The Direction of Updatable Encryption Does Matter

Ryo Nishimaki[1]

NTT Corporation, Tokyo, Japan
ryo.nishimaki.zk@hco.ntt.co.jp

Abstract. We introduce a new definition for key updates, called backward-leak uni-directional key updates, in updatable encryption (UE). This notion is a variant of uni-directional key updates for UE. We show that existing secure UE schemes in the bi-directional key updates setting are not secure in the backward-leak uni-directional key updates setting. Thus, security in the backward-leak uni-directional key updates setting is *strictly* stronger than security in the bi-directional key updates setting. This result is in sharp contrast to the equivalence theorem by Jiang (Asiacrypt 2020), which says security in the bi-directional key updates setting is equivalent to security in the existing uni-directional key updates setting. We call the existing uni-directional key updates "forward-leak uni-directional" key updates to distinguish two types of uni-directional key updates in this paper.

We also present two UE schemes with the following features.

- The first scheme is post-quantum secure in the backward-leak uni-directional key updates setting under the learning with errors assumption.
- The second scheme is secure in the no-directional key updates setting and based on indistinguishability obfuscation and one-way functions. This result solves the open problem left by Jiang (Asiacrypt 2020).

Keywords. updatable encryption, key update, lattice

1 Introduction

1.1 Background

Updatable Encryption. Updatable encryption (UE) is a variant of secret key encryption (SKE) where we can periodically update a secret key and a ciphertext. More specifically, a secret key k_{e} is generated at each period, called epoch. Here, e denotes an index of an epoch. We can generate a conversion key $\Delta_{\mathsf{e}+1}$ that converts a ciphertext under k_{e} (key at epoch e) to one under $\mathsf{k}_{\mathsf{e}+1}$ (key at epoch $\mathsf{e}+1$). Such a conversion key is called update token and generated from two successive secret keys $\mathsf{k}_{\mathsf{e}}, \mathsf{k}_{\mathsf{e}+1}$. Roughly speaking, UE security guarantees that confidentiality holds even after some old (and even new) keys and tokens are corrupted as long as trivial winning conditions are not triggered. Adversaries trivially win if a target secret key is corrupted or a target ciphertext can be converted into a ciphertext under a corrupted secret key. In this study, we focus on ciphertext-independent updates UE, where we can generate an update token only from two secret keys [LT18, KLR19, BDGJ20, Jia20].[1]

[1] The other variant is ciphertext-dependent updates UE, where we need not only two secret keys but also a part of ciphertext (called header) to generate a token [BLMR13, EPRS17, BEKS20]. Ciphertext-independent updates UE is more efficient.

A serious threat to encryption is key leakage. In that case, no security is guaranteed by standard encryption. Key updating is a standard solution to guarantee security even after key leakage. However, the issue is how to update a ciphertext generated by an old key. A naive solution is decrypting all ciphertexts by the old key and re-encrypt them by a new key. However, it incurs significant efficiency loss. Moreover, if we save encrypted data in outsourced storage such as cloud servers, we need to download all ciphertexts from the server, decrypt and re-encrypt them, and upload them again to keep the new key secret. Update tokens of UE solve this problem since if we provide the server with an update token, it can directly convert old ciphertexts into new ones without the new key.

Confidentiality is the primary concern in UE. Confidentiality of UE has been improved to capture realistic attack models [EPRS17, LT18, KLR19, BDGJ20, CLT20] since after UE was introduced [BLMR13]. In particular, Lehman and Tackmann formalized trivially leaked information from corrupted keys and tokens as *the direction of key updates* [LT18]. Although previous works proposed UE schemes with improved confidentiality, most do not focus on preventing information leakage from corrupted keys and tokens. We will explain the detail of the information leakage below. In this work, we focus on the direction of key updates and try to minimize leaked information from update tokens to improve UE confidentiality.

Direction of key and ciphertext updates. Directions of key updates describe information leakage that UE schemes cannot avoid. If an adversary has $\Delta_{\mathsf{e}+1}$ and k_{e}, it might be able to obtain $\mathsf{k}_{\mathsf{e}+1}$. Most existing UE schemes cannot prevent this attack. In particular, in all existing (ciphertext-independent) UE schemes, we cannot avoid leaking a secret key from both directions [LT18, KLR19, BDGJ20, Jia20]. That is, we can extract $\mathsf{k}_{\mathsf{e}+1}$ (resp. k_{e}) from $\Delta_{\mathsf{e}+1}$ and k_{e} (resp. $\mathsf{k}_{\mathsf{e}+1}$). This setting is defined as *bi-directional* key updates [EPRS17, LT18]. Lehman and Tackmann also defined *uni-directional* key updates, where we can extract $\mathsf{k}_{\mathsf{e}+1}$ from k_{e} and $\Delta_{\mathsf{e}+1}$ (forward direction inference). In other words, this setting means adversaries might not be able to infer k_{e} from $\mathsf{k}_{\mathsf{e}+1}$ and $\Delta_{\mathsf{e}+1}$. Uni-directional key updates are more preferable than bi-directional ones since a token leaks less information. More information leakage triggers more trivial winning conditions in confidentiality games for UE.

At first glance, secure UE with uni-directional key updates is stronger than one with bi-directional key updates. However, Jiang proved that secure UE with bi-directional key updates *is equivalent to* one with uni-directional key updates [Jia20] (we call Jiang's equivalence theorem in this paper). Jiang also presented the first post-quantum UE scheme with bi-directional key updates [Jia20].

A natural question is: Why do we consider only one-way uni-directional key updates? That is, we can consider a variant of uni-directional key updates where we can extract k_{e} from $\mathsf{k}_{\mathsf{e}+1}$ and $\Delta_{\mathsf{e}+1}$ (backward direction inference). To distinguish two versions of uni-directional key updates, we call the existing definition *forward-leak uni-directional* key updates and our new one *backward-leak uni-directional* key updates. The backward-leak uni-directional key updates setting has never been studied in the UE literature, but it seems to be a valid setting. It is natural to think the latest key is the most important since the reason why we update keys is that the current and older keys might be leaked. In the forward-leak setting, we must protect older keys to protect newer keys *even if older ciphertexts are deleted*. This is undesirable. However, in the backward-leak setting, we need to protect only the latest key if older ciphertexts are properly deleted. Therefore, the backward-leak key updates are more suitable for UE than the forward-leak key updates.

A related issue is the direction of ciphertext updates. It describes whether we can convert ciphertext into one in an older epoch (downgrading ciphertext) by using an update token or not. If we can both update and downgrade ciphertexts by using a token, we say a UE scheme provides bi-directional ciphertext updates. If we can update but cannot downgrade ciphertexts by using a token, we say a UE scheme provides uni-directional ciphertext updates. UE with uni-directional ciphertext updates is more desirable since older epoch keys might be leaked, and downgrading ciphertexts leaks more information. However, all existing (ciphertext-independent) UE schemes provide bi-directional ciphertext updates.

Thus, the first main question of this study is as follows.

Q1. *Is UE with backward-leak uni-directional key updates strictly stronger than UE with bi-directional key updates?*

We affirmatively answer the first question in this work. Then, the next natural question is as follows.

Q2. *Can we achieve a (post-quantum) UE scheme with backward-leak uni-directional key updates and uni-directional ciphertext updates?*

We also affirmatively answer the second question.

Another natural question is whether we can prevent adversaries from inferring secret keys from both directions or not. That is, even if adversaries have $\mathsf{k}_{\mathsf{e}+1}$ (resp. k_{e}) and $\Delta_{\mathsf{e}+1}$, they cannot infer k_{e} (resp. $\mathsf{k}_{\mathsf{e}+1}$). Such key updates are called *no-directional* key updates [Jia20]. Jiang left this question as an open problem. Thus, the last question in this work is as follows.

Q3. *Can we achieve a UE scheme with no-directional key updates (and uni-directional ciphertext updates)?*

We solve this open question in this work.

1.2 Our Contribution

The first contribution of our work is a definitional work. We define a new definition of key updates, which we call backward-leak uni-directional key updates. In addition, we prove that UE with backward-leak uni-directional key updates is *strictly stronger* than bi-directional key updates (and forward-leak uni-directional key updates). More specifically, we show that there are UE schemes with bi-directional key updates that are not secure in the *backward-leak* uni-directional key updates setting. This is in sharp contrast to Jiang's equivalence theorem [Jia20] explained above.

The second contribution is that we present two new constructions of UE. The features of our UE schemes are as follows.

– The first scheme is a UE scheme with backward-leak uni-directional key updates and secure under the learning with errors (LWE) assumption, which is known as a post-quantum assumption. This scheme satisfies confidentiality against CPA and ciphertext updates are randomized.
– The second scheme is a UE scheme with no-directional key updates and based on one-way functions (OWFs) and indistinguishability obfuscation (IO). This scheme satisfies confidentiality against CPA and ciphertext updates are randomized.

These are the first UE schemes with stronger key updates. Note that all our schemes provide uni-directional ciphertext updates (i.e., cannot downgrade ciphertext into older epoch ones). The first scheme is implementable since it is directly constructed from lattices. Although the second scheme is a theoretical construction,[2] it solves the open question left by Jiang [Jia20].

Both schemes satisfy r-IND-UE-CPA security, which was defined by Boyd, Davies, Gjøsteen, and Jiang [BDGJ20]. However, we consider the backward-leak uni-directional or no-directional settings. See Sec. 2 for the definitions.

1.3 Related Work

We often use "forward-leak uni-/backward-leak uni-/bi-/no-directional UE" to refer to UE with forward-leak uni-/backward-leak uni-/bi-/no-directional key updates in this paper.

Ciphertext-independent updates UE. Lehman and Tackmann introduce post-compromise security for UE and refine previous security notions. Those are close to the definitions in this paper. They also present an efficient *bi-directional* UE scheme based on the DDH assumption [LT18]. Klooß, Lehmann, and Rupp present a CCA-secure *bi-directional* UE scheme based on the DDH assumption in the ROM and RCCA-secure *bidirectional* UE schemes based on the SXDH assumption [KLR19]. Boyd et al. integrate and refine previous security notions and present CCA-secure *bi-directional* UE schemes with deterministic ciphertext updates based on the DDH assumption in the ideal cipher model [BDGJ20]. Jiang studies relationships among various models for UE and presents a *bi-directional* UE scheme based on the LWE assumption [Jia20]. All these schemes provide bi-directional ciphertext updates (a token enables us to update and downgrade a ciphertext).

[2] Note that Jain, Lin,and Sahai achieve IO from well-founded assumptions, the SXDH, LWE, a variant of LPN, and PRG in NC^0 [JLS21]. See their paper for the detail of the assumptions.

Ciphertext-dependent updates UE. Boneh, Lewi, Montgomery, and Raghunathan introduce the notion of UE in the ciphertext-dependent updates setting and present a *bi-directional* UE scheme based on key homomorphic PRFs [BLMR13]. Everspaugh, Paterson, Ristenpart, and Scott define stronger security notions for UE and present *bi-directional* UE schemes that satisfy those notions [EPRS17]. Chen, Li, and Tang introduce a stronger CCA security notion by considering malicious re-encryption attacks and present *bi-directional* UE schemes that satisfy the stronger CCA security [CLT20]. Boneh, Eskandarian, Kim, and Shih improve security notions by Everspaugh et al. [EPRS17] and present efficient *bi-directional* UE schemes [BEKS20].

UE in constructive cryptography. Levy-dit-Vehel and Roméas study security notions for UE in the constructive cryptography framework and explore the right security notion for UE [LR21]. Fabrega, Maurer, and Mularczyk also study security notions for UE in the constructive cryptography framework, generalize previous definitions, and discover new security-efficiency trade-offs. [FMM21].

Concurrent and independent work. Slamanig and Striecks [SS21] concurrently and independently proposed a UE scheme.[3] Their scheme is a pairing-based no-directional scheme. They define a stronger model for UE, where we can set an expiry epoch $\mathsf{e}_\perp$ to a ciphertext. If we update a ciphertext with expiry epoch $\mathsf{e}_\perp$ by using a token $\Delta_{\mathsf{e}+1}$ such that $\mathsf{e}+1 > \mathsf{e}_\perp$, the updated ciphertext can no longer be decrypted. Due to this stronger model, Jiang's equivalence theorem [Jia20] does not necessarily hold. The scheme provides uni-directional ciphertext updates. The sharp differences between their work and ours are as follows. Let T be the maximum number of epochs.

- Their no-directional scheme is secure *in the expiry model under the SXDH assumption*, and the ciphertext and key size are $O(\log^2 T)$ and $O(\log^2 T)$, respectively. Our no-directional scheme is secure if IO exists, but the ciphertext and key size do *not depend on* T. Our no-directional scheme is not practical since it relies on IO. Our uni-directional scheme is *post-quantum secure with backward-leak* key updates, and the ciphertext and key size do *not depend on* T.

1.4 Technical Overview

In this section, we present a high-level overview of our technique.

Direction of key updates. As we introduce in Sec. 1.1, we can consider two types of uni-directional tokens, forward-leak and backward-leak uni-directional tokens. If we can infer in both directions, we call bi-directional token. In the definitions

[3] Their paper [SS21] appeared on Cryptology ePrint archive right after the initial version of this paper (https://eprint.iacr.org/2021/221/20210311:210911) appeared on Cryptology ePrint archive. The comparison here is based on the latest versions of their and our papers.

of confidentiality for UE, trivial winning conditions of adversaries depend on those token variations.

We show the following adversary against existing bi-directional UE schemes: (1) s/he triggers the trivial winning condition of the forward-leak uni-directional key updates setting. (2) s/he does not trigger the trivial winning condition of the backward-leak uni-directional key updates. (3) s/he trivially breaks confidentiality of the schemes in the backward-leak uni-directional key updates. Therefore, existing bi-directional UE schemes are not secure in the backward-leak uni-directional key updates setting. The best way to understand the separation result is looking at an example described in Sec. 3.3.

In this section, we explain the source of the difference between the two settings. First, we recall that UE needs the power of public key encryption (PKE) such as the DDH assumption. We can find this fact in all existing ciphertext-independent UE schemes [LT18, KLR19, BDGJ20, Jia20]. Alamati, Montgomery, and Patranabis [AMP19] prove that ciphertext-independent UE implies PKE. By this fact, we can assume that an epoch key k_e consists of a secret part sk_e and a public key part pk_e. As an example, in RISE scheme [LT18], $\mathsf{sk}_\mathsf{e} = x_\mathsf{e} \in \mathbb{Z}_p$, $\mathsf{pk}_\mathsf{e} = g^{x_\mathsf{e}} \in \mathbb{G}$, and $\Delta_{\mathsf{e}+1} = x_{\mathsf{e}+1}/x_\mathsf{e}$ where g is a generator of a prime-order group $\mathbb{G}$. It is easy to see the token is a bi-directional token.

