The Direction of Updatable Encryption *Does* Matter

Ryo Nishimaki [1]

[1] NTT Secure Platform Laboratories, 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 there exist UE schemes that are secure in the bi-directional key updates setting, but not secure in the backward-leak uni-directional key updates setting. This result is a 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 uni-directional key updates setting (we call the latter "forward-leak uni-directional" key updates to distinguish two types of uni-directional key updates in this paper).

We also construct 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 solves the open problem left by Jiang at Asiacrypt 2020.

Keywords: updatable encryption, key update, lattice

Contents

1 Introduction

1.1 Background

Updatable Encryption. Updatable encryption (UE) is a version of secret key encryption where we can periodically update a secret key and a ciphertext. More specifically, a secret key k_e is generated at each time 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, security of UE guarantees that confidentiality holds even after some old (and even new) keys and tokens are corrupted as long as a target secret key is not corrupted and a target ciphertext cannot be converted into a ciphertext under a corrupted secret key.

There are two versions of UE by considering two types of updates. The first one is ciphertext-independent updates UE, where we can generate an update token only from two secret keys [LT18, KLR19, BDGJ20, Jia20a]. The other one is ciphertext-dependent updates UE, where we need not only two secret keys but also a part of ciphertext (called header of ciphertext) to generate an update token [BLMR13, EPRS17b, BEKS20]. The former is preferable in terms of efficiency. We focus on ciphertext-independent updates UE in this work.

A typical application of UE is encrypting data stored in outsourced storage such as cloud servers. A serious threat to encryption is that a secret key is leaked. In that case, no security is guaranteed by standard encryption. Key updating is a common 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 loss. In the cloud storage application above, to hide plaintext and a new key from a server, we need to download all ciphertexts from the server, decrypt and re-encrypt them, and upload again. 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.

A natural question is whether an adversary can obtain information about secret keys from update tokens. This question depends on settings and concrete constructions. Adversaries can easily obtain a secret key from an update token and another secret key in some constructions and settings. Several definitions have been proposed since after UE was introduced [BLMR13]. In particular, confidentiality of UE have been improved to capture realistic attack models [EPRS17b, LT18, KLR19, BDGJ20]. Lehman and Tackmann formalized trivially leaked information from corrupted keys and tokens as key updates direction [LT18]. However, most of previous works do not focus on decreasing information trivially leaked from corrupted keys and tokens. In this work, we focus on the direction of key updates.

Direction of key updates. Directions of key updates describe information that UE schemes cannot avoid leaking. 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. That is, we can extract $\mathsf{k}_{\mathsf{e}+1}$ (resp. k_e) from Δ_e and k_e (resp. $\mathsf{k}_{\mathsf{e}+1}$). This setting is defined as *bi-directional* key updates [EPRS17b, 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 adversary might not be able to infer k_e from $\mathsf{k}_{\mathsf{e}+1}$ and $\Delta_{\mathsf{e}+1}$.

At first look, secure UE with bi-directional key updates is stronger than one with uni-directional key updates. However, Jiang prove that secure UE with bi-directional key updates *is equivalent to* one with uni-directional key updates [Jia20a]. Jiang also presented the first post-quantum UE scheme with bi-directional key updates [Jia20a].

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 $\mathsf{k}_{\mathsf{e}-1}$ from k_{e} and Δ_{e} (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. We argue that the backward-leak uni-directional key updates setting is meaningful in this work. Thus, the first main question of this study are as follows.

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

We affirmatively answer to 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?*

We also affirmatively answer to 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 an adversary has k_{e} (resp. $\mathsf{k}_{\mathsf{e}-1}$) and Δ_{e}, it cannot infer $\mathsf{k}_{\mathsf{e}-1}$ (resp. k_{e}). Such key updates are called *no-directional* key updates [Jia20a]. 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?*

We solve this open question in this work.

1.2 Our Contribution

In this section, we explain our contributions. The first contribution 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 stronger than forward-leak uni-directional key updates (and bi-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. Note that UE with bi-directional key updates is equivalent to UE with *forward -leak* uni-directional key updates by Jiang's results [Jia20a].

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.
- The second scheme is a UE scheme with no-directional key updates and based on one-way functions (OWFs) and indistinguishability obfuscation (IO).

Both schemes satisfies r-IND-UE-CPA security, which was defined by Boyd, Davies, Gjøsteen, and Jiang [BDGJ20].

Organization. In Section 2, we provide preliminaries and basic definitions. In Section 3, we review the syntax and security definitions of UE. Section 4 defines a new definition of uni-directional key updates (backward-leak uni-directional key updates) and show that it is stronger than those of bi-directional and forward-leak uni-directional key updates.. In Section 5, we present our UE scheme with backward-leak uni-directional key updates based on the LWE problem and prove its security. In Section 6, we present our UE scheme with no-directional key updates based on OWFs and IO and prove its security.

2 Preliminaries

We define some notations and introduce cryptographic notions in this section.

Notations and basic concepts. In this paper, $x \leftarrow X$ denotes selecting an element from a finite set X uniformly at random, and $y \leftarrow \mathsf{A}(x)$ denotes assigning to y the output of a probabilistic or deterministic algorithm A on an input x. When we explicitly show that A uses randomness r, we write $y \leftarrow \mathsf{A}(x;r)$. For a finite set S, $U(S)$ denotes the uniform distribution over S. For strings x and y, $x\|y$ denotes the concatenation of x and y. Let $[\ell]$ and $[\ell, r]$ denote the set of integers $\{1, \cdots, \ell\}$ and $\{\ell, \ldots, r\}$, respectively, λ denote a security parameter, and $y := z$ denote that y is set, defined, or substituted by z. PPT stands for probabilistic polynomial time.

- A function $f : \mathbb{N} \to \mathbb{R}$ is a negligible function if for any constant c, there exists $\lambda_0 \in \mathbb{N}$ such that for any $\lambda > \lambda_0$, $f(\lambda) < \lambda^{-c}$. We write $f(\lambda) \leq \mathsf{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function.
- If $\mathcal{X}^{(b)} = \{X_\lambda^{(b)}\}_{\lambda \in \mathbb{N}}$ for $b \in \{0,1\}$ are two ensembles of random variables indexed by $\lambda \in \mathbb{N}$, we say that $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ are computationally indistinguishable if for any PPT distinguisher $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that

$$\Delta := |\Pr[\mathcal{D}(X_\lambda^{(0)}) = 1] - \Pr[\mathcal{D}(X_\lambda^{(1)}) = 1]| \leq \mathsf{negl}(\lambda).$$

  We write $\mathcal{X}^{(0)} \overset{\mathsf{c}}{\approx} \mathcal{X}^{(1)}$ to denote that the advantage Δ is negligible.
- The statistical distance between $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ over a countable set S is defined as $\Delta_{\mathsf{s}}(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}) := \frac{1}{2}\sum_{\alpha \in S} |\Pr[X_\lambda^{(0)} = \alpha] - \Pr[X_\lambda^{(1)} = \alpha]|$. We say that $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ are statistically/perfectly indistinguishable (denoted by $\mathcal{X}^{(0)} \overset{\mathsf{s}}{\approx} \mathcal{X}^{(1)}$/$\mathcal{X}^{(0)} \overset{\mathsf{p}}{\approx} \mathcal{X}^{(1)}$) if $\Delta_{\mathsf{s}}(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}) \leq \mathsf{negl}(\lambda)$ and $\Delta_{\mathsf{s}}(\mathcal{X}^{(0)}, \mathcal{X}^{(1)}) = 0$, respectively.

Basic cryptographic tools.

Definition 2.1 (Pseudorandom Generator). *A pseudorandom generator (PRG)* $\mathsf{PRG} : \{0,1\}^\tau \to \{0,1\}^{\tau+\ell(\lambda)}$ *with stretch* $\ell(\lambda)$ *(ℓ is some polynomial function) is a polynomial-time computable function that satisfies*

$$\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{PRG},\mathcal{A}} := |\Pr[\mathcal{A}(t) = 1 \mid t \leftarrow \mathsf{PRG}(r), r \leftarrow U_\tau] - \Pr[\mathcal{A}(t) = 1 \mid t \leftarrow U_{\tau+\ell(\lambda)}]| \leq \mathsf{negl}(\lambda),$$

where U_m denotes the uniform distribution over $\{0,1\}^m$.

Definition 2.2 (Pseudorandom functions). *For sets $\mathcal{D}$ and $\mathcal{R}$, let $\{\mathsf{F}_\mathsf{K}(\cdot) : \mathcal{D} \to \mathcal{R} \mid \mathsf{K} \in \{0,1\}^\lambda\}$ be a family of polynomially computable functions. We say that F is pseudorandom if for any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}^{\mathsf{F}(\mathsf{K},\cdot)}(1^\lambda) = 1 \mid \mathsf{K} \leftarrow \{0,1\}^\lambda] - \Pr[\mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1 \mid \mathsf{R} \leftarrow \mathcal{F}_U]| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{F}_U$ is the set of all functions from $\mathcal{D}$ to $\mathcal{R}$.

Theorem 2.3 ([GGM86]). *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a pseudorandom function that maps $n(\lambda)$ bits to $m(\lambda)$ bits (i.e., $\mathcal{D} := \{0,1\}^{n(\lambda)}$ and $\mathcal{R} := \{0,1\}^{m(\lambda)}$).*

Definition 2.4 (Puncturable pseudorandom function). *For sets $\mathcal{D}$ and $\mathcal{R}$, a puncturable pseudorandom function* PPRF *consists of a tuple of algorithms* $(\mathsf{F}, \mathsf{Punc})$ *that satisfies the following two conditions.*

Functionality preserving under puncturing: *For all polynomial size subset* $\{x_i\}_{i\in[k]}$ *of* $\mathcal{D}$*, and for all* $x \in \mathcal{D} \setminus \{x_i\}_{i\in[k]}$*, we have* $\Pr[\mathsf{F}(\mathsf{K},x) = \mathsf{F}(\mathsf{K}^*,x) : \mathsf{K} \leftarrow \{0,1\}^\lambda, \mathsf{K}^* \leftarrow \mathsf{Punc}(\mathsf{K},\{x_i\}_{i\in[k]})] = 1$.

