Stronger and Simpler Updatable Encryption

Huanhuan Chen[1], Yao Jiang Galteland[2], and Kaitai Liang[1]

[1] Delft University of Technology, Delft, The Netherlands
{h.chen-2, kaitai.liang}@tudelft.nl
[2] Norwegian University of Science and Technology, NTNU, Norway
yao.jiang@ntnu.no

Abstract. We define a stronger confidentiality notion for the ciphertext-dependent updatable encryption. The new notion captures the adaptive security that was not covered in prior works (CRYPTO2017, ASIACRYPT 2020), but also supports both deterministic and randomized constructions. We revise the public key encryption introduced by Micciancio et al. (EUROCRYPT 2012) to a simpler scheme using the lattice trapdoor techniques. Based on the resulting scheme, we further propose a new updatable encryption construction that achieves the new notion under the Learning with Errors assumption.

Keywords: updatable encryption · adaptive security · trapdoor for lattices · LWE

1 Introduction

Updatable encryption (UE) enables a cloud server to update ciphertext via only an update token received from a client so that the updated ciphertext is decryptable by a new key. The ciphertexts are previously encrypted by an old key and outsourced to the cloud server by the client, who prefers to switch the key regularly to reduce the risk of key compromise. The client can randomly generate a new key and a token, and further sends the token to the server for the ciphertext updates. Depending on if the client needs to download some part of ciphertexts in the token generation, there are two types of UE schemes: ciphertext-independent [6,11,13,14,16,18] and ciphertext-dependent schemes [4,5,7,8]. In this paper, we focus on ciphertext-dependent UE.

Security Notions. The confidentiality of UE requires ciphertexts should not leak any information to the adversary under the condition that the adversary may corrupt some keys, update tokens, and ciphertexts. Like the standard semantic security for public key encryption (PKE) schemes, the *message confidentiality* [5,8] have been proposed to guarantee that the adversary cannot reveal anything about the underlying plaintext for a given ciphertext. In UE schemes, ciphertexts can be further generated by the update algorithm. This triggers the definition for the *re-encryption indistinguishability* [8] that aims to prevent the adversary distinguishing from which ciphertext the challenge ciphertext is updated. To avoid leaking the "age" of the ciphertext, the Confidentiality [4] is

defined, requiring that any ciphertext generated by the encryption algorithm should be indistinguishable from the ciphertext generated by the updated algorithm. It has been proved to be stronger than the previous two security notions (i.e., message confidentiality and re-encryption indistinguishability).

A crucial consideration, which should be covered in security notions, is to avoid "trivial" wins. In order to capture the real-world security requirements of UE, the adversary in each of the above notions is provided with a number of oracles, enabling it to corrupt keys, update tokens, and ciphertexts. Some combinations of oracles may easily lead to a trivial win and thus, the conditions should be checked carefully at the end of a security game. We point out two shortcomings in the confidentiality notion from prior works [5,8,4].

- Various oracles provided to the adversary may make the definition overly complex.
- Previous works unfortunately only capture selective security, that is, the adversary are given compromised keys in the beginning of the security game, instead of adaptive corruption. Furthermore, none of the prior notions enables the adversary to access to a decryption oracle; and meanwhile, all of them are only applied to randomized update encryption.

In this paper, we define a new confidentiality notion for the ciphertext-dependent UE that is simpler than Confidentiality [4] by reducing the number of oracles given to the adversary. We also prove the equivalence of these two notions with no-directional key updates. We further present a confidentiality notion that is stronger than any existing definitions since we maximize the ability of the adversary by providing a decryption oracle and the ability to corrupt keys adaptively. We follow the trivial condition analysis in [14,6,11], which deal with ciphertext-independent UE, but greatly simplify the process of checking trivial win conditions by recording look-up tables which track the information leaked to the adversary in the game. A remarkable improvement is that the challenger is able to instantly know if a trivial win condition is triggered, and there is no extra computation incurred by the extended leakage sets as compared to [14,6,11]. A brief comparison of the proposed notions with prior works is presented in Fig. 1.

A New UE Construction. We design a new UE scheme based on the Learning With Errors (LWE) problem [17] that meets the new stronger confidentiality. The proposed scheme therefore is more secure than the prior ones. The underlying PKE scheme is inspired by [15], which we will point out a potential risk in their decryption algorithm. We revise the scheme to yield a new and simple PKE. At a high level, we adopt the re-encryption key generation technique from [12], which constructs a proxy re-encryption (PRE) scheme from lattices, to switch ciphertexts. PRE techniques have similar processes of re-encryption key generation and re-encryption, which correspond to token generation and ciphertext update in UE. The power of PRE is to enable a ciphertext to be re-encrypted, in which it is encrypted under a key, decryptable by another key. But PRE does not require the re-encrypted ciphertext independent from the ciphertext it was updated from, let along meeting the IND-UE-atk security. A subtle risk in the

Confidentiality Notion	Oracles To the Adversary	Compromised Key	Challenge Input	Update Procedure
UP-IND [8]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{m}}_1)$	rand
UP-REENC [8]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{c}}_0, \bar{\mathsf{c}}_1)$	rand
Confidentiality [4]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	rand
sConfidentiality Sect. 3.2	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpdate}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	rand
xxIND-UE-CPA Sect. 3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpdate}}, \mathcal{O}_{\mathsf{Corr}},$	Adaptive	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	xx
xxIND-UE-CCA Sect. 3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpdate}}, \mathcal{O}_{\mathsf{Corr}}, \mathcal{O}_{\mathsf{Dec}}$	Adaptive	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	xx

Fig. 1. A summary of confidentiality notions, where $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}\}$ represents the update procedure can be either randomized or deterministic. The adversary in each confidentiality game provides two challenge inputs based on the oracles it has access to and tries to distinguish the challenge outputs. Confidentiality is proven stronger than both UP-IND and UP-REENC given in [4].

proof of [12] was identified by [9] which further proposed a new construction to address the risk. But their solution could be problematic. We clear all the risks by carefully designing our UE. We also provide a detailed proof by introducing firewall techniques from ciphertext-independent to ciphertext-dependent UE schemes.

1.1 Related Work

Ciphertext-Independent UE. The security notions for ciphertext-independent UE have been well studied, which might be known as the same as those for ciphertext-dependent UE, at first glance. Lehmann and Tackmann [14] defined two practical notions: IND-Enc that requests the adversary to distinguish two ciphertexts generated by the encryption algorithm, and IND-Upd that requires to distinguish two ciphertexts generated by the update algorithm. Boyd et al. [6] later introduced a new notion IND-UE to identify a ciphertext generated by the encryption algorithm from an updated ciphertext. The challenge input in the definition of IND-Enc, IND-Upd and IND-UE[3] are the same as that in the definition of UP-IND, UP-REENC, and Confidentiality, respectively.

However, there are noticeable difference between the security notions for two types of UE schemes, reflecting in the ways of recording leakage sets. For ciphertext-independent UE, a single update token can be used to update all ciphertexts; therefore, the bookkeeping techniques developed in [14] and [13] only track the list of epochs, in which the adversary knows an update token, ciphertext, or key, and the later two can be extended via known and inferred update tokens. But for ciphertext-dependent UE, each token is correlated to a ciphertext; only the adversary knows the token related to the ciphertext can update the ciphertext. The definition in [8,4] and this paper records look-up tables to

[3] We also name the new notion IND-UE in Sect. 3.3, aiming to unify the confidentiality notion for ciphertext-independent and ciphertext-dependent UE schemes.

keep track of the leaked ciphertexts, which map key index and ciphertext header (needed to download for token generation) pairs to ciphertext bodies. The tables will be updated automatically, if the adversary queries update tokens related to the known ciphertexts, Furthermore, we will show in Sect. 3.3 that the challenger can check immediately if trivial win conditions are triggered during the game by the recorded table, without an extra computation of the extended sets as that in the security definition for ciphertext-independent UE schemes.

Proxy Re-Encryption. The ciphertext of the PRE scheme [12], which has a re-encryption process similar to the ciphertext update in UE, is of the form $c = (\mathbf{H}_u, \mathbf{b})$, where $\mathbf{H}_u$ is an invertible matrix and

$$\begin{aligned}\mathbf{b}^t =& 2\left(\mathbf{s}^t\mathbf{A}_\mu \bmod q\right) + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathrm{enc}(\mathbf{m}))^t \bmod 2q \\ =& 2\left(\mathbf{s}^t\left[\mathbf{A}_0 \,|\mathbf{A}_1 + \mathbf{H}\mathbf{G}|\, \mathbf{A}_2 + \mathbf{H}_u\mathbf{G}\right] \bmod q\right) + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathrm{encode}(\mathbf{m}))^t \bmod 2q,\end{aligned}$$

in which the matrices $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2$ are the public key, and $\mathbf{e}$ represents the error item. The re-encrypted ciphertext is $c' = (\mathbf{H}_u, \mathbf{b}')$ where

$$\mathbf{b}'^t = 2\left(\mathbf{s}^t\mathbf{A}'_\mu \bmod q\right) + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathrm{enc}(\mathbf{m}))^t \bmod 2q,$$

so that $\mathbf{c}'$ has the same form of c and can be decrypted by the trapdoor for $\mathbf{A}'_\mu$, i.e., the new secret key. The re-encryption proceeds by generating the transition matrix from $\mathbf{A}_u$ to $\mathbf{A}'_u$ whose last nk rows is $[\mathbf{0} \;\; \mathbf{0} \;\; \mathbf{I}]$ as the re-encryption key, i.e., $\mathbf{A}_u \cdot \mathbf{M} = \mathbf{A}'_u$, and re-encrypting $\mathbf{c}$ by multiplying $\mathbf{b}^t$ and $\mathbf{M}$, that is

$$\begin{aligned}\mathbf{b}^t\mathbf{M} =& 2\left(\mathbf{s}^t\mathbf{A}_\mu \cdot \mathbf{M} \bmod q\right) + \mathbf{e}^t \cdot \mathbf{M} + (\mathbf{0}, \mathbf{0}, \mathrm{enc}(\mathbf{m}))^t \cdot \mathbf{M} \bmod 2q \\ =& 2\left(\mathbf{s}^t\mathbf{A}'_\mu \bmod q\right) + \mathbf{e}^t \cdot \mathbf{M} + (\mathbf{0}, \mathbf{0}, \mathrm{enc}(\mathbf{m}))^t \bmod 2q.\end{aligned}$$

1.2 Our Approach

In general, we first build a new PKE scheme inspired by [15] using trapdoor techniques for lattices, but we tackle the potential risk in the decryption algorithm. This PKE scheme serves as the underlying encryption for our UE construction. We leverage the "re-encryption key generation" technique from [12] to generate one part of the update token. The other part is carefully designed to make sure that the UE scheme achieves the stronger IND-UE-CCA-1 notion.

