namely ciphertext and key will leak information in the same direction. In UE schemes with ciphertext and key update settings that both have at least forward-leak direction, we have that the token Δ_{fwl} can upgrade early corrupted keys to learn keys inside the insulated region. For these UE schemes we have that the token Δ_{fwr} can upgrade challenge-equal ciphertexts to an epoch outside the insulated region where the adversary knows a corrupted key. Hence, both tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are required to be unknown to the adversary to make a cryptographic separation.

3.3 Forward Security and Post-Compromise Security

Forward and post-compromise security for UE schemes were discussed by Lehmann and Tackmann [12]. However, all confidentiality games in their work are restricted by not triggering trivial win conditions, which means there exists a

cryptographic separation between the leaked key region and the "safe" region (see the discussion of cryptographic separation in Section 3.2). That is, there exists two tokens one before and one after the epoch where the adversary aims to break the confidentiality such that these tokens cannot be corrupted.

We consider the standard definitions for forward and post-compromise security, in which we do not have the restrictions of tokens which cannot be corrupted. We say a UE scheme have

- *forward security* if the confidentiality in early epochs are not broken even if an adversary compromises keys and tokens in some later epochs.
- *post-compromise security* if the confidentiality in later epochs are not broken even if an adversary compromises keys and tokens in some early epochs.

We observe that only some specific UE schemes can achieve post-compromise security. No UE scheme can achieve forward security.

The first class of UE schemes, discussed in Section 3.2, have update settings such that keys cannot upgrade and ciphertexts cannot downgrade. Since the ciphertext update is forward direction, an adversary can upgrade challenge-equal ciphertexts by the help of tokens to an epoch where the adversary knows a key to break the confidentiality in early epochs. Therefore, such UE schemes cannot achieve forward security. However, an adversary compromises keys and tokens in some early epochs cannot learn keys to break the confidentiality in later epochs. Additionally, the adversary cannot downgrade a challenge-equal ciphertext to an early epoch where the adversary knows a key to break the confidentiality in later epochs. Hence, such UE schemes can have post-compromise security.

The second class of UE schemes, discussed in Section 3.2, have update settings where keys and ciphertexts both can leak information in the forward direction. An adversary can infer keys by tokens in the forward direction to break the confidentiality in later epochs, therefore, such UE schemes cannot achieve post-compromise security. Moreover, since the ciphertext update is forward direction, similar to the discussion in above paragraph, such UE schemes cannot achieve forward security either.

The above discussion matches with the equivalence results of confidentiality notions (see Section 4). It implies that the first class of UE schemes (including UE schemes with backward-leak uni-directional key updates and no-directional key updates) can achieve the strongest confidentiality notion (with post-compromise security).

3.4 Directionality for UE schemes and Confidentiality Notions

We specify that directionality for UE schemes and directionality for confidentiality notions are two different concepts. The update directionality for UE schemes was defined to measure how much keys and ciphertexts are leaked because of update tokens. However, such leakage can be the whole information leakage or just partial information leakage. The partial leakage is not captured in the directionality for UE schemes, there is no definition for partial information leakage.

Under current definitions, we consider update direction for UE schemes to be without partial information leakage. However, partial information leakage may be used to break the confidentiality. An adversary may be able to decrypt ciphertexts where it only knows partial information about the corresponding key. Such partial information leakage is considered in the security notion and it can be seen as equivalent to the whole information leakage. For example, suppose tokens in a UE scheme can be used to infer partial key information in both update direction, such UE schemes are considered as a no-directional key update UE scheme. However, if an adversary can use this partially leaked key to break confidentiality, then such UE scheme is not secure in the no-directional update variant of confidentiality, it can at most be secure in the bi-directional update variant of confidentiality. Overall, UE schemes with one kind of update setting do not immediately have the same update direction variant of security.

A specific update variant of security notion is suitable for examining UE schemes with the same update setting. But we stress that our security definitions are not restricted to UE schemes with some particular update setting, such UE schemes are simply insecure in a less updating (or leakage) variant of notion.

4 Relations among Confidentiality Notions

In the work of [10], Jiang showed that all variants of the same integrity notions are equivalent, hence, we do not consider integrity notions in this work and focus on discussing the relations among confidentiality notions in this section. We prove that the backward-leak uni- and no directional key update variant of the same confidentiality notion are equivalent. As a result, UE schemes with backward-leak uni-directional key updates is as strong as UE schemes with no directional key update. It implies that we can construct a less hard (backward-leak uni-directional key updates) UE scheme to achieve the same security result.

4.1 Equivalence for Trivial Win Conditions

We prove four equivalence of the trivial win conditions. The proofs of these equivalence lemmas follow the proof strategy in [9], and they are provided in the full version [7]. The trivial win conditions considered in this section are checked in a confidentiality game. In conclusion, if the trivial win conditions in the backward-leak uni-directional key update setting are triggered then the same trivial win conditions in the no directional key update setting would be triggered. We will use these trivial win equivalences to prove the relation in Theorem 1.

Lemma 1 (Equivalence for Trivial Win Condition "$\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$"). *For any sets $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., l\}$, we have $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni},\mathsf{uni}} \neq \emptyset \iff \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no},\mathsf{uni}} \neq \emptyset$.*

Lemma 2 (Equivalence for Trivial Win Condition "$\tilde{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}.\mathsf{Upd}(\bar{c})$ is queried"). *For any $\mathcal{K}, \mathcal{T}, \mathcal{C}$. Suppose $\mathcal{K}^*_{\mathsf{kk}} \cap \mathcal{C}^*_{\mathsf{kk},\mathsf{cc}} = \emptyset$, where $\mathsf{kk} \in \{\mathsf{b\text{-}uni}, \mathsf{no}\}$, $\mathsf{cc} = \mathsf{uni}$, then*

$$\tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{no}} \textbf{ or } \mathcal{O}.\mathsf{Upd}(\bar{c}) \textit{ is queried} \iff \tilde{\mathsf{e}} \in \mathcal{T}^*_{\mathsf{b\text{-}uni}} \textbf{ or } \mathcal{O}.\mathsf{Upd}(\bar{c}) \textit{ is queried.}$$

Lemma 3 (Equivalence for Trivial Win Condition " $(c, \mathsf{e}) \in \tilde{\mathcal{L}}^*$"). *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. *Suppose* $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,uni}} = \emptyset$, *then*

$$(c, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{b\text{-}uni,uni}} \iff (c, \mathsf{e}) \in \tilde{\mathcal{L}}^*_{\mathsf{no,uni}}.$$

Lemma 4 (Equivalence for Trivial Win Condition " $(m', \mathsf{e}) \in \tilde{\mathcal{Q}}^*$"). *For any sets* $\mathcal{K}, \mathcal{T}, \mathcal{C} \subseteq \{0, ..., \mathsf{e}\}$. $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,uni}} = \emptyset$, *then* $(m', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{b\text{-}uni,uni}} \iff (m', \mathsf{e}) \in \tilde{\mathcal{Q}}^*_{\mathsf{no,uni}}$.

4.2 Relations among Confidentiality Notions

Jiang [10] showed that the bi-directional key update and the forward-leak uni-directional key update variants of the same confidentiality notions are equivalent, and confidentiality in the no-directional key update is strictly stronger than that in the above mentioned two key update settings. However, the relation between the backward-leak uni-directional key update and the no-directional key update variants of the same confidentiality notion is missing in the previous work. We complete the relations among the eight variants of the same confidentiality notion, they are described as in Fig. 3. This is proven via Theorem 1, given below, and results in [10].

Recall discussions in Section 3.2 and 3.3, where we consider update settings in $\{(\mathsf{b\text{-}uni}, \mathsf{uni}), (\mathsf{no}, \mathsf{uni})\}$ are in one class and the rest update settings are in another class. Intuitive observation shows notions in the same class are equivalent and the prior class is strictly stronger than the latter. Interestingly, the result showed in Fig. 3 matches with this intuition and provides a rigorous proof.

$$(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{notion} \xleftrightarrow{\text{Theorem 1}} (\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion} \underset{*}{\overset{*}{\rightleftarrows}} (\mathsf{kk}, \mathsf{cc})\text{-}\mathsf{notion}$$

Fig. 3: Relations among the eight variants of the same confidentiality notion, where $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$, $\mathsf{kk} \in \{\mathsf{no}, \mathsf{b\text{-}uni}, \mathsf{f\text{-}uni}, \mathsf{bi}\}$ and $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$, the notation of $(\mathsf{kk}, \mathsf{cc})$ are except for values in $\{(\mathsf{b\text{-}uni}, \mathsf{uni}), (\mathsf{no}, \mathsf{uni})\}$. Results in work of [10] are marked with $*$.