Pseudorandomness at punctured points: *For all polynomial size subset* $\{x_i\}_{i\in[k]}$ *of* $\mathcal{D}$*, and any PPT adversary* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{pprf}}_{\mathsf{F},\mathcal{A}} := \Pr[\mathcal{A}(\mathsf{K}^*,\{\mathsf{F}(\mathsf{K},x_i)\}_{i\in[k]}) = 1] - \Pr[\mathcal{A}(\mathsf{K}^*,\mathcal{U}^k) = 1] \leq \mathsf{negl}(\lambda) \ ,$$

where $\mathsf{K} \leftarrow \{0,1\}^\lambda$, $\mathsf{K}^* \leftarrow \mathsf{Punc}(\mathsf{K},\{x_i\}_{i\in[k]})$*, and* $\mathcal{U}$ *denotes the uniform distribution over* $\mathcal{R}$*.*

Theorem 2.5 ([GGM86, BW13, BGI14, KPTZ13]). *If one-way functions exist, then for all efficiently computable functions* $n(\lambda)$ *and* $m(\lambda)$*, there exists a puncturable pseudorandom function that maps* $n(\lambda)$ *bits to* $m(\lambda)$ *bits (i.e.,* $\mathcal{D} := \{0,1\}^{n(\lambda)}$ *and* $\mathcal{R} := \{0,1\}^{m(\lambda)}$*).*

Advanced cryptographic tools.

Definition 2.6 (Indistinguishability Obfuscator). *A PPT algorithm* $i\mathcal{O}$ *is an IO for a circuit class* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *if it satisfies the following two conditions.*

Functionality: *For any security parameter* $\lambda \in \mathbb{N}$*, circuit* $C \in \mathcal{C}_\lambda$*, and input* x*, we have that*

$$\Pr[C'(x) = C(x) \mid C' \leftarrow i\mathcal{O}(C)] = 1 \ .$$

Indistinguishability: *For any PPT distinguisher* $\mathcal{D}$ *and for any pair of circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *such that for any input* x*,* $C_0(x) = C_1(x)$ *and* $|C_0| = |C_1|$*, it holds that*

$$\mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},\mathcal{A}}(\lambda) := |\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(\lambda) \ .$$

3 Updatable Encryption

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

3.1 Syntax

Definition 3.1. *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*.*

$\mathsf{UE.KeyGen}(\mathsf{pp}) \to \mathsf{k_e}$*: The key generation algorithm takes as input the public parameter and outputs an epoch key* $\mathsf{k_e}$*. (This algorithm is an option for UE.)*

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

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

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

$\mathsf{UE.Upd}(\Delta_{\mathsf{e}+1},\mathsf{ct_e}) \to \mathsf{ct_{e+1}}$*: 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.*

3.2 Security Experiments

We review security definitions for UE in this section.

Definition 3.2 (Correctness). *For any* $\mu \in \mathcal{M}$, *for* $0 \le \mathsf{e}_1 \le \mathsf{e}_2 \le T$, *it holds that*

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

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 3.3 (Confidentiality for Updatable Encryption [BDGJ20, Jia20a]). *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* $((\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset) \vee (\mathsf{x} = \mathsf{d} \wedge (\mathsf{e}^* \in \mathcal{T}^* \vee \mathcal{O}.\mathsf{Upd}(\mathsf{ct}^*)$ *is queried*$)))$ *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*

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

The definitions of oracles are described in Figure 1.

Definition 3.4 (Integrity for Updatable Encryption [BDGJ20, Jia20a]). *For* $\mathsf{atk} \in \{\mathsf{ctxt}, \mathsf{ptxt}\}$, *the game* $\mathsf{Exp}^{\mathsf{int\text{-}atk}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *is formalized as follows.*

- *Invoke* Setup *and set* $\mathsf{win} := 0$.
- *Let* $\mathcal{O} := \mathcal{O}.\{\mathsf{Enc}, \mathsf{Next}, \mathsf{Upd}, \mathsf{Corr}, \mathsf{Try}\}$.
- *Run* $\mathcal{A}^{\mathcal{O}}(1^\lambda)$.
- *If* $\mathsf{twf} = 1$ *then* $\mathsf{win} := 0$
- *return* win

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

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

The definitions of oracles are described in Figure 1.

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

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

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

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

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

- $\mu' / \bot \leftarrow \mathsf{UE.Dec}(\mathsf{k_e}, \mathsf{ct})$
- **if** $\Big((\mathsf{xx} = \mathsf{det} \wedge (\mathsf{ct}, \mathsf{e}) \in \widetilde{\mathcal{L}}^*)$ $\vee (\mathsf{xx} = \mathsf{rand} \wedge (\mu', \mathsf{e}) \in \widetilde{\mathcal{Q}}^*)\Big)$ **then** $\mathsf{twf} := 1$
- **return** μ' **or** $\bot$

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

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

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

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

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

- **if** $\widehat{\mathsf{e}} > \mathsf{e}$ **then return** $\bot$
- **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** $\bot$
- $\mathsf{phase} := 1; \mathsf{e}^* := \mathsf{e}$
- **if** $(\cdot, \overline{\mathsf{ct}}, \mathsf{e}^* - 1; \overline{\mu}_1) \notin \mathcal{L}$ **then return** $\bot$
- **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** $\bot$
- $\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}\}$
- $\widetilde{\mathcal{L}} := \widetilde{\mathcal{L}} \cup \{(\mathsf{ct}^*_\mathsf{e}, \mathsf{e})\}$
- **return** $\mathsf{ct}^*_{\mathsf{e}^*}$

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

- $\mu' / \bot \leftarrow \mathsf{UE.Dec}(\mathsf{k_e}, \mathsf{ct}^*)$
- **if** $\Big(\mathsf{e} \in \mathcal{K}^* \vee (\mathsf{atk} = \mathsf{ctxt} \wedge (\mathsf{ct}^*, \mathsf{e}) \in \mathcal{L}^*)$ $\vee (\mathsf{atk} = \mathsf{ptxt} \wedge (\mu', \mathsf{e}) \in \widetilde{\mathcal{Q}}^*)\Big)$ **then** $\mathsf{twf} := 1$
- **if** $\mu' \neq \bot$ **then** $\mathsf{win} := 1$

Figure 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 Sections 3.3 and 3.4.

3.3 Leakage Sets

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

- $\mathcal{K}$: 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 $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$.

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.

3.4 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 3.5 (Direction of Key Update). *We define inferred leakage key sets. The sets depends 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 3.6 (Inferred Token Sets). *For* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$,

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

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 3.7 (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:*

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

  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:*

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

  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 [Jia20a].

Definition 3.8 ($(\mathsf{kk}, \mathsf{cc})$-variant of confidentiality [Jia20a]). *Let* UE *be a UE scheme. Then the* $(\mathsf{kk}, \mathsf{cc})$-notion *advantage, for* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ *and* $\mathsf{notion} \in \{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{d\text{-}ind\text{-}ue\text{-}cpa}, \mathsf{r\text{-}ind\text{-}ue\text{-}cca}, \mathsf{d\text{-}ind\text{-}ue\text{-}cca}\}$, *of an adversary* $\mathcal{A}$ *against* UE *is defined as*

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

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 Definition 3.3 except for all leakage sets are both in the* kk*-directional key updates and* cc*-directional ciphertext updates.*

Definition 3.9 ($(\mathsf{kk}, \mathsf{cc})$-variant of integrity [Jia20a]). *Let* UE *be a UE scheme. Then the* $(\mathsf{kk}, \mathsf{cc})$-notion *advantage, for* $\mathsf{kk} \in \{\mathsf{no}, \mathsf{uni}, \mathsf{bi}\}$, $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$ *and* $\mathsf{notion} \in \{\mathsf{int\text{-}ctxt}, \mathsf{int\text{-}ptxt}\}$, *of an adversary* $\mathcal{A}$ *against* UE *is defined as*

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

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 Definition 3.4 except for all leakage sets are both in the* kk*-directional key updates and* cc*-directional ciphertext updates.*

3.5 Trivial Winning Condition

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

For all confidentiality games in Definition 3.3, 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 $\widetilde{\mathsf{e}} \in \mathcal{T}^* \vee$ "$\mathcal{O}.\mathsf{Upd}(\overline{\mathsf{ct}})$ is queried" is checked since if the condition does not hold, 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 $\mathcal{O}.\mathsf{Chall}$.

For CCA security and integrity, we need to consider other trivial winning conditions. However, such other conditions are not essential in this work. Thus, we only provide those conditions and do not explain the detail. See the paper by Jiang [Jia20a] for the detail. In the CCA setting for deterministic updates, the trivial winning condition $(\mathsf{ct}, \mathsf{e}) \in \widetilde{\mathcal{L}}^*$ is checked when $\mathcal{O}.\mathsf{Dec}(\mathsf{ct})$ is invoked. In the CCA setting for randomized updates, the trivial winning condition $(\mu', \mathsf{e}) \in \widetilde{\mathcal{Q}}^*$ is checked when $\mathcal{O}.\mathsf{Dec}(\mathsf{ct})$ is invoked and $\mu' \leftarrow \mathsf{Dec}(\mathsf{k}_\mathsf{e}, \mathsf{ct})$. In the integrity security (both for CTXT and PTXT), the trivial winning condition $\mathsf{e} \in \mathcal{K}^*$ is checked when $\mathcal{O}, \mathsf{Try}(\widetilde{\mathsf{ct}})$ is invoked. In the integrity security against CTXT, the trivial winning condition $(\widetilde{\mathsf{ct}}, \mathsf{e}) \in \mathcal{L}^*$ is also checked when $\mathcal{O}, \mathsf{Try}(\widetilde{\mathsf{ct}})$ is invoked. In the integrity security against PTXT, the trivial winning condition $(\mu', \mathsf{e}) \in \mathcal{Q}^*$ is also checked when $\mathcal{O}, \mathsf{Try}(\widetilde{\mathsf{ct}})$ is invoked and $\mu' \leftarrow \mathsf{Dec}(\mathsf{k}_\mathsf{e}, \widetilde{\mathsf{ct}})$.