Risk in [15]. The ciphertext in the PKE scheme proposed in [15] is $c = (\mathbf{H}_\mu, \mathbf{b})$ where

$$\mathbf{b}^t = 2(\mathbf{s}^t\mathbf{A}_\mu \bmod q) + (\mathbf{e}_0, \mathbf{e}_1)^t + (\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod 2q,$$

and $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1] \in \mathbb{Z}^{\bar{m}+nk}$ can be calculated from the public key and the invertible matrix $\mathbf{H}_\mu$. The secret key $\mathbf{R} \in \mathbb{Z}^{\bar{m}\times nk}$ is a trapdoor for $\mathbf{A}_u$, with which it can be efficient to recover $\mathbf{s}'$ and $(\mathbf{e}'_0, \mathbf{e}'_1)$ from

$$\mathbf{b}^t = (\mathbf{s}')^t\mathbf{A}_\mu + (\mathbf{e}'_0, \mathbf{e}'_1)^t \mod q.$$

Then the decryption procedure works by regarding $(\mathbf{e}_0, \mathbf{e}_1) = (\mathbf{e}'_0, \mathbf{e}'_1)$ and outputs the plaintext by a encode^{-1} mapping. We show in Lemma 5 that the case $(\mathbf{e}_0, \mathbf{e}_1) = (\mathbf{e}'_0, \mathbf{e}'_1)$ contradicts the key generation algorithm and therefore ciphertexts cannot be decrypted correctly. The problem results from $(\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod q$ cannot be expressed by $(\mathbf{s}')^t \mathbf{A}_\mu$ for some $\mathbf{s}$.

A New PKE Scheme. Our main idea is to avoid the plaintext being involved in the computation of error items. We constructs a 1×3 block matrix as the encryption matrix $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_1] \in \mathbb{Z}^{\bar{m}+2nk}$ and the secret key $\mathbf{R}$ is the trapdoor for the first two block matrices. The ciphertext is $c = (\mathbf{H}_\mu, \mathbf{b})$ where

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q. \tag{1}$$

The decryption proceeds by first recovering $(\mathbf{e}_0, \mathbf{e}_1)$ and $\mathbf{s}$ with the trapdoor $\mathbf{R}$ from the first $\bar{m} + nk$ coordinates of $\mathbf{b}$, and then recovering $\mathbf{m}$ and $\mathbf{e}_3$ from the last nk coordinates of $\mathbf{b}$ with the recovered parameter $\mathbf{s}$.

A New UE Scheme. We state that there are two technical challenges on directly using the re-encryption technique in [12] to construct a secure UE scheme satisfying the $\mathsf{IND\text{-}UE\text{-}notion}$ (recall the notion requires the indistinguishability between "fresh" and updated ciphertexts). The first observation is that $\mathbf{H}_\mu$, as part of the ciphertext, is never changed in the update process. The adversary can distinguish the challenge ciphertexts by comparing the invertible matrices extracted from the challenge output and input, and win the game easily. Beyond that, $\mathbf{s}$ is also unchanged during update process. If the adversary is able to corrupt an old key, then it can invert $\mathbf{s}$ from the ciphertext. Using $\mathbf{s}$ and the new public key, the adversary can recover the plaintext from the new ciphertext.

We have to change the invertible matrix $\mathbf{H}_\mu$ in next epoch. One may replace $\mathbf{H}_\mu$ in the re-encryption process by a random invertible matrix $\mathbf{H}'_\mu$ to construct the encryption matrix $\mathbf{A}'_u$. But we notice that the token (re-encryption key) is generated before the update (re-encryption) process and a single token is used to update all ciphertexts. It is hard to generate a transition matrix in advance before the update process such that $\mathbf{A}_u \cdot \mathbf{M} = \mathbf{A}'_u$ works for all random $\mathbf{A}_u$ and $\mathbf{A}'_u$. Our approach is to generate such a random invertible matrix in the token (re-encryption key) generation algorithm, that is, all ciphertexts (refresh or updated) in one epoch have the same encryption matrix $\mathbf{A}_\mu$ (randomly changed in the next epoch), so that we can correctly generate the transition matrix from one epoch to the next, and also ensure the randomness of $\mathbf{H}_\mu$ in each epoch. To improve the randomness of $\mathbf{s}$, we add $\mathbf{b}_0$ the encryption of $\mathbf{0}$ under the new key to the updated ciphertext. In summary, the update token is $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$ and the updated ciphertext of $c = (\mathbf{H}_\mu, \mathbf{b})$ is $c' = (\mathbf{H}'_u, \mathbf{b}')$, where

$$\begin{aligned}(\mathbf{b}')^t &= \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ &= \left[\mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right]\mathbf{M} + (\mathbf{s}')^t\mathbf{A}'_\mu + (\mathbf{e}')^t \\ &= (\mathbf{s} + \mathbf{s}')^t\mathbf{A}'_\mu + \left(\mathbf{e}^t\mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod q. \end{aligned} \tag{2}$$

The updated ciphertext has the same form of c with new independent invertible matrix and new random factor $\mathbf{s}+\mathbf{s}'$. Thus, we solve the two challenges and maintain the security when adopting the re-encryption technique [12] to UE and meanwhile, ensure that the token (especially, $\mathbf{M}$) does not leak any information about secret keys.

2 Preliminaries

We use upper-case and lower-case bold letters to denote matrices and column vectors, respectively. The real numbers and the integers are denoted by $\mathbb{R}$ and $\mathbb{Z}$. For a vector $\mathbf{x} \in \mathbb{R}^n$, let $\|\mathbf{x}\|$ and $\|\mathbf{x}\|_\infty$ denote the 2-norm and infinity norm of $\mathbf{x}$. For any $\mathbf{B} \in \mathbb{R}^{n\times k}$, the largest singular value of $\mathbf{B}$ is defined by $s_1(\mathbf{B}) := \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\|$, where the maxima is taken over all unit vectors $\mathbf{u} \in \mathbb{R}^k$ and $\mathbf{B}^t$ is the transposition of $\mathbf{B}$. For two matrices $\mathbf{A} \in \mathbb{R}^{m\times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m\times n_2}$, $[\mathbf{A} \mid \mathbf{B}] \in \mathbb{R}^{m\times(n_1+n_2)}$ denotes the concatenation of the columns of $\mathbf{A}$ and $\mathbf{B}$.

2.1 Lattices, Trapdoors, and Algorithms

Definition 1. *For integer numbers q, n and m, and an arbitrary matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, define the full-rank m-dimensional integer lattices*

$$\Lambda(\mathbf{A}^t) = \{\mathbf{y} \in \mathbb{Z}^m : \exists\ \mathbf{x} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{y} = \mathbf{A}^t\mathbf{x} \bmod q\}.$$

Throughout this paper, $q \geq 2$ is an integer modulus and $k = \lceil \log_2 q \rceil$. For an integer $n \geq 1$, $\mathbf{G}$ is defined as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n\times nk}$, where $\mathbf{g}^t = [1\ 2\ 4\ \cdots\ 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$, i.e. $\mathbf{G} = diag(\mathbf{g}^t, \cdots, \mathbf{g}^t)$. The following theorem enables two efficient algorithms to solve SIS and LWE problems (please see Appendix A for details) relative to $\mathbf{G}$.

Theorem 1 ([15], Theorem 4.1). *For any integer $q \geq 2, n \geq 1, k = \lceil \log_2 q \rceil$ and $m = nk$, $\mathbf{G}$ has the following properties:*

- *Inverting $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \mod q$ can be performed in quasilinear time for any $\mathbf{s} \in \mathbb{Z}_q^n$ and $\|\mathbf{e}\|_\infty \leq q/4$.*
- *Preimage sampling for $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\mathbf{x} \mod q$ can be performed in quasilinear time.*

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for some $m \geq nk \geq n$. A $\mathbf{G}$-trapdoor for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}_q^{(m-nk)\times nk}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}\mathbf{G}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{nk\times nk}$. As an example, in [15], one can generate a random $\mathbf{A} = [\mathbf{A}_0 | -\mathbf{A}_0\mathbf{R} + \mathbf{H}\mathbf{G}]$ where $\mathbf{A}_0$ is a uniform matrix in $\mathbb{Z}_q^{n\times\bar{m}}$, $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ is an invertible matrix and $\mathbf{R}$ is chosen from a distribution $\mathcal{D}$ over $\mathbb{Z}_q^{\bar{m}\times nk}$. Clearly, $\mathbf{R}$ is a $\mathbf{G}$-trapdoor for $\mathbf{A}$ for any $\bar{m}$ and distribution $\mathcal{D}$, which will be also used in our construction. The following two properties ensure to solve LWE and SIS relative to $\mathbf{A}$ by using a trapdoor for $\mathbf{A}$.

Lemma 1 ([15], Theorem 5.4). *For a* $\mathbf{G}$*-trapdoor* $\mathbf{R}$ *for* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and an* LWE *instance* $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$*, if* $\|[\mathbf{R}^t \ \mathbf{I}] \cdot \mathbf{e}\|_\infty \leq q/4$ *, then there is an efficient algorithm called* $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}, \mathbf{H}, \mathbf{b})$ *that recovers* $\mathbf{s}$ *and* $\mathbf{e}$ *from the function* $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$*.*

Lemma 2 ([15], Theorem 5.5). *For a* $\mathbf{G}$*-trapdoor* $\mathbf{R}$ *for* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and any* $\mathbf{u} \in \mathbb{Z}_q^n$*, there is an efficient algorithm called* $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}, \mathbf{H}, \mathbf{u}, s)$ *that samples a Gaussian vector* $\mathbf{x}$ *from* $D_{\mathbb{Z},s}^m$ *such that* $\mathbf{Ax} = \mathbf{u}$*, where* s *can be as small as* $\sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\sum_{\mathbf{G}}) + 1} \cdot \omega(\sqrt{\log n})$ *and* $s_1(\sum_{\mathbf{G}})$ *is a constant for given* $\mathbf{G}$*(equal to 4 if* q *is a power of 2, and 5 otherwise.).*

Remark 1. More generally, the algorithm $\mathsf{SampleD}^{\mathcal{O}}$ can be extended from vectors to matrices. Specifically, for a matrix $\mathbf{U} \in \mathbb{Z}_q^{n\times l}$, we call $\mathsf{SampleD}^{\mathcal{O}}$ on each column of $\mathbf{U}$ and then obtain l m-dimensional columns $\{x_i\}_1^l$ from $D_{\mathbb{Z},s}^m$. Let $\mathbf{X} = [x_1 \mid \cdots \mid x_l]$, then $\mathbf{AX} = \mathbf{U}$ and $\mathbf{X} \in D_{\mathbb{Z},s}^{m\times l}$, where s can be as small as $\sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\sum_{\mathbf{G}}) + 1} \cdot \omega(\sqrt{\log n})$.

2.2 Updatable Encryption

We briefly review the syntax of ciphertext-dependent UE and the prior notions of confidentiality.

Definition 2 ([5,8,4]). *A ciphertext-dependent UE scheme includes a tuple of PPT algorithms* {UE.KG, UE.Enc, UE.Dec, UE.TokenGen, UE.Update} *that operate in epochs starting from 0.*

- $\mathsf{UE.KG}(1^\lambda)$*: the key generation algorithm outputs an epoch key* $\mathsf{k_e}$*.*
- $\mathsf{UE.Enc}(\mathsf{k_e}, \mathsf{m})$*: the encryption algorithm takes as input an epoch key* $\mathsf{k_e}$ *and a message* m *and outputs a ciphertext header* $\hat{\mathsf{ct}}_\mathsf{e}$ *and a ciphertext body* $\mathsf{ct_e}$*, i.e.,* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$*.*
- $\mathsf{UE.Dec}(\mathsf{k_e}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: the decryption algorithms takes as input an epoch key* $\mathrm{k_e}$ *and a ciphertext* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$ *and outputs a message* m' *or* $\perp$*.*
- $\mathsf{UE.TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$*: the token generation algorithm takes as input two epoch keys* $\mathsf{k_e}$ *and* $\mathsf{k_{e+1}}$ *and a ciphertext header* $\hat{\mathsf{ct}}_\mathsf{e}$*, and outputs an update token* $\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}_\mathsf{e}}$ *or* $\perp$*.*
- $\mathsf{UE.Update}(\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}_\mathsf{e}}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: the update algorithm takes as input a token* $\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}}$ *with related to the ciphertext* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$*, and outputs an updated ciphertext* $(\hat{\mathsf{ct}}_{\mathsf{e+1}}, \mathsf{ct_{e+1}})$ *or* $\perp$*.*

In an updatable encryption scheme, there are two ways to generate a ciphertext: either via the encryption algorithm to produce the fresh ciphertext, or the update algorithm to produce the updated ciphertext. The correctness of a UE requires these two types of ciphertexts to decrypt correctly to the underlying message except with a low failure probability.