The direction of key updates depends on how to generate a token. A simple but crucial observation is that we must use sk_e to generate $\Delta_{\mathsf{e}+1}$. Otherwise, $\Delta_{\mathsf{e}+1}$ does not have the power of decrypting and converting a ciphertext at epoch e. On the other hand, we do not necessarily need $\mathsf{sk}_{\mathsf{e}+1}$ to generate $\Delta_{\mathsf{e}+1}$ since we can generate a ciphertext at epoch $\mathsf{e}+1$ by using $\mathsf{pk}_{\mathsf{e}+1}$.

The relation between the direction types and how to generate a token is as follows. A forward-leak uni-directional token means $\Delta_{\mathsf{e}+1}$ explicitly contains information about $\mathsf{sk}_{\mathsf{e}+1}$. By combining the observation above, $\Delta_{\mathsf{e}+1}$ should contain information about sk_e and $\mathsf{sk}_{\mathsf{e}+1}$ in the forward-leak uni-directional key updates setting. In addition, we can update an older epoch ciphertext into a newer epoch ciphertext and attack the new one if the newer epoch key is revealed. In other words, we can attack older epoch ciphertext even if older epoch keys are not revealed (backward-leak inference is not possible in this setting). The key inference direction could be the same as the ciphertext update direction. By this observation, it is natural that Jiang's equivalence theorem holds.

On the other hand, a backward-leak uni-directional token means $\Delta_{\mathsf{e}+1}$ explicitly contains information about sk_e. It is possible to generate $\Delta_{\mathsf{e}+1}$ from sk_e and $\mathsf{pk}_{\mathsf{e}+1}$ based on the observations so far. Thus, a backward-leak uni-directional token could hide information about $\mathsf{sk}_{\mathsf{e}+1}$ and prevent the forward inference. In addition, this property prevents downgrading a ciphertext into an older epoch ciphertext. Thus, even if an older epoch key is revealed, we cannot necessarily attack the newer epoch ciphertexts since downgrading ciphertext and forward-leak inference are impossible. The key inference direction is opposite to the ciphertext update direction. This property is in sharp contrast to the forward-leak setting. Therefore, triggers of trivial winning conditions are different in these two settings. An intuition behind our separation result is based on those observations.

See Sec. 3.3 for the detail. Those observations are the starting points of our UE scheme in the backward-leak uni-directional key updates setting. See the next paragraph for an overview.

Our backward-leak uni-directional key updates scheme. Roughly speaking, a token $\Delta_{\mathsf{e}+1}$ is a homomorphic encryption of sk_e under a public key $\mathsf{pk}_{\mathsf{e}+1}$ in our backward-leak uni-directional UE scheme. To update a ciphertext $\mathsf{ct}_\mathsf{e} \leftarrow \mathsf{Enc}(\mathsf{pk}_\mathsf{e}, \mu)$ at epoch e, we homomorphically decrypt ct_e by using $\Delta_{\mathsf{e}+1} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{e}+1}, \mathsf{sk}_\mathsf{e})$ and obtain $\mathsf{Enc}(\mathsf{pk}_{\mathsf{e}+1}, \mu)$. It is easy to see that if we have $\Delta_{\mathsf{e}+1}$ and $\mathsf{sk}_{\mathsf{e}+1}$, we can obtain sk_e by decryption. However, it is difficult to infer $\mathsf{sk}_{\mathsf{e}+1}$ from $\Delta_{\mathsf{e}+1}$ and sk_e since $\mathsf{sk}_{\mathsf{e}+1}$ is not used to generate $\Delta_{\mathsf{e}+1}$. By the security of PKE, it is difficult to obtain $\mathsf{sk}_{\mathsf{e}+1}$ from $\mathsf{pk}_{\mathsf{e}+1}$. To achieve confidentiality for UE, we need to re-randomize tokens and updated ciphertext. This is also possible by using the homomorphic property. Although we use the homomorphic property of lattice-based encryption in our construction, we do not need fully homomorphic encryption (FHE). We use the key-switching technique [BV14, BV11] and the noise smudging technique [AJL+12] to directly achieve secure UE from the LWE assumption. This idea is inspired by uni-directional proxy re-encryption schemes based on lattices [Gen09, ABPW13, CCL+14, NX15].

To prove confidentiality, we need to erase information about $\mathsf{sk}_{\mathsf{e}^*}$ where e^* is the target epoch (otherwise, we cannot use confidentiality under $\mathsf{pk}_{\mathsf{e}^*}$). However, secret keys are linked to update tokens. Thus, we need to gradually erase secret keys in update tokens from new ones to old ones. That is, we change $\mathsf{Enc}(\mathsf{pk}_{\mathsf{e}+1}, \mathsf{sk}_\mathsf{e})$ into $\mathsf{Enc}(\mathsf{pk}_{\mathsf{e}+1}, 0^{|\mathsf{sk}_\mathsf{e}|})$. Once this change is done, we can change $\mathsf{Enc}(\mathsf{pk}_\mathsf{e}, \mathsf{sk}_{\mathsf{e}-1})$ into $\mathsf{Enc}(\mathsf{pk}_\mathsf{e}, 0^{|\mathsf{sk}_{\mathsf{e}-1}|})$, and so forth. Note that there exists an epoch e_r where $\Delta_{\mathsf{e}_\mathsf{r}+1}$ is not corrupted such that $\mathsf{e}^* \leq \mathsf{e}_\mathsf{r}$ as long as adversaries do not trigger the trivial winning conditions. We can start the erasing process from e_r since $\mathsf{sk}_{\mathsf{e}_\mathsf{r}}$ is not used anywhere. This proof outline is reminiscent of the proof technique for multi-hop universal proxy re-encryption [DN21].

Our no-directional key updates scheme. A no-directional token leaks information about neither k_e nor $\mathsf{k}_{\mathsf{e}+1}$. To protect k_e and $\mathsf{k}_{\mathsf{e}+1}$, we obfuscate an update circuit. We consider a secret key encryption (SKE) scheme $\mathsf{SKE}.(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and the following circuit R. Two different secret keys $\mathsf{sk}_\mathsf{e}, \mathsf{sk}_{\mathsf{e}+1} \leftarrow \mathsf{SKE}.\mathsf{Gen}(1^\lambda)$ are hard-coded in R. R takes a ciphertext $\mathsf{ct}_\mathsf{e} \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathsf{sk}_\mathsf{e}, \mu)$ as an input, computes $\mu = \mathsf{SKE}.\mathsf{Dec}(\mathsf{sk}_\mathsf{e}, \mathsf{ct}_\mathsf{e})$, and outputs $\mathsf{ct}_{\mathsf{e}+1} \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathsf{sk}_{\mathsf{e}+1}, \mu)$. A token is an obfuscated circuit of $\mathsf{R}[\mathsf{sk}_\mathsf{e}, \mathsf{sk}_{\mathsf{e}+1}]$ (notation $[\mathsf{sk}_\mathsf{e}, \mathsf{sk}_{\mathsf{e}+1}]$ denotes that $(\mathsf{sk}_\mathsf{e}, \mathsf{sk}_{\mathsf{e}+1})$ are hard-coded). This scheme works as a UE scheme. Intuitively, a token does not leak information about hard-coded secret keys due to obfuscation security. However, we do not know how to prove confidentiality of the scheme above.

To prove security, we instantiate the SKE scheme and obfuscation above with puncturable pseudorandom functions (PRFs) and IO [SW21], respectively. That is, a secret key is a PRF key K, and a ciphertext is $(t, y \oplus \mu) \coloneqq (\mathsf{PRG}(r), \mathsf{PRF}(\mathsf{K}, \mathsf{PRG}(r)) \oplus \mu)$ where PRG is a pseudorandom generator (PRG) and $r \leftarrow \{0,1\}^\tau$. We slightly modified the update circuit above so that it takes not only a ciphertext at epoch e but also randomness $r_{\mathsf{e}+1}$ for a ciphertext at the next epoch. That is, we use a circuit $\mathsf{C}_{\mathsf{re}}[\mathsf{K}_{\mathsf{e}}, \mathsf{K}_{\mathsf{e}+1}]((t, c), r_{\mathsf{e}+1})$ that decrypts (t, c) by K_{e} and encrypts the result by $\mathsf{K}_{\mathsf{e}+1}$ and $r_{\mathsf{e}+1}$. By using this particular scheme and the punctured programming technique with IO security [SW21], we can prove confidentiality of our no-directional UE scheme.

The issue is how to simulate update tokens in security proofs. Note that a UE secret key at epoch e is linked only to UE tokens Δ_{e} and $\Delta_{\mathsf{e}+1}$ in the construction above. In our no-directional scheme, to change target ciphertexts into random ones, we use pseudorandomness of a PRF key $\mathsf{K}_{\mathsf{e}^*}$, which is a UE key $\mathsf{k}_{\mathsf{e}^*}$ at epoch e^*. In the security game of pseudorandomness at punctured points, the adversary is given y^* and a punctured key $\mathsf{K}_{\mathsf{e}^*}\{t^*\}$ where t^* is chosen by the adversary and tries to distinguish y^* is $\mathsf{PRF}(\mathsf{K}_{\mathsf{e}^*}, t^*)$ or random. The punctured key enables us to evaluate the PRF at all inputs except the punctured point t^*. By using $\mathsf{K}_{\mathsf{e}^*}\{t^*\}$, we can simulate tokens Δ_{e} and $\Delta_{\mathsf{e}+1}$ for all inputs except (r, y) such that $t^* = \mathsf{PRG}(r)$. The issue is that we cannot evaluate the PRF at t^*. However, we can overcome this issue by the standard exception handling technique since t^* can be randomly chosen by the reduction due to PRG security and $y^* = \mathsf{PRF}(\mathsf{K}_{\mathsf{e}^*}, t^*)$ is given as a target in the pseudorandomness game. We can construct functionally equivalent circuits by using $\mathsf{K}_{\mathsf{e}^*}\{t^*\}$, t^*, y^*, and exceptional handling. The exceptional handling cannot be detected by IO security. Thus, we can simulate update tokens and use pseudorandomness to prove confidentiality.

Organization. In Sec. 2, we review the syntax and security definitions of UE. Sec. 3 defines a new definition of uni-directional key updates (backward-leak uni-directional key updates) and shows that it is strictly stronger than those of bi-directional and forward-leak uni-directional key updates. In Sec. 4, we present our UE scheme with backward-leak uni-directional key updates based on the LWE problem and prove its security. In Sec. 5, we present our UE scheme with no-directional key updates. Due to space limitations, we omit many details in this version. Please see the full version [Nis21] for them.

2 Updatable Encryption

In this section, we briefly review the syntax and definitions of UE.

Syntax.

Definition 2.1. *An updatable encryption scheme* UE *for message space* $\mathcal{M}$ *consists of a tuple of PPT algorithms* $(\mathsf{UE.Setup}, \mathsf{UE.KeyGen}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.TokGen}, \mathsf{UE.Upd})$.

$\mathsf{UE.Setup}(1^\lambda) \to \mathsf{pp}$*: The setup algorithm takes as input the security parameter and outputs a public parameter* pp*. (This algorithm is an option for UE.)*

$\mathsf{UE.KeyGen}(\mathsf{pp}) \to \mathsf{k}_{\mathsf{e}}$*: The key generation algorithm takes as input the public parameter and outputs an epoch key* k_{e}*.*

$\mathsf{UE.Enc}(\mathsf{k}, \mu) \to \mathsf{ct}$*: The encryption algorithm takes as input an epoch key and a plaintext* μ *and outputs a ciphertext* ct*.*

$\mathsf{UE.Dec}(\mathsf{k}, \mathsf{ct}) \rightarrow \mu'$: *The decryption algorithm takes as input an epoch key and a ciphertext and outputs a plaintext* μ' *or* $\perp$.

$\mathsf{UE.TokGen}(\mathsf{k_e}, \mathsf{k_{e+1}}) \rightarrow \Delta_{\mathsf{e}+1}$: *The token generation algorithm takes as input two keys of successive epochs* e *and* $\mathsf{e}+1$ *and outputs a token* $\Delta_{\mathsf{e}+1}$.

$\mathsf{UE.Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct_e}) \rightarrow \mathsf{ct_{e+1}}$: *The update algorithm takes as input a token* $\Delta_{\mathsf{e}+1}$ *and a ciphertext* $\mathsf{ct_e}$ *and outputs a ciphertext* $\mathsf{ct_{e+1}}$.

Let T *be the maximum number of the epoch.*

Security experiments. We review security definitions for UE in this section.

Definition 2.2 (Correctness). *For any* $\mu \in \mathcal{M}$, *for* $0 \leq \mathsf{e}_1 \leq \mathsf{e}_2 \leq T$, *it holds that*

$$\Pr[\mathsf{UE.Dec}(\mathsf{k}_{\mathsf{e}_2}, \mathsf{ct}_{\mathsf{e}_2}) \neq \mu] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{pp} \leftarrow \mathsf{UE.Setup}(1^\lambda)$, $\mathsf{k}_{\mathsf{e}_1}, \ldots, \mathsf{k}_{\mathsf{e}_2} \leftarrow \mathsf{UE.KeyGen}(\mathsf{pp})$, $\mathsf{ct}_{\mathsf{e}_1} \leftarrow \mathsf{UE.Enc}(\mathsf{k}_{\mathsf{e}_1}, \mu)$, *and* $\Delta_{i+1} \leftarrow \mathsf{UE.TokGen}(\mathsf{k}_i, \mathsf{k}_{i+1})$, $\mathsf{ct}_{i+1} \leftarrow \mathsf{UE.Upd}(\Delta_{i+1}, \mathsf{ct}_i)$ *for* $i \in [\mathsf{e}_1, \mathsf{e}_2 - 1]$.

Definition 2.3 (Confidentiality for Updatable Encryption [BDGJ20, Jia20]). *For* $\mathsf{x} \in \{\mathsf{d}, \mathsf{r}\}$, $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca}\}$, *the game* $\mathsf{Exp}^{\mathsf{x\text{-}ind\text{-}ue\text{-}atk}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *is formalized as follows.*

- *Invoke* Setup *and set* $\mathsf{phase} := 0$.
- *Let* $\mathcal{O} := \mathcal{O}.\{\mathsf{Enc}, \mathsf{Next}, \mathsf{Upd}, \mathsf{Corr}, \mathsf{Chall}, \mathsf{Upd}\widetilde{\mathsf{C}}\}$ *if* $\mathsf{atk} = \mathsf{cpa}$. *If* $\mathsf{atk} = \mathsf{cca}$, $\mathcal{O}.\mathsf{Dec}$ *is also added in* $\mathcal{O}$.
- *Run* $\mathsf{coin}' \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda)$.
- *If* $((\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset) \vee (\mathsf{x} = \mathsf{d} \wedge (\mathsf{e}^* \in \mathcal{T}^* \vee \mathcal{O}.\mathsf{Upd}(\overline{\mathsf{ct}})$ *is invoked*))) *then* $\mathsf{twf} := 1$
- *If* $\mathsf{twf} = 1$ *then* $\mathsf{coin}' \leftarrow \{0, 1\}$
- *return* coin'

We say a UE scheme is x*-IND-UE-*atk *secure if it holds*

$$\mathsf{Adv}^{\mathsf{x\text{-}ind\text{-}ue\text{-}atk}}_{\Sigma,\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{x\text{-}ind\text{-}ue\text{-}atk}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{Exp}^{\mathsf{x\text{-}ind\text{-}ue\text{-}atk}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda).$$

The definitions of oracles are described in Fig. 1.