3.6 Firewall and Insulated Region

Definition 3.10 (Firewall [LT18, KLR19, BDGJ20, Jia20a]). *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* $(\Delta_{\mathsf{fwl}+1}, \ldots, \mathsf{fwr})$ *are corrupted. That is,* $[\mathsf{fwl}+1, \mathsf{fwr}] \subseteq \mathcal{T}$.

Definition 3.11 (Insulated Region [LT18, KLR19, BDGJ20, Jia20a]). *The union of all insulated regions is defined as* $\mathcal{IR} := \bigcup_{[\mathsf{fwl},\mathsf{fwr}] \in \mathcal{FW}} [\mathsf{fwl}, \mathsf{fwr}]$.

4 Backward-Leak Uni-Directional Key Update and Relations

4.1 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 Definition 3.5.

Definition 4.1 (Uni-Directional Key Update (revisited)). *We define two types of uni-directional key updates. One is the same as that in Definition 3.5. To distinguish two types of uni-directional key updates, we rename the original one in Definition 3.5 to* forward-leak uni-directional *key updates. The definitions of two notions are as follows.*

- *backward-leak uni-directional key updates:*

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

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

• *forward-leak uni-directional key updates:*

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

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

By using the definition above, for $\mathsf{kk} \in \{\mathsf{no}, \mathsf{f\text{-}uni}, \mathsf{b\text{-}uni}, \mathsf{bi}\}$, we can consider Definitions 3.6 and 3.7.

We illustrate leaked information in the setting of forward/backward-leak uni-directional key updates settings in Figures 2 and 3.

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

Figure 2: Inferred keys in the forward-leak uni-directional key updates setting. 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. That is, we can obtain $\mathsf{k}_{\mathsf{e}+1}$ but not $\mathsf{k}_{\mathsf{e}-1}$.

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

Figure 3: Inferred keys in the backward-leak uni-directional key updates setting. 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. That is, we can obtain $\mathsf{k}_{\mathsf{e}-1}$ but not $\mathsf{k}_{\mathsf{e}+1}$.

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, Jia20a].[1] 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}. That is, $\mathsf{k}_{\mathsf{e}} = (\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}})$.

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 what we need to generate a ciphertext is the public key part of an epoch key 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 an encryption of sk_{e} under $\mathsf{pk}_{\mathsf{e}+1}$ and homomorphic properties. In fact, this is what we do in Section 5. 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 a natural setting. 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 $\mathsf{sk}_{\mathsf{e}+1}$ as in the existing schemes [LT18, BDGJ20, Jia20a] and our backward-leak uni-directional scheme. 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 $\Delta_{\mathsf{e}+1}$ and 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

[1] Everspaugh et al. [EPRS17b] 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 [ABPP14, HLL16]. In addition, Everspaugh et al. retracted the ciphertext-independent construction in their full version paper [EPRS17a].

in Δ_{e+1} [LT18, KLR19, BDGJ20, Jia20a]. In addition, generating a token Δ_{e+1} from sk_{e+1} and pk_e is unnatural since it is unlikely such Δ_{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. In fact, this is what we do in Section 6 to achieve a no-directional key updates scheme.

On meaningful combination with bi/uni-directional ciphertext updates. For ciphertext updates, it is natural to consider only the uni-directional ciphertext updates in Definition 3.7 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 so 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 Δ_{e+1} to convert a ciphertext under k_{e+1} into one under k_e in the backward-leak uni-directional key updates setting. This observation affects a theorem proved by Jiang [Jia20b, Theorem 3.2] (Theorem 4.6 in this paper), which we explain later.

Summary of observations. We summarize possible combinations for token generation in Figure 4. Based on the equivalence theorem by Jiang [Jia20a] (will appear in the next section), in the row of $\mathsf{TokGen}(\mathsf{pk}_e, \mathsf{sk}_{e+1})$, both fields must be "bi" and sk_e is essentially used in a token. This is natural since UE can update a ciphertext in the forward direction.

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	?

Figure 4: Possible combinations for token generation from pk or sk and its relationship to directions of key updates and ciphertext updates. In this table, we do not consider using obfuscation. 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 the case, it might not be able to update a ciphertext in the forward direction or it should be bi-directional. In the ciphertext update column, "backward-leak? or bi?" means that it can be backward, but it does not fit to the nature of UE and if it can be forward, it essentially has the power of bi-directional updates. Lastly, "?" means that we do not know whether this type can update a ciphertext or not.

4.2 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 4.2. *There exist secure UE schemes in the bi-directional key updates setting that are not secure in the backward-leak uni-directional key updates setting.*

On equivalence between bi-directional and uni-directional key updates. First, we review Jiang's equivalence theorem.

Theorem 4.3 ([Jia20a, 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* $\mathsf{kk}, \mathsf{cc}, \mathsf{kk}', \mathsf{cc}' \in \{\mathsf{uni}, \mathsf{f\text{-}uni}\}$ *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 (Theorem 4.3) for the confidentiality case is the following.

Lemma 4.4 ([Jia20a, Lemma 6]). *For any* $\mathcal{K}, \mathcal{T}, \mathcal{C}$*, we have* $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni},\mathsf{uni}} \neq \emptyset \Leftrightarrow \mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi},\mathsf{bi}} \neq \emptyset$.

See Definitions 3.7 and 4.1 for the sets in the lemma. Note that this lemma holds for *forward-leak* uni-directional key updates. We show an counterexample to this lemma (for confidentiality) in the case of the *backward-leak* uni-directional key updates setting.

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 Figure 5.

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

Figure 5: 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, × and ✓ 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.

In the example in Figure 5, 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 Figure 5.) 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, $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi}} = \{2,3,4,5\} \neq \emptyset$ and $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni}} = \{4,5\} \neq \emptyset$. 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 Lemma 4.4 (Jiang's Lemma 6 [Jia20a]). 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 $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \{2,3,4,5\} \neq \emptyset$ and $\mathcal{K}^*_{\mathsf{b\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{b\text{-}uni,uni}} = \{3,4\} \neq \emptyset$. However, if we set $\mathcal{C} = \{5\}$, the difference between forward/backward directional key updates is clear. Now, $\mathcal{K}^*_{\mathsf{bi}} \cap \mathcal{C}^*_{\mathsf{bi,bi}} = \{2,3,4,5\} \neq \emptyset$, but $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni,uni}} = \emptyset$ 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 Lemma 4.4 (Jiang's Lemma 6 [Jia20a]) when we use the definition of *backward-leak* uni-directional key updates.

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

Theorem 4.5. *The ciphertext-independent UE schemes by Everspaugh et al. [EPRS17b], Lehman and Tackmann [LT18], Boyd et al. [BDGJ20], and Jiang [Jia20a] do not satisfy confidentiality in the* backward-leak *uni-directional setting.*

Proof. We use the leakage sets example $\mathcal{K}$ and $\mathcal{T}$ in Figure 5 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 schemes described in the theorem statement (the schemes in the statement are bi-directional key updates schemes). Thus, the adversary trivially win the confidentiality game in the backward-leak uni-directional setting since a challenge ciphertext is encrypted under k_5. ■

By Theorem 4.5 and the results by Everspaugh et al [EPRS17b], Lehman and Tackmann [LT18], Boyd et al. [BDGJ20], and Jiang [Jia20a], we immediately obtain Theorem 4.2 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.

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 prove the following theorem.

Theorem 4.6 ([Jia20b, Theorem 3.2]). *Let* UE *be an UE scheme and* notion $\in \{$d-ind-ue-cpa, r-ind-ue-cpa, d-ind-ue-cca, r-ind-ue-cca$\}$. *For any* (no, bi)-notion *adversary* $\mathcal{A}$ *against* UE, *there exists a* (f-uni, bi)-notions *adversary* $\mathcal{B}$ *against* UE *such that*

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

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

The source of the puzzle above comes from the theorem holds for *bi-directional ciphertext* updates. The key lemma for proving Jiang's theorem above (Theorem 4.6) is the following.

Lemma 4.7 ([Jia20b, Lemma 3.15]). *For any* $\mathcal{K}, \mathcal{T}, \mathcal{C}$, *we have* $\mathcal{K}^*_{\mathsf{f\text{-}uni}} \cap \mathcal{C}^*_{\mathsf{f\text{-}uni,bi}} \neq \emptyset \Rightarrow \mathcal{K}^*_{\mathsf{no}} \cap \mathcal{C}^*_{\mathsf{no,bi}} \neq \emptyset$.

The proof of the lemma above heavily relies on the bi-directional ciphertext update setting. As we argued in Section 4.1, 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 setting, the equivalence theorem above (Theorem 4.6), which is counterintuitive, does not hold in the case of the backward-leak uni-directional key updates setting.

Extrusion area of firewall. As we see in the proof of Theorem 4.5, 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. In the example in Figure 5, epoch $\{5\}$ is such an area. We call it an extrusion area of firewall $[\mathsf{fwl}, \mathsf{fwr}]$ in the backward-leak uni-directional (and no-directional) key update setting.

Definition 4.8 (Extrusion Area). *An extrusion region with firewalls* fwl *and* fwr *is a consecutive sequence of epochs* $[\mathsf{eal}, \mathsf{fwl}-1]$ *for which:*

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

We can treat this area as firewall area in the backward-leak uni-directional (and no-directional) setting since we cannot convert a challenge ciphertext in this are into one under a corrupted key (due to backward-leak uni-directional (or no-directional) property). In addition, we cannot convert the challenge ciphertext in the extrusion area into one under keys in the firewall area since a token between these two areas is not given by the definition of the firewall.

Why does the direction matters? The question is: "What is the source of the difference between forward/backward-leak uni-directional key updates?" The point is that ciphertext updates can always go forward, but cannot necessarily go backward. Look at the example in Figure 5 again. If we set $\mathcal{C} = \{3\}$ in the forward-leak uni-directional setting, we can go forward by using corrupted tokens Δ_4, Δ_5. Thus, even if epoch 3 is not corrupted, we can convert a ciphertext under k_3 into one under k_5, which is corrupted. However, in the backward-leak uni-directional setting, if we set $\mathcal{C} = \{5\}$, we cannot go backward even if we use Δ_4, Δ_5. Note that we focus on the uni-directional ciphertext updates setting here. This asymmetry incurs the difference.