Prior Notions of Confidentiality. Boneh et al. [4] defined a stronger notion of confidentiality than any previous works [5,8], called Confidentiality, for ciphertext-dependent UE schemes, which captures the indistinguishability of the fresh and the updated ciphertexts.

Their notion shows the selective security, where the adversary in the security game is given some epoch keys in the setup phase, but it is not allowed to commit adaptive corruption. During the query phase, the adversary is given $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{TokenGen}, \mathcal{O}.\mathsf{Update}\}$ to obtain ciphertexts. Then it submits two challenge inputs $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ in the challenge phase based on the information it has acquired and receives the challenge output from the challenger. The goal of the adversary is to guess that if the challenge output is encrypted from m or updated from $(\hat{\mathsf{ct}}, \mathsf{ct})$. The adversary can continue querying those oracles and eventually give a guessing bit. A look-up table is maintained during the game to prevent the adversary from obtaining the challenge-equal ciphertexts (which are encrypted or updated from the challenge input) and the epoch key in a same epoch; otherwise, the adversary can trivially win the game by decrypting the challenge-equal ciphertexts with the epoch key and comparing the received underlying message with the challenge input. Note we will present the formal definition in Sect. 3.2.

3 New Confidentiality Notions for Updatable Encryption

To simplify the security notion given in [4], we define a new confidentiality notion called sConfidentiality, where we replace $\mathcal{O}.\mathsf{ReKeyGen}$ and $\mathcal{O}.\mathsf{ReEncrypt}$ in the security game with a single $\mathcal{O}.\mathsf{sReEncrypt}$ that returns both the update token and updated ciphertext to the adversary simultaneously. We prove in Theorem 2 that sConfidentiality and Confidentiality are equal for UE schemes with no-directional key updates.

Meanwhile, to give as much power to the adversary as possible, we define a new confidentiality notion called xxIND-UE-atk[4], where the adversary is given extra access to $\mathcal{O}.\mathsf{Dec}$ and $\mathcal{O}.\mathsf{Corr}$ that allows it to corrupt epoch keys at any time during the game. We also fully address the trivial win conditions for ciphertext-dependent UE schemes and provide simplified approaches to check if trivial win conditions are triggered in the security game.

Note that we present the notion of sConfidentiality in a way that tokens are generated between two nodes (which are more often used in PRE representing different clients) to make it consistent and easy to have comparisons with the work proposed in [4]; whilst we define the notion of xxIND-UE-atk in a commonly used way in UE schemes that tokens are computed by two consecutive epoch keys as we defined in Sect. 2.2, which in practical means ciphertexts are updated periodically.

[4] The same notion for ciphertext-independent UE scheme was proposed in [6]. We try to unify the notions. As we have shown in the comparison table in the introduction, there are intrinsic differences between ciphertext-independent UE and ciphertext-dependent UE.

3.1 UE Schemes with No-Directional Key Updates

In ciphertext-independent UE schemes, update tokens are generated by two successive epoch keys through the token generation algorithm $\Delta = \mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}})$; therefore, one key may be derived by the other if the token is known by the adversary. However, in ciphertext-dependent UE schemes, tokens are also determined by the ciphertext header: $\Delta = \mathsf{UE.TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$, so that keys may not be derived via corrupted tokens. We generalise the definition of no-directional key updates from ciphertext-independent UE to ciphertext-dependent UE as follows.

Definition 3. *A UE scheme, either ciphertext-independent or ciphertext-dependent, is said to have no-directional key updates if epoch keys cannot be inferred from known tokens.*

Constructing ciphertext-independent UE schemes with no-directional keys updates was an interesting open problem left by Jiang [11], and the difficulty relies on the requirement that the update token should be able to update all ciphertexts, without leaking information about either the old key or the new key. There are currently only two ciphertext-independent UE schemes with no-directional key updates [16,18].

But constructing ciphertext-dependent UE schemes with the same feature is relatively easy, since an update token is only required to update a corresponding ciphertext. In fact, this is the case for all known ciphertext-dependent UE schemes in [5,8,4], where the token generation procedure is actually a Dec-then-Enc process: decrypting the ciphertext header under the old key and generating new random variants which are then encrypted by the new key. The confidentiality of the underlying encryption ensures the information of keys is not leaked by the token. Our construction in this paper also provides no-directional keys updates, but we develop a new approach that is totally different from the previous Dec-then-Enc process, to generate the update token.

3.2 A Simplified Confidentiality Notion

We now define a new simplified confidentiality notion (Definition 4) by substituting the oracles $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Upd}}$ that the adversary has assess to in the $\mathsf{Confidentiality}$ game with a single $\mathcal{O}_{\mathsf{sUpd}}$. In the challenge phase, the adversary submits the challenge inputs $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ to the challenger according to the keys (by which it is given, those dishonest keys) and the oracles (with which it interacts), and its goal is to distinguish either the challenge ciphertext is a fresh encryption of the message m or it is a updated ciphertext of $(\hat{\mathsf{ct}}, \mathsf{ct})$. Before submitting a guessing bit, the adversary can keep querying those given oracles.

To track the challenge-equal ciphertexts that are obtained by the adversary in the game, we record a look-up table TC that maps a key index and challenge-equal ciphertext header pair to the corresponding challenge-equal ciphertext body. The adversary is prohibited from learning challenge-equal ciphertexts under any of the dishonest keys. This knowledge will lead to a trivial win since the

adversary can use the dishonest key to decrypt the challenge-equal ciphertext and compare the its underlying message with the challenge message.

Definition 4 (sConfientiality). *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Update}, \mathsf{Decrypt})$ *be an updatable encryption scheme. For a security parameter* λ*, positive integers* $h, d \in \mathbb{N}$*, an adversary* $\mathcal{A}$*, and a binary bit* $b \in \{0,1\}$*, we define the confidentiality experiment* $\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, h, d, \mathcal{A}, b)$ *and oracles* $\mathcal{O} = (\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Chall}})$ *in Fig. 2. The experiment maintains a look-up table* TC *as defined above.*

We say that an updatable encryption scheme UE *satisfies* sConfidentiality *if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $h, d \leq \mathsf{poly}(\lambda)$ *and efficient adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr\left[\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, h, d, \mathcal{A}, 0) = 1\right] - \Pr\left[\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, h, d, \mathcal{A}, 1) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

$\underline{\mathrm{Expt}^{\mathbf{sConf}}_{\mathsf{UE}}(\lambda, h, d, \mathcal{A}, b):}$

$\mathsf{k}_1, \ldots, \mathsf{k}_{h+d} \leftarrow \mathsf{KG}(1^\lambda)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{k}_{h+1}, \ldots, \mathsf{k}_{h+d})$

return $b' = b$

$\underline{\mathcal{O}_{\mathsf{Enc}}(i, \mathsf{m}):}$

return $\mathsf{Enc}(\mathsf{k}_i, \mathsf{m})$

$\underline{\mathcal{O}_{\mathsf{sUpd}}(i, j, (\hat{\mathsf{ct}}, \mathsf{ct})):}$

if $j > h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**

  return $\perp$

$\Delta_{i,j,\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_i, \mathsf{k}_j, \hat{\mathsf{ct}})$

$(\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{i,j,\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$

if $j \leq h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**

  $\mathsf{TC}[j, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$

return $(\Delta_{i,j,\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}', \mathsf{ct}'))$

$\underline{\mathcal{O}_{\mathsf{Chall}}(i, j, \mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct})):}$

if $j > h$ **then**

  return $\perp$

$(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}(\mathsf{k}_j, \mathsf{m})$

$\Delta_{i,j,\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_i, \mathsf{k}_j, \hat{\mathsf{ct}})$

$(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{i,j,\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$

if $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) = \perp$ **then**

  return $\perp$

if $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $|\mathsf{ct}'_0| \neq |\mathsf{ct}'_1|$ **then**

  return $\perp$

$\mathsf{TC}[j, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$

return $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Fig. 2. Security game for sConfidentiality. The dishonest keys $\mathsf{k}_{h+1}, \cdots, \mathsf{k}_{h+d}$ are provided to the adversary in the startup, while the honest keys $\mathsf{k}_1, \cdots, \mathsf{k}_h$ are kept private from the adversary. Both the judgements in $\mathcal{O}.\mathsf{sUpd}$ and $\mathcal{O}.\mathsf{Chall}$ are set to avoid the adversary knowing the key and valid challenge-equal ciphertexts in the same node.

Remark 2. The definition of Confidentiality is the same as sConfidentiality but the adversary is given more oracles $\mathcal{O} = (\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{Chall}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Upd}})$, in which the first two oracles are the same as those in Fig. 2 and the rest are defined in Fig. 3.

$\mathcal{O}_{\mathsf{TokenGen}}(i, j, \hat{\mathsf{ct}})$:

if $j > h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**
  return $\perp$
$\Delta_{i,j,\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathrm{k}_i, \mathrm{k}_j, \hat{\mathsf{ct}})$
$(\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{i,j,\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
if $j \leq h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**
  $\mathsf{TC}[j, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
return $\Delta_{i,j,\hat{\mathsf{ct}}}$

$\mathcal{O}_{\mathsf{Upd}}(i, j, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

$\Delta_{i,j,\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathrm{k}_i, \mathrm{k}_j, \hat{\mathsf{ct}})$
$(\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{i,j,\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
if $j > h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**
  return $\perp$
if $j \leq h$ and $\mathsf{TC}[i, \hat{\mathsf{ct}}] \neq \perp$ **then**
  $\mathsf{TC}[j, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
return $((\hat{\mathsf{ct}}', \mathsf{ct}'))$

Fig. 3. Different Oracles in Confidentiality.

Remark 3. Both Confidentiality and sConfidentiality only capture the confidentiality security on UE schemes with randomized ciphertext updates[5]. For a deterministic Confidentiality-secure UE scheme, the adversary is able to obtain the challenge output before the challenge phrase by querying $\mathcal{O}$.Update on the challenge ciphertext input, which makes it trivially win the game. We will provide a more secure confidentiality notion in the next section to consider both randomized and deterministic UE schemes.

As we discussed above, there cannot exist a dishonest key index in which the adversary knows a challenge-equal ciphertext, as this will lead to a trivial win. In the query phase, the adversary can obtain challenge-equal ciphertexts via querying the oracles $\mathcal{O}_{\mathsf{Chall}}$ or $\mathcal{O}_{\mathsf{sUpd}}$. In addition, the adversary may infer more challenge-equal ciphertexts via its known update tokens, and this behavior cannot be tracked. We have the following lemma to consider these two cases.