The prefix d and r in the definition above indicate that we consider UE schemes with *deterministic* and *randomized* update algorithms, respectively.

Leakage sets. We introduce leakage sets. Adversaries can obtain secret keys, update tokens, challenge-equal ciphertexts from oracles. We record epochs in the following sets to maintain which epoch key/token/challenge-equal-ciphertext was given to adversaries.

- $\mathcal{K}$: Set of epochs where $\mathcal{A}$ corrupted the epoch key via $\mathcal{O}.\mathsf{Corr}$.
- $\mathcal{T}$: Set of epochs where $\mathcal{A}$ corrupted the update token via $\mathcal{O}.\mathsf{Corr}$.
- $\mathcal{C}$: Set of epochs where $\mathcal{A}$ obtained a challenge-equal ciphertext via $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$.

$\underline{\mathsf{Setup}(1^\lambda)}$:

- $\mathsf{k}_0 \leftarrow \mathsf{UE.KeyGen}(1^\lambda)$
- $\Delta_0 := \perp; \mathsf{e}, \mathsf{cnt}, \mathsf{twf} := 0$
- $\mathcal{L}, \widetilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} := \emptyset$

$\underline{\mathcal{O}.\mathsf{Enc}(\mu)}$:

- $\mathsf{cnt} := \mathsf{cnt} + 1$
- $\mathsf{ct} \leftarrow \mathsf{UE.Enc}(\mathsf{k}_\mathsf{e}, \mu)$
- $\mathcal{L} := \mathcal{L} \cup \{(\mathsf{cnt}, \mathsf{ct}, \mathsf{e}; \mu)\}$
- **return** ct

$\underline{\mathcal{O}.\mathsf{Dec}(\mathsf{ct})}$:

- $\mu'/\perp \leftarrow \mathsf{UE.Dec}(\mathsf{k}_\mathsf{e}, \mathsf{ct})$
- **if** $\Big((\mathsf{x} = \mathsf{d} \land (\mathsf{ct}, \mathsf{e}) \in \widetilde{\mathcal{L}}^*)$
  $\lor (\mathsf{x} = \mathsf{r} \land (\mu', \mathsf{e}) \in \widetilde{\mathcal{Q}}^*)\Big)$
  then $\mathsf{twf} := 1$
- **return** μ' **or** $\perp$

$\underline{\mathcal{O}.\mathsf{Next}()}$:

- $\mathsf{e} := \mathsf{e} + 1$
- $\mathsf{k}_\mathsf{e} \leftarrow \mathsf{UE.KeyGen}(1^\lambda)$
- $\Delta_\mathsf{e} \leftarrow \mathsf{UE.TokGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e})$
- **if** $\mathsf{phase} = 1$
  then $\mathsf{ct}^*_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathsf{ct}^*_{\mathsf{e}-1})$

$\underline{\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_{\mathsf{e}-1})}$:

- **if** $(j, \mathsf{ct}_{\mathsf{e}-1}, \mathsf{e} - 1; \mu) \notin \mathcal{L}$
  then return $\perp$
- $\mathsf{ct}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathsf{ct}_{\mathsf{e}-1})$
- $\mathcal{L} := \mathcal{L} \cup \{(\mathsf{cnt}, \mathsf{ct}_\mathsf{e}, \mathsf{e}; \mu)\}$
- **return** ct_e

$\underline{\mathcal{O}.\mathsf{Corr}(\mathsf{mode}, \widehat{\mathsf{e}})}$:

- **if** $\widehat{\mathsf{e}} > \mathsf{e}$ **then return** $\perp$
- **if** $\mathsf{mode} = \mathsf{key}$
  then $\mathcal{K} := \mathcal{K} \cup \{\widehat{\mathsf{e}}\}$
  return $\mathsf{k}_{\widehat{\mathsf{e}}}$
- **if** $\mathsf{mode} = \mathsf{token}$
  then $\mathcal{T} := \mathcal{T} \cup \{\widehat{\mathsf{e}}\}$
  return $\Delta_{\widehat{\mathsf{e}}}$

$\underline{\mathcal{O}.\mathsf{Chall}(\overline{\mu}, \overline{\mathsf{ct}})}$:

- **if** $\mathsf{phase} = 1$ **then return** $\perp$
- $\mathsf{phase} := 1; \mathsf{e}^* := \mathsf{e}$
- **if** $(\cdot, \overline{\mathsf{ct}}, \mathsf{e}^* - 1; \overline{\mu}_1) \notin \mathcal{L}$
  then return $\perp$
- **if** $b = 0$
  then $\mathsf{ct}^*_{\mathsf{e}^*} \leftarrow \mathsf{UE.Enc}(\mathsf{k}_{\mathsf{e}^*}, \overline{\mu})$
  else $\mathsf{ct}^*_{\mathsf{e}^*} \leftarrow \mathsf{UE.Upd}(\Delta_{\mathsf{e}^*}, \overline{\mathsf{ct}})$
- $\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}^*\}$
- $\widetilde{\mathcal{L}} := \widetilde{\mathcal{L}} \cup \{(\mathsf{ct}^*_{\mathsf{e}^*}, \mathsf{e}^*)\}$
- **return** $\mathsf{ct}^*_{\mathsf{e}^*}$

$\underline{\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}()}$:

- **if** $\mathsf{phase} \neq 1$ **then return** $\perp$
- $\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}\}$
- $\widetilde{\mathcal{L}} := \widetilde{\mathcal{L}} \cup \{(\mathsf{ct}^*_\mathsf{e}, \mathsf{e})\}$
- **return** ct^*_e

$\underline{\mathcal{O}.\mathsf{Try}(\mathsf{ct}^*)}$:

- $\mu'/\perp \leftarrow \mathsf{UE.Dec}(\mathsf{k}_\mathsf{e}, \mathsf{ct}^*)$
- **if** $(\mathsf{e} \in \mathcal{K}^* \lor (\mathsf{atk} = \mathsf{ctxt} \land (\mathsf{ct}^*, \mathsf{e}) \in \mathcal{L}^*)$
  $\lor (\mathsf{atk} = \mathsf{ptxt} \land (\mu', \mathsf{e}) \in \mathcal{Q}^*))$
  then $\mathsf{twf} := 1$
- **if** $\mu' \neq \perp$ **then** $\mathsf{win} := 1$

Fig. 1: The behavior of oracles in security experiments for updatable encryption. Leakages sets $\mathcal{L}, \widetilde{\mathcal{L}}, \mathcal{L}^*, \widetilde{\mathcal{L}}^*, \mathcal{C}, \mathcal{K}, \mathcal{K}^*, \mathcal{T}, \mathcal{T}^*, \mathcal{Q}, \mathcal{Q}^*, \widetilde{\mathcal{Q}}^*$ are defined in Sec. 2.

We also record ciphertexts given via oracles to maintain which (updated) ciphertexts adversaries obtained.

- $\mathcal{L}$: Set of non-challenge ciphertexts $(\mathsf{cnt}, \mathsf{ct}, \mathsf{e}; \mu)$ returned via $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$, where cnt is a query index incremented by each invocation of $\mathcal{O}.\mathsf{Enc}$, ct is the given ciphertext, e is the epoch where the query happens, and μ is the queried plaintext or the plaintext in the queried ciphertext.

- $\widetilde{\mathcal{L}}$: Set of challenge-equal ciphertexts $(\mathsf{ct}^*_\mathsf{e}, \mathsf{e})$ returned via $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$, where ct^*_e is the given challenge-equal ciphertext and e is the epoch where the query happens.

In the deterministic update setting, where algorithm Upd is deterministic, an updated ciphertext is uniquely determined by a token and a ciphertext. Thus, we consider extended ciphertext sets $\mathcal{L}^*$ and $\widetilde{\mathcal{L}}^*$ inferred from $\mathcal{L}$ and $\widetilde{\mathcal{L}}$, respectively, by using $\mathcal{T}$. Regarding $\mathcal{L}^*$, we only need information about the ciphertext and epoch. That is, $\mathcal{L}^*$ consists of sets of a ciphertext and an epoch index.

In the randomized update setting, where algorithm Upd is probabilistic, an update ciphertext is not uniquely determined. Thus, we consider sets of plaintexts of which adversaries have ciphertexts.

- $\mathcal{Q}^*$: Set of plaintexts (μ, e) such that the adversary obtained or could generate a ciphertext of μ at epoch e.
- $\widetilde{\mathcal{Q}}^*$: Set of challenge plaintexts $\{(\overline{\mu}, \mathsf{e}), (\overline{\mu}_1, \mathsf{e})\}$, where $(\overline{\mu}, \overline{\mathsf{ct}})$ is the query to $\mathcal{O}.\mathsf{Chall}$ and $\overline{\mu}_1$ is the plaintext in $\overline{\mathsf{ct}}$. The adversary obtained or could generate a challenge-equal ciphertext of $\overline{\mu}$ or $\overline{\mu}_1$ at epoch e.

Inferred leakage sets. Lehman and Tackmann [LT18] presented the bookkeeping technique to analyze the epoch leakage sets. We maintain leaked information by the technique in security games.

Key leakage. Adversaries can infer some information from leakage sets $\mathcal{K}$ and $\mathcal{T}$. Here, "infer" means that adversaries can trivially extract some secret information from given keys and tokens. For example, in the ElGamal-based UE scheme by Lehman and Tackmann (called RISE) [LT18], a secret key at epoch e is $k_\mathsf{e} \in \mathbb{Z}_p$ where p is a prime and a token is $\Delta_{\mathsf{e}+1} = k_{\mathsf{e}+1}/k_\mathsf{e} \in \mathbb{Z}_p$. Thus, we can easily extract k_e from $\Delta_{\mathsf{e}+1}$ and $k_{\mathsf{e}+1}$ (and vice versa).

Inferred information depends on the direction of key updates. In previous works on UE, there are three types of directions of key updates, called bi/uni/no-directional key updates. Formally, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, we consider the following kk-directional key update setting.

Definition 2.4 (Direction of Key Update). *We define inferred leakage key sets. The sets depend on the setting of key updates.*

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

$$\mathcal{K}^*_\mathsf{uni} := \{\mathsf{e} \in [0, \ell] \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}$$

  where $\mathsf{CorrK}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T})$
- *Bi-directional key updates:*

$$\mathcal{K}^*_\mathsf{bi} := \{\mathsf{e} \in [0, \ell] \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}$$

  where $\mathsf{CorrK}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T})$

Token leakage. If two successive keys are leaked, a token generated from those keys is also inferred.

Definition 2.5 (Inferred Token Sets). *For* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$*,*

$$\mathcal{T}^*_{\mathsf{kk}} \coloneqq \{\mathsf{e} \in [0, \ell] \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{kk}} \wedge \mathsf{e} - 1 \in \mathcal{K}^*_{\mathsf{kk}})\}$$

Challenge-equal ciphertext leakage. We can update ciphertexts by using tokens. That is, we can obtain updated ciphertexts generated from a challenge ciphertext via leaked tokens. To check whether a challenge ciphertext can be converted into a ciphertext under a corrupted key, we maintain challenge-equal ciphertext epochs defined below.

Definition 2.6 (Direction of Ciphertext Update). *We define two types of challenge-equal ciphertext epoch sets. For* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$*,*

– *Uni-directional ciphertext updates:*

$$\mathcal{C}^*_{\mathsf{kk},\mathsf{uni}} \coloneqq \{\mathsf{e} \in [0, \ell] \mid \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}$$

where $\mathsf{ChallEq}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}})$

– *Bi-directional ciphertext updates:*

$$\mathcal{C}^*_{\mathsf{kk},\mathsf{bi}} \coloneqq \{\mathsf{e} \in [0, \ell] \mid \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}$$

where $\mathsf{ChallEq}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*_{\mathsf{kk}}) \vee (\mathsf{ChallEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}^*_{\mathsf{kk}})$

By considering directions of key/ciphertext updates, we can consider variants of security notions for UE [Jia20].

Definition 2.7 ($(\mathsf{kk}, \mathsf{cc})$-variant of confidentiality [Jia20]). *Let* UE *be a UE scheme. Then the* $(\mathsf{kk}, \mathsf{cc})$-$\mathsf{notion}$ *advantage, for* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$*,* $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ *and* $\mathsf{notion} \in \{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{d\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{r\text{-}ind\text{-}ue\text{-}cca}, \mathsf{d\text{-}ind\text{-}ue\text{-}cca}\}$*, of an adversary* $\mathcal{A}$ *against* UE *is defined as*

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

where $\mathsf{Exp}^{(\mathsf{kk},\mathsf{cc})\text{-}\mathsf{notion}}_{\mathsf{UE},\mathcal{A}}(\lambda, b)$ *is the same as the experiment* $\mathsf{Expt}^{\mathsf{notion}}_{\mathsf{UE},\mathcal{A}}(\lambda, b)$ *in Def. 2.3 except for all leakage sets are both in the* kk*-directional key updates and* cc*-directional ciphertext updates.*

Trivial winning condition. Adversaries trivially win the security game if we can convert a challenge ciphertext into a ciphertext under a corrupted key. Thus, we need to define trivial winning conditions.

For all confidentiality games in Def. 2.3, the trivial winning condition $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ is checked since if the condition holds, adversaries can win the game by decrypting a challenge-equal ciphertext by using a corrupted key.

For all confidentiality games for deterministic update UE, the trivial winning condition $\tilde{\mathsf{e}} \in \mathcal{T}^* \vee$ "$\mathcal{O}.\mathsf{Upd}(\overline{\mathsf{ct}})$ is queried" is checked since if the condition holds, adversaries can win the game by checking the challenge ciphertext is equal to an updated ciphertext generated from the token and a queried ciphertext to $\mathcal{O}.\mathsf{Chall}$.

We need to consider other trivial winning conditions in the CCA setting (both for randomized and deterministic updates) and integrity setting. However, we do not consider these settings in this work. We do not explain those conditions. See the paper by Jiang [Jia20] for the detail.

Firewall and insulated region.

Definition 2.8 (Firewall [LT18, KLR19, BDGJ20, Jia20]). *An insulated region with firewalls* fwl *and* fwr *is a consecutive sequence of epochs* $[\mathsf{fwl}, \mathsf{fwr}]$ *for which:*

- *No key in the sequence of epochs* $[\mathsf{fwl}, \mathsf{fwr}]$ *is corrupted. That is, it holds* $[\mathsf{fwl}, \mathsf{fwr}] \cap \mathcal{K} = \emptyset$.
- *The tokens* Δ_{fwl} *and* $\Delta_{\mathsf{fwr}+1}$ *are not corrupted if they exist. That is, it holds* $\mathsf{fwl}, \mathsf{fwr}+1 \notin \mathcal{T}$.
- *All tokens* $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ *are corrupted. That is,* $[\mathsf{fwl}+1, \mathsf{fwr}] \subseteq \mathcal{T}$.