Based on this observation, if we consider "backward uni-directional ciphertext updates", a similar equivalence result to Jiang's one holds even in the backward-leak uni-directional key updates setting. However, note that the backward uni-directional ciphertext updates setting is quite unnatural as we discussed in Section 4.1.

5 Construction: Backward-Leak Uni-Directional Key Update

We present a backward-leak uni-directional key update scheme from the LWE assumption.

5.1 Lattice Preliminaries

First, we introduce basic notations, notions, and tools for lattice-based cryptography.

Notations. We say that a distribution over $\mathbb{R}$ is B-bounded if a sample from the distribution is in $[-B, B]$ with overwhelming probability. For $x \in \mathbb{R}$, let $\lfloor x \rceil := \lceil x - 1/2 \rceil$. For $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{R}^\ell$, let $\lfloor \boldsymbol{x} \rceil := (\lfloor x_1 \rceil, \ldots, \lfloor x_\ell \rceil) \in \mathbb{Z}^\ell$. For any integer $q \geq 2$, we write $\mathbb{Z}_q$ for the ring $\{\lceil -q/2 \rceil, \ldots, -1, 0, 1, \ldots, \lfloor q/2 \rfloor\}$ with addition and multiplication modulo q.

For two matrices $\boldsymbol{X} \in \mathbb{R}^{m \times n_1}$ and $\boldsymbol{Y} \in \mathbb{R}^{m \times n_2}$, $[\boldsymbol{X} \mid \boldsymbol{Y}] \in \mathbb{R}^{m \times (n_1 + n_2)}$ denotes the concatenation of the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$. For two matrices $\boldsymbol{X} \in \mathbb{R}^{m_1 \times n}$ and $\boldsymbol{Y} \in \mathbb{R}^{m_2 \times n}$, $[\boldsymbol{X}; \boldsymbol{Y}] \in \mathbb{R}^{(m_1 + m_2) \times n}$ denotes the concatenation of the rows of $\boldsymbol{X}$ and $\boldsymbol{Y}$. For a vector $\boldsymbol{x} \in \mathbb{R}^m$, $\|\boldsymbol{x}\|_p$ denotes the ℓ_p norm of $\boldsymbol{x}$. We omit subscript if $p = 2$ for simplicity.

Distributions. We review the basic definitions of the distributions in lattice-based cryptography. Let $N(0,\sigma^2)$ denote the Gaussian distribution whose mean is 0 and variance is σ^2. The Gaussian distribution is defined by density function $(1/\sigma\sqrt{2\pi})\cdot\exp(-x^2/2\sigma^2)$ over $\mathbb{R}$. We also define the followings.

- *Discretized* Gaussian $\bar{\Psi}_\alpha$: For $\alpha\in(0,1)$ and a positive integer q, we sample x from $N(0,\alpha^2/2\pi)$ and output $\lfloor qx\rceil \bmod q$.
- *Discrete* Gaussian: For a positive real s, the n-dimensional Gaussian function is defined as $\rho_s(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2/s^2)$.
- Discrete Gaussian distribution $D_{A,s}$: For a positive real s and a countable set A, $D_{A,s}(\boldsymbol{x}) := \frac{\rho_s(\boldsymbol{x})}{\sum_{\boldsymbol{y}\in A}\rho_s(\boldsymbol{y})}$.

Gentry, Peikert, and Vaikuntanathan [GPV08] gave an efficient sampler, $\mathsf{SampleD}$, for $D_{\mathbb{Z},s}$.

We use the following statistical properties in this paper.

Lemma 5.1 ([Ban93, Lemma 1.5], [LP11, Lemma 2.1]). *Let $c\geq 1$, $C = c\cdot\exp((1-c^2)/2)$. For any real $s>0$ and any integer $n\geq 1$, we have that*

$$\Pr_{\boldsymbol{e}\leftarrow D_{\mathbb{Z}^n,s}}\left[\|\boldsymbol{e}\|_2 \geq cs\sqrt{n/(2\pi)}\right] \leq C^n.$$

In particular, letting $c=\sqrt{2\pi}$ and $C<1/4$, we have that $\Pr_{\boldsymbol{e}\leftarrow D_{\mathbb{Z}^n,s}}\left[\|\boldsymbol{e}\|_2\geq s\sqrt{n}\right] < 2^{-2n}$.

Lemma 5.2 ([AJW11, Lemma 2.1 (smudging lemma)]). *Let $B_1 = B_1(\lambda)$, and $B_2 = B_2(\lambda)$ be positive integers and let $e_1\in[-B_1,B_1]$ be a fixed integer. Let $e_2\leftarrow[-B_2,B_2]$ be chosen uniformly at random. If $B_1/B_2 = negl(\lambda)$, the distribution of e_2 is statistically indistinguishable from that of e_2+e_1.*

Lemma 5.3 (Adapted version of the leftover hash lemma). *Let q be an odd prime. Let D be a distribution over $\mathbb{Z}_q^m$ of min-entropy at least $(n+\ell)\lg q + g(n)$. Then, we have that $\Delta((\boldsymbol{A},\boldsymbol{eA}),(\boldsymbol{A},\boldsymbol{u}^\top)) \leq 2^{-g(n)/2}$, where $\boldsymbol{A}\leftarrow\mathbb{Z}_q^{m\times(n+\ell)}$, $\boldsymbol{e}\leftarrow D$, and $\boldsymbol{u}\leftarrow\mathbb{Z}_q^{n+\ell}$*

This lemma holds if we use $\boldsymbol{E}\leftarrow D^k$ and $\boldsymbol{U}\leftarrow\mathbb{Z}_q^{k\times(n+\ell)}$ instead of $\boldsymbol{e}$ and $\boldsymbol{u}$ where $k=\mathrm{poly}(n)$.

5.2 Learning with Errors

Regev introduced the LWE problem [Reg09]. Let $A(\boldsymbol{s},\chi)$ be a distribution over $\mathbb{Z}_q^n\times\mathbb{Z}_q$ defined as follows. For a vector $\boldsymbol{s}\in\mathbb{Z}_q^n$ and a distribution χ over $\mathbb{Z}_q$, we sample $\boldsymbol{a}\leftarrow\mathbb{Z}_q^n$ and $x\leftarrow\chi$, and output $(\boldsymbol{a},\langle\boldsymbol{a},\boldsymbol{s}\rangle+x)$.

Definition 5.4 (The LWE problem and assumption). *For an integer $q=q(n)$, and distributions χ over $\mathbb{Z}_q$ and ψ over $\mathbb{Z}_q^n$, the learning with errors problem, $\mathrm{LWE}(n,q,\chi)$ for the distribution ψ, is distinguishing oracle $A(\boldsymbol{s},\chi)$ from oracle $A(\boldsymbol{s},U(\mathbb{Z}_q))$, where $\boldsymbol{s}\leftarrow\psi$. We say the $\mathrm{LWE}(n,q,\chi)$ assumption holds for ψ if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{\mathsf{lwe}}_{\mathcal{A},(n,q,\chi,\psi)}(n) = \left|\Pr[\mathcal{A}^{A(\boldsymbol{s},\chi)}(1^n)=1 \mid \boldsymbol{s}\leftarrow\psi] - \Pr[\mathcal{A}^{A(\boldsymbol{s},U(\mathbb{Z}_q))}(1^n)=1 \mid \boldsymbol{s}\leftarrow\psi]\right|$$

is negligible in n where $\boldsymbol{s}\leftarrow\psi$.

We note that $A(\boldsymbol{s},U(\mathbb{Z}_q)) = U(\mathbb{Z}_q^n\times\mathbb{Z}_q)$ for any $\boldsymbol{s}\in\mathbb{Z}_q^n$. We also note that even if we use matrices $\boldsymbol{A}\leftarrow\mathbb{Z}_q^{m\times n}$ and $\boldsymbol{S}\in\mathbb{Z}_q^{n\times\ell}$ instead of vectors $\boldsymbol{a}$ and $\boldsymbol{s}$, respectively, the assumption holds (if the original assumption holds) by a simple hybrid argument. In that case, we omit parameter m,ℓ from the notation for simplicity. Applebaum, Cash, Peikert, and Sahai [ACPS09] showed that we can use χ^n instead of $U(\mathbb{Z}_q^n)$ for the distribution of $\boldsymbol{s}$.

Theorem 5.5 (Adapted version of [ACPS09, Lemma 2]). *Let $q = p^e$ be a prime power. If the* $\mathrm{LWE}(n, q, \chi)$ *assumption holds for* $U(\mathbb{Z}_q^n)$*, then the* $\mathrm{LWE}(n, q, \chi)$ *assumption holds for* χ^n.

Solving the LWE problem with $\chi = \bar{\Psi}_\alpha$ or $D_{\mathbb{Z},s}$ *on average* is as hard as the *worst* case of the approximation version of the shortest independent vector problem, SIVP_γ, and the decision version of the shortest vector problem, GapSVP_γ, under a classical/quantum reduction, where γ is an approximation factor [Reg09, Pei09, BLP+13].

5.3 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 [BV11a, BV11b, BGV12]. In particular, we use that for multi-bit plaintexts [BGH13]. In the following, we denote a plaintext by $\boldsymbol{\mu} \in \{0,1\}^\ell$.