Lemma 3. *For UE schemes with no-directional key updates, the adversary cannot learn a challenge-equal ciphertext under a dishonest key in the confidentiality game for* sConfidentiality *as defined in Fig. 2, if the invalid symbol* $\perp$ *is not returned during the game.*

Proof. In the confidentiality game of no-directional keys updates UE schemes, all keys that are known to the adversary are the dishonest keys given in the setup, and cannot be extended. However, apart from the obtained challenge-equal ciphertexts during the query phase, the adversary can infer more challenge-equal ciphertexts via its known tokens. For the former case, we avoid the trivial win by disallowing the adversary to query the challenge oracle on a dishonest key index and to update the challenge-equal ciphertext to any of the dishonest key. For the latter case, the adversary can update challenge-equal ciphertext from a honest key k_i to a dishonest key k_j by its known token $\Delta_{i,j,\hat{\mathsf{ct}}}$. Since the key k_j

[5] In this paper, we assume the update algorithm is deterministic; when we refer to a randomized UE schemes, we mean the token generation algorithm is randomized.

is dishonest, the adversary cannot learn the token $\Delta_{i,j,\hat{\mathsf{ct}}}$ from the challenger, so it must be inferred by k_i and k_j, which contradicts with the condition that k_i is honest.

Theorem 2. *Let* $\mathsf{UE} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{ToenGen}, \mathsf{Update}, \mathsf{Decrypt})$ *be an updatable encryption scheme. For any* sConfidentiality *adversary* $\mathcal{A}$ *against* UE*, there is a* Confidentiality *adversary* $\mathcal{B}$ *against* UE *([BEKS20], Def. 3.4) such that*

$$\mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda) \tag{3}$$

In addition, if UE *is a UE scheme with no-directional key updates, then for any* Confidentiality *adversary* $\mathcal{B}$ *against* UE*, there is a* sConfidentiality *adversary* $\mathcal{A}$ *against* UE *such that*

$$\mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda) = \mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda)$$

Proof. We construct a reduction $\mathcal{B}$ that runs the Confidentiality game and simulates all responses to the queries of $\mathcal{A}$. The reduction sends all its known keys to $\mathcal{A}$, all $\mathcal{A}$'s queries except $\mathcal{O}_{\mathsf{sUpd}}$ to its challenger, and returns the challenger's responses to $\mathcal{A}$. When $\mathcal{A}$ queries the orcale $\mathcal{O}_{\mathsf{sUpd}}$ on some input, the reduction $\mathcal{B}$ submits the same input to $\mathcal{O}_{\mathsf{Upd}}$. If the response of the challenger is $\perp$, $\mathcal{B}$ also sends $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ calculates the updated ciphertext by the received update token and forwards the update token, together with the updated ciphertext, to the adversary $\mathcal{A}$. At last, $\mathcal{B}$ forward $\mathcal{A}$'s guess to its challenger. Since the reduction has an extra access to $\mathcal{O}_{\mathsf{Update}}$, we have the Inequality (3). [6]

If UE is a UE scheme with no-directional key updates, we construct a reduction $\mathcal{A}$ that runs the sConfidentiality game and simulates all the responses to the queries of the given $\mathcal{B}$. The reduction $\mathcal{A}$ sends all its known keys to $\mathcal{B}$ and all $\mathcal{B}$'s queries except those on $\mathcal{O}_{\mathsf{TokenGen}}$ and $\mathcal{O}_{\mathsf{Upd}}$ to its challenger, and returns its challenger's responses to $\mathcal{B}$. When $\mathcal{B}$ queries the oracle $\mathcal{O}_{\mathsf{TokenGen}}$ (or $\mathcal{O}_{\mathsf{Upd}}$) on some input, the reduction $\mathcal{A}$ submits the same input to the $\mathcal{O}_{\mathsf{sUpd}}$ oracle, and returns the update token (or updated ciphertext, respectively) received from its challenger to $\mathcal{B}$ if the response is not $\perp$; otherwise, $\mathcal{A}$ returns $\perp$ to $\mathcal{B}$.

Therefore, the reduction $\mathcal{A}$ simulates all response to $\mathcal{B}$'s queries. The only difference between $\mathcal{A}$ and $\mathcal{B}$ happens when $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{Upd}}$ in a honest key index: the reduction $\mathcal{A}$ receives the updated ciphertext, together with the update token from its challenger, but it only returns the updated ciphertext to the adversary $\mathcal{B}$. Since updating ciphertexts to a honest key is permitted, the reduction $\mathcal{A}$ does not trigger the trivial win condition by Lemma 3. Thus, we have

$$\mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda)$$

In combination with (3), we conclude the advantage of $\mathcal{A}$ is equal to that of $\mathcal{B}$.

[6] $\mathcal{O}_{\mathsf{sUpd}}$ is actually equal to $\mathcal{O}_{\mathsf{Upd}}$.

3.3 A Stronger Confidentiality Notion

We provide a new confidentiality notion for ciphertext-dependent schemes in the Definition 5 to give much more power to the adversary than the notion of $\mathsf{sConfidentiality}$. All available oracles that the adversary has access to are described in Fig. 4. The stronger notion captures the adaptive security by allowing the adversary to adaptively corrupt keys via $\mathcal{O}_{\mathsf{Corr}}$, and considers UE schemes in both cases: deterministic and randomized ciphertext update.

To track non-challenge and challenge-equal ciphertexts known to the adversary, we record two look-up tables: $\mathsf{TC}_{\mathsf{non}}$ that maps an epoch and non-challenge ciphertext header pair to the corresponding non-challenge ciphertext body, and $\mathsf{TC}_{\mathsf{chall}}$ that maps an epoch and challenge-equal ciphertext header pair to the corresponding challenge-equal ciphertext body, e.g. $\mathsf{TC}_{\mathsf{non}}[i, \hat{\mathsf{ct}}] = \mathsf{ct}$ indicating the adversary knows non-challenge ciphertext $(\hat{\mathsf{ct}}, \mathsf{ct})$ in epoch i. We denote $\mathsf{T}[0], \mathsf{T}[1]$ as the column and row index set of the table T, respectively. Therefore, $\mathsf{TC}_{\mathsf{chall}}[0]$ is the set of epochs, in which the adversary learns a challenge-equal ciphertext. We also record an epoch set $\mathcal{K}$ that contains the set of epochs in which the adversary corrupts an epoch key. All of the above are used to check if trivial win conditions are triggered in the experiment, which will be discussed below[7].

Comparison with $\mathsf{sConfidentiality}$. The adversary is provided with an extra ability to query the decryption oracle, and it can also corrupt keys at any time during the game by querying $\mathcal{O}_{\mathsf{Corr}}$, instead of selecting the compromised keys in the setup phrase. Moreover, these oracles also consider deterministic UE schemes: the challenge ciphertext input should not be such ciphertexts whose updates are already known to the adversary before the challenge, otherwise the adversary can easily win the game by comparing the challenge output with the update of the challenge ciphertext input. If a UE scheme is deterministic, the previous behaviors can be checked on the table $\mathsf{TC}_{\mathsf{non}}$ in the challenge phase, as all the ciphertexts known to the adversary before the challenge are recorded in $\mathsf{TC}_{\mathsf{non}}$.

Lemma 4. *For UE schemes with no-directional key updates, if* $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$*, then within all the epochs in which the adversary knows a challenge-equal ciphertext is* $\mathsf{TC}_{\mathsf{chall}}[0]$*, even if it is given access to all the oracles in Fig 4.*

Proof. The proof is similar to that of Lemma 3. Suppose $\mathsf{TC}_{\mathsf{chall}}[0] = \cup\{\mathsf{e}_{start}, \cdots, \mathsf{e}_{end}\}$. We prove that the adversary cannot know a challenge-equal ciphertext in epoch e_{end+1}. Since e_{end} is the last epoch in the epoch continuum, the adversary cannot know a challenge-equal ciphertext in epoch e_{end+1} via querying $\mathcal{O}_{\mathsf{sUpd}}$, since the received update ciphertext will be recorded in the table $\mathsf{TC}_{\mathsf{chall}}$, which conflicts with the condition that e_{end} is the last epoch in the epoch continuum. Alternatively, it can update challenge-equal ciphertext in epoch e_{end} with its inferred

[7] We follow the analysis of trivial win conditions of ciphertext-independent UE schemes [14,13,6,11], while providing much simpler approaches to check the conditions by look-up tables. This also indicates the differences between ciphertext-independent and ciphertext-dependent UE schemes

$\mathcal{O}_{\mathsf{Enc}}(\mathsf{e}, \mathsf{m})$:

$(\hat{\mathsf{ct}}, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{k}_\mathsf{e}, \mathsf{m})$
$\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}] \leftarrow \mathsf{ct}$
return $(\hat{\mathsf{ct}}, \mathsf{ct})$

$\mathcal{O}_{\mathsf{sUpd}}(\mathsf{e}, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

if $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ and
  $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ **then**
  return $\perp$
$\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathrm{k}_{\mathsf{e}-1}, \mathrm{k}_\mathsf{e}, \hat{\mathsf{ct}})$
$(\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
if $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$ **then**
  $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
else $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
return $(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}', \mathsf{ct}'))$

$\mathcal{O}_{\mathsf{Corr}}(\mathsf{e})$:

$\mathcal{K} = \mathcal{K} \cup \{\mathsf{e}\}$
return k_e

$\mathcal{O}_{\mathsf{Dec}}(\mathsf{e}, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

m' or $\perp \leftarrow \mathsf{Dec}(\mathsf{k}_\mathsf{e}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
if ($\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{chall}}[i, \hat{\mathsf{ct}}] = \mathsf{ct}$) or
  (($\mathsf{xx} = \mathsf{rand}$ and $\mathsf{e} \in \mathsf{TC}_{\mathsf{chall}}[0]$) and
  ($\mathsf{m}' = \mathsf{m}$ or m_1)) **then**
  return $\perp$
return $\mathsf{Dec}\left(\mathsf{k}_\mathsf{e}, (\hat{\mathsf{ct}}, \mathsf{ct})\right)$

$\mathcal{O}_{\mathsf{Chall}}(\mathsf{e}, \mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$:

$(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}\left(\mathsf{k}_\mathsf{e}, \mathsf{m}\right)$
if $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \mathsf{ct}$ **then**
  return $\perp$
$\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}\left(\mathrm{k}_{\mathsf{e}-1}, \mathrm{k}_\mathsf{e}, \hat{\mathsf{ct}}\right)$
$(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
if $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $|\mathsf{ct}'_0| \neq |\mathsf{ct}'_1|$ **then**
  return $\perp$
if ($\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1$) **then**
  return $\perp$
$\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$
return $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Fig. 4. An overview of the oracles that are access to the adversary in the Definition 5.

token. But from $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, we know the epoch key k_{end} is unknown to the adversary, which is needed to infer the token in e_{end+1}. The proof is the same for the challenge-equal ciphertext in the epoch e_{start}.

Remark 4. Lemma 4 shows that the adversary cannot infer a challenge-equal ciphertext in an epoch that it does not obtain a challenge-equal ciphertext during the query phase. But that does not mean all the ciphertexts known to the adversary are stored in the table $\mathsf{TC}_{\mathsf{chall}}$. For randomized UE schemes, the adversary can create arbitrary number of valid challenge-equal ciphertexts in any epoch in $\mathsf{TC}_{\mathsf{chall}}[0]$ by performing the update with its known ciphertexts and tokens. However, for the deterministic context, all the challenge-equal ciphertexts that the adversary knows are completely stored in the table $\mathsf{TC}_{\mathsf{chall}}$.