In Theorem 1, we compare two types of UE notions: the no-directional key update setting and the backward-leak uni-directional setting. To illustrate the intuition for our equivalence between these two settings we have two scenarios: the first is where the adversary has corrupted a key located in an epoch earlier that the challenge epoch, and the second is where the adversary has corrupted a key located in an epoch later that the challenge epoch. In the first scenario, the adversary cannot update the key even if she had all update token because in both update settings forward key update is impossible. Similarly, the challenge ciphertext cannot be downgraded. Thus, in the first scenario, both update settings are equivalent. For the second scenario, we have that the adversary will

win the game in both update settings if she has access to enough tokens, because now the key can either be downgraded, using the tokens, or the challenge ciphertext can be upgraded, using the same tokens. So, in the second scenario, both update settings are equivalent.

Theorem 1. *Let* $\mathsf{UE} = (\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ *be an updatable encryption scheme and* $\mathsf{notion} \in \{\mathsf{detIND\text{-}UE\text{-}CPA}, \mathsf{randIND\text{-}UE\text{-}CPA}, \mathsf{detIND\text{-}UE\text{-}CCA}, \mathsf{randIND\text{-}UE\text{-}CCA}\}$. *For any* $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{notion}$ *adversary* $\mathcal{A}$ *against* UE*, there exists a* $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ *adversary* $\mathcal{B}_1$ *against* UE *such that*

$$\mathbf{Adv}^{(\mathsf{b\text{-}uni},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) = \mathbf{Adv}^{(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_1}(\lambda).$$

Proof. The proof follows the same method as the proof of Theorem 3.1 in [10]. We construct a reduction $\mathcal{B}_1$ running the $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ experiment which will simulate the responses of queries made by the $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{notion}$ adversary $\mathcal{A}$. The reduction will send all queries received from $\mathcal{A}$ to its $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ 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 $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ 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 equivalences of trivial win conditions in Section 4.1 (Lemma 1 to 4), if $\mathcal{A}$ does not trigger the trivial win conditions in the $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{notion}$ game, then the reduction will not trigger the trivial win conditions in the $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ game either. If $\mathcal{A}$ triggers the trivial win conditions in the $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{notion}$ game, then the reduction will also trigger the trivial win conditions in the $(\mathsf{no}, \mathsf{uni})\text{-}\mathsf{notion}$ game. Therefore, the reduction perfectly simulates the responses to adversary $\mathcal{A}$. And we have $\mathbf{Adv}^{(\mathsf{no},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{B}_1}(\lambda) = \mathbf{Adv}^{(\mathsf{b\text{-}uni},\mathsf{uni})\text{-}\mathsf{notion}}_{\mathsf{UE},\ \mathcal{A}}(\lambda)$.

5 UE from Key Homomorphic PKE

We use key homomorphic PKE schemes to construct UE schemes in this section. The idea of key homomorphic by Boneh et al. [3] inspired this construction idea. Our key homomorphic is more general than the key homomorphic construction presented by Boneh et al. [3], where we can rotate information based on a public value and a secret value instead of two secret values.

5.1 Key Homomorphic PKE

We define key homomorphic PKE in this section, which will be used to build updatable encryption schemes in the later two sections.

Definition 5 (Key Homomorphic PKE). *We say public key encryption* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *is* key homomorphic *if:*

1. *there exists an efficiently computable secret key to public key algorithm* $[\cdot] : \mathcal{SK} \to \mathcal{PK}$ *such that for any* $(sk, pk) \leftarrow \mathsf{KG}(\lambda)$*, the following two pairs of distributions are statistically close:*

$$(sk, [sk]) \overset{s}{\approx} (sk, pk). \tag{4}$$

$$\{(sk_1 \otimes sk_2, [sk_1 \otimes sk_2]) \mid sk_1, sk_2 \overset{\$}{\leftarrow} \mathcal{SK}\} \overset{s}{\approx} \{(sk, [sk]) \mid sk \overset{\$}{\leftarrow} \mathcal{SK}\}. \tag{5}$$

   where $\otimes$ *is an operation (which can be addition, multiplication, etc.) over the secret key space.*
2. *there exists an efficiently computable key homomorphic to key algorithm defined as:*
   $\mathsf{KHK} : \mathcal{SK} \times \mathcal{PK} \to \mathcal{PK}$ *takes a secret key and a public key as input and outputs a public key, such that for any secret key* $sk_2 \in \mathcal{SK}$ *and public key* $pk_1 \in \mathcal{PK}$*, the following two distributions are statistically close:*

$$\mathsf{KHK}(sk_2, pk_1) \overset{s}{\approx} [sk_1 \otimes sk_2], \tag{6}$$

   where sk_1 *is the secret key of* pk_1*.*
3. *there exists an efficiently commutable key homomorphic to ciphertext algorithm* KHC*, defined as:*
   $\mathsf{KHC} : \mathcal{SK} \times \mathcal{CS} \to \mathcal{CS}$ *takes a secret value and a ciphertext as input and outputs a ciphertext, such that for any keys* $(sk_1, pk_1) \leftarrow \mathsf{KG}$*, secret value* $sk_2 \overset{\$}{\leftarrow} \mathcal{SK}$*, and any message* m*, the following two distributions are statistically close:*

$$(c, \mathsf{KHC}(sk_2, c)) \overset{s}{\approx} (c, \mathsf{Enc}([sk_1 \otimes sk_2], m)), \tag{7}$$

   where $c = \mathsf{Enc}(pk_1, m)$*.*

Remark 1. Equation (4) guarantees that we can compute a public key from a secret key. Equation (5) makes sure the distribution generated from the homomorphism of keys is statistically close to the original key distribution. Equation (6) allows us to compute a new public key from a secret key (assume sk_2) and an old public key (assume $pk_1 = [sk_1]$), the underlying secret key of the newly generated public key matches the output of the homomorphic operation applied to the input secret keys (sk_1 and sk_2). It is essential for our security proof of Theorem 2. We need an algorithm to simulate new public keys from the corresponding old public keys and some secret values. In the proof of Theorem 2, when we use a reduction to simulate the game to an adversary within the firewall, the reduction has no knowledge of any secret keys, but it can simulate public keys with KHK from its own challenge public key and simulated secrets (cf. Fig. 11 in the full version [7]). Without these simulated public keys, the reduction cannot simulate valid ciphertexts within the firewall region. Additionally, we require the two distributions to be statistically close to make sure any adversary cannot distinguish simulated public keys from the real ones. All the PKE considered here

satisfy this property. This statistical property is crucial, since the correctness of our scheme requires Equation (7) to hold statistically. This, in turn, requires the two public keys are statistically indistinguishable, since otherwise Equation (7) cannot hold for an unbounded adversary. Equation (7) enables us to compute a new ciphertext from one public key to another public key without changing the underlying message.

Examples of key homomorphic public key encryption schemes are ElGamal encryption and LWE-based PKE (for some parameter choice) schemes. The security of such PKE schemes are based on DDH and LWE problems, resp..

- ElGamal encryption: Let $\mathbb{G}$ be a cyclic group of order q with generator g. For any secret key $x \in \mathbb{Z}_q$, the corresponding public key is $[x] = g^x$. Given any public value $[y]$ and any secret value x, we can compute $\mathsf{KHK}(x, [y]) = [y]^x = g^{xy} = [x \otimes y]$. Here, $\otimes$ is multiplication of integers and function KHK is computed as $\mathsf{KHK}(x, y) = y^x$. For any ciphertext $c = ([rx], [r] \cdot m)$, any $(y, [y]) \xleftarrow{\$} \mathsf{KG}$ and a random value $r' \xleftarrow{\$} \mathbb{Z}_q^*$, note that $[xy] = \mathsf{KHK}(y, [x]) = [x]^y$, then $\mathsf{KHC}(y, c) = (c_1^y \cdot [xy]^{r'}, [r'] \cdot c_2) = ([(r + r')xy], [r + r'] \cdot m)$ is a ciphertext encrypted under public key $[xy]$ with the same underlying message m of the original ciphertext c.
- Lattice based PKE: Denote $[s] = as + e$, then we can compute $\mathsf{KHK}(s, [t]) = [s] + [t] \overset{s}{\approx} [s \otimes t]$, the right side public key has a bigger error. Here, $\otimes$ is group addition. For any ciphertext $c = (ar, [s]r + e' + m)$ and any $(t, [t]) \xleftarrow{\$} \mathsf{KG}$, note that $[s \otimes t] \overset{s}{\approx} [s] + [t]$, then $\mathsf{KHC}(t, c) = (c_1 + ar', c_2 + c_1 t + [s \otimes t]r' + e'') \overset{s}{\approx} (a(r + r'), [s \otimes t](r + r') + (e' + e'') + m)$ is a ciphertext encrypted under public key $[s \otimes t]$.