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

The key-switching technique. We review the key-switching technique in the multi-bit version for our update algorithm. Let $\eta = \lceil \lg q \rceil$. 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 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 an 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

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

To decrypt ciphertext by secret key S_{e+1}, we compute

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

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

backward-leak uni-directional update. In fact, we do not need the secret key S_{e+1} at epoch $e+1$ for update. We set $B_{e+1} = AS_{e+1} + Y_{e+1}$, which we call the public key part of the key at epoch $e+1$. We choose $R_{e+1} \leftarrow \{-1,+1\}^{n\eta\times m}$ and compute an update token

$$\begin{aligned} M_{e+1} &= R_{e+1}[A \mid B_{e+1}] + [O \mid -\mathsf{P2}(S_e)] \\ &= [A' \mid A'S_{e+1} + Y'] + [O \mid -\mathsf{P2}(S_e)], \end{aligned}$$

where $A' = R_{e+1}A$ and $Y' = R_{e+1}Y_j$. By using M_{e+1}, we can update ciphertext (u,c) at epoch e. Thus, even if given the key S_e at epoch e and the token M_{e+1}, we cannot infer S_{e+1} since only the public key part B_{e+1} (this is pseudorandom by the LWE assumption) of the key at epoch $e+1$ is embedded in M_{e+1}. Note that S_e and S_{e+1} are independently chosen. However, if given the key S_{e+1} at epoch $e+1$ and the token M_{e+1}, we can easily infer S_e since S_e is encrypted under S_{e+1}. Thus, this update mechanism is backward-leak uni-directional.

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

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

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

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

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

1. $(B_e, S_e) \leftarrow \mathsf{Reg.Gen}(1^\lambda)$.
2. Output $\mathsf{k}_e := (\mathsf{sk}_e, \mathsf{pk}_e) := (S_e, B_e)$.

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

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

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

1. Parse $\mathsf{k}_e = (S_e, B_e)$ $\mathsf{ct} = (u,c)$.

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 $(\boldsymbol{u}', \boldsymbol{c}') := \mathsf{BD}(\boldsymbol{u})\boldsymbol{M}_{\mathsf{e}+1}$;
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 $\mathsf{ct}_{\mathsf{e}+1} := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) := (\boldsymbol{u}' + \tilde{\boldsymbol{u}}, \boldsymbol{c} + \boldsymbol{c}' + \tilde{\boldsymbol{v}})$.

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 5.4 and 5.5. Let T be the maximum number of the epoch.

Theorem 5.6. *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 $(1 + n\eta + m)mB + B' \le q/4T$. Then* RtR *is correct.*

Theorem 5.7. *Suppose that $m \ge (n+\ell)\lg q + \omega(\lg \lambda)$. Under the* $\mathrm{LWE}(n, q, \chi)$ *assumption,* RtR *is r-IND-UE-CPA secure.*

5.4 Correctness

We give rough estimations on B-bounded and B' 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 Theorem 5.6. The theorem follows from Propositions 5.8 and 5.9 below. ■

Proposition 5.8. *The scheme is correct for the encryption algorithm if $mB + B' < q/4$.*

The correctness easily follows from the proof by Regev [Reg09]. For completeness, we include the proof.

Proof. Let $(\boldsymbol{S}, \boldsymbol{B}) = (\boldsymbol{S}, \boldsymbol{A}\boldsymbol{S} + \boldsymbol{X})$, where $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n \times \ell}$ and $\boldsymbol{X} \leftarrow \chi^{m \times \ell}$ be a key k_{e} of epoch e. To decrypt a ciphertext $(\boldsymbol{u}, \boldsymbol{c}) = (\boldsymbol{r}\boldsymbol{A}, \boldsymbol{r}\boldsymbol{B} + \lfloor q/2 \rfloor \boldsymbol{\mu})$ of the message $\boldsymbol{\mu} \in \{0,1\}^\ell$ with $\boldsymbol{S}$ under k_{e}, we compute

$$\begin{aligned} \boldsymbol{d} = \boldsymbol{c} - \boldsymbol{u}\boldsymbol{S} &= \lfloor q/2 \rfloor \boldsymbol{\mu} + \boldsymbol{e}' + \boldsymbol{r}\boldsymbol{B} - \boldsymbol{r}\boldsymbol{A}\boldsymbol{S} \\ &= \lfloor q/2 \rfloor \boldsymbol{\mu} + \boldsymbol{e}' + \boldsymbol{r}\boldsymbol{X}. \end{aligned}$$

If $\|\boldsymbol{e}'\|_\infty + \|\boldsymbol{r}\boldsymbol{X}\|_\infty < q/4$, we can obtain $\boldsymbol{\mu} = \lfloor 2\boldsymbol{d}/q \rceil \bmod 2$. Since $\boldsymbol{r} \in \{-1,+1\}^m$ and $\boldsymbol{X}$ consists of samples from a B-bounded distribution, $\|\boldsymbol{r}\boldsymbol{X}\|_\infty$ is at most mB. In addition, $\boldsymbol{e}'$ are from a B'-bounded distribution, $\|\boldsymbol{e}'\|_\infty$ is at most B'. From the parameter setting $(1 + n\eta + m)mB + B' \le q/4T$, we have $\|\boldsymbol{e}' + \boldsymbol{r}\boldsymbol{X}\|_\infty \le \|\boldsymbol{e}'\|_\infty + \|\boldsymbol{r}\boldsymbol{X}\|_\infty \le mB + B' < q/4T$. This completes the proof. ■

Proposition 5.9. *The scheme is correct for the update algorithm if* $(1+n\eta+m)mB+B' < q/4T$.

Proof. Let $\boldsymbol{B}_{\mathsf{e}} = \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}} + \boldsymbol{X}_{\mathsf{e}}$ and $\boldsymbol{B}_{\mathsf{e}+1} = \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{X}_{\mathsf{e}+1}$ be public keys of epoch e and $\mathsf{e}+1$, respectively, where $\boldsymbol{S}_{\mathsf{e}}, \boldsymbol{S}_{\mathsf{e}+1} \leftarrow \mathbb{Z}_q^{n\times\ell}$ and $\boldsymbol{X}_{\mathsf{e}}, \boldsymbol{X}_{\mathsf{e}+1} \leftarrow \chi^{m\times\ell}$. The update token from e to $\mathsf{e}+1$ is generated as

$$\boldsymbol{M}_{\mathsf{e}+1} = \boldsymbol{R}_{\mathsf{e}+1}[\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}] + [\boldsymbol{O} \mid -\mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})] \text{ and } \boldsymbol{N}_{\mathsf{e}+1} = \boldsymbol{R}'_{\mathsf{e}+1} \cdot [\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}+1}],$$

where $\boldsymbol{R}_{\mathsf{e}+1} \leftarrow \{-1,+1\}^{n\eta\times m}$ and $\boldsymbol{R}'_{\mathsf{e}+1} \leftarrow \{-1,+1\}^{m\times m}$. We consider a ciphertext $(\boldsymbol{u},\boldsymbol{c}) = \boldsymbol{r}[\boldsymbol{A} \mid \boldsymbol{B}_{\mathsf{e}}] + (\boldsymbol{0}, \lfloor q/2 \rfloor \boldsymbol{\mu})$ of the message $\boldsymbol{\mu} \in \{0,1\}^{\ell}$, and an updated ciphertext $(\bar{\boldsymbol{u}},\bar{\boldsymbol{c}}) = (\boldsymbol{u}'+\tilde{\boldsymbol{u}}, \boldsymbol{c}+\boldsymbol{c}'+\tilde{\boldsymbol{c}}) = (\boldsymbol{0},\boldsymbol{c}) + \mathsf{BD}(\boldsymbol{u})\cdot\boldsymbol{M}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}}\cdot\boldsymbol{N}_{\mathsf{e}+1}$, where $\tilde{\boldsymbol{r}} \leftarrow \{-1,+1\}^m$.

The decryption algorithm $\mathsf{Dec}(\boldsymbol{S}_{\mathsf{e}+1}, (\bar{\boldsymbol{u}},\bar{\boldsymbol{c}}))$ computes

$$\begin{aligned}
\boldsymbol{d} &= \bar{\boldsymbol{c}} - \bar{\boldsymbol{u}}\boldsymbol{S}_{\mathsf{e}+1} \\
&= \boldsymbol{c} + \mathsf{BD}(\boldsymbol{u})\boldsymbol{M}_{\mathsf{e}+1}[-\boldsymbol{S}_{\mathsf{e}+1}; \boldsymbol{I}_\ell] + \tilde{\boldsymbol{r}}\boldsymbol{N}_{\mathsf{e}+1}[-\boldsymbol{S}_{\mathsf{e}+1}; \boldsymbol{I}_\ell] \\
&= \boldsymbol{c} + \mathsf{BD}(\boldsymbol{u})\left(\boldsymbol{R}_{\mathsf{e}+1}(\boldsymbol{B}_{\mathsf{e}+1} - \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}+1}) - \mathsf{P2}(\boldsymbol{S}_{\mathsf{e}})\right) + \tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1}(\boldsymbol{B}_{\mathsf{e}+1} - \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}+1}) \\
&= \boldsymbol{c} - \boldsymbol{u}\boldsymbol{S}_{\mathsf{e}} + (\mathsf{BD}(\boldsymbol{u})\boldsymbol{R}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1})\boldsymbol{X}_{\mathsf{e}+1} \\
&= \lfloor q/2 \rfloor \boldsymbol{\mu} + \underbrace{\boldsymbol{e}' + \boldsymbol{r}\boldsymbol{X}_{\mathsf{e}} + (\mathsf{BD}(\boldsymbol{u})\boldsymbol{R}_{\mathsf{e}+1} + \tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1})\boldsymbol{X}_{\mathsf{e}+1}}_{=:\delta},
\end{aligned}$$

where we used the equation $\boldsymbol{B}_{\mathsf{e}+1} = \boldsymbol{A}\boldsymbol{S}_{\mathsf{e}+1} + \boldsymbol{X}_{\mathsf{e}+1}$. As in the proof of Proposition 5.8, if $\|\delta\|_\infty < q/4$, we can correctly decrypt the update ciphertext into $\boldsymbol{\mu}$ by $\boldsymbol{S}_{\mathsf{e}+1}$. We have that following bounds by routine calculation:

$$\left\|\boldsymbol{e}'\right\|_\infty \le B', \|\boldsymbol{r}\boldsymbol{X}_{\mathsf{e}}\|_\infty \le mB, \|\mathsf{BD}(\boldsymbol{u})\boldsymbol{R}_{\mathsf{e}+1}\boldsymbol{X}_{\mathsf{e}+1}\|_\infty \le n\eta\cdot mB, \text{ and } \|\tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1}\boldsymbol{X}_{\mathsf{e}+1}\|_\infty \le m^2 B.$$

From the triangle inequality and the parameter setting, we have that

$$\begin{aligned}
\|\delta\|_\infty &\le \left\|\boldsymbol{e}'\right\|_+ \|\boldsymbol{r}\boldsymbol{X}_{\mathsf{e}}\|_\infty + \|\mathsf{BD}(\boldsymbol{u})\boldsymbol{R}_{\mathsf{e}+1}\boldsymbol{X}_{\mathsf{e}+1}\|_\infty + \|\tilde{\boldsymbol{r}}\boldsymbol{R}'_{\mathsf{e}+1}\boldsymbol{X}_{\mathsf{e}+1}\|_\infty \\
&\le B' + mB + n\eta mB + m^2 B = (1+n\eta+m)mB + B' \\
&< q/4T \le q/4.
\end{aligned}$$

Thus, when we update a ciphertext once, a noise is added and its magnitude is at most $q/4T$. This means that we can update a ciphertext T times. We complete the proof. ■

5.5 Confidentiality

We show RtR is r-IND-UE-CPA. First, we look at the detail of the update procedure again. By simple calculation, we obtain

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

The last equation (statistical indistinguishability) holds by Lemma 5.2. This equation shows that we can simulate an update ciphertext from 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.