Trivial Wins via Keys and Ciphertexts. If there exists an epoch such that the adversary knows the epoch key and a valid challenge-equal ciphertext in the epoch, it leads to a trivial win as we discussed about the prior confidentiality work of [4] in Sect. 2.2. More specifically, this condition is equal to

$\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$ for UE schemes with no-directional key updates by Lemma 4.

Trivial Wins via Updates. We consider UE schemes with deterministic ciphertext updates. The adversary can query the oracle $\mathcal{O}_{\mathsf{sUpd}}$ to know the updated ciphertexts of the challenge input $(\hat{\mathsf{ct}}, \mathsf{ct})$ before the challenge phrase, which are recorded in the table $\mathsf{TC}_{\mathsf{non}}$, and win the game by comparing the updated ciphertext in challenge epoch with the challenge ciphertext. Furthermore, the adversary may also infer the updated ciphertext of $(\hat{\mathsf{ct}}, \mathsf{ct})$ in challenge epoch via inferred token $\Delta_{\mathsf{e},\hat{\mathsf{ct}}}$. But that is impossible since the key $\mathsf{k_e}$ in the challenge epoch is unknown to the adversary by $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$ and the challenge epoch $\mathsf{e} \in \mathsf{TC}_{\mathsf{chall}}[0]$. We track the above behaviors via the table $\mathsf{TC}_{\mathsf{non}}$, which are further checked in the challenge oracle $\mathcal{O}_{\mathsf{Chall}}$.[8]

Trivial Wins via Decryptions. We now also consider UE schemes with deterministic ciphertext updates. The table $\mathsf{TC}_{\mathsf{chall}}$ records all the challenge-equal ciphertexts known to the adversary in the game. The adversary can trivially win the game by querying the decryption oracle on the challenge-equal ciphertexts recorded on the table $\mathsf{TC}_{\mathsf{chall}}$.

For UE schemes with randomized ciphertext updates, the epoch set $\mathsf{TC}_{\mathsf{chall}}[0]$ records all the epochs in which the adversary can generate a valid challenge-equal ciphertext. The adversary can trivially win the game if the returned message of the decryption oracle in epochs in $\mathsf{TC}_{\mathsf{chall}}[0]$ is the challenge message or the plaintext of the challenge input ciphertext. These two trivial win conditions are checked in the oracle $\mathcal{O}_{\mathsf{Dec}}$.

Definition 5 (xxIND-UE-atk). *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Update}, \mathsf{Decrypt})$ *be a ciphertext-dependent updatable encryption scheme with no-directional key updates. For an adversary* $\mathcal{A}$ *and* $\mathrm{b} \in \{0, 1\}$*, we define the confidentiality experiment* $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ *in Fig. 5 for* $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ *and* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$.

We say UE *meets the* xxIND-UE-atk *confidentiality if there is a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathsf{Adv}^{\mathsf{xxIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*, where*

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

3.4 Firewall Techniques in Ciphertext-Dependent UE Schemes

Firewall Technique. In ciphertext-independent UE schemes, firewall technique was developed in [14,13] to facilitate the security proof by separating epochs into different regions. Inside an insulated regions (defined below), the simulation in the proof should appropriately respond the queries of the adversary, since it corrupts all tokens within this region. While outside, the simulation can generate tokens and epoch keys freely.

[8] These conditions are checked immediately in the confidentiality game to avoid the trivial wins. After the query phase, there is only one check left $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset$, without extra calculations and further checks of the extended leakages sets of keys, tokens and ciphertext in [6,11].

$\mathsf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$:

1 : $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct})) \leftarrow \mathcal{A}^{\mathcal{O}_1}(1^\lambda)$ // setup phase
2 : $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{chall}}$ on $(\mathsf{m}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ // challenge phase
3 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}_2}(1^\lambda)$ // response phase
4 : **if** $(\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset)$ **then**
5 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
6 : **return** b'

Fig. 5. The confidentiality game $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$. The adversary generates a challenge plaintext and a challenge ciphertext in the setup phase according to the oracles set $\mathcal{O}_1$ it has access to, which are then submitted to the challenger in the challenge phase. The adversary can continue to querying the oracle set $\mathcal{O}_2$ and eventually provides a guessing bit. The oracle sets $\mathcal{O}_1$ and $\mathcal{O}_2$ are shown in Table 1 and Fig. 4. The adversary loses the game if $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset$.

atk	$\mathcal{O}_1$	$\mathcal{O}_2$
CPA	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}}$	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}}$
CCA-1	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \boxed{\mathcal{O}_{\mathsf{Dec}}}, \mathcal{O}_{\mathsf{Corr}}$	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}}$
CCA	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \boxed{\mathcal{O}_{\mathsf{Dec}}}, \mathcal{O}_{\mathsf{Corr}}$	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \boxed{\mathcal{O}_{\mathsf{Dec}}}, \mathcal{O}_{\mathsf{Corr}}$

Table 1. Oracles that the adversary has access to before and after the challenge phase in the confidentiality game for different attacks. It can corrupt keys at any time during the game in all attacks via querying $\mathsf{O}_{\mathsf{Corr}}$, but is not allowed to query the decryption oracle in the CPA attack, limited to query the decryption oracle before the challenge in the CCA-1 attack, and free to query the decryption oracle in the CCA attack.

Definition 6 (Firewall, [14,13]). *An insulated region with firewalls* fwl *and* fwr*, denoted by* $\mathcal{FW}$*, is consecutive sequence of epochs* $(\mathsf{fwl}, \cdots, \mathsf{fwr})$ *for which:*

- *no key in the sequence of epochs* $\{\mathsf{fwl}, \cdots, \mathsf{fwr}\}$ *is corrupted;*
- *the tokens* Δ_{fwl} *and* $\Delta_{\mathsf{fwr}+1}$ *are not corrupted;*
- *all tokens* $\{\Delta_{\mathsf{fwl}+1}, \cdots, \Delta_{\mathsf{fwr}}\}$ *are corrupted.*

In ciphertext-dependent UE schemes, however, there are two kinds of tokens: challenge-equal tokens and non-challenge tokens, which are used to update challenge-equal and non-challenge ciphertexts respectively. We similarly define the insulated region, inside which all challenge-equal tokens are corrupted but no epoch key is corrupted.

Definition 7 (Firewall). *In ciphertext-dependent UE schemes, an insulated region with firewalls* fwl *and* fwr*, denoted by* $\mathcal{FW}$*, is consecutive sequence of epochs* $(\mathsf{fwl}, \cdots, \mathsf{fwr})$ *for which:*

- *no key in the sequence of epochs* $\{\mathsf{fwl}, \cdots, \mathsf{fwr}\}$ *is corrupted;*
- *no challenge-equal token in epoch* fwl *and* $\mathsf{fwr}+1$ *is corrupted;*
- *all challenge-equal tokens in epochs* $\{\mathsf{fwl}+1, \cdots, \mathsf{fwr}\}$ *are corrupted.*

For an adversary $\mathcal{A}$ in the xxIND-UE-atk game, suppose it asks for the challenge in epoch $\tilde{\mathsf{e}}$ and does not trigger the trivial win conditions. From the proof of Lemma 4, we know $\mathcal{A}$ cannot update a ciphertext from the epoch e_{end} to the start epoch e'_{start} of the next continuum. Therefore, we have $\mathsf{TC}_{\mathsf{chall}}[0] = \{\mathsf{e}_{start}, \cdots, \mathsf{e}_{end}\}$, i.e. the epoch set in which $\mathcal{A}$ knows a challenge-equal ciphertext is a consecutive continuum, which starts from the challenge epoch (that is $\mathsf{e}_{start} = \tilde{\mathsf{e}}$), and ends in the epoch e_{end} - the last epoch that the adversary queries the update oracle $\mathcal{O}_{\mathsf{sUpd}}$ on the challenge-equal ciphertext.

The epoch keys and tokens in the epoch in $\mathsf{TC}_{\mathsf{chall}}[0]$ have the following properties.

- $\mathcal{A}$ does not know the challenge-equal token in epoch $\mathsf{e}_{end}+1$, which follows from the proof of Lemma 4;
- $\mathcal{A}$ knows all challenge-equal tokens in epochs in $\{\mathsf{e}_{start}+1, \cdots, \mathsf{e}_{end}\}$, which is obtained when $\mathcal{A}$ queries the updates of challenge-equal ciphertexts via $\mathcal{O}_{\mathsf{sUpd}}$;
- $\mathcal{A}$ does not know any key in epochs in $\{\mathsf{e}_{start}, \cdots, \mathsf{e}_{end}\}$, since $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$,
- $\mathcal{A}$ does not know the challenge-equal token in epoch e_{start} from querying or inferring. Since the key in e_{start} is unknown to $\mathcal{A}$, it cannot infer any token in epoch e_{start}. For deterministic UE schemes, the adversary is prohibited to query the oracle $\mathcal{O}_{\mathsf{sUpd}}$, which will lead to $\mathsf{TC}_{\mathsf{non}}[\tilde{\mathsf{e}}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1$ (see the challenge oracle $\mathcal{O}_{\mathsf{chall}}$ in Fig. 4). For randomized UE schemes, the probability that two random tokens are equal is negligible.

We have the following lemma that follows from the discussion above, and Lemma 6 is a corollary of Lemma 5.

Lemma 5. *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Update}, \mathsf{Decrypt})$ *be a ciphertext-dependent updatable encryption scheme with no-directional key updates, and* $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ *and* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$*. For an* xxIND-UE-atk *adversary* $\mathcal{A}$ *against* UE*, the set of epoch in which* $\mathcal{A}$ *knows a challenge-equal ciphertext is one insulated region, starting from the challenge epoch and ending at the last epoch in which the adversary queries the* $\mathcal{O}_{\mathsf{sUpd}}$*.*

Lemma 6. *In ciphertext-dependent UE schemes, if the* xxIND-UE-atk *adversary knows a challenge-equal ciphertext in epoch* e*, then* e *must be in an insulated region.*

4 A CCA-1 Secure PKE Scheme

In this section, we first investigate a potential risk in the decryption algorithm of the PKE [15] and further propose a new simpler PKE scheme TDP, which will be used in Sect. 5 as the underlying primitive to build our updatable encryption.

Lemma 7. *The decryption algorithm in the PKE scheme [15] cannot decrypt ciphertexts correctly.*

Proof. Let $c = (\mu, \mathbf{b})$ be a valid ciphertext generated by the encryption algorithm, that is

$$\mathbf{b}^t = 2(\mathbf{s}^t\mathbf{A}_\mu \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod 2q, \tag{4}$$

for some vectors $\mathbf{s}$ and $\mathbf{e}$, and $\mathbf{A}_\mu = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G}]$. The decryption algorithm proceeds by calling $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}_\mu, \mathbf{H}_\mu, \mathbf{b} \bmod q)$, which returns $\mathbf{z}$ and $\mathbf{e}' = (\mathbf{e}_0, \mathbf{e}_1)$ such that

$$\mathbf{b}^t = \mathbf{z}^t\mathbf{A}_\mu + (\mathbf{e}')^t \mod q. \tag{5}$$

This step is feasible since the secret key $\mathbf{R}$ is a $\mathbf{G}$-trapdoor for $\mathbf{A}_\mu$. However, the decryption algorithm further regards $\mathbf{e}'$ as $\mathbf{e}$ and outputs $\mathsf{encode}^{-1}((\mathbf{b} - \mathbf{e}')^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])$ as the plaintext $\mathbf{m}$. In the following, we show $\mathbf{e} \not\equiv \mathbf{e}'$.