5.2 Bi-Directional UE from Key Homomorphic PKE

We construct a UE scheme PKEUE, which is constructed from a key homomorphic PKE scheme PKE, the construction is described in Fig. 4. Note that the next key pair $(sk_{\mathsf{e}+1}, pk_{\mathsf{e}+1})$ is statistically close to a real key pair generated from $\mathsf{PKE.KG}(\lambda)$ by Equation (4) and (5). Note that RISE [12] and LWEUE [9] are constructed by this method, they are built from ElGamal encryption and LWE-based PKE.

Correctness. The correctness of PKEUE follows from the correctness of PKE. It is sufficient to prove that the updated ciphertext is a valid ciphertext which keeps the same underlying message as the original ciphertext. Since PKE is key homomorphic (see Definition 5), we have that $c_{\mathsf{e}+1} = \mathsf{PKE.KHC}(\Delta_{\mathsf{e}+1}, c_{\mathsf{e}}) \underset{Equation\ (7)}{\overset{s}{\approx}} \mathsf{PKE.Enc}([sk_{\mathsf{e}} \otimes \Delta_{\mathsf{e}+1}], m) = \mathsf{PKE.Enc}(pk_{\mathsf{e}+1}, m)$.

Next, we prove that the UE scheme constructed above satisfies the weaker variant of $\mathsf{rand\text{-}IND\text{-}UE}$ security, namely the $(\mathsf{bi}, \mathsf{bi})$-$\mathsf{rand\text{-}IND\text{-}UE}$ security.

$\underline{\mathsf{Setup}(\lambda):}$
$(sk_1, pk_1) \leftarrow \mathsf{PKE.KG}(\lambda)$
return (sk_1, pk_1)

$\underline{\mathsf{Next}(sk_e):}$
$(\Delta_{e+1}, [\Delta_{e+1}]) \leftarrow \mathsf{PKE.KG}(\lambda)$
$sk_{e+1} \leftarrow sk_e \otimes \Delta_{e+1}$
$pk_{e+1} \leftarrow [sk_{e+1}]$
return $\Delta_{e+1}, (sk_{e+1}, pk_{e+1})$

$\underline{\mathsf{Enc}(pk_e, m):}$
$c_e \leftarrow \mathsf{PKE.Enc}(pk_e, m)$
return c_e

$\underline{\mathsf{Dec}(sk_e, c_e):}$
$m' \leftarrow \mathsf{PKE.Dec}(sk_e, c_e)$
return m'

$\underline{\mathsf{Upd}(\Delta_{e+1}, c_e):}$
$c_{e+1} \leftarrow \mathsf{PKE.KHC}(\Delta_{e+1}, c_e)$
return c_{e+1}

Fig. 4: $\mathsf{PKEUE} = (\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ is a UE scheme constructed from a key homomorphic PKE PKE.

Theorem 2. *Let* PKEUE *be the updatable encryption scheme described in Fig. 4. For any* (bi, bi)-rand-IND-UE *adversary* $\mathcal{A}$ *against* PKEUE*, there exists an* IND-CPA *adversary* $\mathcal{B}_2$ *against* PKE *such that*

$$\mathbf{Adv}^{(\mathsf{bi},\mathsf{bi})\text{-}\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{PKEUE},\ \mathcal{A}}(\lambda) \leq l^3 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{B}_2} + \mathsf{negl}(\lambda),$$

where l *is the upper bound on the last epoch.*

Proof. In this proof, we use three steps to reach our desired goal. In the first step, 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 an updated ciphertext as the challenge output, to the right of epoch i it gives an encryption of the challenge input message as output. Therefore, we can move the (bi, bi)-randIND-UE-CPA game from left to right across the epoch space. In the second step, we apply the firewall technique [4, 11, 12] (recall the discussion in Section 3.2) so that we can construct a reduction (in step 3) playing the IND-CPA game by simulating the hybrid game to the adversary. Due to space limitations, the detailed security proof is shown in the full version [7].

5.3 Uni-Directional UE from Key and Message Homomorphic PKE

In this section, we construct a UE scheme with backward-leak uni-directional key update, which is called UNIUE. The UNIUE scheme is built from a key and message homomorphic PKE scheme. Recall that key homomorphic PKE is defined in Definition 5. The message homomorphic PKE is defined as the standard homomorphic encryption, we name it message homomorphic to distinguish two types of homomorphism (key homomorphism and message homomorphism) in this paper.

Definition 6 (Message Homomorphic PKE). *We say public key encryption* $\mathsf{PKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *is* message homomorphic *if for any message* $m_1, m_2 \in \mathcal{M}$ *and any public key* pk*,* $\mathsf{Enc}(pk, m_1) \otimes \mathsf{Enc}(pk, m_2) = \mathsf{Enc}(pk, m_1 \oplus m_2)$*, where* $\otimes$ *is an operation over the ciphertext space and* $\oplus$ *is an operation over the message space.*

Notation. For a vector $\mathbf{A} = (a_1, ..., a_n)$, we define $[\mathbf{A}] = ([a_1], ..., [a_n])$. For vectors $\mathbf{A}, \mathbf{B}$ and function f, we define $f(\mathbf{A}, \mathbf{B}) = (f(a_1, b_1),\ ..., f(a_n, b_n))$.

Constructing Uni-Directional UE. We construct a backward-leak uni-directional key update and uni-directional ciphertext update UE scheme UNIUE from a key and message homomorphic PKE scheme. The construction is described in Fig. 5. The idea of this construction is that only the public value $pk_{\mathsf{e}+1,\mathsf{e}+1}$ is included in the update token $\mathbf{\Delta}_{\mathsf{e}+1}$, secret key $sk_{\mathsf{e}+1,\mathsf{e}+1}$ is not included, in other words, no information of this secret key can be revealed from the token. We can deploy this key pair to protect the confidentiality in the new epoch. The detailed construction is shown in Fig. 5. When epoch period turns into the next epoch period, the new epoch key will increase one key element. That is, epoch key $\mathbf{k}_{\mathsf{e}}$ has e pairs of secret key and public key, epoch key $\mathbf{k}_{\mathsf{e}+1}$ has $\mathsf{e}+1$ pairs of secret key and public key. To update a ciphertext, we use the difference of $\mathbf{sk}_{\mathsf{e}}$ and $\mathbf{sk}_{\mathsf{e}+1}$ (except for the last element) to move encryption of random elements from epoch e to epoch $\mathsf{e}+1$. Due to PKE is message homomorphic, we can perform a re-randomization to refresh the underlying random values. Then, we add an encryption of a new random value, which is encrypted under $pk_{\mathsf{e}+1,\mathsf{e}+1}$, into the updated ciphertext. This new random value can be used to hide the message. Note that if we do not include this additional randomness, the above construction is bi-directional.

Correctness. It is sufficient to prove that the updpated ciphertext is a ciphertext in epoch $\mathsf{e}+1$, with underlying message m. We compute the updpated ciphertext as follows. $\mathbf{c}^1 = \mathsf{PKE.KHC}(\mathbf{\Delta}^{sk}_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e},1}) \underset{Equation\ (7)}{\overset{s}{\approx}} \mathsf{PKE.Enc}([\mathbf{sk}_{\mathsf{e}} \otimes \mathbf{\Delta}^{sk}_{\mathsf{e}+1}], \mathbf{R}_{\mathsf{e}})$ and $\mathbf{c}_{\mathsf{e}+1,1} \leftarrow (\mathbf{c}^1, 0) + \mathsf{PKE.Enc}(\mathbf{pk}_{\mathsf{e}+1}, \mathbf{R}) = \mathsf{PKE.Enc}(\mathbf{pk}_{\mathsf{e}+1}, (\mathbf{R}_{\mathsf{e}}, 0) + \mathbf{R})$. Denote $\mathbf{R}_{\mathsf{e}+1} = (r_{\mathsf{e}+1,1}, ..., r_{\mathsf{e}+1,\mathsf{e}+1}) = (\mathbf{R}_{\mathsf{e}}, 0) + \mathbf{R}$, due to the randomness of $\mathbf{R}$ we know $\mathbf{R}_{\mathsf{e}+1}$ is a random vector. Furthermore, $c_{\mathsf{e}+1,2} \leftarrow c_{\mathsf{e},2} \oplus r_1 \oplus \cdots \oplus r_{\mathsf{e}+1} = r_{\mathsf{e}+1,1} \oplus \cdots \oplus r_{\mathsf{e}+1,\mathsf{e}+1} \oplus m$. Therefore, the updated ciphertext $\mathbf{c}_{\mathsf{e}+1}$ is a valid ciphertext in epoch $\mathsf{e}+1$. Note that we consider epoch bounded UE, which implies that the noise in lattice-based constructions will not grow too large.