Definition 2.9 (Insulated Region [LT18, KLR19, BDGJ20, Jia20]). *The union of all insulated regions is defined as* $\mathcal{IR} := \bigcup_{[\mathsf{fwl},\mathsf{fwr}] \in \mathcal{FW}} [\mathsf{fwl}, \mathsf{fwr}]$, *where* $\mathcal{FW}$ *is the set of insulated region with firewalls.*

On security definitions. Boyd et al. prove that r-IND-UE-CPA implies both the standard CPA security for UE and unlinkability of updated ciphertext. See their paper [BDGJ20] for the detail.

3 Backward-Leak Uni-Directional Key Update and Relations

3.1 Definition

We introduce a new notion for the direction of key updates in this section. The notion is categorized in uni-directional key updates, but the direction is the opposite of the uni-directional key updates in Def. 2.4.

Definition 3.1 (Uni-Directional Key Update (revisited)). *We define two types of uni-directional key updates. One is the same as that in Def. 2.4. To distinguish two types of uni-directional key updates, we rename the original one in Def. 2.4 to* forward-leak uni-directional *key updates. The definitions of two notions are as follows.*

- *forward-leak uni-directional key updates:* $\mathcal{K}^*_{\mathsf{f\text{-}uni}} := \mathcal{K}^*_{\mathsf{uni}}$.

– <u>*backward-leak uni-directional key updates:*</u>

$$\mathcal{K}^*_{\mathsf{b\text{-}uni}} := \{\mathsf{e} \in [0, \ell] \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}$$

where $\mathsf{CorrK}(\mathsf{e}) = \mathsf{true} \Leftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T})$

By using the definition above, we can consider Def. 2.5 and 2.6 for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$. We illustrate leaked information in the setting of forward/backward-leak uni-directional key updates settings in Fig. 2.

set	$\mathsf{e}-1$	e	$\mathsf{e}+1$
$\mathcal{K}^*_{\mathsf{f\text{-}uni}}$	×	✓	inferred
$\mathcal{T}^*_{\mathsf{f\text{-}uni}}$		✓	✓

set	$\mathsf{e}-1$	e	$\mathsf{e}+1$
$\mathcal{K}^*_{\mathsf{b\text{-}uni}}$	inferred	✓	×
$\mathcal{T}^*_{\mathsf{b\text{-}uni}}$		✓	✓

Fig. 2: Inferred keys in the forward-leak/backward-leak uni-directional key updates settings. Symbol ✓ means the key/token was given via $\mathcal{O}.\mathsf{Corr}$. Symbol × means we cannot trivially obtain the information. The text "inferred" means we can trivially extract the information from given values.

3.2 Observations on Definitions

On the meaningfulness of backward-leak uni-directional key updates. First of all, all ciphertext-independent UE schemes rely on public key encryption power in some sense [LT18, BDGJ20, Jia20].[4] This fact is endorsed by the result by Alamati, Montgomery, and Patranabis [AMP19], which shows any ciphertext-independent UE scheme that is forward and post-compromise secure implies PKE. Thus, we can assume that an epoch key consists of a secret key part sk_e and a public key part pk_e.

To achieve the ciphertext update mechanism of UE, a token $\Delta_{\mathsf{e}+1}$ must include information about sk_e since an update algorithm essentially decrypts a ciphertext at epoch e and generates a ciphertext for epoch $\mathsf{e}+1$. The question is: "Do we really need $\mathsf{sk}_{\mathsf{e}+1}$ for updating a ciphertext from e to $\mathsf{e}+1$?". The answer is no. The point is that we need only the public key part of an epoch key to generate a ciphertext in most existing ciphertext-independent UE schemes. Thus, we might be able to construct an update token by using only sk_e and $\mathsf{pk}_{\mathsf{e}+1}$. More specifically, we might be able to transform a ciphertext for epoch e by using

[4]Everspaugh et al. [EPRS17] presented a ciphertext-independent UE scheme from authenticated encryption (AE). However, they assume an AE scheme is secure against related key attacks. So far, it seems that we need the power of public key encryption (such as DDH) to achieve related key secure AE [HLL16]. In addition, Everspaugh et al. retracted the ciphertext-independent construction in their full version paper (https://eprint.iacr.org/2017/527/20180903:192110).

encryption of sk_e under $\mathsf{pk}_{\mathsf{e}+1}$ and homomorphic properties. This is what we do in Sec. 4. This insight comes from a few constructions of uni-directional proxy re-encryption [Gen09, ABPW13, CCL+14, NX15].

Based on the observations above, we can say the backward-leak uni-directional key updates setting is natural. If a token $\Delta_{\mathsf{e}+1}$ is generated by using $(\mathsf{sk}_\mathsf{e}, \mathsf{pk}_{\mathsf{e}+1})$, it is likely we can infer sk_e from $\Delta_{\mathsf{e}+1}$ and $\mathsf{sk}_{\mathsf{e}+1}$ (our backward-leak uni-directional scheme is an example). However, it might be difficult to extract information about $\mathsf{sk}_{\mathsf{e}+1}$ from sk_e and $\Delta_{\mathsf{e}+1}$ since only $\mathsf{pk}_{\mathsf{e}+1}$ is embedded in $\Delta_{\mathsf{e}+1}$. In fact, it is difficult in our backward-leak uni-directional scheme.

In the forward-leak uni-directional key updates setting, we assume that it is easy to infer $\mathsf{sk}_{\mathsf{e}+1}$ from $\Delta_{\mathsf{e}+1}$ and sk_e. In some sense, this says $\mathsf{sk}_{\mathsf{e}+1}$ is directly embedded in $\Delta_{\mathsf{e}+1}$. We might be able to execute bi-directional key/ciphertext updates if a token enables us to update a ciphertext (in the forward direction). Here, "directly embedded" means that a secret key is not encrypted. In fact, in all existing UE schemes bi-directional (and forward-leak uni-directional) key updates, $\mathsf{sk}_{\mathsf{e}+1}$ is directly embedded in $\Delta_{\mathsf{e}+1}$ [LT18, KLR19, BDGJ20, Jia20]. In addition, generating a token $\Delta_{\mathsf{e}+1}$ from $\mathsf{sk}_{\mathsf{e}+1}$ and pk_e is unnatural since it is unlikely such $\Delta_{\mathsf{e}+1}$ can update a ciphertext under pk_e.

Note that the argument above does not consider obfuscation [BGI+12]. If we can somehow obfuscate secret keys in a token, it could be difficult to infer secret keys in the token even if we use those secret keys to generate the token. This is what we do in Sec. 5 to achieve a no-directional key updates scheme.

As we argue in Sec. 1.1, backward-leak uni-directional key updates are more suitable than forward-leak ones in practice. In fact, we prove that confidentiality in the backward-leak uni-directional key updates setting is *strictly* stronger than that in the forward-leak uni-directional key updates setting.

On meaningful combination with bi/uni-directional ciphertext updates. For ciphertext updates, it is natural to consider only the uni-directional ciphertext updates in Def. 2.6 since updating ciphertext should go forward direction due to the nature of UE. Of course, we can define another uni-directional ciphertext updates (called "backward uni-directional" or "downgrade-only" ciphertext updates), but it is not meaningful.

Jiang considered a setting where key updates are uni-directional (this is forward-leak uni-directional by our definition) and ciphertext updates are bi-directional. This is meaningful only in the forward-leak uni-directional key updates since forward-leak uni-directional and bi-directional key updates are equivalent by Jiang's result. However, it is unnatural to consider bi-directional ciphertext updates with *backward-leak* uni-directional key updates. This is because we show that backward-leak uni-directional key updates are strictly stronger than bi-directional key updates. In addition, it is difficult to use $\Delta_{\mathsf{e}+1}$ to convert a ciphertext under $\mathsf{k}_{\mathsf{e}+1}$ into one under k_e in the backward-leak uni-directional key updates setting. This observation affects a theorem proved by Jiang [Jia20, Theorem 3.2 in the ePrint ver.] (Thm. 3.5 in this paper), which we explain later.

Relaxed firewall. As we observed above, it is natural to consider uni-directional ciphertext updates in the backward uni-directional key updates setting. In this setting, adversaries cannot convert a ciphertext at the challenge epoch into a ciphertext at an older epoch by using tokens. Thus, even if a token Δ_{fwl} at a left firewall fwl is given to adversaries when a challenge epoch is in between fwl and fwr, adversaries cannot obtain a challenge-equal ciphertext at an epoch whose secret key is corrupted. We define this modified firewall notion as relaxed firewall below.

Definition 3.2 (Relaxed Firewall). *A relaxed insulated region with relaxed firewalls* fwl *and* fwr *is a consecutive sequence of epochs* $[\mathsf{fwl}, \mathsf{fwr}]$ *for which:*

- *No key in the sequence of epochs* $[\mathsf{fwl}, \mathsf{fwr}]$ *is corrupted. That is, it holds* $[\mathsf{fwl}, \mathsf{fwr}] \cap \mathcal{K} = \emptyset$*.*
- *The token* $\Delta_{\mathsf{fwr}+1}$ *is not corrupted if they exist. That is, it holds* $\mathsf{fwr} + 1 \notin \mathcal{T}$*.*
- *All tokens* $(\Delta_{\mathsf{fwl}}, \ldots, \Delta_{\mathsf{fwr}})$ *can be corrupted. That is,* $[\mathsf{fwl}, \mathsf{fwr}] \subseteq \mathcal{T}$*.*

The difference from Def. 2.8 is that Δ_{fwl} can be corrupted.

Definition 3.3 (Relaxed Insulated Region). *The union of all relaxed insulated regions is defined as* $\mathsf{r}\mathcal{IR} := \bigcup_{[\mathsf{fwl},\mathsf{fwr}] \in \mathsf{r}\mathcal{FW}} [\mathsf{fwl}, \mathsf{fwr}]$*, where* $\mathsf{r}\mathcal{FW}$ *is the set of relaxed insulated region with relaxed firewalls.*

As we will see in the proof of Thm. 3.4, there exists an epoch such that it is set as the challenge ciphertext epoch (does not trigger the trivial winning condition), but not in a firewall area under Def. 2.8 (the original definition of firewall). In the example in Fig. 3, which will appear later, epoch $\{5\}$ is such an area. Therefore, we introduce the modified notion.

Summary of observations. We summarize possible combinations for token generation and directions of key and ciphertext updates in Table 1. Note that we do not consider using obfuscation in this table. In each field, possible types are written. In the key update column, "forward-leak? or bi?" means that it can be forward-leak, but in this case, it might not be able to update a ciphertext in the forward direction. If it can update, it essentially includes $\mathsf{sk_e}$ and should be bi-directional. In the ciphertext update column, "backward-leak? or bi?" means that it can be backward, but it does not fit the nature of UE, and if it can be forward, it essentially has the power of bi-directional updates. That is, the second-row case could collapse to the first-row case in Table 1 if the second case works as UE (ciphertext updates are in the forward direction). Lastly, "?" means that we do not know whether this type can update a ciphertext or not (or it is unlikely that the type can update a ciphertext).

All previous ciphertext-independent updates UE schemes fall into the first row category. Our scheme in Sec. 4 falls into the third row category. There might be a hope that we can achieve a no-directional UE scheme by using obfuscation-like techniques (but without obfuscation) in the third row case. It is an interesting open question.

Table 1: Possible combinations for token generation from pk or sk and its relationship to possible directions of key updates and ciphertext updates.

use pk or sk	key update type	ct update type
$\mathsf{TokGen}(\mathsf{sk_e}, \mathsf{sk_{e+1}})$	bi	bi
$\mathsf{TokGen}(\mathsf{pk_e}, \mathsf{sk_{e+1}})$	forward-leak? or bi?	backward? or bi?
$\mathsf{TokGen}(\mathsf{sk_e}, \mathsf{pk_{e+1}})$	backward-leak	forward
$\mathsf{TokGen}(\mathsf{pk_e}, \mathsf{pk_{e+1}})$	no	?

3.3 Relationships

We show that bi-directional key updates does not imply backward-leak uni-directional key updates in this section. More precisely, we prove the following

Theorem 3.1. *There exist secure r-IND-UE-CPA UE schemes in the bi-directional key updates setting that are not r-IND-UE-CPA in the backward-leak uni-directional key updates setting.*

On the equivalence between bi-directional and uni-directional key updates. First, we review a simple fact. It is easy to see that the following theorem holds by the definition of confidentiality (Def. 2.3).

Theorem 3.2. *If a UE scheme is r-IND-UE-CPA in the backward-leak uni-directional, forward-leak uni-directional, or no-directional key updates setting, it is also r-IND-UE-CPA secure in the bi-directional key updates setting.*

Next, we review Jiang's equivalence theorem.

Theorem 3.3 ([Jia20, Theorem 2]). *Let* UE *be an UE scheme and* $\mathsf{notion} \in \{\mathsf{d\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{r\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{d\text{-}ind\text{-}ue\text{-}cca}, \mathsf{r\text{-}ind\text{-}ue\text{-}cca}, \mathsf{int\text{-}ctxt}, \mathsf{int\text{-}ptxt}\}$*. For any* $\mathsf{kk}, \mathsf{kk}' \in \{\mathsf{f\text{-}uni}, \mathsf{bi}\}$*,* $\mathsf{cc}, \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{bi}\}$*, and any* $(\mathsf{kk}, \mathsf{cc})$-notion *adversary* $\mathcal{A}$ *against* UE*, there exists a* $(\mathsf{kk}', \mathsf{cc}')$-notions *adversary* $\mathcal{B}$ *against* UE *such that*

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

The key lemma for proving Jiang's theorem (Thm. 3.3) for the confidentiality case is the following.

Lemma 3.1 ([Jia20, Lemma 6]). *For any* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, we have* $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni,uni}} \neq \emptyset \Leftrightarrow \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} \neq \emptyset$.

See Def. 2.6 and 3.1 for the sets in the lemma. Note that this lemma holds for *forward-leak* uni-directional key updates. We show a counterexample to this lemma (for confidentiality) in the case of the *backward-leak* uni-directional key updates setting.

		0		$\{1\}$		2		3		4		5		$\{6$		$7\}$		8
$\mathcal{K}$		✓		×		×		×		✓		×		×		×		✓
$\mathcal{T}$	×		×		×		✓		✓		✓		×		✓		×	
$\mathcal{K}^*_{\mathsf{bi}}$		✓		×		**✔**		**✔**		✓		**✔**		×		×		✓
$\mathcal{T}^*_{\mathsf{bi}}$	×		×		×		✓		✓		✓		×		✓		×	
$\mathcal{K}^*_{\mathsf{f\text{-}uni}}$		✓		×		$\underline{\times}$		$\underline{\times}$		✓		**✔**		×		×		✓
$\mathcal{T}^*_{\mathsf{f\text{-}uni}}$	×		×		×		✓		✓		✓		×		✓		×	
$\mathcal{K}^*_{\mathsf{b\text{-}uni}}$		✓		×		**✔**		**✔**		✓		$\underline{\times}$		×		×		✓
$\mathcal{T}^*_{\mathsf{b\text{-}uni}}$	×		×		×		✓		✓		✓		×		✓		×	

Fig. 3: Example of leakage sets in the setting of bi/forward/backward-leak uni-directional key updates where $\mathcal{K} := \{0,4,8\}$, $\mathcal{T} := \{3,4,5,7\}$, $\mathcal{IR} = \{1,6,7\}$. Here, $\times$ and $\checkmark$ indicates an epoch key or token is not corrupted and corrupted, respectively. The boldface check mark **✔** indicates an epoch key or token is inferred from other corrupted keys/tokens.