$\mathsf{Hyb.Upd}(\mathsf{ct}_e, \boldsymbol{B}_{e+1}, \boldsymbol{\mu}; \boldsymbol{e}'_e, (\boldsymbol{R}_{e+1}, \boldsymbol{R}'_{e+1}))$**:**

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

By Equation (1), we can simulate $\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_e)$ by using $\mathsf{Hyb.Upd}(\mathsf{ct}_e, \boldsymbol{B}_{e+1}, \boldsymbol{\mu}; \boldsymbol{e}'_e, (\boldsymbol{R}_{e+1}, \boldsymbol{R}'_{e+1}))$.

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

- Choose $\boldsymbol{B}_e^+ \leftarrow \mathbb{Z}_q^{m \times \ell}$.
- Output $\mathsf{pk}_e := \boldsymbol{B}_e^+$.

$\mathsf{Sim.TokGen}(\mathsf{pp})$**:**

- Choose $\boldsymbol{M}_{e+1}^+ \leftarrow \mathbb{Z}_q^{n\eta \times (n+\ell)}$ and $\boldsymbol{N}_{e+1}^+ \leftarrow \mathbb{Z}_q^{m \times (n+\ell)}$.
- Output $\Delta_{e+1}^+ := (\boldsymbol{M}_{e+1}^+, \boldsymbol{N}_{e+1}^+)$.

$\mathsf{Sim.Upd}(\mathsf{ct}_e)$**:**

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

$\mathsf{Sim.Enc}(\mathsf{pp})$**:**

- Choose $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.
- Output $\mathsf{ct}_e := (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}})$.

We follow the firewall technique [LT18, KLR19, BDGJ20, Jia20a] to prove security.

Proof of Theorem 5.7. 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 $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda, b)$ 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 $\mathsf{coin} = 0$, $\mathsf{UE.Enc}(\mathsf{k}_{\widetilde{e}}, \overline{\mu})$ else $\mathsf{UE.Upd}(\Delta_{\widetilde{e}}, \overline{\mathsf{ct}})$.
- 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 $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda, b)$.

Let $U(\lambda)$ be a random variable distributed uniformly in $[0,T]$, we have

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{RtR},\mathcal{A}}(\lambda)\\
&= |\Pr[\mathsf{Hyb}_{T+1}(1)=1] - \Pr[\mathsf{Hyb}_{T+1}(0)=1]|\\
&= \Bigg|\sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{i+1}(1)=1] - \Pr[\mathsf{Hyb}_{i}(1)=1])\\
&\quad + \sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{i+1}(0)=1] - \Pr[\mathsf{Hyb}_{i}(0)=1])\Bigg|\\
&= \Bigg|\sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{U(\lambda)+1}(1)=1 \mid U(\lambda)=i] - \Pr[\mathsf{Hyb}_{i}(1)=1 \mid U(\lambda)=i])\\
&\quad + \sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{i+1}(0)=1 \mid U(\lambda)=i] - \Pr[\mathsf{Hyb}_{i}(0)=1 \mid U(\lambda)=i])\Bigg|\\
&= (T+1)\Bigg|\sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{U(\lambda)+1}(1)=1 \wedge U(\lambda)=i] - \Pr[\mathsf{Hyb}_{U(\lambda)}(1)=1 \wedge U(\lambda)=i]) \qquad (2)\\
&\quad + \sum_{i=0}^{T}(\Pr[\mathsf{Hyb}_{U(\lambda)+1}(0)=1 \wedge U(\lambda)=i] - \Pr[\mathsf{Hyb}_{U(\lambda)}(0)=1 \wedge U(\lambda)=i])\Bigg|\\
&\leq (T+1)|\Pr[\mathsf{Hyb}_{U(\lambda)+1}(1)=1] - \Pr[\mathsf{Hyb}_{U(\lambda)}(1)=1]|\\
&\quad + (T+1)|\Pr[\mathsf{Hyb}_{U(\lambda)+1}(0)=1] - \Pr[\mathsf{Hyb}_{U(\lambda)}(0)=1]|,
\end{aligned}
$$

where we use $\Pr[U(\lambda)=i] = 1/(T+1)$ for Equation (2). 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 $j \in [\mathsf{fwl}, \mathsf{fwr}]$, Δ_{fwl}, or $\Delta_{\mathsf{fwr}+1}$, the game aborts.

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

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

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 Theorem 5.7. ■

Lemma 5.10. *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 $\perp$

- $(\mathsf{ct}_\mathsf{e}, \boldsymbol{e}'_\mathsf{e}) \leftarrow \mathsf{Hyb.Upd}(\mathsf{ct}_{\mathsf{e}-1}, \boldsymbol{B}_\mathsf{e}, \boldsymbol{\mu}; \boldsymbol{e}'_{\mathsf{e}-1}, (\boldsymbol{R}_\mathsf{e}, \boldsymbol{R}'_\mathsf{e}))$.
- $\mathcal{L} \coloneqq \mathcal{L} \cup \{(\cdot, \mathsf{ct}_\mathsf{e}, \mathsf{e}; \boldsymbol{e}'_\mathsf{e}, \boldsymbol{\mu})\}$.

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. Thus, by Equation (1), it is easy to see the following proposition holds.

Proposition 5.11. $|\Pr[\mathsf{Hyb}'_i(b) = 1] - \Pr[\mathsf{Hyb}^\mathsf{r}_i(b) = 1]| \leq \mathsf{negl}(\lambda)$.

$\mathsf{G}_j(i,b)$**:** 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.

We note that $j \in [i, \mathsf{fwr}+1]$ and i is fixed. By the definition, we have

$$\mathsf{G}_{\mathsf{fwr}+1}(i+1,b) = \mathsf{Hyb}^\mathsf{r}_{i+1}(b) \text{ and } \mathsf{G}_{\mathsf{fwr}+1}(i,b) = \mathsf{Hyb}^\mathsf{r}_i(b). \tag{3}$$

We prove that

$$|\Pr[\mathsf{G}_{j+1}(i+1,b) = 1] - \Pr[\mathsf{G}_j(i+1,b) = 1]| \leq \mathsf{negl}(\lambda) \text{ for } j \in [i, \mathsf{fwr}] \tag{4}$$
$$|\Pr[\mathsf{G}_i(i+1,b) = 1] - \Pr[\mathsf{G}_i(i,b) = 1]| \leq \mathsf{negl}(\lambda) \tag{5}$$
$$|\Pr[\mathsf{G}_{j+1}(i,b) = 1] - \Pr[\mathsf{G}_j(i,b) = 1]| \leq \mathsf{negl}(\lambda) \text{ for } j \in [i, \mathsf{fwr}]. \tag{6}$$

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 Proposition 5.11 and Equation (3), we obtain what we want to prove (Lemma 5.10). Thus, all we must do is proving Equations (4) to (6).

First, we prove Equation (4). 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 $\Delta_j \coloneqq (\boldsymbol{M}^+_j, \boldsymbol{N}^+_j) \leftarrow \mathbb{Z}_q^{n\eta \times (n+\ell)} \times \mathbb{Z}_q^{m \times (n+\ell)}$ 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 Equation (4).

Proposition 5.12. *It holds that* $|\Pr[\mathsf{Game}\text{-}1(b) = 1] - \Pr[\mathsf{G}_{j+1}(i+1,b) = 1]| \leq \mathsf{Adv}^{\mathsf{lwe}}_{\mathcal{A},(n,q,\chi,\psi)}(n)$

Proposition 5.13. *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 Equation (5). 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 5.14. *It holds that* $|\Pr[\mathsf{G}_i(i+1,b)=1]-\Pr[\mathsf{G}_i(i,b)=1]| \le \mathsf{negl}(\lambda)$.

Lastly, we prove Equation (6). Once the challenge-equal ciphertext at epoch i becomes random, we go back to games where public keys and tokens are real. These backward transitions are possible by using the proof of Equation (4) in a reverse manner. Thus, we complete the proof of Lemma 5.10 if we prove Propositions 5.12 to 5.14. We write those proofs below. We summarize how public keys, update tokens, and challenge-equal ciphertexts at epoch i are generated in Figure 5. ■

	$\mathsf{G}_{i+1}(i+1,b)$	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	$\mathsf{TokGen}(\mathsf{sk}_{i-1},\mathsf{pk}_i)$	$\mathsf{TokGen}(\mathsf{sk}_{i-1},\mathsf{pk}_i)$	$\mathsf{Sim.TokGen}(\mathsf{pp})$	$\mathsf{Sim.TokGen}(\mathsf{pp})$
$\mathsf{ct}^*_{i,1}$	$\mathsf{Upd}(\Delta_i,\mathsf{ct}_{i-1})$	$\mathsf{Upd}(\Delta_i,\mathsf{ct}_{i-1})$	$\mathsf{Upd}(\Delta_i^+,\mathsf{ct}_{i-1})$	$\mathsf{Sim.Upd}(\mathsf{ct}_{i-1})$
$\mathsf{ct}^*_{i,0}$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Enc}(\mathsf{pk}_i,\overline{\mu}_0)$	$\mathsf{Sim.Enc}(\mathsf{pp})$

Figure 6: 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 Propositions 5.12 to 5.14.