Backward-Leak Uni-Directional Key Updates. Any new key $\mathbf{sk}_{\mathsf{e}+1}$ has a random key element $sk_{\mathsf{e}+1,\mathsf{e}+1}$ which is independent from the update token $\mathbf{\Delta}_{\mathsf{e}+1}$ and the previous key $\mathbf{sk}_{\mathsf{e}}$, hence, any adversary cannot upgrade keys.

Uni-Directional Ciphertext Updates. If there exists an adversary $\mathcal{A}$ which can infer a valid previous ciphertext $\mathbf{c}_{\mathsf{e}}$ from token $\mathbf{\Delta}_{\mathsf{e}+1}$ and ciphertext $\mathbf{c}_{\mathsf{e}+1}$. Then we claim that $\mathcal{A}$ can use this ability to win the $\mathsf{IND\$\text{-}CPA}$ game for PKE. Initially, $\mathcal{A}$ receives a public key $pk_{\mathsf{e}+1,\mathsf{e}+1}$ from its $\mathsf{IND\$\text{-}CPA}$ challenger. $\mathcal{A}$ generates a secret key $\mathbf{sk}_{\mathsf{e}}$ and a token $\mathbf{\Delta}^{sk}_{\mathsf{e}+1}$ as algorithms of UNIUE specify. $\mathcal{A}$ computes the public key $\mathbf{pk}_{\mathsf{e}+1}$ and token $\mathbf{\Delta}_{\mathsf{e}+1}$ by embedding the public key $pk_{\mathsf{e}+1,\mathsf{e}+1}$. Suppose $\mathcal{A}$ asks for a challenge query with input $r_{\mathsf{e}+1,\mathsf{e}+1}$, it gets the challenge ciphertext $\tilde{c}$. Then $\mathcal{A}$ uses $r_{\mathsf{e}+1,\mathsf{e}+1}$ and $\tilde{c}$ to create a ciphertext in epoch $\mathsf{e}+1$, say $\tilde{\mathbf{c}}_{\mathsf{e}+1}$. $\mathcal{A}$ can move the ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}+1}$ to a ciphertext $\tilde{\mathbf{c}}_{\mathsf{e}}$ by the token $\mathbf{\Delta}_{\mathsf{e}+1}$

$\underline{\mathsf{Setup}(\lambda):}$
$(sk_{1,1}, pk_{1,1}) \leftarrow \mathsf{PKE.KG}(\lambda)$
return $(sk_{1,1}, pk_{1,1})$

$\underline{\mathsf{Next}(\mathbf{sk}_\mathsf{e}):}$
parse $\mathbf{sk}_\mathsf{e} = (sk_{\mathsf{e},1}, ..., sk_{\mathsf{e},\mathsf{e}})$
for $i \in \{1, ..., \mathsf{e}\}$ **do**
$(\Delta_i, [\Delta_i]) \leftarrow \mathsf{PKE.KG}(\lambda)$
$sk_{\mathsf{e}+1,i} \leftarrow sk_{\mathsf{e},i} \otimes \Delta_i$
$pk_{\mathsf{e}+1,i} \leftarrow [sk_{\mathsf{e}+1,i}]$
$(sk_{\mathsf{e}+1,\mathsf{e}+1}, pk_{\mathsf{e}+1,\mathsf{e}+1}) \leftarrow \mathsf{PKE.KG}(\lambda)$
$\mathbf{sk}_{\mathsf{e}+1} \leftarrow (sk_{\mathsf{e}+1,1}, ..., sk_{\mathsf{e}+1,\mathsf{e}+1})$
$\mathbf{pk}_{\mathsf{e}+1} \leftarrow (pk_{\mathsf{e}+1,1}, ..., pk_{\mathsf{e}+1,\mathsf{e}+1})$
$\mathbf{\Delta}^{sk}_{\mathsf{e}+1} \leftarrow (\Delta_1, ..., \Delta_\mathsf{e})$
$\mathbf{\Delta}_{\mathsf{e}+1} \leftarrow (\mathbf{\Delta}^{sk}_{\mathsf{e}+1}, pk_{\mathsf{e}+1,\mathsf{e}+1})$
return $\mathbf{\Delta}_{\mathsf{e}+1}, (\mathbf{sk}_{\mathsf{e}+1}, \mathbf{pk}_{\mathsf{e}+1})$

$\underline{\mathsf{Enc}(\mathbf{pk}_\mathsf{e}, m):}$

$\mathbf{R}_\mathsf{e} \xleftarrow{\$} \mathcal{M}^{\mathsf{e}\times 1}$
parse $\mathbf{R}_\mathsf{e} = (r_{\mathsf{e},1}, ..., r_{\mathsf{e},\mathsf{e}})$
$\mathbf{c}_{\mathsf{e},1} \leftarrow \mathsf{PKE.Enc}(\mathbf{pk}_\mathsf{e}, \mathbf{R}_\mathsf{e})$
$c_{\mathsf{e},2} \leftarrow r_{\mathsf{e},1} \oplus \cdots \oplus r_{\mathsf{e},\mathsf{e}} \oplus m$
return $\mathbf{c}_\mathsf{e} = (\mathbf{c}_{\mathsf{e},1}, c_{\mathsf{e},2})$

$\underline{\mathsf{Dec}(\mathbf{sk}_\mathsf{e}, \mathbf{c}_\mathsf{e}):}$
parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}_{\mathsf{e},1}, c_{\mathsf{e},2})$
$\mathbf{R}_\mathsf{e} \leftarrow \mathsf{PKE.Dec}(\mathbf{sk}_\mathsf{e}, \mathbf{c}_{\mathsf{e},1})$
parse $\mathbf{R}_\mathsf{e} = (r_{\mathsf{e},1}, ..., r_{\mathsf{e},\mathsf{e}})$
$m' \leftarrow c_{\mathsf{e},2} \oplus^{-1} (r_{\mathsf{e},1} \oplus \cdots \oplus r_{\mathsf{e},\mathsf{e}})$
return m'

$\underline{\mathsf{Upd}(\mathbf{\Delta}_{\mathsf{e}+1}, \mathbf{c}_\mathsf{e}):}$
parse $\mathbf{\Delta}_{\mathsf{e}+1} = (\mathbf{\Delta}^{sk}_{\mathsf{e}+1}, pk_{\mathsf{e}+1})$
parse $\mathbf{c}_\mathsf{e} = (\mathbf{c}_{\mathsf{e},1}, c_{\mathsf{e},2})$
$\mathbf{R} \xleftarrow{\$} \mathcal{M}^{(\mathsf{e}+1)\times 1}$
$\mathbf{c}^1 \leftarrow \mathsf{PKE.KHC}(\mathbf{\Delta}^{sk}_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e},1})$
$\mathbf{c}_{\mathsf{e}+1,1} \leftarrow (\mathbf{c}^1, 0) + \mathsf{PKE.Enc}(\mathbf{pk}_{\mathsf{e}+1}, \mathbf{R})$
parse $\mathbf{R} = (r_1, ..., r_{\mathsf{e}+1})$
$c_{\mathsf{e}+1,2} \leftarrow c_{\mathsf{e},2} \oplus r_1 \oplus \cdots \oplus r_{\mathsf{e}+1}$
$\mathbf{c}_{\mathsf{e}+1} \leftarrow (\mathbf{c}_{\mathsf{e}+1,1}, c_{\mathsf{e}+1,2})$
return $\mathbf{c}_{\mathsf{e}+1}$

Fig. 5: $\mathsf{UNIUE} = (\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ is a UE scheme built from a key and message homomorphic PKE scheme PKE. $\oplus$ is an operation on the message space and assume its inverse operation exists.

and then decrypt $\tilde{\mathbf{c}}_\mathsf{e}$ by the secret key $\mathbf{sk}_\mathsf{e}$. Eventually, $\mathcal{A}$ compares the message with the message used when it creates $\tilde{\mathbf{c}}_{\mathsf{e}+1}$. If they are the same then $\mathcal{A}$ guesses it received a real encryption from its $\mathsf{IND\$\text{-}CPA}$ challenger, otherwise, it guesses it received a random ciphertext from its $\mathsf{IND\$\text{-}CPA}$ challenger. The advantage of $\mathcal{A}$ winning the $\mathsf{IND\$\text{-}CPA}$ game is equal to the probability of $\mathcal{A}$ successfully downgrades the ciphertext $\mathbf{c}_{\mathsf{e}+1}$ to a ciphertext $\mathbf{c}_\mathsf{e}$ by the token $\mathbf{\Delta}_{\mathsf{e}+1}$.