By Eq. (4), we know

$$\mathbf{b}^t = 2(\mathbf{s}^t\mathbf{A}_\mu \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q. \tag{6}$$

If $\mathbf{e}' = \mathbf{e}$, then there exists some $\mathbf{s}'$ such that

$$(\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t = (\mathbf{s}')^t\mathbf{A}_\mu = (\mathbf{s}')^t[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G}] \mod q. \tag{7}$$

As a necessary condition, we have

$$(\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} = (\mathbf{s}')^t\mathbf{A}_\mu \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} \mod q, \tag{8}$$

which implies $\mathbf{v}^t\mathbf{G} = (\mathbf{s}')^t\mathbf{H}_\mu\mathbf{G} \bmod q$ for which $\mathbf{v}^t\mathbf{G} = \mathsf{encode}(\mathbf{m})^t$ by the definition of the encode function, and therefore $\mathbf{v}^t\mathbf{H}_\mu^{-1} = (\mathbf{s}')^t \bmod q$. Then

$$(\mathbf{s}')^t\mathbf{A}_0 = \mathbf{v}^t\mathbf{H}_\mu^{-1}\mathbf{A}_0 \mod q.$$

By Eq. (7), we know $(\mathbf{s}')^t\mathbf{A}_0 = 0 \bmod q$ for any message $\mathbf{m}$, which equals $\mathbf{v}^t\mathbf{H}_\mu^{-1}\mathbf{A}_0 = 0 \bmod q$ for any $\mathbf{v}$. Therefore, $\mathbf{A}_0 \equiv 0 \bmod q$, which is a contradiction to the key generation algorithm.

4.1 A New PKE Scheme

In our PKE scheme TDP we construct a 1×3 block matrix as the encryption matrix $\mathbf{A}_\mu$. The secret key serves as the trapdoor for the first two blocks of $\mathbf{A}_\mu$ and is used in the decryption algorithm to ensure the correct output of the error items. In addition, we only encode the message to $\Lambda(\mathbf{G}^t)$ (by comparison, $\Lambda(\mathbf{G}^t)/2\Lambda(\mathbf{G}^t)$ in [15]), and perform the modular arithmetic operation on q instead of $2q$.

Some parameters involved in the construction are introduced as follows, in which we use standard asymptotic notations of O, Ω, ω. Let 1^λ be the security parameter and $\omega(\sqrt{\log n})$ is a fixed function.

- $\mathbf{G} = \mathbb{Z}_q^{n \times nk}$ is the gadget matrix in Sect. 2 to make the oracles $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{SampleD}^{\mathcal{O}}$ efficient. We choose sufficiently large prime power $q = p^e = \mathsf{poly}(\lambda)$ and $k = O(\log q) = O(\log n)$.
- $\bar{m} = O(nk)$ and $\mathcal{D} = D_{\mathbb{Z},\omega(\sqrt{\log n})}^{\bar{m} \times nk}$ so that $(\mathbf{A}, \mathbf{AR})$ is $\mathsf{negl}(\lambda)$-far from uniform for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$ from the leftover hash lemma. We define σ as the upper bound for $s_1(\mathbf{R})$ by Corollary 1, i.e., $\sigma := \omega(\sqrt{\log n}) \cdot O(\sqrt{nk})$.
- An efficient encoding $\mathsf{encode} : \{0,1\}^{nk} \to \Lambda(\mathbf{G}^t)$, which is defined by $\mathsf{encode}(\mathbf{m}) = \mathbf{Bm} \in \mathbb{Z}^{nk}$ where $\mathbf{m} \in \{0,1\}^{nk}$ and $\mathbf{B}$ is any basis of $\Lambda(\mathbf{G}^t)$. Note that this mapping can be efficiently inverted.
- α is an error rate for LWE such that $1/\alpha = 4 \cdot O(nk) \cdot \omega(\sqrt{\log n})$.

The PKE scheme TDP is described as follows:

- $\mathsf{TDP.KG}(1^\lambda)$: choose $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n \times \bar{m}}$, $\mathbf{R}_1$, $\mathbf{R}_2 \xleftarrow{\$} \mathcal{D}$ and let the encryption matrix $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2] \in \mathbb{Z}_q^{n \times m}$. The public key is $\mathsf{pk} = \mathbf{A}$ and the secret key is $\mathsf{sk} = \mathbf{R}_1$.

- $\mathsf{TDP.Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m} \in \{0,1\}^{nk})$: choose an invertible matrix $\mathbf{H}_\mu \in \mathbb{Z}_q^{n \times n}$, and let $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. Choose a random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and an error vector $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in D_{\mathbb{Z},\alpha q}^{\bar{m}} \times D_{\mathbb{Z},d}^{nk} \times D_{\mathbb{Z},d}^{nk}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m} \cdot (\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$. Let

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{9}$$

  where the first $\mathbf{0}$ has dimension $\bar{m}$ and the second has dimension nk. Output the ciphertext $c = (\mathbf{H}_\mu, \mathbf{b})$. Notice that $\mathbf{R}_1$ is a trapdoor for $[\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}]$.

- $\mathsf{TDP.Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: let $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$.
  1. **Recover the error item from the ciphertext.** If c or $\mathbf{b}$ does not parse, or $\mathbf{H}_\mu = \mathbf{0}$, output $\perp$. Otherwise parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)^t$. If $(\mathbf{b}_0, \mathbf{b}_1) \notin \Lambda([\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}]^t)$, output $\perp$. Otherwise, call $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}], [\mathbf{b}_0, \mathbf{b}_1], \mathbf{H}_\mu)$ by Lemma 1, which returns $\mathbf{z}$ and $(\mathbf{e}_0, \mathbf{e}_1)$ such that
$$(\mathbf{b}_0, \mathbf{b}_1)^t = \mathbf{z}^t[\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}] + (\mathbf{e}_0, \mathbf{e}_1)^t \mod q.$$
     If $\mathsf{Invert}^{\mathcal{O}}$ fails, output $\perp$. Invert $\mathbf{b}_2^t - \mathbf{z}^t\mathbf{A}_2$ and find the unique solution $\mathbf{u}, \mathbf{e}_2$ to the equation
$$\mathbf{b}_2^t - \mathbf{z}^t\mathbf{A}_2 = \mathbf{u}^t\mathbf{G} + \mathbf{e}_2^t \mod q,$$
     by Theorem 1, if they exist. If $\|\mathbf{e}_0\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\mathbf{e}_j\| \geq \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $j = 1, 2$, output $\perp$ (Lemma 11).
  2. **Recover the plaintext.** Output
$$\mathsf{encode}^{-1}\left((\mathbf{u}^t\mathbf{G})^t\right) \in \mathbb{Z}_2^{nk},$$
     if the result exists, otherwise output $\perp$.

4.2 Correctness and Security

We provide a full proof of the correctness (Lemma 8) and security (Lemma 9) in Sect. 5 for the updatable encryption scheme TDUE, which is based on TDP, regarded as a subcase of TDUE.

Lemma 8. *Our PKE scheme* TDP *decrypts correctly except with* $2^{-\Omega(n)}$ *failure probability.*

Proof. The proof is the same as Lemma 10 except the bound for the error vectors. By Corollary 1, we have $s_1(\mathbf{R}) = \omega(\sqrt{\log n}) \cdot O(\sqrt{nk})$. By Lemma 11, we have $\|\mathbf{e}_0\| \leq \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_i\| \leq \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $j = 1, 2$, except with negligible probability $2^{-\Omega(n)}$, where $\bar{m} = O(nk)$. Therefore,

$$\left\|(\mathbf{e}_0, \mathbf{e}_1)^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\right\|_\infty \leq \left\|(\mathbf{e}_0, \mathbf{e}_1)^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\right\| \leq \left\|\mathbf{e}_0^t\mathbf{R}\right\| + \|\mathbf{e}_1\| \leq \alpha q \cdot O(nk) \cdot \omega(\sqrt{\log n}),$$

which is further smaller than $q/4$ by the α we choose, and $\|\mathbf{e}_2\|_\infty \leq q/4$ for the same reason.

Lemma 9. *Our PKE scheme* TDP *is* CCA-1*secure if the* LWE *problem is hard.*

5 A CCA-1 Secure Updatable Encryption Scheme

Based on the TDP scheme in Sect. 4, we construct a new UE scheme, which is IND-UE-CCA-1 secure under the assumption of the LWE hardness.

5.1 Construction

At a high level, the ciphertext of a plaintext $\mathbf{m}$ is of the form $(\hat{\mathsf{ct}}, \mathsf{ct}) = (\mathbf{H}_\mu, \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m})^t)$ generated by the TDP.Enc, where the encryption matrix $\mathbf{A}_\mu$ can be computed from the public key and the invertible matrix $\mathbf{H}_\mu$. The secret key $\mathbf{R}_1$ serves as a trapdoor for the first two block matrices of $\mathbf{A}_\mu$, and therefore can be used to generate a transition matrix $\mathbf{M}$ between the previous and new encryption matrices, i.e. $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ where $\mathbf{A}'_\mu$ is the new encryption matrix. The invertible matrix $\mathbf{H}_\mu$ is randomly changed in next epoch to avoid the adversary trivially winning the security game by comparing $\mathbf{H}_\mu$. Meanwhile, the ciphertext of message $\mathbf{0}$ is added to the new ciphertext to ensure the randomness. Then the update algorithm runs by substituting $\mathbf{H}_\mu$ by $\mathbf{H}_\mu{}'$ and computing $\mathbf{b}^t \cdot \mathbf{M} + \mathsf{Enc}(\mathbf{0})$, which equals $\mathbf{s}'^t\mathbf{A}'_\mu + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m})^t)$. Therefore the updated ciphertext is of the same shape as the encrypted ciphertext with new randomness and error, which guarantees the forward and post-compromise security [14], i.e., preserving the confidentiality in the presence of temporary key compromises.

We use the same parameters as in Sect. 4.1 except

- α is an error rate for LWE such that $1/\alpha = 4l \cdot \omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}$ where l is the maximal number of update that the scheme can support.

– $\tau = \sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\sum_{\mathbf{G}}) + 1} \cdot \omega(\sqrt{\log n})$ is smallest Gaussian parameter for the discrete Gaussian distribution from which the sampling algorithm $\mathsf{SampleD}^{\mathcal{O}}$ can sample vectors, where $s_1(\sum_{\mathbf{G}}) = 5$ by Theorem 1.

The UE scheme TDUE is described as follows.