Counterexample in backward-leak uni-directional key updates setting. Looking at an example is the best thing to understand relationships. We consider an example of epoch key leakage sets in Fig. 3.

In the example in Fig. 3, the firewall area is $\mathcal{IR} = \{1,6,7\}$. The difference between the bi-directional setting and forward-leak uni-directional setting is the epochs 2 and 3. The difference between the bi-directional setting and backward-leak uni-directional setting is the epoch 5. (Both differences are underlined in Fig. 3.) We investigate each difference in the forward/backward-leak uni-directional settings.

The case of bi/forward-leak uni-directional key updates: First, we consider the bi/forward-leak uni-directional key updates settings. If we set $\mathcal{C} = \{3\}$, it holds $\mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\}$ and $\mathcal{C}^*_{\mathsf{f\text{-}uni},\mathsf{uni}} = \{3,4,5\}$. Thus, $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\} \neq \emptyset$ and $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni},\mathsf{uni}} = \{4,5\} \neq \emptyset$. If we set $\mathcal{C} = \{5\}$, it holds that $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\} \neq \emptyset$ and $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni},\mathsf{uni}} = \{5\} \neq \emptyset$. This is consistent with Lem. 3.1 (Jiang's Lemma 6 [Jia20]). Note that if we set $\mathcal{C} = \{2\}$, we obtain a similar result to $\mathcal{C} = \{3\}$.

The case of bi/backward-leak uni-directional key updates: Next, we consider the bi/backward-leak uni-directional key updates settings. If we set $\mathcal{C} = \{3\}$, it holds $\mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\}$ and $\mathcal{C}^*_{\mathsf{b\text{-}uni},\mathsf{uni}} = \{3,4,5\}$ since Δ_5 is given even though k_5 is not given in the backward-leak uni-directional setting. Thus, it holds $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\} \neq \emptyset$ and $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni},\mathsf{uni}} = \{3,4\} \neq \emptyset$. However, if we set $\mathcal{C} = \{5\}$, the difference between forward/backward directional key updates is clear. Now, $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} = \{2,3,4,5\} \neq \emptyset$, but $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni},\mathsf{uni}} = \emptyset$ since we cannot infer k_5 (the key at epoch 5) due to the definition of backward-leak uni-directional key updates (we cannot go to forward direction even if we are given k_4 and Δ_5.). This means that even if we set $\mathcal{C} = \{5\}$, the trivial winning condition is not triggered in the backward-leak uni-directional setting. However, the trivial winning condition in the bi-directional setting is triggered. Therefore, this is a counterexample to Lem. 3.1 (Jiang's Lemma 6 [Jia20]) when we use the definition of *backward-leak* uni-directional key updates.

By using the example above, we immediately obtain the following theorem.

Theorem 3.4. *The ciphertext-independent UE schemes Lehman and Tackmann [LT18], Boyd et al. [BDGJ20], and Jiang [Jia20] do not satisfy confidentiality in the* backward-leak *uni-directional setting.*

Proof. We use the leakage sets example $\mathcal{K}$ and $\mathcal{T}$ in Fig. 3 and set $\mathcal{C} = \{5\}$. This does not trigger the trivial winning condition in the backward-leak uni-directional setting. However, an adversary can infer k_5 by using k_4 and Δ_5 in the *bi-directional key updates* schemes described in the theorem statement. Thus, the adversary trivially wins the confidentiality game in the backward-leak uni-directional setting since a challenge ciphertext is encrypted under k_5. ∎

By Thm. 3.4 and the results by Lehman and Tackmann [LT18], Boyd et al. [BDGJ20], and Jiang [Jia20], we immediately obtain Thm. 3.1 since they show that their schemes satisfy confidentiality in the bi-directional key updates setting. Therefore, surprisingly (or unsurprisingly), UE with backward-leak uni-directional (and no-directional) key updates is *strictly stronger* than UE with bi-directional key updates by Thms. 3.1 and 3.2.

On equivalence between no/uni/bi-directional key updates in bi-directional ciphertext update setting. We give an observation on the equivalence theorem about no-directional key updates. Jiang also proves the following theorem.

Theorem 3.5 ([Jia20, Theorem 3.2 in the ePrint ver.]). *Let* UE *be an UE scheme and* $\mathsf{notion} \in \{\mathsf{d\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{r\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{d\text{-}ind\text{-}ue\text{-}cca}, \mathsf{r\text{-}ind\text{-}ue\text{-}cca}\}$*. For any* $(\mathsf{no}, \mathsf{bi})\text{-}\mathsf{notion}$ *adversary* $\mathcal{A}$ *against* UE*, there exists a* $(\mathsf{f\text{-}uni}, \mathsf{bi})\text{-}\mathsf{notions}$ *adversary* $\mathcal{B}$ *against* UE *such that*

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

This theorem seems to contradict our conclusion above, which says UE with no-directional key updates is strictly stronger than UE with forward-leak uni-directional key updates. Recall that no-directional key updates is stronger than backward-leak uni-directional key updates. We also note that bi-directional key updates and forward-leak uni-directional key updates are equivalent.

The source of the puzzle above comes from the fact that the theorem holds for *bi-directional ciphertext* updates. The key lemma for proving Jiang's theorem above (Thm. 3.5) is the following.

Lemma 3.2 ([Jia20, Lemma 3.15 in the ePrint ver.]). *For any* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, we have* $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni},\mathsf{bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no},\mathsf{bi}} \neq \emptyset$.

The proof of the lemma above heavily relies on the bi-directional ciphertext update setting. As we argued in Sec. 3.2, it is unnatural to consider bi-directional ciphertext updates with backward-leak uni-directional (and no-directional) key updates. Thus, if we exclude such an unnatural or artificial setting, the equivalence theorem above (Thm. 3.5), which is counterintuitive, does not hold in the case of the backward-leak uni-directional key updates setting.

4 Construction with Backward-Leak Uni-Directional Key Update

In this section, we present a backward-leak uni-directional key update scheme from the LWE assumption.

4.1 Scheme Description and Design Idea

We present a UE scheme with backward-leak uni-directional key updates based on the Regev PKE scheme [Reg09], and denoted by RtR. A proxy re-encryption scheme by Nishimaki and Xagawa [NX15] inspired this construction idea.

The ciphertext update technique is based on the key-switching technique [BV14, BV11, BGV14]. In particular, we use that for multi-bit plaintexts [BGH13]. In the following, we denote a plaintext by $\boldsymbol{\mu} \in \{0,1\}^\ell$ and error distributions by χ and χ_{ns}.

A variant of Regev PKE scheme. We review a variant of Regev PKE scheme [Reg09] in the multi-user settings.

- $\mathsf{Setup}(1^\lambda)$: Choose $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m \times n}$ and output $\mathsf{pp} \coloneqq (\boldsymbol{A}, 1^\lambda, 1^n, 1^m, 1^\ell, q, \chi, \chi_{\mathsf{ns}})$.
- $\mathsf{Reg.Gen}(\mathsf{pp})$: Choose $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n \times \ell}$ and $\boldsymbol{X} \leftarrow \chi^{m \times \ell}$, compute $\boldsymbol{B} \coloneqq \boldsymbol{AS} + \boldsymbol{X} \in \mathbb{Z}_q^{m \times \ell}$, and outputs $\mathsf{pk} = \boldsymbol{B}$ and $\mathsf{sk} = \boldsymbol{S}$.
- $\mathsf{Reg.Enc}(\mathsf{pk}, \boldsymbol{\mu})$: Choose $\boldsymbol{r} \leftarrow \{-1, +1\}^m$ and $\boldsymbol{e}' \leftarrow \chi_{\mathsf{ns}}^\ell$ and output $(\boldsymbol{u}, \boldsymbol{c}) \coloneqq (\boldsymbol{rA}, \boldsymbol{rB} + \boldsymbol{e}' + \lfloor q/2 \rfloor \boldsymbol{\mu})$.
- $\mathsf{Reg.Dec}(\mathsf{sk}, (\boldsymbol{u}, \boldsymbol{c}))$ Compute $\boldsymbol{d} \coloneqq \boldsymbol{c} - \boldsymbol{uS}$ and output $\boldsymbol{\mu} \coloneqq \lfloor (2/q)\boldsymbol{d} \rceil \bmod 2$.

Key-switching technique. We review the key-switching technique in the multi-bit version for our update algorithm. Let $\eta \coloneqq \lceil \lg q \rceil$. We give the definitions of the binary-decomposition algorithm $\mathsf{BD}(\cdot)$ and the powers-of-2 algorithm $\mathsf{P2}(\cdot)$.

- $\mathsf{BD}(\boldsymbol{x} \in \mathbb{Z}_q^n)$: It decomposes $\boldsymbol{x} = \sum_{k=1}^{\eta} 2^{k-1} \boldsymbol{u}_k$, where $\boldsymbol{u}_k \in \{0,1\}^n$, and outputs $(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\eta) \in \{0,1\}^{n\eta}$.
- $\mathsf{P2}(\boldsymbol{s} \in \mathbb{Z}_q^{n \times 1})$: It outputs $[1, 2, \ldots, 2^{\eta-1}]^\top \otimes \boldsymbol{s} = [\boldsymbol{s}; 2\boldsymbol{s}; \ldots; 2^{\eta-1}\boldsymbol{s}] \in \mathbb{Z}_q^{n\eta \times 1}$, where $\otimes$ denotes the standard tensor product. We extend the domain of $\mathsf{P2}$ by setting $\mathsf{P2}([\boldsymbol{s}_1 \ \ldots \ \boldsymbol{s}_\ell] \in \mathbb{Z}_q^{n \times \ell}) = [\mathsf{P2}(\boldsymbol{s}_1) \ \ldots \ \mathsf{P2}(\boldsymbol{s}_\ell)] \in \mathbb{Z}_q^{n\eta \times \ell}$.

By the definition, it holds that $\mathsf{BD}(\boldsymbol{x}) \cdot \mathsf{P2}(\boldsymbol{S}) = \boldsymbol{x} \cdot \boldsymbol{S} \in \mathbb{Z}_q^\ell$ for any $\boldsymbol{x} \in \mathbb{Z}_q^n$ and $\boldsymbol{S} \in \mathbb{Z}_q^{n \times \ell}$.

Let $\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{S}_{\mathsf{e}+1} \in \mathbb{Z}_q^{n\times\ell}$ be two secret keys at epoch $\mathsf{e}, \mathsf{e}+1$, respectively. The key-switching technique enables us to homomorphically decrypt a ciphertext at epoch e and obtain a ciphertext at epoch $\mathsf{e}+1$ by using encryption of $\boldsymbol{S}_{\mathsf{e}}$ under the key at epoch $\mathsf{e}+1$. More formally, the key-switching matrix $\boldsymbol{M}_{\mathsf{e}+1}$ is $[\boldsymbol{A}' \mid \boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{Y}] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})]$, where $\boldsymbol{A}' \leftarrow \mathbb{Z}_q^{n\eta\times n}$, $\boldsymbol{Y} \leftarrow \chi^{n\eta\times\ell}$. To update a ciphertext $(\boldsymbol{u}, \boldsymbol{c})$ under $\boldsymbol{S}_{\mathsf{e}}$ to one under $\boldsymbol{S}_{\mathsf{e}+1}$, we compute $(\boldsymbol{u}', \boldsymbol{c}') = (\boldsymbol{0}, \boldsymbol{c}) + \mathsf{BD}(\boldsymbol{u})\boldsymbol{M}_{\mathsf{e}+1}$. By simple calculation, we have that

$$\begin{aligned}(\boldsymbol{u}', \boldsymbol{c}') &= (\boldsymbol{0}, \boldsymbol{c}) + \mathsf{BD}(\boldsymbol{u})\left([\boldsymbol{A}' \mid \boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{Y}] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})]\right)\\ &= (\mathsf{BD}(\boldsymbol{u})\boldsymbol{A}', \boldsymbol{c} - \boldsymbol{u}\boldsymbol{S}_{\mathsf{e}} + \mathsf{BD}(\boldsymbol{u})\boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1} + \mathsf{BD}(\boldsymbol{u}) \cdot \boldsymbol{Y}).\end{aligned}$$

To decrypt ciphertext by secret key $\boldsymbol{S}_{\mathsf{e}+1}$, we compute

$$\begin{aligned}\boldsymbol{c}' - \boldsymbol{u}'\boldsymbol{S}_{\mathsf{e}+1} &= \boldsymbol{c} - \boldsymbol{u}\boldsymbol{S}_{\mathsf{e}} + \mathsf{BD}(\boldsymbol{u})\boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1} + \mathsf{BD}(\boldsymbol{u}) \cdot \boldsymbol{Y} - \mathsf{BD}(\boldsymbol{u})\boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1}\\ &= \boldsymbol{c} - \boldsymbol{u}\boldsymbol{S}_{\mathsf{e}} + \mathsf{BD}(\boldsymbol{u}) \cdot \boldsymbol{Y}.\end{aligned}$$

Thus, the decryption is correct if the magnitude of additional noises $\mathsf{BD}(\boldsymbol{u}) \cdot \boldsymbol{Y}$ is small.

backward-leak uni-directional update. In fact, we do not need the secret key $\boldsymbol{S}_{\mathsf{e}+1}$ at epoch $\mathsf{e}+1$ for update. We set $\boldsymbol{B}_{\mathsf{e}+1} = \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{Y}_{\mathsf{e}+1}$, which we call the public key part of the key at epoch $\mathsf{e}+1$. We choose $\boldsymbol{R}_{\mathsf{e}+1} \leftarrow \{-1, +1\}^{n\eta\times m}$ and compute an update token

$$\begin{aligned}\boldsymbol{M}_{\mathsf{e}+1} &= \boldsymbol{R}_{\mathsf{e}+1}[\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})]\\ &= [\boldsymbol{A}' \mid \boldsymbol{A}'\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{Y}'] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})],\end{aligned}$$

where $\boldsymbol{A}' = \boldsymbol{R}_{\mathsf{e}+1}\boldsymbol{A}$ and $\boldsymbol{Y}' = \boldsymbol{R}_{\mathsf{e}+1}\boldsymbol{Y}_j$. By using $\boldsymbol{M}_{\mathsf{e}+1}$, we can update ciphertext $(\boldsymbol{u}, \boldsymbol{c})$ at epoch e. Thus, even if given the key $\boldsymbol{S}_{\mathsf{e}}$ at epoch e and the token $\boldsymbol{M}_{\mathsf{e}+1}$, we cannot infer $\boldsymbol{S}_{\mathsf{e}+1}$ since only the public key part $\boldsymbol{B}_{\mathsf{e}+1}$ (this is pseudorandom by the LWE assumption) of the key at epoch $\mathsf{e}+1$ is embedded in $\boldsymbol{M}_{\mathsf{e}+1}$. Note that $\boldsymbol{S}_{\mathsf{e}}$ and $\boldsymbol{S}_{\mathsf{e}+1}$ are independently chosen. However, if given the key $\boldsymbol{S}_{\mathsf{e}+1}$ at epoch $\mathsf{e}+1$ and the token $\boldsymbol{M}_{\mathsf{e}+1}$, we can easily infer $\boldsymbol{S}_{\mathsf{e}}$ since $\boldsymbol{S}_{\mathsf{e}}$ is encrypted under $\boldsymbol{S}_{\mathsf{e}+1}$. Thus, this update mechanism is a backward-leak uni-directional key update and uni-directional ciphertext update.