Next, we prove that the UE scheme constructed in Fig. 5 satisfies the stronger variant of $\mathsf{rand\text{-}IND\text{-}UE}$ security, namely the $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{rand\text{-}IND\text{-}UE}$ security.

Theorem 3. *Let* UNIUE *be the updatable encryption scheme described in Fig. 5. For any* $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{rand\text{-}IND\text{-}UE}$ *adversary* $\mathcal{A}$ *against* UNIUE*, there exists an* $\mathsf{IND\$\text{-}CPA}$ *adversary* $\mathcal{B}_3$ *against* PKE *such that*

$$\mathbf{Adv}^{(\mathsf{b\text{-}uni},\mathsf{uni})\text{-}\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UNIUE},\ \mathcal{A}}(\lambda) \leq 2l^2 \cdot \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{PKE},\mathcal{B}_3} + \mathsf{negl}(\lambda),$$

where l *is the upper bound on the last epoch.*

Remark 2. The difference between proving a UE scheme is $(\mathsf{b\text{-}uni}, \mathsf{uni})\text{-}\mathsf{rand\text{-}IND\text{-}UE}$ secure and $(\mathsf{bi}, \mathsf{bi})\text{-}\mathsf{rand\text{-}IND\text{-}UE}$ secure are trivial win conditions, and how the reduction runs the simulation.

Consider the backward-leak uni-directional key updates variant of a confidentiality notion, there will exist a the relaxed insulated region (recall the discussion in Section 3.2). Assume $\tilde{\mathsf{e}}$ is the challenge epoch, where key in this epoch should not be corrupted. Hence, there exists an epoch after $\tilde{\mathsf{e}}$, say fwr, such that keys in epoch $\{\tilde{\mathsf{e}}, ..., \mathsf{fwr}\}$ and token in epoch $\mathsf{fwr}+1$ are not corrupted, in addition, tokens in epoch $\{\tilde{\mathsf{e}}+1, ..., \mathsf{fwr}\}$ are corrupted. By Definition 4, we know that epoch region $\{\tilde{\mathsf{e}}, ..., \mathsf{fwr}\}$ is a relaxed insulated region. Tokens and keys before $\tilde{\mathsf{e}}$ can be corrupted, which will not trigger the trivial win condition. Because knowing any token and any key before epoch $\tilde{\mathsf{e}}$ will not break the confidentiality in epoch $\tilde{\mathsf{e}}$, the key element $sk_{\tilde{\mathsf{e}},\tilde{\mathsf{e}}}$ in $\mathbf{sk}_{\tilde{\mathsf{e}}}$ is an independent and random value compared to all previous update tokens and keys, hence, ciphertexts in epoch $\tilde{\mathsf{e}}$ are random looking without the knowledge of the key element $sk_{\tilde{\mathsf{e}},\tilde{\mathsf{e}}}$.

Proof. The proof is similar to the proof in Theorem 2. We construct hybrid games and apply the firewall technique on relaxed insulated regions.

Step 1. In the initial hybrid games, we move challenges from real to random over epochs. We construct a sequence of hybrid games $H_1, ..., H_l$. For $\mathrm{b} \in \{0,1\}$, experiment H_i^{b} is defined as follows, 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 $(\bar{m}, \bar{c})$, in epoch j:

- if $j \leq i$, for $\mathrm{b}=1$ return an updated ciphertext of $\bar{c}$, for $\mathrm{b}=0$ return (an updated ciphertext which is updated from) an encrypted ciphertext of $\bar{m}$.
- if $j > i$, return a random ciphertext.

Thus H_l^1 is $\mathbf{Exp}^{\mathsf{(b\text{-}uni,uni)\text{-}randIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{UNIUE},\,\mathcal{A}}$, i.e. all challenge responses are challenge-equal ciphertexts of $\mathsf{Upd}(\bar{c})$. And H_l^0 is $\mathbf{Exp}^{\mathsf{(b\text{-}uni,uni)\text{-}randIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{UNIUE},\,\mathcal{A}}$, i.e. all challenge responses are challenge-equal ciphertexts of $\mathsf{Enc}(\bar{m})$. Notice that $H_0^0 = H_0^1$, in which all challenge ciphertexts are random ciphertexts. We have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{(b\text{-}uni,uni)\text{-}randIND\text{-}UE\text{-}CPA}}_{\mathsf{UNIUE},\,\mathcal{A}} &= \big|\mathbf{Pr}[H_l^1 = 1] - \mathbf{Pr}[H_l^0 = 1]\big| \\
&\leq \sum_{i=1}^{l} |\mathbf{Pr}[H_i^1 = 1] - \mathbf{Pr}[H_{i-1}^1 = 1]| \\
&\quad + \sum_{i=1}^{l} |\mathbf{Pr}[H_i^0 = 1] - \mathbf{Pr}[H_{i-1}^0 = 1]|.
\end{aligned}$$

Step 2. We define a new game $\mathcal{G}_i$ that is the same as game H_i, except for the game randomly picks a number $\mathsf{fwr} \xleftarrow{\$} \{0, ..., l\}$. If the adversary corrupts a key in the sequence of epochs $(i, ..., \mathsf{fwr})$ or a token in epoch $\mathsf{fwr}+1$, the game aborts. This loss is upper bounded by l.

Then we have $|\mathbf{Pr}[H_i^{\mathrm{b}} = 1] - \mathbf{Pr}[H_{i-1}^{\mathrm{b}} = 1]| \leq l|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]|$.

Step 3. In this step, we prove that $|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]| = \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{PKE},\mathcal{B}_3} + \mathsf{negl}(\lambda)$. Assume $\mathcal{A}_i$ is an adversary attempting to distinguish $\mathcal{G}_i^{\mathrm{b}}$ from $\mathcal{G}_{i-1}^{\mathrm{b}}$. We construct a reduction $\mathcal{B}_3$, detailed in Fig. 6, playing the IND\$-CPA game by simulating the responses to adversary $\mathcal{A}_i$.

Initially, the reduction guesses a number fwr. If $\mathcal{A}_i$ corrupts $\mathbf{k}_i, ..., \mathbf{k}_{\mathsf{fwr}}$, or $\boldsymbol{\Delta}_{\mathsf{fwr}+1}$ the reduction aborts the game.

A summary of the technical simulations are as follows.

- In the setup phrase, $\mathcal{B}_3$ generates all keys and tokens, except for $\mathbf{k}_i, ..., \mathbf{k}_{\mathsf{fwr}}$, $\boldsymbol{\Delta}_{\mathsf{fwr}+1}$, as follows.
  - The key pairs and tokens outside of the relaxed insulated regions are generated as in $\mathcal{G}_i$.
  - The public keys within relaxed firewalls are generated by embedding public key pk to the i-th term of $\mathbf{pk}_i$, where pk is the public key received from its IND\$-CPA game.
- To simulate non-challenge ciphertexts: $\mathcal{B}_3$ uses public keys to simulate encrypted ciphertexts and updated ciphertexts.
- To simulate challenge-equal ciphertexts in an epoch that is:
  - $j < i$: $\mathcal{B}_3$ uses public keys to simulate encryption and updating.
  - $j = i$: $\mathcal{B}_3$ embeds the challenge ciphertext $\tilde{c}$ received from its IND\$-CPA challenger to the challenge-equal ciphertext in epoch i. More precisely, suppose $\mathcal{B}_3$ receives a challenge query $\mathcal{O}.\mathsf{Chall}$ with input $(\bar{m}_0, \bar{\mathbf{c}})$ in challenge epoch $\tilde{\mathsf{e}}$, where the underlying message of $\bar{\mathbf{c}}$ is $\bar{m}_1$. $\mathcal{B}_3$ sends a random value r_i to its IND\$-CPA challenger and obtains $\tilde{c}_\beta$. $\mathcal{B}_3$ embeds $\tilde{c}_\beta$ to the i-th term of the challenge ciphertext and uses r_i and m_{b} to compute the last term of the challenge ciphertext. Afterwards, $\mathcal{B}_3$ returns $\tilde{\mathbf{c}}_i$ to the adversary $\mathcal{A}$. Again, by Equation (7), $\mathcal{B}_3$ perfectly simulate the challenge ciphertexts in game $\mathcal{G}_{i-1+\beta}$ except for a negligible probability $\mathsf{negl}(\lambda)$.
  - $j > i$: $\mathcal{B}_3$ outputs random ciphertext as challenge ciphertext.