Proof of Proposition 5.12. 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 $\boldsymbol{B}_j := \boldsymbol{B}$. 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. 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 $j' \in [\mathsf{fwl},j]$. 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 tokes 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 $\boldsymbol{B}=\boldsymbol{A}\boldsymbol{S}+\boldsymbol{X}$ 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 Proposition 5.13. The difference between these two games is as follows:

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

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

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 (Lemma 5.3) and these differences are statistically indistinguishable. This completes the proof. ■

Proof of Proposition 5.14. 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 $\tilde{\boldsymbol{r}} \leftarrow \{-1,+1\}^m$.

$\mathsf{G}_i(i,1)$: $\mathsf{ct}^*_{i,1} = (\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}})$:

$$(\bar{\boldsymbol{u}}, \bar{\boldsymbol{c}}) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell.$$

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 (Lemma 5.3) 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 (Lemma 5.3) and these differences are statistically indistinguishable. This completes the proof. ■

6 Construction: No-Directional Key Update

6.1 Scheme Description

We present a no-directional key update scheme $\mathsf{UE}_{\mathsf{io}}$ from puncturable PRFs and IO.

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

- 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 $\Delta_{e+1} \leftarrow i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k_e}, \mathsf{k_{e+1}}])$ where circuit $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}$ is described in Figure 7.

$\mathsf{Enc}(\mathsf{k_e}, \mu)$:

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

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

- Parse $\Delta_{\mathsf{e}+1} = i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k}_\mathsf{e}, \mathsf{k}_{\mathsf{e}+1}])$ and choose $r_{\mathsf{e}+1} \leftarrow \{0,1\}^\tau$.
- Compute and output $(t, c) := i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k}_\mathsf{e}, \mathsf{k}_{\mathsf{e}+1}])(\mathsf{ct}_\mathsf{e}, r_{\mathsf{e}+1})$.

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

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 $\mathsf{ct}_{\mathsf{e}+1} := (t', y' \oplus \mu')$.

Figure 7: The description of $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}$

6.2 Correctness

It is easy to see the $\mathsf{UE}_{\mathsf{io}}$ satisfies the correctness. The decryption algorithm computes $y \oplus \mathsf{PRF}(\mathsf{K}, t)$ from a ciphertext $\mathsf{ct} = (t, y) = (t, \mathsf{PRF}(\mathsf{K}, t))$. Thus, $\mu' = \mathsf{PRF}(\mathsf{K}, t) \oplus \mu \oplus \mathsf{PRF}(\mathsf{K}, t) = \mu$.

The output of $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct}_\mathsf{e})$ is completely the same as $\mathsf{Enc}(\mathsf{k}_{\mathsf{e}+1}, \mu)$ by the definition of $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k}_\mathsf{e}, \mathsf{k}_{\mathsf{e}+1}]$. Thus, the correctness for updates also holds.

6.3 Confidentiality

Theorem 6.1. *If $i\mathcal{O}$ is secure IO,* PRG *is a secure PRG, and* PRF *is a punctured PRF,* $\mathsf{UE}_{\mathsf{io}}$ *is r-IND-UE-CPA secure.*

Proof of Theorem 6.1. Let T be the upper bound of the number of epoch. We consider a similar sequence of hybrid games to those in Section 5.5 at the early stage of this proof. In fact, we use the same hybrid games.

$\mathsf{Hyb}_i(b)$**:** This is the same as $\mathsf{Exp}^{\mathsf{r\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE}_{\mathsf{io}},\mathcal{A}}(\lambda, b)$ 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 $\mathsf{coin} = 0$, $\mathsf{UE.Enc}(\mathsf{k}_{\widetilde{\mathsf{e}}}, \overline{\mu})$ else $\mathsf{UE.Upd}(\Delta_{\widetilde{\mathsf{e}}}, \overline{\mathsf{ct}})$.
- for $j \geq i$, return a random ciphertext.

$\mathsf{Hyb}'_i(b)$**:** 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 $j \in [\mathsf{fwl}, \mathsf{fwr}]$, Δ_{fwl}, or $\Delta_{\mathsf{fwr}+1}$, the game aborts.

Our goal is to prove $|\Pr[\mathsf{Hyb}'_{U(\lambda)+1}(b) = 1] - \Pr[\mathsf{Hyb}'_{U(\lambda)}(b) = 1]| \leq \mathsf{negl}(\lambda)$ as in the proof Theorem 5.7 in Section 5.5. ■

Hereafter, we write $\mathsf{Hyb}'_i(b)$ instead of $\mathsf{Hyb}'_{U(\lambda)}(b)$ for simplicity.

Lemma 6.2. *If there exists a secure IO and a punctured PRF, it holds that* $|\Pr[\mathsf{Hyb}'_{i+1}(b) = 1] - \Pr[\mathsf{Hyb}'_i(b) = 1]| \leq \mathsf{negl}(\lambda)$.

Hereafter, the proof deviates from the proof of Theorem 5.7. We note that we can simulate $\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_\mathsf{e})$ by using recorded plaintexts and key $\mathsf{K}_{\mathsf{e}+1}$ since the update algorithm outputs $\mathsf{Enc}(\mathsf{K}_{\mathsf{e}+1}, \mu)$ as we see in Section 6.2.

Proof. We define a sequence of games.

Game-0**:** This game chooses a random coin $\mathsf{coin} \leftarrow \{0,1\}$. If $\mathsf{coin} = 0$, it simulates $\mathsf{Hyb}'_i(b)$. Else if $\mathsf{coin} = 1$, it simulates Hyb'_{i+1}. That is, the game sends a real challenge-equal ciphertext for the challenge query at epoch i if $\mathsf{coin} = 0$ and a uniformly random ciphertext at epoch i. We define an event E_x as the adversary correctly guess coin in Game-x.

Game-1**:** This is the same as Game-0 except that we modify the challenge-equal ciphertext at epoch i for $\mathsf{coin} = 0$. It chooses $t^*_i \leftarrow \{0,1\}^n$ instead of using $t^*_i := \mathsf{PRG}(r^*_i)$.

Game-2**:** This is the same as Game-1 except that we modify the token generation algorithm for token Δ_i. It uses a punctured key $\mathsf{K}_i\{t^*_i\} \leftarrow \mathsf{Punc}(\mathsf{K}_i, t^*_i)$ and $\mathsf{FP}^i[\mathsf{K}_{i-1}, \mathsf{K}_i\{t^*_i\}]$ instead of $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k}_{i-1}, \mathsf{k}_i]$. The description of FP^i is given in Figure 8.

Game-3**:** This is the same as Game-2 except that we modify the token generation algorithm for token Δ_{i+1}. It uses the punctured key $\mathsf{K}_i\{t^*_i\} \leftarrow \mathsf{Punc}(\mathsf{K}_i, t^*_i)$ and $\mathsf{BP}^i[\mathsf{K}_i\{t^*_i\}, \mathsf{K}_{i+1}, t^*_i, y^*_i]$ instead of $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}[\mathsf{k}_i, \mathsf{k}_{i+1}]$. The description of BP^i is given in Figure 9.

Game-4**:** This is the same as Game-3 except that we modify the challenge-equal ciphertext at epoch i for $\mathsf{coin} = 0$. It chooses $y^*_i \leftarrow \{0,1\}^m$ and answers (t^*_i, y^*_i) for the challenge query at epoch i.

Forward Punctured Hybrid Update Function $\mathsf{FP}^j[\mathsf{K}_{j-1}, \mathsf{K}_j\{t^*_j\}](\mathsf{ct}_{j-1}, r_j)$

Hardwired: $\mathsf{K}_{j-1}, \mathsf{K}_j\{t^*_j\}$.

Input: A ciphertext ct_{j-1} and randomness $r_j \in \{0,1\}^\tau$.

Padding: This circuit is padded to size $\mathsf{pad}_\mathsf{T} := \mathsf{pad}_\mathsf{T}(\lambda)$, which is determined in analysis.

1. Parse $\mathsf{ct}_{j-1} = (t_{j-1}, c_{j-1})$
2. Compute $\mu' := c_{j-1} \oplus \mathsf{PRF}(\mathsf{K}_{j-1}, t_{j-1})$.
3. Compute $t_j := \mathsf{PRG}(r_j)$ and $y_j := \mathsf{PRF}(\mathsf{K}_j\{t^*_j\}, t_j)$
4. Return $\mathsf{ct}_j := (t_j, y_j \oplus \mu')$.

Figure 8: The description of FP^j

By the definition of Game-0, we have

$$\Pr[\mathsf{E}_0] = |\Pr[\mathsf{Hyb}'_{U(\lambda)+1}(b) = 1] - \Pr[\mathsf{Hyb}'_{U(\lambda)}(b) = 1]|.$$

In addition, since the challenge-equal ciphertext at epoch i for $\mathsf{coin} = 0$ is random in Game-4, it trivially holds that

$$\Pr[\mathsf{E}_4] \leq \mathsf{negl}(\lambda).$$

Thus, if we prove the propositions below, we complete the proof of Lemma 6.2.

Proposition 6.3. *It holds that* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_0]| \leq \mathsf{Adv}^{\mathsf{prg}}_{\mathcal{B}_1,\mathsf{PRG}}(\lambda)$.

Backward Punctured Hybrid Update Function $\mathsf{BP}^j[\mathsf{K}_j\{t_j^*\}, \mathsf{K}_{j+1}, t_j^*, y_j^*](\mathsf{ct}_j, r_{j+1})$

Hardwired: $\mathsf{K}_j\{t_j^*\}, \mathsf{K}_{j+1}, t_j^*, y_j^* \in \{0,1\}^m$.

Input: A ciphertext ct_j and randomness $r_{j+1} \in \{0,1\}^\tau$.

Padding: This circuit is padded to size $\mathsf{pad}_\mathsf{T} := \mathsf{pad}_\mathsf{T}(\lambda)$, which is determined in analysis.