How to achieve randomized update. The update algorithm above is deterministic. To re-randomize an updated ciphertext, we set the update token as $\boldsymbol{M}_{\mathsf{e}+1}$ and $\boldsymbol{B}_{\mathsf{e}+1}$, which is the public key part at epoch $\mathsf{e}+1$. First, we convert ciphertext $(\boldsymbol{u}, \boldsymbol{c})$ at epoch e into $(\boldsymbol{u}', \boldsymbol{c}')$ using $\boldsymbol{M}_{\mathsf{e}+1}$ as above and masking $(\boldsymbol{u}', \boldsymbol{c}')$ with a new ciphertext $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) \coloneqq \tilde{\boldsymbol{r}}[\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}]$ of the plaintext $\boldsymbol{0}$. This is not enough for confidentiality since it includes information about $\boldsymbol{B}_{\mathsf{e}+1}$ and is not random. To overcome this issue, we randomize $[\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}]$ into $\boldsymbol{N}_{\mathsf{e}+1} = \boldsymbol{R}'_{\mathsf{e}+1} \cdot [\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}]$, where $\boldsymbol{R}'_{\mathsf{e}+1} \leftarrow \{-1, +1\}^{m\times m}$ and add it to $\Delta_{\mathsf{e}+1}$. Since the matrix $\boldsymbol{N}_{\mathsf{e}+1}$ consists of m ciphertexts of the message $\boldsymbol{0}$, this is pseudorandom. The update token consists of key-switching matrix $\boldsymbol{M}_{\mathsf{e}+1}$ and randomized matrix $\boldsymbol{N}_{\mathsf{e}+1}$.

Backward-leak uni-directional key update scheme. A UE scheme, RtR, is defined as follows:

$\mathsf{Setup}(1^\lambda)$**:**

1. Choose $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times n}$.
2. Output $\mathsf{pp} := (\boldsymbol{A}, 1^\lambda, 1^n, 1^m, 1^\ell, q, \chi, \chi_{\mathsf{ns}})$.

$\mathsf{Gen}(\mathsf{pp})$**:**

1. Generate $(\boldsymbol{B}_{\mathsf{e}}, \boldsymbol{S}_{\mathsf{e}}) \leftarrow \mathsf{Reg.Gen}(1^\lambda)$.
2. Output $\mathsf{k}_{\mathsf{e}} := (\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}}) := (\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{B}_{\mathsf{e}})$.

$\mathsf{Enc}(\mathsf{k}_{\mathsf{e}}, \boldsymbol{\mu} \in \{0,1\}^\ell)$**:**

1. Parse $\mathsf{k}_{\mathsf{e}} = (\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{B}_{\mathsf{e}})$.
2. Generate $(\boldsymbol{u}, \boldsymbol{c}) \leftarrow \mathsf{Reg.Enc}(\boldsymbol{B}_{\mathsf{e}}, \boldsymbol{\mu})$.
3. Output $\mathsf{ct} := (\boldsymbol{u}, \boldsymbol{c}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

$\mathsf{Dec}(\mathsf{k}_{\mathsf{e}}, \mathsf{ct})$**:**

1. Parse $\mathsf{k}_{\mathsf{e}} = (\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{B}_{\mathsf{e}})$ $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{c})$.
2. Compute and output $\boldsymbol{\mu} \leftarrow \mathsf{Reg.Dec}(\boldsymbol{S}_{\mathsf{e}}, \mathsf{ct})$.

$\mathsf{TokGen}(\mathsf{k}_{\mathsf{e}}, \mathsf{k}_{\mathsf{e}+1})$**:**

1. Parse $\mathsf{k}_{\mathsf{e}} = (\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{B}_{\mathsf{e}})$ and $\mathsf{k}_{\mathsf{e}+1} = (\boldsymbol{S}_{\mathsf{e}+1}, \boldsymbol{B}_{\mathsf{e}+1})$.
2. Compute $\boldsymbol{M}_{\mathsf{e}+1} := \boldsymbol{R}_{\mathsf{e}+1} \cdot [\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})]$, where $\boldsymbol{R}_{\mathsf{e}+1} \leftarrow \{-1,+1\}^{n\eta\times m}$.
3. Compute $\boldsymbol{N}_{\mathsf{e}+1} := \boldsymbol{R}'_{\mathsf{e}+1} \cdot [\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}]$, where $\boldsymbol{R}'_{\mathsf{e}+1} \leftarrow \{-1,+1\}^{m\times m}$.
4. Output $\Delta_{\mathsf{e}+1} := (\boldsymbol{M}_{\mathsf{e}+1}, \boldsymbol{N}_{\mathsf{e}+1})$.

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct}_{\mathsf{e}})$**:**

1. Parse $\Delta_{\mathsf{e}+1} = (\boldsymbol{M}_{\mathsf{e}+1}, \boldsymbol{N}_{\mathsf{e}+1})$ and $\mathsf{ct}_{\mathsf{e}} = (\boldsymbol{u}_{\mathsf{e}}, \boldsymbol{c}_{\mathsf{e}})$.
2. Compute $(\boldsymbol{u}', \boldsymbol{c}') := \mathsf{BD}(\boldsymbol{u}_{\mathsf{e}})\boldsymbol{M}_{\mathsf{e}+1}$;
3. Compute $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) := \tilde{\boldsymbol{r}} \cdot \boldsymbol{N}_{\mathsf{e}+1}$, where $\tilde{\boldsymbol{r}} \leftarrow \{-1,+1\}^m$;
4. Output $\mathsf{ct}_{\mathsf{e}+1} := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) := (\boldsymbol{u}' + \tilde{\boldsymbol{u}}, \boldsymbol{c}_{\mathsf{e}} + \boldsymbol{c}' + \tilde{\boldsymbol{v}})$.

For notational convenience, we call $\mathsf{pk}_{\mathsf{e}} = \boldsymbol{B}_{\mathsf{e}}$ and $\mathsf{sk}_{\mathsf{e}} = \boldsymbol{S}_{\mathsf{e}}$ public key and secret key of epoch e, respectively. Note that we can run Enc without $\mathsf{sk}_{\mathsf{e}} = \boldsymbol{S}_{\mathsf{e}}$ (we need only $\mathsf{pk}_{\mathsf{e}} = \boldsymbol{B}_{\mathsf{e}}$). We also note that we can run $\mathsf{TokGen}(\mathsf{k}_{\mathsf{e}}, \mathsf{k}_{\mathsf{e}+1})$ without $\mathsf{sk}_{\mathsf{e}+1}$ (we need only $\mathsf{pk}_{\mathsf{e}+1}$ and sk_{e}).

The scheme is correct and r-IND-UE-CPA secure. We prove the following theorems in Sections 4.2 and 4.3. Let T be the maximum number of the epoch.

Theorem 4.1. *Let χ and χ_{ns} be B-bounded and B'-bounded distributions, respectively, such that $B/B' = \mathsf{negl}(\lambda)$ and $m = 2n\lg q + \omega(\sqrt{\lg\lambda})$. Suppose that $(1 + n\eta + m)mB + B' \leq q/4T$. Then RtR is correct.*

Theorem 4.2. *Suppose that $m \geq (n+\ell)\lg q + \omega(\lg\lambda)$. Under the $\mathrm{LWE}(n,q,\chi)$ assumption, RtR is r-IND-UE-CPA secure in the backward-leak uni-directional setting. That is, $\mathsf{Adv}^{\mathsf{(b\text{-}uni,uni)\text{-}r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda)$.*

4.2 Correctness

We give rough estimations on B-bounded and B'-bounded distributions χ and χ_{ns}, respectively, for simplicity. However, if we set $\chi = \bar{\Psi}_\alpha$ or $D_{\mathbb{Z},s}$, we can obtain tighter bounds.

Proof of Thm. 4.1. The theorem follows from Prop. 4.1 and 4.2 below. ■

Proposition 4.1. *The scheme is correct for the encryption algorithm if $mB + B' < q/4$.*

Proposition 4.2. *The scheme is correct for the update algorithm if $(1 + n\eta + m)mB + B' < q/4T$.*

Those correctness easily follows from the proof by Regev [Reg09]. We omit them due to space limitations. See the full version for the proofs.

4.3 Confidentiality

We show RtR is r-IND-UE-CPA in the backward-leak uni-directional setting. Although it is trivial that RtR satisfies uni-directional ciphertext updates from its security, we confirm it below.

Lemma 4.1. *If* $(\mathsf{Setup}, \mathsf{Reg.Gen}, \mathsf{Reg.Enc}, \mathsf{Reg.Dec})$ *is IND-CPA secure PKE, adversaries cannot convert a ciphertext under a public key $\mathsf{pk}_{\mathsf{e}+1}$ into one under a public key pk_{e} even if they are given $\Delta_{\mathsf{e}+1}$.*

Proof. We construct an algorithm $\mathcal{B}$ that breaks IND-CPA security under $\mathsf{pk}_{\mathsf{e}+1}$ by using an adversary $\mathcal{D}$ that converts a ciphertext under $\mathsf{pk}_{\mathsf{e}+1}$ into one under pk_{e} by using $(\mathsf{pk}_{\mathsf{e}}, \mathsf{sk}_{\mathsf{e}})$, $\mathsf{pk}_{\mathsf{e}+1}$, and $\Delta_{\mathsf{e}+1}$.

First, $\mathcal{B}$ is given $\mathsf{pk}_{\mathsf{e}+1}$. $\mathcal{B}$ generates $(\mathsf{pk}_{\mathsf{e}}, \mathsf{sk}_{\mathsf{e}})$ and $\Delta_{\mathsf{e}+1} \leftarrow \mathsf{TokGen}(\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}+1})$, selects any (m_0, m_1), sends (m_0, m_1) to its challenger, and receives a target ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Reg.Enc}(\mathsf{pk}_{\mathsf{e}+1}, m_b)$ where $b \leftarrow \{0,1\}$. Next, $\mathcal{B}$ sends $((\mathsf{pk}_{\mathsf{e}}, \mathsf{sk}_{\mathsf{e}}), \Delta_{\mathsf{e}+1}, \mathsf{ct}^*)$ to $\mathcal{D}$. $\mathcal{D}$ outputs a ciphertext ct' under pk_{e}. Then, $\mathcal{B}$ computes $m' \leftarrow \mathsf{Reg.Dec}(\mathsf{sk}_{\mathsf{e}}, \mathsf{ct}')$ by using sk_{e} and if $m' = m_{b'}$, it outputs b'.

It is easy to see that if $\mathcal{D}$ can convert ct^* into a ciphertext under pk_{e}, $\mathcal{B}$ outputs $b' = b$. This completes the proof. ■

Second, we look at the detail of the update procedure again. By simple calculation, we obtain

$$\begin{aligned}(\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) &= (\boldsymbol{0}, \boldsymbol{c}_{\mathsf{e}}) + \mathsf{BD}(\boldsymbol{u}_{\mathsf{e}}) \cdot \boldsymbol{M}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}} \cdot \boldsymbol{N}_{\mathsf{e}+1} \\ &= (\boldsymbol{r}^\dagger \boldsymbol{A}, \boldsymbol{r}^\dagger \boldsymbol{B}_{\mathsf{e}+1} + \boldsymbol{e}'_{\mathsf{e}} + \boldsymbol{r}\boldsymbol{X}_{\mathsf{e}} + \lfloor q/2 \rfloor \boldsymbol{\mu}) \text{ where } \boldsymbol{r}^\dagger := \mathsf{BD}(\boldsymbol{u}_{\mathsf{e}})\boldsymbol{R}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1} \\ &\overset{\mathsf{s}}{\approx} (\boldsymbol{r}^\dagger \boldsymbol{A}, \boldsymbol{r}^\dagger \boldsymbol{B}_{\mathsf{e}+1} + \boldsymbol{e}'_{\mathsf{e}} + \lfloor q/2 \rfloor \boldsymbol{\mu}). \end{aligned} \tag{1}$$

The last equation (statistical indistinguishability) holds by the noise smuding lemma [AJL+12]. This equation shows that we can simulate an update ciphertext by using the original ciphertext, its plaintext and randomness, the new epoch public key, and *randomness* for generating the token $\Delta_{\mathsf{e}+1}$ (not the token itself).

To show the security, we define auxiliary algorithms for simulation.

$\mathsf{Hyb.Upd}(\mathsf{ct_e}, \boldsymbol{B}_{\mathsf{e}+1}, \boldsymbol{\mu}; \boldsymbol{e}'_{\mathsf{e}}, (\boldsymbol{R}_{\mathsf{e}+1}, \boldsymbol{R}'_{\mathsf{e}+1}))$:
- Parse $\mathsf{ct_e} = (\boldsymbol{u}_{\mathsf{e}}, \boldsymbol{c}_{\mathsf{e}})$.
- Choose $\tilde{\boldsymbol{r}} \leftarrow \{-1, +1\}^m$ and set $\boldsymbol{r}^\dagger := \mathsf{BD}(\boldsymbol{u}_{\mathsf{e}})\boldsymbol{R}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1}$.
- Set $\mathsf{ct}_{\mathsf{e}+1} := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) := (\boldsymbol{r}^\dagger \boldsymbol{A}, \boldsymbol{r}^\dagger \boldsymbol{B}_{\mathsf{e}+1} + \boldsymbol{e}'_{\mathsf{e}} + \lfloor q/2 \rfloor \boldsymbol{\mu})$.
- Output $(\mathsf{ct}_{\mathsf{e}+1}; \boldsymbol{e}'_{\mathsf{e}})$.

$\mathsf{Sim.Gen}(\mathsf{pp})$:
- Choose and output $\mathsf{pk_e} := \boldsymbol{B}^+_{\mathsf{e}} \leftarrow \mathbb{Z}_q^{m \times \ell}$.

$\mathsf{Sim.TokGen}(\mathsf{pp})$:
- Choose and output $\Delta^+_{\mathsf{e}+1} := (\boldsymbol{M}^+_{\mathsf{e}+1}, \boldsymbol{N}^+_{\mathsf{e}+1}) \leftarrow \mathbb{Z}_q^{n\eta \times (n+\ell)} \times \mathbb{Z}_q^{m \times (n+\ell)}$.