Eventually, $\mathcal{B}_3$ receives the output bit from $\mathcal{A}_i$ and if $\mathcal{A}_i$ guesses it is playing $\mathcal{G}_i$ (suppose it represents the guess response of $\mathcal{A}_i$ is 1), then $\mathcal{B}_3$ guesses it received a real encryption and sends 1 to its IND\$-CPA challenger. Otherwise, sends 0 to its IND\$-CPA challenger. We have $|\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1] - \mathbf{Pr}[\mathcal{G}_{i-1}^{\mathrm{b}} = 1]| = \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathsf{PKE}} + \mathsf{negl}(\lambda)$.

6 UE from Bootstrappable PKE

Bootstrappability can be used to refresh ciphertexts without revealing the underlying message, which implies updatable encryption.

Definition 7 (Bootstrappable PKE). *We say a public key encryption* $\mathsf{BPKE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *is* bootstrappable *if it can evaluate its own decryption circuit D. More precisely, there exists a re-encryption algorithm* $\mathsf{Recrypt}$ *that takes a public*

For b $\in \{0,1\}$ $\mathcal{B}_3$ plays
IND$-CPA game by running $\mathcal{A}_i$:
  receive pk
  do Setup
  $\mathrm{b}' \leftarrow \mathcal{A}_i^{oracles}(\lambda)$
  if ABORT occurred **or** $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
   or $i >$ fwr **then**
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
  return b'

Setup(λ)

  $\Delta_1 \leftarrow \perp$; e $\leftarrow 1$; phase, twf $\leftarrow 0$;
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
  fwr $\xleftarrow{\$} \{0, ..., l\}$
  $(sk_{1,1}, pk_{1,1}) \leftarrow$ PKE.KG(λ)
  for $j \in \{2, ..., i\text{-}1\}$ **do**
    $\mathbf{\Delta}_j, (\mathbf{sk}_j, \mathbf{pk}_j) \leftarrow$ UNIUE.Next($\mathbf{sk}_{j\text{-}1}$)

  for $j \in \{i, ...,$ fwr$\}$ **do**
    parse $\mathbf{pk}_{j\text{-}1} = (pk_{j\text{-}1,1}, ..., pk_{j\text{-}1,j\text{-}1})$
    for $t \in \{1, ..., j\text{-}1\}$ **do**
      $(\Delta_t, \cdot) \leftarrow$ PKE.KG(λ)
      $pk_{j,t} \leftarrow$ KHK$(\Delta_t, pk_{j\text{-}1,t})$
      if $j = i$ **then**
        $pk_{i,i} \leftarrow pk$
      else
        $(\cdot, pk_{j,j}) \leftarrow$ PKE.KG(λ)
    $\mathbf{\Delta}_j^{sk} \leftarrow (\Delta_1, ..., \Delta_{j\text{-}1})$
    $\mathbf{pk}_j = (pk_{j,1}, ..., pk_{j,j})$
  for $j =$ fwr$+1$ **do**
    for $t \in \{1, ...,$ fwr$+1\}$ **do**
      $(sk_{\mathsf{fwr}+1,t}, pk_{\mathsf{fwr}+1,t}) \leftarrow$ PKE.KG(λ)
    $\mathbf{sk}_{\mathsf{fwr}+1} = (sk_{\mathsf{fwr}+1,1}, ..., sk_{\mathsf{fwr}+1,\mathsf{fwr}+1})$
    $\mathbf{pk}_{\mathsf{fwr}+1} = (pk_{\mathsf{fwr}+1,1}, ..., pk_{\mathsf{fwr}+1,\mathsf{fwr}+1})$
  for $j \in \{$fwr$+2, ..., l\}$ **do**
    $\mathbf{\Delta}_j, (\mathbf{sk}_j, \mathbf{pk}_j) \leftarrow$ UNIUE.Next($\mathbf{sk}_{j\text{-}1}$)

$\mathcal{O}$.Enc(m) :
  $\mathbf{c}_\mathsf{e} \leftarrow$ UNIUE.Enc($\mathbf{pk}_\mathsf{e}, m$)
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_\mathsf{e}, \mathsf{e}; m)\}$
  return $\mathbf{c}_\mathsf{e}$

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

$\mathcal{O}$.Upd($\mathbf{c}_{\mathsf{e}\text{-}1}$) :
  if $(\cdot, \mathbf{c}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; m) \notin \mathcal{L}$ **then**
    return $\perp$

  $\mathbf{c}_\mathsf{e} \leftarrow$ UNIUE.Enc($\mathbf{pk}_\mathsf{e}, m$)
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathbf{c}_\mathsf{e}, \mathsf{e}; m)\}$
  return $\mathbf{c}_\mathsf{e}$

$\mathcal{O}$.Corr(inp, ê) :
  if (inp $=$ key **and** ê $\in \{i, ...,$ fwr$\}$)
   or (inp $=$ token **and** ê $=$ fwr$+1$) **then**
    ABORT
  else
    do as $\mathcal{O}$.Corr(inp, ê) specifies

$\mathcal{O}$.Chall($\bar{m}_0, \bar{\mathbf{c}}$) :
  if phase $= 1$ **then**
    return $\perp$
  phase $\leftarrow 1$; ẽ $\leftarrow$ e
  if $(\cdot, \bar{\mathbf{c}}, \tilde{\mathsf{e}}\text{-}1; \bar{m}_1) \notin \mathcal{L}$ **then**
    return $\perp$
  for $j \in \{1, ..., i\text{-}1\}$ **do**
    $\tilde{\mathbf{c}}_j \leftarrow$ UNIUE.Enc($\mathbf{pk}_j, m_\mathrm{b}$)
  for $j = i$ **do**
    $r_i \xleftarrow{\$} \mathcal{M}$
    Send r_i to the IND$-CPA challenger,
    get $\tilde{c}_\beta$
    parse $\tilde{\mathbf{pk}}_i = (pk_{i,1}, ..., pk_{i,i-1})^\intercal$
    $(\tilde{\mathbf{c}}'_{i,1}, c_{i,2}) \leftarrow$ UNIUE.Enc($\tilde{\mathbf{pk}}_i, m_\mathrm{b}$)
    $\tilde{\mathbf{c}}_{i,1} \leftarrow (\tilde{\mathbf{c}}'_{i,1}, \tilde{c}_\beta)$
    $\tilde{c}_{i,2} \leftarrow c_{i,2} \oplus r_i$
    $\tilde{\mathbf{c}}_i \leftarrow (\tilde{\mathbf{c}}_{i,1}, \tilde{c}_{i,2})$
  for $j \in \{i\text{+}1, ..., l\}$ **do**
    $\tilde{\mathbf{c}}_j \xleftarrow{\$} \mathcal{CS}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{j\}$
    $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathbf{c}}_j, j)\}$
  return $\tilde{\mathbf{c}}_{\tilde{\mathsf{e}}}$

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

Fig. 6: Reduction $\mathcal{B}_3$ for proof of Theorem 3.

key, the decryption circuit D, an encryption of a secret key and a ciphertext as input and outputs a new ciphertext, such that for any keys $(sk_1, pk_1), (sk_2, pk_2) \leftarrow \mathsf{KG}(\lambda)$ and any message m, the following two distributions are statistically close:

$$(c, \mathsf{Recrypt}(pk_2, D, \mathsf{Enc}(pk_2, sk_1), c)) \overset{s}{\approx} (c, \mathsf{Enc}(pk_2, m)), \tag{8}$$

where $c = \mathsf{Enc}(pk_1, m)$.

Note that bootstrappable PKE is simpler than a FHE scheme. Most FHE scheme requires bootstrappability, while only bootstrappability is not enough for FHE. Gentry [8, Chapter 4] constructed a re-encryption algorithm $\mathsf{Recrypt}$ (see Fig. 7), which allows us to update a ciphertext under pk_1 to a ciphertext under pk_2.

$\mathsf{Recrypt}(pk_2, D, \langle\overline{sk_{1,j}}\rangle, c_1)$:
  $\overline{c_{1,j}} \xleftarrow{\$} \mathsf{BPKE.Enc}(pk_2, c_{1,j})$
  $c_2 \leftarrow \mathsf{BPKE.Evaluate}(pk_2, D, \langle\langle\overline{sk_{1,j}}\rangle, \langle\overline{c_{1,j}}\rangle\rangle)$
  return c_2

Fig. 7: $\mathsf{Recrypt}$ algorithm. For any key pairs $(sk_1, pk_1), (sk_2, pk_2) \leftarrow \mathsf{KG}(\lambda)$. Let $sk_{1,j}$ be the j-th bit of sk_1 and $\overline{sk_{1,j}} = \mathsf{Enc}(pk_2, sk_{1,j})$. For any plaintext $m \in \mathcal{M}$, let $c_1 = \mathsf{Enc}(pk_1, m)$, and $c_{1,j}$ denote the j-th bit of c_1. The output c_2 is an encryption of $\mathsf{Dec}(sk_1, c_1) = m$ under pk_2.