1. Parse $\mathsf{ct}_j = (t_j, c_j)$
2. If $t_j = t_j^*$, compute $\mu' := c_j \oplus y_j^*$.
3. Else compute $\mu' := c_j \oplus \mathsf{PRF}(\mathsf{K}_j\{t_j^*\}, t_j)$.
4. Compute $t_{j+1} := \mathsf{PRG}(r_{j+1})$ and $y_{j+1} := \mathsf{PRF}(\mathsf{K}_{j+1}, t_{j+1})$
5. Return $\mathsf{ct}_{j+1} := (t_j, y_{j+1} \oplus \mu')$.

Figure 9: The description of BP^j

Proposition 6.4. *It holds that* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_1]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathcal{B}_2, i\mathcal{O}}(\lambda)$.

Proposition 6.5. *It holds that* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_2]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathcal{B}_3, i\mathcal{O}}(\lambda)$.

Proposition 6.6. *It holds that* $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_3]| \leq \mathsf{Adv}^{\mathsf{pprf}}_{\mathcal{B}_4, \mathsf{PRF}}(\lambda)$.

We summarize those transitions in Figure 10. We give the proofs of Propositions 6.3 to 6.6 below. ■

values	Δ_i	Δ_{i+1}	k_i	$\mathsf{ct}^* = (t_i^*, c_i^*)$	security
Game-0	$i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}})$	$i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}})$	K_i	(real, real)	N/A
Game-1	$i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}})$	$i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}})$	K_i	(\$, real)	PRG
Game-2	$i\mathcal{O}(\mathsf{FP}^i)$	$i\mathcal{O}(\mathsf{C}^{\mathsf{io}}_{\mathsf{re}})$	$\mathsf{K}_i\{t_i^*\}$	(\$, real)	IO
Game-3	$i\mathcal{O}(\mathsf{FP}^i)$	$i\mathcal{O}(\mathsf{BP}^i)$	$\mathsf{K}_i\{t_i^*\}$	(\$, real)	IO
Game-4	$i\mathcal{O}(\mathsf{FP}^i)$	$i\mathcal{O}(\mathsf{BP}^i)$	$\mathsf{K}_i\{t_i^*\}$	(\$, \$)	PPRF

Figure 10: The differences of values for epoch i tokens in Game-0 to Game-4. We omit the hardwired values in $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}, \mathsf{FP}^i, \mathsf{BP}^i$ for simplicity. In the column of ct^*, real means this value is honestly generated as in the real game and \$ means this value is uniformly random.

Padding Parameter. The proof of security relies on the indistinguishability of the obfuscated circuits of $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}$, FP^j, and BP^j defined in Figures 7 to 9. We need to set $\mathsf{pad}_\mathsf{T} := \max(|\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}|, |\mathsf{FP}^j|, |\mathsf{BP}^j|)$.

Proofs of core propositions. First of all, we note that we can simulate all tokens and updated ciphertext at epochs except epochs i and $i+1$ since we can generate K_j for $j \in [0, T] \setminus \{i\}$ in the reductions. In addition, we can simulate updated ciphertext from epoch i to $i+1$ if we know the underlying plaintext since the distribution of updated ciphertext is completely the same as the normal encryption. We observed in Section 6.2.

Proof of Proposition 6.3. It is easy to obtain this proposition since the only difference between these games is the challenge-equal ciphertext at epoch i consists of random t_i^* or real $t_i^* = \mathsf{PRG}(r_i^*)$. Note that r_i^* is not used in any other part. This value is an internal randomness to generate the challenge-equal ciphertext. Thus, the proposition follows by the security of the PRG. ■

Proof of Proposition 6.4. The difference between these games is the token Δ_i is generated by using $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}$ or FP^i. If these two circuits are functionally equivalent, we obtain the statement by using IO security.

The two program differ if $t_i^* = \mathsf{PRG}(r_i)$ for input (ct_{i-1}, r_i) since in FP^i we use a punctured key $\mathsf{K}_i\{t_i^*\}$. (Other parts are completely the same.) However, $t_i^* \leftarrow \{0,1\}^\tau$ in these games, $t_i^* = \mathsf{PRG}(r_i)$ happens only with $1/2^\lambda$ when we set $\tau := \lambda$ and $n := 2\lambda$. Thus, these two circuits are functionally equivalent with probability $1 - 1/2^\lambda$. By using the security of IO, we complete the proof. ■

Proof of Proposition 6.5. The difference between these games is the token Δ_{i+1} is generated by using $\mathsf{C}^{\mathsf{io}}_{\mathsf{re}}$ or BP^i. If these two circuits are functionally equivalent, we obtain the statement by using IO security.

Note that in these two games, the hardwired value y_i^* is equal to $\mathsf{PRF}(\mathsf{K}_i, t_i^*)$ since we never used the security of the PPRF so far.

The two program differ if $t_i = t_i^*$ for input $(\mathsf{ct}_i = (t_i, c_i), r_{i+1})$ since in BP^i we use a punctured key $\mathsf{K}_i\{t_i^*\}$. (Other parts are completely the same.) However, if $t_i = t_i^*$ in BP^i, it computes $\mu' := c_i \oplus y_i^*$. As we noted above, $y_i^* = \mathsf{PRF}(\mathsf{K}_i, t_i^*)$ in these games. This means $\mu' = c_i \oplus \mathsf{PRF}(\mathsf{K}_i, t_i^*)$ even when $t_i = t_i^*$. Thus, these two circuits are functionally equivalent. By using the security of IO, we complete the proof. ■

Proof of Proposition 6.6. We construct an adversary $\mathcal{B}_4$ for PPRF by using a distinguisher $\mathcal{A}$ for these two games. $\mathcal{B}_4$ chooses $t_i^* \leftarrow \{0,1\}^\tau$ and sends it to the challenger of PPRF and receives $\mathsf{K}\{t_i^*\}$ and y. $\mathcal{B}_4$ sets (implicitly) $\mathsf{k}_i := \mathsf{K}$ and $y_i^* := y$ and simulates the game for $\mathcal{A}$.

The issue is how to simulate tokens related to k_i since $\mathcal{B}_4$ does not have "non-punctured" PRF key $\mathsf{k}_i = \mathsf{K}_i = \mathsf{K}$. However, by the game transitions so far, we never use K_i to generate tokens Δ_i and Δ_{i+1} (other tokens do not need K_i). $\mathcal{B}_4$ uses $\mathsf{K}\{t_i^*\}$ instead of K_i. Thus, $\mathcal{B}_4$ can simulate all tokens and updated ciphertext by using $\mathsf{k}_i = \mathsf{K}\{t_i^*\}$ and $y_i^* = y$ given from the challenger.

If $y = \mathsf{PRF}(\mathsf{K}, t_i^*)$, the distribution of $y_i^* = y$ is the same as in Game-3. If $y \leftarrow \{0,1\}^m$, the distribution of $y_i^* = y$ is the same as in Game-4. Therefore, if $\mathcal{A}$ distinguishes these two games, $\mathcal{B}_4$ can break the security of PPRF. This completes the proof. ■

Acknowledgments

The author would like to thank Fuyuki Kitagawa for giving a pointer to the reference about an RKA-secure AE scheme.

References

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 77–94. Springer, Heidelberg, August 2014. (Cited on page 10.)

[ABPW13] Yoshinori Aono, Xavier Boyen, Le Trieu Phong, and Lihua Wang. Key-private proxy re-encryption under LWE. In Goutam Paul and Serge Vaudenay, editors, *INDOCRYPT 2013*, volume 8250 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2013. (Cited on page 10.)

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009. (Cited on page 15, 16.)

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. Cryptology ePrint Archive, Report 2011/613, 2011. http://eprint.iacr.org/2011/613. (Cited on page 15.)

[AMP19] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Symmetric primitives with structured secrets. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 650–679. Springer, Heidelberg, August 2019. (Cited on page 10.)

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 4(296):625–635, 1993. (Cited on page 15.)

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 464–493. Springer, Heidelberg, August 2020. (Cited on page 1, 2, 5, 9, 10, 11, 13, 20.)

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving speed and security in updatable encryption schemes. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 559–589. Springer, Heidelberg, December 2020. (Cited on page 1.)

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 1–13. Springer, Heidelberg, February / March 2013. (Cited on page 16.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 11.)

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 4.)

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. (Cited on page 16.)

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013. (Cited on page 1.)

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. (Cited on page 16.)

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. (Cited on page 16.)

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011. (Cited on page 16.)

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 4.)

[CCL+14] Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 95–112. Springer, Heidelberg, March 2014. (Cited on page 10.)

[EPRS17a] Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key rotation for authenticated encryption. Cryptology ePrint Archive, Report 2017/527, 2017. http://eprint.iacr.org/2017/527. (Cited on page 10.)

[EPRS17b] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 98–129. Springer, Heidelberg, August 2017. (Cited on page 1, 10, 13.)

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig. (Cited on page 10.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. (Cited on page 3, 4.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 15.)

[HLL16] Shuai Han, Shengli Liu, and Lin Lyu. Efficient KDM-CCA secure public-key encryption for polynomial functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 307–338. Springer, Heidelberg, December 2016. (Cited on page 10.)

[Jia20a] Yao Jiang. The direction of updatable encryption does not matter much. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 529–558. Springer, Heidelberg, December 2020. (Cited on page 1, 2, 5, 8, 9, 10, 11, 12, 13, 20.)

[Jia20b] Yao Jiang. The direction of updatable encryption does not matter much. Cryptology ePrint Archive, Report 2020/622, 2020. https://eprint.iacr.org/2020/622. (Cited on page 11, 13.)

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 68–99. Springer, Heidelberg, May 2019. (Cited on page 1, 9, 11, 20.)

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013. (Cited on page 4.)

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011. (Cited on page 15.)

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 685–716. Springer, Heidelberg, April / May 2018. (Cited on page 1, 7, 9, 10, 11, 13, 20.)

[NX15] Ryo Nishimaki and Keita Xagawa. Key-private proxy re-encryption from lattices, revisited. *IEICE Transactions*, 98-A(1):100–116, 2015. (Cited on page 10, 16.)

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009. (Cited on page 16.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 15, 16, 18.)