– $\mathsf{TDUE.KG}(1^\lambda)$: output $\mathsf{TDP.KG}(1^\lambda)$.
– $\mathsf{TDUE.Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m} \in \{0,1\}^{nk})$: output $\mathsf{TDP.Enc}(\mathbf{A}, \mathbf{m})$.
– $\mathsf{TDUE.Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: output $\mathsf{TDP.Dec}(\mathbf{R}_1, (\mathbf{H}_\mu, \mathbf{b}))$.
– $\mathsf{TDUE.TokenGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}', \mathbf{H}_\mu)$: parse $\mathsf{pk} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2]$, $\mathsf{sk} = \mathbf{R}_1$, and $\mathsf{pk}' = [\mathbf{A}'_0 \mid \mathbf{A}'_1 \mid \mathbf{A}'_2]$.
  1. Generate a random invertible matrix $\mathbf{H}'_\mu$ and let $\mathbf{A}'_\mu = [\mathbf{A}'_0 \mid \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G} \mid \mathbf{A}'_2]$ be the new encryption matrix. We first generate a transition matrix $\mathbf{M}$ for which $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ in the following steps 2, 3, 4, and then compute the encryption of the message $\mathbf{0}$ under $\mathbf{A}'_\mu$ in step 5.
  2. Call $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_\mu, \mathbf{A}'_0, \tau)$ (Remark 1 and $\mathbf{R}_1$ is a trapdoor for $[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}]$), which returns an $(\bar{m} + nk) \times \bar{m}$ matrix, parsed as $\mathbf{X}_{00} \in \mathbb{Z}^{\bar{m} \times \bar{m}}$ and $\mathbf{X}_{10} \in \mathbb{Z}^{nk \times \bar{m}}$ with Gaussian entries of parameter τ, satisfying
$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{00} \\ \mathbf{X}_{10} \end{bmatrix} = \mathbf{A}'_0. \tag{10}$$
  3. Call $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_\mu, \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G}, \tau\sqrt{\bar{m}/2})$, which returns $\mathbf{X}_{01} \in \mathbb{Z}_q^{\bar{m} \times nk}$ and $\mathbf{X}_{11} \in \mathbb{Z}_q^{nk \times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$ such that
$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{01} \\ \mathbf{X}_{11} \end{bmatrix} = \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G}. \tag{11}$$
  4. Continue calling the sample oracle $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_1, \mathbf{A}'_2 - \mathbf{A}_2, \tau\sqrt{\bar{m}/2})$ and obtain $\mathbf{X}_{02} \in \mathbb{Z}_q^{\bar{m} \times nk}$ and $\mathbf{X}_{12} \in \mathbb{Z}_q^{nk \times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$ such that
$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix} \mathbf{X}_{02} \\ \mathbf{X}_{12} \end{bmatrix} = \mathbf{A}'_2 - \mathbf{A}_2. \tag{12}$$
     Note that $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. Define the transition matrix from $\mathbf{A}_\mu$ to $\mathbf{A}'_\mu$ as
$$\mathbf{M} = \begin{bmatrix} \mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02} \\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{13}$$
     Then $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ from Eq. (10), (11), (12).
  5. Let $\mathbf{b}_0$ be the ciphertext of message $\mathbf{m} = \mathbf{0}$ under the public key pk' with the invertible matrix $\mathbf{H}'_\mu$ generated in step 1. That is,
$$\mathbf{b}_0^t = (\mathbf{s}')^t\mathbf{A}'_\mu + (\mathbf{e}')^t \mod q.$$

6. Output $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$.

– $\mathsf{Update}(\Delta, c = (\mathbf{H}_\mu, \mathbf{b}))$: parse $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$ and compute

$$(\mathbf{b}')^t = \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \mod q,$$

and output $c' = (\mathbf{H}'_\mu, \mathbf{b}')$.

5.2 Correctness

We prove that the decryption algorithm in our scheme can perform correctly with overwhelming probability. Note that the second component in the ciphertext generated by the update algorithm (updated ciphertext) is as follows.

$$\begin{aligned}(\mathbf{b}')^t &= \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ &= \left[\mathbf{s}^t \mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right] \mathbf{M} + (\mathbf{s}')^t \mathbf{A}'_\mu + (\mathbf{e}')^t \\ &= (\mathbf{s} + \mathbf{s}')^t \mathbf{A}'_\mu + \left(\mathbf{e}^t \mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod q. \end{aligned} \tag{14}$$

The third equation holds because $\mathbf{A}_\mu \mathbf{M} = \mathbf{A}'_\mu$ and the last nk rows in $\mathbf{M}$ is $[\mathbf{0}\ \mathbf{0}\ \mathbf{I}]$. Therefore the item $(\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t$ stays the same when multiplied by $\mathbf{M}$. Then the updated ciphertext shares the same form with the fresh ciphertext (generated by the encryption algorithm), except that the update algorithm enlarges the error terms by $\mathbf{e}^t \mathbf{M} + (\mathbf{e}')^t$, which may bring the failure to the invert $\mathsf{Invert}^{\mathcal{O}}$ and further influence the correctness of the decryption algorithm. In the following, we show the "increased" errors in the updated ciphertexts can be tolerated by the decryption algorithm via the appropriate selection on the LWE parameter α.

Lemma 10. *Our UE scheme TDUE decrypts correctly except with $2^{-\Omega(n)}$ failure probability.*

Proof. Since the decryption on the fresh ciphertext (from Enc) is a subcase of that on the updated ciphertext (from Upd), we choose to prove that the decryption algorithm can output a correct plaintext after l updates from epoch 0, where l is the maximal update number.

Let $(\mathrm{pk}_\mathrm{e}, \mathrm{sk}_\mathrm{e} = (\mathbf{R}_{1,\mathrm{e}}, \mathbf{R}_{2,\mathrm{e}}))_{0 \le \mathrm{e} \le l} \leftarrow \mathsf{KG}(1^n)$ be the public and secret key in epoch e. For a random plaintext $\mathbf{m} \in \{0,1\}^{nk}$, let c_e be the ciphertext of $\mathbf{m}$ in epoch e and $c_0 = \mathsf{Enc}(\mathbf{m}) = (\mathbf{H}_{\mu,0}, \mathbf{s}_0^t \mathbf{A}_{\mu,0} + \mathbf{e}_0^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t)$. For $1 \le i \le l$, let the token in epoch i be $\Delta_i = (\mathbf{M}_i, \mathbf{b}_{0,i}, \mathbf{H}_{\mu,i})$, where $\mathbf{b}_{0,i}$ is the fresh ciphertext of message $\mathbf{0}$ in epoch i, i.e. $\mathbf{b}_{0,i}^t = \mathbf{s}_i^t \mathbf{A}_{\mu,i} + \mathbf{e}_i^t$ and $\mathbf{A}_{\mu,i} = [\mathbf{A}_{0,i} \mid \mathbf{A}_{1,i} + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}]$ defined in the encryption algorithm. Then iteratively using Eq. (14), we know the updated ciphertext of $\mathbf{m}$ at epoch l is $c_l = (\mathbf{H}_{\mu,l}, \mathbf{b}_l)$ where

$$\mathbf{b}_l^t = \left(\sum_{i=0}^{l} \mathbf{s}_i\right)^t \mathbf{A}_{\mu,l} + \sum_{i=0}^{l} \left(\mathbf{e}_i^t \cdot \prod_{j=i+1}^{l} \mathbf{M}_j\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t.$$

Let $\mathbf{e}_l^t = \sum_{i=0}^{l}(\mathbf{e}_i^t \prod_{j=i+1}^{l} \mathbf{M}_j) = (\mathbf{e}_l^0, \mathbf{e}_l^1, \mathbf{e}_l^2)^t$. We provide in Appendix B the upper bound for the error in $\mathbf{b}_l$ that

$$\left\|(\mathbf{e}_l^0, \mathbf{e}_l^1, \mathbf{e}_l^2)^t \cdot \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix}\right\|_\infty < q/4 \quad \text{and} \quad \left\|(\mathbf{e}_l^2)\right\|_\infty < q/4, \tag{15}$$

except with probability $2^{-\Omega(n)}$ via the appropriate parameter selection for the scheme. Let $\mathbf{b}_l^t = (\mathbf{b}_l^0, \mathbf{b}_l^1, \mathbf{b}_l^2)^t$. Then the call to $\mathsf{Invert}^{\mathcal{O}}$ made by $\mathsf{Dec}(\mathsf{sk}_l, (\mathbf{H}_{\mu,l}, \mathbf{b}_l))$ returns $\mathbf{z}$ and $(\mathbf{e}_l^0, \mathbf{e}_l^1)$ correctly, for which

$$(\mathbf{b}_l^0, \mathbf{b}_l^1)^t = \mathbf{z}^t[\mathbf{A}_{0,l} \mid \mathbf{A}_{1,l} + \mathbf{H}_{\mu,l}\mathbf{G}] + (\mathbf{e}_l^0, \mathbf{e}_l^1)^t \mod q.$$

It follows that

$$(\mathbf{b}_l^2)^t - \mathbf{z}^t\mathbf{A}_{2,l} = (\mathbf{e}_l^2)^t + \mathsf{encode}(\mathbf{m})^t,$$

where $\left\|\mathbf{e}_l^2\right\| < q/4$ by Inequality (15) and $\mathsf{encode}(\mathbf{m})^t = \mathbf{u}^t\mathbf{G}$ for some $\mathbf{u} \in \mathbb{Z}_q^{nk}$ by the definition of encode. Inverting $(\mathbf{b}_l^2)^t - \mathbf{z}^t\mathbf{A}_{2,l}$, we will find the unique $\mathbf{e}_l^2$ and $\mathbf{u}$. Finally, we have

$$\mathsf{encode}^{-1}\left((\mathbf{u}^t\mathbf{G})^t\right) = \mathsf{encode}^{-1}\left(\mathsf{encode}(\mathbf{m})\right) = \mathbf{m}.$$

Therefore, the decryption algorithm Dec outputs $\mathbf{m}$ as desired.

5.3 Security Proof

In this section, we show that our scheme is $\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}$ secure under the hardness assumption of LWE. In general, we take three steps, see Fig. 2, to bound the advantage of the adversary. In the first step, we build a hybrid game H_i for any epoch i, which follows [11,16]. To the left of i, the game H_i returns the real challenge-equal ciphertexts; while, to the right of i, it returns random ciphertexts. In the second step, we adopt the firewall technique, which was developed in ciphertext-independent schemes [14,13,6,11] to set up a modified game of H_i that enables a simulation of the modified game in the third step. This is feasible due to our new confidentiality notion and Lemma 6. At last, we play a sequence of games. In the first game, we simulate valid epoch keys and tokens in firewall areas, showing how to simulate the responses to all the queries. This is the main part we attempt to overcome from [12], where the adversary in the security game of a PRE scheme can easily distinguish the simulated and the real games and also the simulation cannot respond multiple times of update queries. We then modify the challenge-equal ciphertext in the second game, which delivers the construction of a reduction that solves the LWE by simulating the second game to the adversary.

Theorem 3. *Let* UE *be the updatable encryption scheme described in Sect. 5. For any* $\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}$ *adversary* $\mathcal{A}$ *against* UE*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{LWE}_{n,q,\alpha}$ *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}}_{\mathsf{UE},\mathcal{A}}(1^\lambda) \leq 2(l+1)^3 \cdot \left[(l+2) \cdot \mathsf{negl}(\lambda) + (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}}) \cdot 2^{-\Omega(n)} + \mathsf{Adv}^{\mathsf{LWE}}_{n,q,\alpha}(\mathcal{B})\right].$$

Step	Process
Step 1	H_i : real ct (→ i) random ct → epoch
Step 2	$\mathcal{G}_i$: real ct (fwl → i) random ct (i → fwr) → epoch
Step 3	Game 1: Simulate epoch keys and tokens in $\{\mathsf{fwl}, \cdots, \mathsf{fwr}\}$ Game 2: Modify the challenge-equal ciphertext Game 3: A reduction solving LWE and simulating Game 2

Table 2. Steps in the security proof. Within an insulated region, the reduction should appropriately respond to all the queries made by the adversary, since all challenge-equal tokens in this area are known to it. Outside the region, the reduction can generate epoch keys and tokens freely. ct is the abbreviation of ciphertext.

where l is the maximal number of ciphertext updates that the scheme UE supports, and n_{Dec} and n_{sUpd} are the number of queries to the oracles $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{sUpd}}$, respectively.