The $\mathsf{Recrypt}$ algorithm can be used to update ciphertext, where the update token is the encryption of the current secret key sk_{e} under the next public key $pk_{\mathsf{e}+1}$. We construct an updatable encryption scheme BPKEUE from BPKE, which is shown in Fig. 8.

$\mathsf{Setup}(\lambda)$:
  $(sk_1, pk_1) \leftarrow \mathsf{BPKE.KG}(\lambda)$
  return (sk_1, pk_1)

$\mathsf{Next}(sk_{\mathsf{e}})$:
  $(sk_{\mathsf{e}+1}, pk_{\mathsf{e}+1}) \leftarrow \mathsf{BPKE.KG}(\lambda)$
  $\Delta_{\mathsf{e}+1} \leftarrow \mathsf{BPKE.Enc}(pk_{\mathsf{e}+1}, sk_{\mathsf{e}})$
  return $\Delta_{\mathsf{e}+1}, (sk_{\mathsf{e}+1}, pk_{\mathsf{e}+1})$

$\mathsf{Enc}(pk_{\mathsf{e}}, m)$:
  $c_{\mathsf{e}} \leftarrow \mathsf{BPKE.Enc}(pk_{\mathsf{e}}, m)$
  return c_{e}

$\mathsf{Dec}(sk_{\mathsf{e}}, c_{\mathsf{e}})$:
  $m' \leftarrow \mathsf{BPKE.Dec}(sk_{\mathsf{e}}, c_{\mathsf{e}})$
  return m'

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, c_{\mathsf{e}})$:
  $c_{\mathsf{e}+1} \leftarrow \mathsf{BPKE.Recrypt}(pk_{\mathsf{e}+1}, D, \Delta_{\mathsf{e}+1}, c_{\mathsf{e}})$
  return $c_{\mathsf{e}+1}$

Fig. 8: $\mathsf{BPKEUE} = (\mathsf{Setup}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Upd})$ is a UE scheme constructed from a bootstrappable PKE scheme BPKE.

Correctness. The correctness of encrypting then decrypting follows the correctness of the underlying PKE scheme. The correctness of encrypting then updating then decrypting is because of the bootstrappability of BPKE scheme, the

re-encrypted ciphertext is a new ciphertext encrypted under the new public key with the same message. Note that we consider epoch bounded UE, which implies that the noise will not grow too large.

Backward-Leak Uni-Directional Key Updates. We can see the earlier key sk_{e} and token $\Delta_{\mathsf{e}+1}$ as a plaintext and the corresponding ciphertext under public key $pk_{\mathsf{e}+1}$. Hence, any adversary can not obtain the secret key $sk_{\mathsf{e}+1}$.

Uni-Directional Ciphertext Updates. If there exists an adversary which can infer a valid previous ciphertext c_{e} from token $\Delta_{\mathsf{e}+1}$ and ciphertext $c_{\mathsf{e}+1}$. Then we claim that the adversary can use this ability to win the IND-CPA game for BPKE. Initially, the adversary receives a public key $pk_{\mathsf{e}+1}$ from its IND-CPA challenger. The adversary generates a secret key sk_{e} and computes the token $\Delta_{\mathsf{e}+1}$ by the knowledge of the public key $pk_{\mathsf{e}+1}$ and the secret key sk_{e}. Then the adversary can move the challenge ciphertext $\tilde{c}_{\mathsf{e}+1}$ (encrypted under $pk_{\mathsf{e}+1}$) to a ciphertext $\tilde{c}_{\mathsf{e}}$ by the token $\Delta_{\mathsf{e}+1}$. Note that $\tilde{c}_{\mathsf{e}+1}$ and $\tilde{c}_{\mathsf{e}}$ have the same underlying message. Therefore, the adversary can decrypt $\tilde{c}_{\mathsf{e}}$ by sk_{e} and then compare the output with the challenge messages to win the IND-CPA game.

Remark 3. Nishimaki [14] observed that if the update token $\Delta_{\mathsf{e}+1}$ is generated by the old secret key sk_{e} and the new public key $pk_{\mathsf{e}+1}$, such UE schemes may have backward-leak uni-directional key updates. The reason is that it will be difficult to break the confidentiality in epoch $\mathsf{e}+1$ with only the knowledge of $pk_{\mathsf{e}+1}$, $\Delta_{\mathsf{e}+1}$ and sk_{e}, no information about $sk_{\mathsf{e}+1}$ is revealed.

We observed that such UE schemes may have uni-directional ciphertext updates as well. The proof idea is similar to the proof of BPKEUE has uni-directional ciphertext updates. We claim that if such UE schemes do not have uni-directional ciphertext updates, then any adversary can break the confidentiality of such UE schemes without the knowledge of any epoch key. If an adversary aims to attack the confidentiality in epoch $\mathsf{e}+1$, it can generate a new secret key in epoch e. Then the adversary computes the token $\Delta_{\mathsf{e}+1}$ by the generated secret key sk_{e} and the public key $pk_{\mathsf{e}+1}$. It can move ciphertexts from epoch $\mathsf{e}+1$ to epoch e by token $\Delta_{\mathsf{e}+1}$ and then decrypt it by sk_{e} to win the confidentiality in epoch $\mathsf{e}+1$.

Next, we prove that the UE scheme constructed in Fig. 8 is secure under the (b-uni, uni)-rand-IND-UE notion.

Theorem 4. *Let* BPKEUE *be the UE scheme described in Fig. 8. For any* (b-uni, uni)-rand-IND-UE *adversary* $\mathcal{A}$ *against* BPKEUE*, there exists an* IND-CPA *adversary* $\mathcal{B}_4$ *against* BPKE *such that*

$$\mathbf{Adv}^{(\mathsf{b\text{-}uni},\mathsf{uni})\text{-}\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{BPKEUE},\ \mathcal{A}}(\lambda) \leq 2l^3 \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BPKE},\mathcal{B}_4} + \mathsf{negl}(l),$$

where l is the upper bound on the last epoch.

Before proving the above theorem, we prove a lemma first. In the IND game, any adversary can only ask for tokens and encryption oracles.

Lemma 5. *Let* BPKEUE *be the UE scheme described in Fig. 8. For any* IND *adversary* $\mathcal{A}$ *against* BPKEUE*, there exists an* IND-CPA *adversary* $\mathcal{B}_5$ *against* BPKE *such that*

$$\mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BPKEUE},\ \mathcal{A}}(\lambda) \leq 2l \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BPKE},\mathcal{B}_5},$$

where l *is the upper bound on the last epoch.*

Proof (of Lemma 5). The proof is similar to the proof of Theorem 4.2.3 in [8]. We use a hybrid games to move tokens from real to random. In hybrid i, the last $l - i$ tokens are random from the real keys. More precisely, for $i \in \{1, ..., l\}$ let $\mathcal{G}_i$ be a game that is identical to the IND game against BPKEUE, except for all $j > i$:

$$(sk'_j, pk'_j) \xleftarrow{\$} \mathsf{KG}(\lambda), \Delta_j \xleftarrow{\$} \mathsf{Enc}(pk_j, sk'_{j-1}).$$

Note that the last $l - i$ tokens are not related to the real keys.

We have that $\mathcal{G}_l$ is the IND game and the advantage of any adversary winning $\mathcal{G}_0$ is upper bounded by $l \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BPKE}}$. Next, we claim that for any $i \in \{1, ..., l\}$, $|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| = \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BPKE}}$.

Suppose $\mathcal{A}$ is an adversary aiming to distinguish $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$. We construct a reduction $\mathcal{B}_5$ playing the IND-CPA game (against BPKE) and simulating the response to $\mathcal{A}$. Initially, $\mathcal{B}_5$ receives a public key pk from its IND-CPA challenger. $\mathcal{B}_5$ generates key pairs as in $\mathcal{G}_i$ except for it embeds pk to pk_i. It generates a random key pair $(sk'_{i-1}, pk'_{i-1}) \xleftarrow{\$} \mathsf{KG}(\lambda)$, sets $(m_0, m_1) = (sk'_{i-1}, sk_{i-1})$ and sends (m_0, m_1) to its challenger. The challenger flips a coin $\beta \xleftarrow{\$} \{0, 1\}$ and returns the encryption of m_β. $\mathcal{B}_5$ sets the received challenge ciphertext as Δ_i. Note that $\mathcal{B}_5$ perfectly simulates public keys and tokens in $\mathcal{G}_{i-1+\beta}$ to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge query on $(\bar{m}_0, \bar{m}_1)$, $\mathcal{B}_5$ flips a coin $\mathrm{b} \xleftarrow{\$} \{0, 1\}$ and sends the encryptions of $\bar{m}_\mathrm{b}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ submit a guess, if $\mathcal{A}$ guesses it is $\mathcal{G}_i$ then $\mathcal{B}_5$ returns 1 to its challenger, otherwise, $\mathcal{B}_5$ returns 0.