$\mathsf{Sim.Upd}(\mathsf{pp})$:
- Choose and output $\mathsf{ct}_{\mathsf{e}+1} := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

$\mathsf{Sim.Enc}(\mathsf{pp})$:
- Choose and output $\mathsf{ct_e} := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

Lemma 4.2. $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct_e}) \stackrel{\mathsf{s}}{\approx} \mathsf{Hyb.Upd}(\mathsf{ct_e}, \boldsymbol{B}_{\mathsf{e}+1}, \boldsymbol{\mu}; \boldsymbol{e}'_{\mathsf{e}}, (\boldsymbol{R}_{\mathsf{e}+1}, \boldsymbol{R}'_{\mathsf{e}+1}))$

By Eq. (1), Lem. 4.2 immediately holds. That is, we can simulate $\mathcal{O}.\mathsf{Upd}(\mathsf{ct_e})$ by using $\mathsf{Hyb.Upd}(\mathsf{ct_e}, \boldsymbol{B}_{\mathsf{e}+1}, \boldsymbol{\mu}; \boldsymbol{e}'_{\mathsf{e}}, (\boldsymbol{R}_{\mathsf{e}+1}, \boldsymbol{R}'_{\mathsf{e}+1}))$.

We follow the firewall technique [LT18, KLR19, BDGJ20, Jia20] to prove security, but we use the relaxed firewall notion in Def. 3.2.

Proof of Thm. 4.2. Let T be the upper bound of the number of epoch. We consider a sequence of hybrid games. First, we define the following hybrid game:

$\mathsf{Hyb}_i(b)$: This is the same as $\mathsf{Exp}^{\mathsf{(b\text{-}uni,uni)\text{-}r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda, b)$ except the following difference: When the adversary sends a query $(\overline{\mu}, \overline{\mathsf{ct}})$ to $\mathcal{O}.\mathsf{Chall}$ or an empty query to $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$ at epoch j,
- for $j < i$, return an honestly generated challenge-equal ciphertext. That is, if $b = 0$, $\mathsf{UE.Enc}(\mathsf{k}_{\tilde{\mathsf{e}}}, \overline{\mu})$ else $\mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \overline{\mathsf{ct}})$.
- for $j \geq i$, return a random ciphertext.

It is easy to see that $\mathsf{Hyb}_{T+1}(b)$ is the same as the original r-INE-UE-CPA game in the backward-leak uni-directional setting $\mathsf{Exp}^{\mathsf{(b\text{-}uni,uni)\text{-}r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda, b)$. Let $U(\lambda)$ be a random variable distributed uniformly in $[0, T]$, by the standard hybrid argument, we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{(b\text{-}uni,uni)\text{-}r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda) \leq (T+1)|\Pr[\mathsf{Hyb}_{U(\lambda)+1}(1) = 1] - \Pr[\mathsf{Hyb}_{U(\lambda)}(1) = 1]| \\ + (T+1)|\Pr[\mathsf{Hyb}_{U(\lambda)+1}(0) = 1] - \Pr[\mathsf{Hyb}_{U(\lambda)}(0) = 1]|,\end{aligned}$$

where we use $\Pr[U(\lambda) = i] = 1/(T+1)$. Note that $\mathsf{Hyb}_0(0) = \mathsf{Hyb}_0(1)$ trivially holds since all challenge-equal ciphertexts are random ciphertexts. Thus, our goal is to prove $|\Pr[\mathsf{Hyb}_{U(\lambda)+1}(b) = 1] - \Pr[\mathsf{Hyb}_{U(\lambda)}(b) = 1]| \leq \mathsf{negl}(\lambda)$ for $b \in \{0, 1\}$.

Hereafter, we write $\mathsf{Hyb}_i(b)$ instead of $\mathsf{Hyb}_{U(\lambda)}(b)$ for simplicity. Next, we define the following hybrid game:

$\mathsf{Hyb}'_i(b)$**:** This is the same as $\mathsf{Hyb}_i(b)$ except that the game chooses $\mathsf{fwl}, \mathsf{fwr} \leftarrow [0, T]$. If the adversary corrupts k_j such that $j \in [\mathsf{fwl}, \mathsf{fwr}]$ or $\Delta_{\mathsf{fwr}+1}$, the game aborts.

The guess is correct with probability $1/(T+1)^2$. We have

$$|\Pr[\mathsf{Hyb}_i(b) = 1] - \Pr[\mathsf{Hyb}_{i-1}(b)]| \leq (T+1)^2 |\Pr[\mathsf{Hyb}'_i(b) = 1] - \Pr[\mathsf{Hyb}'_{i-1}(b) = 1]|.$$

If $|\Pr[\mathsf{Hyb}'_{U(\lambda)+1}(b) = 1] - \Pr[\mathsf{Hyb}'_{U(\lambda)}(b) = 1]| \leq \mathsf{negl}(\lambda)$, we complete the proof of Thm. 4.2. ■

Lemma 4.3. *If the LWE assumption holds, it holds that* $|\Pr[\mathsf{Hyb}'_{i+1}(b) = 1] - \Pr[\mathsf{Hyb}'_i(b) = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that the difference between these two games appears when the challenge query is sent at epoch i, so we can assume $\widetilde{\mathsf{e}} = i$. We start from $\mathsf{Hyb}'_{i+1}(b)$ and gradually change it to $\mathsf{Hyb}'_i(b)$. We define another sequence of games.

$\mathsf{Hyb}^{\mathsf{r}}_i(b)$**:** This is the same as $\mathsf{Hyb}'_i(b)$ except that we use the hybrid update algorithm $\mathsf{Hyb.Upd}$ to simulate $\mathcal{O}.\mathsf{Upd}$. More precisely, $\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_{\mathsf{e}-1})$ act as follows:
- If $(\cdot, \mathsf{ct}_{\mathsf{e}-1}, \mathsf{e}-1; \boldsymbol{e}'_{\mathsf{e}-1}; \boldsymbol{\mu}) \notin \mathcal{L}$, then return $\perp$
- Otherwise, $(\mathsf{ct}_{\mathsf{e}}, \boldsymbol{e}'_{\mathsf{e}}) \leftarrow \mathsf{Hyb.Upd}(\mathsf{ct}_{\mathsf{e}-1}, \boldsymbol{B}_{\mathsf{e}}, \boldsymbol{\mu}; \boldsymbol{e}'_{\mathsf{e}-1}, (\boldsymbol{R}_{\mathsf{e}}, \boldsymbol{R}'_{\mathsf{e}}))$.
- $\mathcal{L} \coloneqq \mathcal{L} \cup \{(\cdot, \mathsf{ct}_{\mathsf{e}}, \mathsf{e}; \boldsymbol{e}'_{\mathsf{e}}, \boldsymbol{\mu})\}$.

Note that $\boldsymbol{R}_{\mathsf{e}}$ and $\boldsymbol{R}'_{\mathsf{e}}$ are randomness used in TokGen, so anyone can choose them. Simulators internally choose and record them.

Proposition 4.3. $|\Pr[\mathsf{Hyb}'_i(b) = 1] - \Pr[\mathsf{Hyb}^{\mathsf{r}}_i(b) = 1]| \leq \mathsf{negl}(\lambda)$.

It is easy to see Prop. 4.3 holds by Lem. 4.2. The next goal is proving $|\Pr[\mathsf{Hyb}^{\mathsf{r}}_{i+1}(b) = 1] - \Pr[\mathsf{Hyb}^{\mathsf{r}}_i(b) = 1]| \leq \mathsf{negl}(\lambda)$. We define the following games.

$\mathsf{G}_j(i, b)$**:** This is the same as $\mathsf{Hyb}^{\mathsf{r}}_i(b)$ except the following difference.
- For $i \leq k < j$, pk_k and Δ_k are honestly generated as in the real.
- For $\mathsf{fwr} \geq k \geq j$, pk_k and Δ_k are uniformly random.

That is, we gradually erase information about UE secret keys from newer epochs to older epochs. We note that $j \in [i, \mathsf{fwr}+1]$ and i is fixed. By the definition, we have

$$\mathsf{G}_{\mathsf{fwr}+1}(i+1, b) = \mathsf{Hyb}^{\mathsf{r}}_{i+1}(b) \text{ and } \mathsf{G}_{\mathsf{fwr}+1}(i, b) = \mathsf{Hyb}^{\mathsf{r}}_i(b). \tag{2}$$

We prove that

$$|\Pr[\mathsf{G}_{j+1}(i+1, b) = 1] - \Pr[\mathsf{G}_j(i+1, b) = 1]| \leq \mathsf{negl}(\lambda) \text{ for } j \in [i, \mathsf{fwr}] \tag{3}$$
$$|\Pr[\mathsf{G}_i(i+1, b) = 1] - \Pr[\mathsf{G}_i(i, b) = 1]| \leq \mathsf{negl}(\lambda) \tag{4}$$
$$|\Pr[\mathsf{G}_{j+1}(i, b) = 1] - \Pr[\mathsf{G}_j(i, b) = 1]| \leq \mathsf{negl}(\lambda) \text{ for } j \in [i, \mathsf{fwr}]. \tag{5}$$

From these equations, we immediately obtain

$$|\Pr[\mathsf{G}_{\mathsf{fwr}+1}(i+1,b)=1]-\Pr[\mathsf{G}_{\mathsf{fwr}+1}(i,b)]|\leq \mathsf{negl}(\lambda).$$

By combining this with Prop. 4.3 and Eq. (2), we obtain what we want to prove (Lem. 4.3). Thus, all we must do is proving Eqs. (3) to (5).

First, we prove Eq. (3). We define a few hybrid games as follows.

- $\mathsf{Game}\text{-}0(b)$: This is the same as $\mathsf{G}_{j+1}(i+1,b)$. At this point, public keys and tokens of epochs in $[i,j]$ are real values while those at epochs in $[j+1,\mathsf{fwr}]$ are already random values.
- $\mathsf{Game}\text{-}1(b)$: This is the same as $\mathsf{Game}\text{-}0(b)$ except that we modify the public key part of epoch j. We use $\boldsymbol{B}_j^+ \leftarrow \mathbb{Z}_q^{m\times \ell}$ instead of $\boldsymbol{B}_j$ such that $(\boldsymbol{S}_j,\boldsymbol{B}_j)\leftarrow \mathsf{Reg.Gen}(1^\lambda)$. Note that we do not use the secret key $\boldsymbol{S}_j$ of epoch j anywhere in this game since Δ_{j+1} is already a random value.
- $\mathsf{Game}\text{-}2(b)$: This is the same as $\mathsf{Game}\text{-}1(b)$ except that we modify the token generation algorithm for token Δ_j. We use $\Delta_j := (\boldsymbol{M}_j^+,\boldsymbol{N}_j^+)\leftarrow \mathbb{Z}_q^{n\eta\times(n+\ell)}\times \mathbb{Z}_q^{m\times(n+\ell)}$ instead of $(\boldsymbol{M}_j,\boldsymbol{N}_j)\leftarrow \mathsf{TokGen}(\mathsf{k}_{j-1},\mathsf{k}_j)$.

Obviously, $\mathsf{Game}\text{-}2(b)$ is the same as $\mathsf{G}_j(i+1,b)$. It is easy to see if we prove the following, we complete the proof of Eq. (3).

Proposition 4.4. *If the LWE assumption holds, it holds that* $|\Pr[\mathsf{Game}\text{-}1(b)=1]-\Pr[\mathsf{G}_{j+1}(i+1,b)=1]|\leq \mathsf{negl}(\lambda)$.

Proposition 4.5. *It holds that* $|\Pr[\mathsf{Game}\text{-}2(b)=1]-\Pr[\mathsf{Game}\text{-}1(b)=1]|\leq \mathsf{negl}(\lambda)$.

We will prove these propositions above later.

Next, we prove Eq. (4). The only difference between $\mathsf{G}_i(i+1,b)$ and $\mathsf{G}_i(i,b)$ is the challenge-equal ciphertext at epoch i. That is, $\mathsf{G}_i(i,b)$ is the same as $\mathsf{G}_i(i+1,b)$ except that we modify the challenge-equal ciphertext for b at epoch i. We use $(\bar{\boldsymbol{u}},\bar{\boldsymbol{c}})\leftarrow \mathbb{Z}_q^n\times\mathbb{Z}_q^\ell$ instead of $(\bar{\boldsymbol{u}},\bar{\boldsymbol{c}})\leftarrow \mathsf{Upd}(\Delta_i^+,\overline{\mathsf{ct}})$ (the case $b=1$) or $(\bar{\boldsymbol{u}},\bar{\boldsymbol{c}})\leftarrow \mathsf{Enc}(\mathsf{k}_i,\overline{\mu}_0)$ (the case $b=0$). We prove the following proposition later.

Proposition 4.6. *It holds that* $|\Pr[\mathsf{G}_i(i+1,b)=1]-\Pr[\mathsf{G}_i(i,b)=1]|\leq \mathsf{negl}(\lambda)$.

Lastly, we prove Eq. (5). Once the challenge-equal ciphertext at epoch i becomes random, we need to go back to games where public keys and tokens are real. In $\mathsf{G}_j(i,b)$ for $j\in[i,\mathsf{fwr}]$, publics keys and tokens (from epochs j to fwr) are also random. We need to change them from random to real since we need to arrive at Hyb_i^r, where public keys and tokens are real (but ciphertext at epoch i is random). Thus, we need to prove Eq. (5). These backward transitions are possible by using the proof of Eq. (3) in a reverse manner. We summarize how public keys, update tokens, and challenge-equal ciphertexts at epoch i are generated in Fig. 4.

Thus, we complete the proof of Lem. 4.3 if we prove Prop. 4.4 to 4.6. We write those proofs below. ■

Value	$\mathsf{G}_{i+1}(i+1,b)$	Game-1	Game-2 $= \mathsf{G}_i(i+1,b)$	$\mathsf{G}_i(i,b)$
pk_i	$\mathsf{Reg.Gen}(1^\lambda)$	$\mathsf{Sim.Gen}(\mathsf{pp})$	$\mathsf{Sim.Gen}(\mathsf{pp})$	$\mathsf{Sim.Gen}(\mathsf{pp})$
Δ_i	$\mathsf{TokGen}(\mathsf{sk}_{i-1},\mathsf{pk}_i)$	$\mathsf{TokGen}(\mathsf{sk}_{i-1},\mathsf{pk}_i)$	$\mathsf{Sim.TokGen}(\mathsf{pp})$	$\mathsf{Sim.TokGen}(\mathsf{pp})$
$\mathsf{ct}^*_{i,1}$	$\mathsf{Upd}(\Delta_i,\mathsf{ct}_{i-1})$	$\mathsf{Upd}(\Delta_i,\mathsf{ct}_{i-1})$	$\mathsf{Upd}(\Delta_i^+,\mathsf{ct}_{i-1})$	$\mathsf{Sim.Upd}(\mathsf{pp})$
$\mathsf{ct}^*_{i,0}$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Sim.Enc}(\mathsf{pp})$

Fig. 4: The differences of public keys, update tokens, challenge-equal ciphertexts at epoch i in hybrid games. We focus the case where $i = \widetilde{\mathsf{e}}$.