Since $\mathcal{B}_5$ perfectly simulate $\mathcal{G}_{i-1+\beta}$ to $\mathcal{A}$. The probability of $\mathcal{A}$ is able to distinguish which game it is playing is equal to $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{BPKE},\mathcal{B}_5}$.

Proof (of Theorem 4). We use firewall technique and construct a sequence of hybrid games to move challenges from left to right over the relaxed insulated regions. Define game $\mathcal{G}_i$ as (b-uni, uni)-randIND-UE-CPA game, except for

- The game randomly choose a number $\mathsf{fwr} \xleftarrow{\$} \{0, ..., l\}$. If fwr is not the i-th right firewall, returns a random bit for b'. This loss is upper bounded by l.
- To the left side of epoch fwr, the game returns a ciphertext with respect to $\bar{c}$, to the right side of epoch fwr returns a encryption of $\bar{m}$.

If fwr is guessed correct, then $\mathcal{G}_0$ is $\mathbf{Exp}^{\text{(b-uni,uni)-randIND-UE-CPA-0}}_{\mathsf{BPKEUE}}$ and $\mathcal{G}_l$ is $\mathbf{Exp}^{\text{(b-uni,uni)-randIND-UE-CPA-1}}_{\mathsf{BPKEUE}}$. So we can bound the (b-uni, uni)-randIND-UE-CPA advantage by the advantage of distinguishing $\mathcal{G}_0$ and $\mathcal{G}_l$.

$$\mathbf{Adv}^{\text{(b-uni,uni)-randIND-UE-CPA}}_{\mathsf{BPKEUE},\ \mathcal{A}}(\lambda) \leq \sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]|,$$

Notice that if $\mathcal{G}_{i-1}$ and $\mathcal{G}_i$ have the same right firewall fwr, then they are the same game, hence, we assume $\mathcal{G}_{i-1}$ and $\mathcal{G}_i$ have different right firewall. Suppose $\mathcal{A}_i$ is an adversary attempting to distinguish $\mathcal{G}_{i-1}$ from $\mathcal{G}_i$. For all queries concerning epochs outside of the i-th relaxed insulated region ($\{i,\ldots,\mathsf{fwr}\}$) the responses will be equal in either game, We construct a reduction $\mathcal{B}_4$ playing the IND game (within the epoch region $\{i,\ldots,\mathsf{fwr}\}$) for BPKEUE and will simulate the responses of queries made by $\mathcal{A}_i$. Initially, the reduction guesses a numbers fwr. If $\mathcal{A}_i$ corrupts $k_i,\ldots,k_{\mathsf{fwr}}$, or $\Delta_{\mathsf{fwr}+1}$ the reduction aborts the game. A summary of the technical simulations are as follows.

- In the setup phrase, $\mathcal{B}_4$ generates all keys and tokens, except for $k_i,\ldots,k_{\mathsf{fwr}}$, $\Delta_{\mathsf{fwr}+1}$, as follows.
  - The key pairs and tokens outside of the relaxed insulated regions are generated as in $\mathcal{G}_i$.
  - The public key within firewalls are generated by embedding public keys $(pk_i,\ldots,pk_{\mathsf{fwr}})$ and tokens $(\Delta_{i+1},\ldots,\Delta_{\mathsf{fwr}})$, which are received from the IND challenger.
- To simulate non-challenge ciphertexts: $\mathcal{B}_4$ uses public keys to simulate encrypted ciphertexts and updated ciphertexts. Due to updated ciphertext is statistically close to the fresh encryption of the same underlying message, the adversary notice this change with negligible probability.
- To simulate challenge-equal ciphertexts in an epoch that is:
  - $j < i$ or $j > \mathsf{fwr}$: $\mathcal{B}_4$ uses public keys to simulate encryption and updating.
  - $j \in \{i,\ldots,\mathsf{fwr}\}$: $\mathcal{B}_4$ sends $(\bar{m}_0,\bar{m}_1)$ to its IND challenger and forwards the response to $\mathcal{A}_i$, where $\bar{m}_1$ is the underlying message of $\bar{c}$.

Eventually, $\mathcal{A}_i$ sends a guess. If $\mathcal{A}_i$ guesses it is playing $\mathcal{G}_{i-1}$ then $\mathcal{B}_4$ guesses 0 to its IND challenger. Otherwise, $\mathcal{B}_4$ sends 1 to its IND challenger. Note that $\mathcal{B}_4$ perfectly simulates $\mathcal{G}_{i-1}$ to $\mathcal{A}_i$ when its challenger encrypts $\bar{m}_0$ and perfectly simulates $\mathcal{G}_i$ to $\mathcal{A}_i$ when its challenger encrypts $\bar{m}_1$.

$$\mathbf{Adv}^{\mathsf{(b\text{-}uni,uni)\text{-}randIND\text{-}UE\text{-}CPA}}_{\mathsf{BPKEUE},\ \mathcal{A}_i}(\lambda) \leq l^2 \cdot \mathbf{Adv}^{\mathsf{IND}}_{\mathsf{BPKEUE},\mathcal{B}_4} + \mathsf{negl}(l),$$

Combing the result of Lemma 5, we have the desired result.

Acknowledgements. We thank the anonymous reviewers of Eurocrypt 2022, Crypto 2022, and PKC 2023 for their useful comments. We also thank Christoph Striecks and Daniel Slamanig for their valuable suggestions to improve the previous version of our paper.

References

1. Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) Proceedings of CRYPTO 2019. LNCS, vol. 11692, pp. 650–679. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_23

2. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) Proceedings of ASIACRYPT 2020. LNCS, vol. 12493, pp. 559–589. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_19
3. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic prfs and their applications. In: Canetti, R., Garay, J.A. (eds.) Proceedings of CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer (2013). https://doi.org/10.1007/978-3-642-40041-4_23
4. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) Proceedings of CRYPTO 2020. LNCS, vol. 12170, pp. 464–493. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_16
5. Chen, L., Li, Y., Tang, Q.: CCA updatable encryption against malicious re-encryption attacks. In: Moriai, S., Wang, H. (eds.) Proceedings of ASIACRYPT 2020. LNCS, vol. 12493, pp. 590–620. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_20
6. Everspaugh, A., Paterson, K.G., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) Proceedings of CRYPTO 2017. LNCS, vol. 10403, pp. 98–129. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_4
7. Galteland, Y.J., Pan, J.: Backward-leak uni-directional updatable encryption from (homomorphic) public key encryption. Cryptology ePrint Archive, Paper 2022/324 (2022), https://eprint.iacr.org/2022/324
8. Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford, CA, USA (2009)
9. Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) Proceedings of ASIACRYPT 2020. LNCS, vol. 12493, pp. 529–558. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_18
10. Jiang, Y.: The direction of updatable encryption does not matter much. Cryptology ePrint Archive, Report 2020/622 (2020), https://ia.cr/2020/622
11. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) Proceedings of EUROCRYPT 2019. LNCS, vol. 11476, pp. 68–99. Springer (2019). https://doi.org/10.1007/978-3-030-17653-2_3
12. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) Proceedings of EUROCRYPT 2018. LNCS, vol. 10822, pp. 685–716. Springer (2018). https://doi.org/10.1007/978-3-319-78372-7_22
13. Miao, P., Patranabis, S., Watson, G.: Unidirectional updatable encryption and proxy re-encryption from ddh or lwe. Cryptology ePrint Archive, Report 2022/311 (2022), https://ia.cr/2022/311
14. Nishimaki, R.: The direction of updatable encryption does matter. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Proceedings of PKC 2022. LNCS, vol. 13178, pp. 194–224. Springer (2022). https://doi.org/10.1007/978-3-030-97131-1_7
15. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603
16. Slamanig, D., Striecks, C.: Puncture 'em all: Stronger updatable encryption with no-directional key updates. IACR Cryptol. ePrint Arch. p. 268 (2021), https://eprint.iacr.org/2021/268