Proofs of core propositions. We give the proofs of Prop. 4.4 to 4.6.

Proof of Prop. 4.4. We construct a reduction $\mathcal{B}$ that solves the LWE problem by using the distinguisher $\mathcal{A}$ for the two games.

Recall that the key k_j of epoch j consists of $(\mathsf{sk}_j, \mathsf{pk}_j)$. $\mathcal{B}$ is given an LWE instance $(\boldsymbol{A}, \boldsymbol{B})$ and set $\boldsymbol{B}_j \coloneqq \boldsymbol{B}$. That is, $\boldsymbol{B}$ is used as the public key pk_j of epoch j. Note that $\mathcal{B}$ can simulate all values in epoch $k \in [0,T] \setminus [\mathsf{fwl},\mathsf{fwr}]$ since all values in epoch k (outside the firewall) are independent of the secret key of epoch j. (Note that such values may be related to the public key of epoch j via tokens.) That is, $\mathcal{B}$ can choose the secret key $\boldsymbol{S}_k$. We also note that $\mathcal{B}$ can simulate $\mathcal{O}.\mathsf{Upd}$ by using $\mathsf{Hyb.Upd}$. In $[\mathsf{fwl},\mathsf{fwr}]$, values are related to the secret key $\boldsymbol{S}$ behind $\boldsymbol{B}$. However, in $\mathsf{G}_{j+1}(i+1,b)$ (and $\mathsf{Game\text{-}1}(b)$), all values in $[j+1,\mathsf{fwr}]$ are uniformly random values. Note that the original update token Δ_{j+1} needs sk_j and pk_{j+1}. However, Δ_{j+1} was already changed to Δ^+_{j+1}, which is uniformly random value, and we do not need sk_j.

Thus, the issue is how to simulate values in epoch j' such that $j' \in [\mathsf{fwl}, j]$ (including the case where $\mathsf{fwl} = j$). As we see in the definition of TokGen, we do not need sk_j to generate Δ_j and $\mathcal{B}$ can simulate Δ_j. Therefore, $\mathcal{B}$ can also simulate $\mathsf{ct}^*_{j,b}$ for both $b = 0,1$. For $j'' \in [\mathsf{fwl}, j-1]$, public keys and tokens are not related to sk_j. Thus, $\mathcal{B}$ chooses $\boldsymbol{S}_{j''}$ and can simulate all values $(\mathsf{pk}_{j''}, \Delta_{j''}, \mathsf{ct}^*_{j'',b})$ by using the normal algorithms.

If $\boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{X}$ where $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n\times\ell}$ and $\boldsymbol{X} \leftarrow \chi^{m\times\ell}$, the distribution is the same as $\mathsf{G}_{i+1}(i+1,b)$. If $\boldsymbol{B}$ is uniformly random, the distribution is the same as $\mathsf{Game\text{-}1}(b)$. Therefore, $\mathcal{B}$ distinguish the instance if $\mathcal{A}$ distinguishes the two games. This completes the proof. ■

Proof of Prop. 4.5. The difference between these two games is as follows:

$\mathsf{Game\text{-}1}(b)$**:** $\Delta_j = (\boldsymbol{M}_j, \boldsymbol{N}_j)$:

$$\boldsymbol{M}_j \coloneqq \boldsymbol{R}_j \cdot [\boldsymbol{A} \mid \boldsymbol{B}_j] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{j-1})], \boldsymbol{N}_j \coloneqq \boldsymbol{R}'_j \cdot [\boldsymbol{A} \mid \boldsymbol{B}_j],$$

where $\boldsymbol{R}_j \leftarrow \{-1,+1\}^{n\eta\times m}$, $\boldsymbol{R}'_j \leftarrow \{-1,+1\}^{m\times m}$.

$\mathsf{Game\text{-}2}(b)$**:** $\Delta^+_j = (\boldsymbol{M}^+_j, \boldsymbol{N}^+_j)$: $(\boldsymbol{M}^+_j, \boldsymbol{N}^+_j) \leftarrow \mathbb{Z}_q^{n\eta\times(n+\ell)} \times \mathbb{Z}_q^{m\times(n+\ell)}$.

In $\mathsf{Game}\text{-}1(b)$ and $\mathsf{Game}\text{-}2(b)$, the public key $\boldsymbol{B}_j \leftarrow \mathbb{Z}_q^{m\times\ell}$ is uniformly random. Thus, we can apply the leftover hash lemma and these differences are statistically indistinguishable. This completes the proof. ∎

Proof of Prop. 4.6. The difference between these two games is as follows: For $b = 1$,

$\mathsf{G}_i(i+1, 1)$**:** $\mathsf{ct}^*_{i,1} = (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}})$: $(\boldsymbol{u}'+\tilde{\boldsymbol{u}}, \boldsymbol{c}_{i-1}+\boldsymbol{c}'+\tilde{\boldsymbol{v}}) = (\boldsymbol{0}, \boldsymbol{c}_{i-1}) + \mathsf{BD}(\boldsymbol{u}_i)\boldsymbol{M}_i^+ + \tilde{\boldsymbol{r}}\boldsymbol{N}_i^+$, where $\tilde{\boldsymbol{r}} \leftarrow \{-1, +1\}^m$.

$\mathsf{G}_i(i, 1)$**:** $\mathsf{ct}^*_{i,1} = (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}})$: $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

n In $\mathsf{G}_i(i+1, b)$ and $\mathsf{G}_i(i, b)$, $\boldsymbol{N}_i^+$ is uniformly random. Thus, we can apply the leftover hash lemma and these differences are statistically indistinguishable. For $b = 0$,

$\mathsf{G}_i(i+1, 0)$**:** $\mathsf{ct}^*_{i,0} = (\boldsymbol{u}, \boldsymbol{c})$: $(\boldsymbol{r}\boldsymbol{A}_i, \boldsymbol{r}\boldsymbol{B}_i^+ + \boldsymbol{e}' + \lfloor q/2 \rfloor \boldsymbol{\mu}_0)$, where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{m\times}$, $\boldsymbol{r} \leftarrow \{-1, +1\}^m$, $\boldsymbol{e}' \leftarrow \chi_{\mathsf{ns}}^\ell$, and $\boldsymbol{B}_i^+ \leftarrow \mathbb{Z}_q^{m\times\ell}$.

$\mathsf{G}_i(i, 0)$**:** $\mathsf{ct}^*_{i,0} = (\boldsymbol{u}, \boldsymbol{c})$: $(\boldsymbol{u}, \boldsymbol{c}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

In $\mathsf{G}_i(i+1, b)$ and $\mathsf{G}_i(i, b)$, the public key $\boldsymbol{B}_i^+ \leftarrow \mathbb{Z}_q^{m\times\ell}$ is uniformly random. Thus, we can apply the leftover hash lemma and these differences are statistically indistinguishable. This completes the proof. ∎

5 Construction with No-Directional Key Update

5.1 Scheme Description

We present a no-directional key update scheme $\mathsf{UE_{io}}$ from puncturable PRFs and IO. Let $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \{0,1\}^\ell$ and $\mathsf{PRG} : \{0,1\}^\tau \rightarrow \{0,1\}^n$. We will set $\tau := \lambda$, $n := 2\lambda$.

$\mathsf{Setup}(1^\lambda)$: Does nothing.
$\mathsf{KeyGen}(1^\lambda)$:
- Generate $\mathsf{K} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and output $\mathsf{k_e} := \mathsf{K}$.

$\mathsf{TokGen}(\mathsf{k_e}, \mathsf{k_{e+1}})$
- Generate and output $\Delta_{\mathsf{e}+1} \leftarrow i\mathcal{O}(\mathsf{C_{re}}[\mathsf{k_e}, \mathsf{k_{e+1}}])$ where circuit $\mathsf{C_{re}}$ is described in Fig. 5.

$\mathsf{Enc}(\mathsf{k_e}, \mu \in \{0,1\}^\ell)$:
- Choose $r \leftarrow \{0,1\}^\tau$ and compute $t := \mathsf{PRG}(r)$.
- Compute $y := \mathsf{PRF}(\mathsf{K}, t)$ and output $\mathsf{ct} := (t, y \oplus \mu)$.

$\mathsf{Dec}(\mathsf{k_e}, \mathsf{ct})$:
- Parse $\mathsf{k_e} = \mathsf{K}$ $\mathsf{ct} = (t, c)$.
- Compute $\mu' := c \oplus \mathsf{PRF}(\mathsf{K}, t)$ and output μ'.

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct_e})$
- Parse $\Delta_{\mathsf{e}+1} = i\mathcal{O}(\mathsf{C_{re}}[\mathsf{k_e}, \mathsf{k_{e+1}}])$ and choose $r_{\mathsf{e}+1} \leftarrow \{0,1\}^\tau$.
- Compute and output $(t, c) := i\mathcal{O}(\mathsf{C_{re}}[\mathsf{k_e}, \mathsf{k_{e+1}}])(\mathsf{ct_e}, r_{\mathsf{e}+1})$.

Theorem 5.1. $\mathsf{UE_{io}}$ *is an r-IND-UE-CPA secure UE scheme in the no-directional key updates setting.*

We omit the proof due to space limitations. See the full version.

Update Function $\mathsf{C}_{\mathsf{re}}[\mathsf{k}_{\mathsf{e}}, \mathsf{k}_{\mathsf{e}+1}](\mathsf{ct}_{\mathsf{e}}, r_{\mathsf{e}+1})$

Hardwired: k_{e}, $\mathsf{k}_{\mathsf{e}+1}$.
Input: A ciphertext ct_{e} and randomness $r_{\mathsf{e}+1} \in \{0,1\}^{\tau}$.
Padding: This circuit is padded to size $\mathsf{pad}_{\mathsf{T}} := \mathsf{pad}_{\mathsf{T}}(\lambda)$, which is determined in analysis.

1. Parse $\mathsf{ct}_{\mathsf{e}} = (t_{\mathsf{e}}, c_{\mathsf{e}})$
2. Compute $\mu' := c_{\mathsf{e}} \oplus \mathsf{PRF}(\mathsf{k}_{\mathsf{e}}, t_{\mathsf{e}})$.
3. Compute $t' := \mathsf{PRG}(r_{\mathsf{e}+1})$ and $y' := \mathsf{PRF}(\mathsf{k}_{\mathsf{e}+1}, t')$
4. Return $\mathsf{ct}_{\mathsf{e}+1} := (t', y' \oplus \mu')$.

Fig. 5: The description of C_{re}

Acknowledgments

The author would like to thank Fuyuki Kitagawa for giving a pointer to the right reference about an RKA-secure AE scheme. The author also thanks anonymous reviewers of PKC 2022 for useful comments.

References

ABPW13. Y. Aono, X. Boyen, L. T. Phong, and L. Wang. Key-Private Proxy Re-encryption under LWE. In *INDOCRYPT 2013*, pages 1–18. 2013.

AJL+12. G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *EUROCRYPT 2012*, pages 483–501. 2012.

AMP19. N. Alamati, H. Montgomery, and S. Patranabis. Symmetric Primitives with Structured Secrets. In *CRYPTO 2019, Part I*, pages 650–679. 2019.

BDGJ20. C. Boyd, G. T. Davies, K. Gjøsteen, and Y. Jiang. Fast and Secure Updatable Encryption. In *CRYPTO 2020, Part I*, pages 464–493. 2020.

BEKS20. D. Boneh, S. Eskandarian, S. Kim, and M. Shih. Improving Speed and Security in Updatable Encryption Schemes. In *ASIACRYPT 2020, Part III*, pages 559–589. 2020.

BGH13. Z. Brakerski, C. Gentry, and S. Halevi. Packed Ciphertexts in LWE-Based Homomorphic Encryption. In *PKC 2013*, pages 1–13. 2013.

BGI+12. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012.

BGV14. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

BLMR13. D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. Key Homomorphic PRFs and Their Applications. In *CRYPTO 2013, Part I*, pages 410–428. 2013.

BV11. Z. Brakerski and V. Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *CRYPTO 2011*, pages 505–524. 2011.

BV14. Z. Brakerski and V. Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

CCL$^+$14. N. Chandran, M. Chase, F.-H. Liu, R. Nishimaki, and K. Xagawa. Re-encryption, Functional Re-encryption, and Multi-hop Re-encryption: A Framework for Achieving Obfuscation-Based Security and Instantiations from Lattices. In *PKC 2014*, pages 95–112. 2014.

CLT20. L. Chen, Y. Li, and Q. Tang. CCA Updatable Encryption Against Malicious Re-encryption Attacks. In *ASIACRYPT 2020, Part III*, pages 590–620. 2020.

DN21. N. Döttling and R. Nishimaki. Universal Proxy Re-Encryption. In *PKC 2021, Part I*, pages 512–542. 2021.

EPRS17. A. Everspaugh, K. G. Paterson, T. Ristenpart, and S. Scott. Key Rotation for Authenticated Encryption. In *CRYPTO 2017, Part III*, pages 98–129. 2017.

FMM21. A. Fabrega, U. Maurer, and M. Mularczyk. A Fresh Approach to Updatable Symmetric Encryption. *IACR Cryptol. ePrint Arch.*, 2021:559, 2021.

Gen09. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

HLL16. S. Han, S. Liu, and L. Lyu. Efficient KDM-CCA Secure Public-Key Encryption for Polynomial Functions. In *ASIACRYPT 2016, Part II*, pages 307–338. 2016.

Jia20. Y. Jiang. The Direction of Updatable Encryption Does Not Matter Much. In *ASIACRYPT 2020, Part III*, pages 529–558. 2020.

JLS21. A. Jain, H. Lin, and A. Sahai. Indistinguishability Obfuscation from Well-Founded Assumptions. In *STOC 2021*, 2021.

KLR19. M. Klooß, A. Lehmann, and A. Rupp. (R)CCA Secure Updatable Encryption with Integrity Protection. In *EUROCRYPT 2019, Part I*, pages 68–99. 2019.

LR21. F. Levy-dit-Vehel and M. Roméas. A Composable Look at Updatable Encryption. *IACR Cryptol. ePrint Arch.*, 2021:538, 2021.

LT18. A. Lehmann and B. Tackmann. Updatable Encryption with Post-Compromise Security. In *EUROCRYPT 2018, Part III*, pages 685–716. 2018.

Nis21. R. Nishimaki. The Direction of Updatable Encryption Does Matter. *IACR Cryptol. ePrint Arch.*, page 221, 2021.

NX15. R. Nishimaki and K. Xagawa. Key-Private Proxy Re-Encryption from Lattices, Revisited. *IEICE Transactions*, 98-A(1):100–116, 2015.

Reg09. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009.

SS21. D. Slamanig and C. Striecks. Puncture ’Em All: Stronger Updatable Encryption with No-Directional Key Updates. *IACR Cryptol. ePrint Arch.*, 2021:268, 2021.

SW21. A. Sahai and B. Waters. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. *SIAM Journal on Computing*, 50(3):857–908, 2021.