Proof. We proceed via three steps that are defined as follows:

Step 1. Consider a sequence of hybrid experiments $H_0^{\mathrm{b}}, \cdots, H_{l+1}^{\mathrm{b}}$ for $\mathrm{b} \in \{0,1\}$, where H_i^{b} is the same as the $\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}$ game except when the adversary sends a query with challenge input $(\bar{\mathbf{m}}, \bar{c})$ to $\mathcal{O}.\mathsf{Chall}$ in epoch j:

- if $j < i$, return an honestly generated challenge-equal ciphertext. That is, for $\mathrm{b} = 0$, return the update ciphertext of $\mathsf{Enc}(\mathsf{pk}_j, \bar{\mathbf{m}})$; otherwise return the update ciphertext of $\mathsf{Upd}(\Delta_j, \bar{c})$.
- if $j \geq i$, return a random ciphertext.

We see that H_{l+1}^{b} is the same as $\mathsf{Exp}_{\mathsf{UE},\mathcal{A}}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1\text{-}b}}$, and $H_0^0 = H_0^1$ since all challenge responses are the same. Then We have

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

Our goal is to prove $\left|\Pr\left[H_{i+1}^{\mathrm{b}} = 1\right] - \Pr\left[H_i^{\mathrm{b}} = 1\right]\right| \leq \mathsf{negl}(\lambda)$ for any i and b.

Step 2. Since the responses in all epochs except i will be the same in both games H_{i+1}^{b} and H_i^{b} for $\mathrm{b} \in \{0,1\}$, we assume the adversary who tries to distinguish the two games asks for the challenge in epoch i (by $\mathcal{O}.\mathsf{Chall}$). Therefore by Lemma 6, there exists an insulated region $[\mathsf{fwl}, \mathsf{fwr}]$ around epoch i such that epoch keys in $(\mathsf{fwl}, \cdots, \mathsf{fwr})$ and no challenge-equal tokens in epochs fwl and $\mathsf{fwr}+1$ are corrupted, otherwise trivial win conditions will be triggered. In fact, we have $\mathsf{fwl} = i$ by Lemma 5.

We then define a new game $\mathcal{G}_i^{\mathrm{b}}$ which is the same as H_i^{b} except that the game chooses two random numbers $\mathsf{fwl}, \mathsf{fwr} \leftarrow \{0, \cdots, l\}$. If the adversary corrupts one of epoch keys in $[\mathsf{fwl}, \mathsf{fwr}]$, or one of tokens in epochs fwl and $\mathsf{fwr}+1$, the game aborts. The guess is correct with probability $1/(l+1)^2$. Then we have

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

Our goal is to prove $\left|\Pr\left[\mathcal{G}_{i+1}^{\mathrm{b}}=1\right]-\Pr\left[\mathcal{G}_i^{\mathrm{b}}=1\right]\right| \leq \mathsf{negl}(\lambda)$ for any i and b.

Step 3. For $\mathrm{b} \in \{0,1\}$ let $\mathcal{A}_i$ be an adversary who tries to distinguish $\mathcal{G}_{i+1}^{\mathrm{b}}$ from $\mathcal{G}_i^{\mathrm{b}}$. To provide an upper bound for the advantage of $\mathcal{A}_i$, we define a sequence of games as follows.

Game 0:

For a random number $d \xleftarrow{\$} \{0,1\}$, if $d=0$ the game plays $\mathcal{G}_i^b$ to $\mathcal{A}_i$; otherwise it plays $\mathcal{G}_{i+1}^b$ to $\mathcal{A}_i$. Denote E_j be the event that the adversary succeeds in guessing d in the **Game** j. Then we have

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

Game 1:

We consider a modified game that is the same as **Game 0**, except how the public keys and tokens are generated in epochs from i (=fwl) to fwr. The overall idea to change the public key in epoch i with a special form that ensures the embedding of LWE samples to the challenge ciphertexts in epoch i. But this form of public key in epoch i may cause the failure of generating token Δ_{i+1}. We leverage a method introduced in [12] to simulate tokens in epoch from $i+1$ to fwr by generating the public keys from $i+2$ to fwr with the same form as pk_i. Our simulation overcomes some proof risks in [12], in which the simulated token can be distinguished[9] but also tackle the shortcoming that ciphertexts can only be updated one time.

(1) At the start of the game, we pre-generate random invertible matrices $\{\mathbf{H}_{\mu,j}^*\}_{j=i}^{\mathsf{fwr}} \in \mathbb{Z}_q^{n\times n}$ that will be used to generate Δ_j and pk_j.

(2) We generate all keys from 0 to l as **Game 0** by running the key generation algorithm, except for $\mathsf{pk}_{\mathrm{i}}, \cdots, \mathsf{pk}_{\mathsf{fwr}}$.

(3) **Public key in epoch** i**.** We choose random $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n\times \bar{m}}$ and secret key $\mathbf{R}_{1,i} \in \mathcal{D}$, and let the public key be

$$\mathsf{pk}_i = \left[\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}_{\mu,i}^*\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\right].$$

Notice that pk_i is still $\mathsf{negl}(\lambda)$-far from the uniform for any choice of $\mathbf{H}_{\mu,i}^*$. This design is to generate challenge ciphertexts of the form in the following Eq. (20) and further facilitate the simulation in Game 3.

[9] This was first observed in [9]. But the distribution of the simulated secret key in their proof is different from that in the real scheme they constructed, which may lead to the potential failure of decryption.

Public key in epoch $i+1$. In epoch $i+1$, we choose two matrices $\mathbf{X}_{00,i+1} \in \mathbb{Z}_q^{\bar{m}\times nk}$ and $\mathbf{X}_{10,i+1} \in \mathbb{Z}_q^{nk\times nk}$ from a Gaussian distribution with parameter τ and let

$$\mathbf{A}_{0,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \cdot \begin{bmatrix} \mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1} \end{bmatrix}, \tag{16}$$

which is still a $\mathsf{negl}(\lambda)$-far from the uniform. Then we choose a random matrix $\mathbf{R}_{1,i+1} \in \mathbb{Z}^{\bar{m}\times nk}$ whose entry equals to 0 with probability 1/2 and ± 1 with probability 1/4 each, and two random matrices $\mathbf{X}_{02,i+1}, \mathbf{X}_{12,i+1} \in \mathbb{Z}_q^{\bar{m}\times nk} \times \mathbb{Z}_q^{nk\times nk}$ with Gaussian entries of parameter $\tau\sqrt{\bar{m}/2}$. Compute

$$\mathbf{A}_{1,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix} \mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1} \end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{17}$$

$$\mathbf{A}_{2,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix} \mathbf{X}_{02,i+1} \\ \mathbf{X}_{12,i+1} \end{bmatrix} - \mathbf{A}_{2,i}, \tag{18}$$

where $\mathbf{A}_{2,i} = -\mathbf{A}_{0,i}\mathbf{R}_{2,i}$. Let the public key in epoch $i+1$ be

$$\mathsf{pk}_{i+1} = [\mathbf{A}_{0,i+1} \mid \mathbf{A}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} \mid \mathbf{A}_{2,i+1}].$$

and the secret key be $\mathsf{sk}_{i+1} = \mathbf{R}_{1,i+1}$.

Remark 5. By Lemma 15, we know $(\mathbf{A}_{0,i+1}, \mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1})$ is $\mathsf{negl}(n)$-close to the uniform if $\bar{m} \geq n\log(q) + 2\log(nk/\delta)$ for some small $\delta = \mathsf{negl}(n)$.

Remark 6. Every entry of $\begin{bmatrix} \mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1} \end{bmatrix} \cdot \mathbf{R}_{1,i+1}$ is an inner product of a $\bar{m}$-vector from a Gaussian distribution with parameter τ and a $\{0, 1, -1\}^{\bar{m}}$ vector with half of the coordinates equal to 0, which is therefore a vector from a Gaussian distribution with parameter $\tau\sqrt{\bar{m}/2}$ by Lemma 16 that is the same distribution as the second-column block matrix of the token in Game 0.

Remark 7. We do not require $\mathbf{A}_{2,i+1}$ to be presented in the form of $\mathbf{A}_{0,i+1}\mathbf{R}_{2,i+1}$ for some matrix $\mathbf{R}_{2,i+1}$ as in the real game, and we will show that this does not affect the responses to the queries.

Public key in epochs from $i+2$ to fwr. For any epoch j in $\{i+2, \cdots, \mathsf{fwr}\}$, we iteratively generate the public key pk_j and the secret key sk_j as pk_{i+1} and sk_{i+1} respectively.

(4) **Simulating challenge-equal queries.** An overview of the oracles that the adversary has access to on challenge-equal ciphertexts is summarised in Table 3. We should respond the queries on challenge-equal ciphertexts and challenge-equal tokens in epochs from i to fwr, but do not need to answer the decryption on the challenge-equal ciphertext.

Ciphertext. For challenge-equal ciphertexts in epoch i, we notice that a random invertible matrix should be generated when updating ciphertexts and encrypting messages; here we use the special pre-generated invertible

Simulation	Challenge-equal	
	i	$i+1$
Public Key	$\begin{matrix}\mathbf{A}_{0,i} \\ -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G} \\ -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\end{matrix}$	$\begin{matrix}\mathbf{A}_{0,i+1} \\ -\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} \\ \mathbf{A}_{2,i+1}\end{matrix}$
$\mathsf{Enc}/\mathbf{A}_\mu$	$\begin{matrix}\mathbf{A}_{0,i} \\ -\mathbf{A}_{0,i}\mathbf{R}_{1,i} \\ -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\end{matrix}$	$\begin{matrix}\mathbf{A}_{0,i+1} \\ -\mathbf{A}_{0,i}\mathbf{R}_{1,i+1} \\ \mathbf{A}_{2,i+1}\end{matrix}$
Dec	–	–
TokenGen	Eq. (23)	
Update	$\mathbf{b}^t_{i+1} = \mathbf{b}^t_i\Delta_{i+1} + \mathbf{b}^t_0$	

Table 3. Simulation of the responses to the queries on challenge-equal ciphertexts. When the adversary queries the oracle $\mathcal{O}.\mathsf{sUpd}$, the simulation returns the output of TokenGen and Update simultaneously. Public keys and ciphertexts (only $\mathbf{A}_\mu$ is shown) are represented by the block matrices.

matrix $\mathbf{H}^*_{\mu,i}$ in both of the two algorithms, which is randomly generated in step (1) and is unknown to the adversary conditioned on the public keys we sampled in step (3). Then the challenge ciphertext in epoch i either from updating or from fresh encryption is in the form $c_i = (\mathbf{H}^*_{\mu,i}, \mathbf{b}_i)$ where (for simplicity, we skip the modular arithmetic operation in the ciphertexts)

$$\begin{aligned}\mathbf{b}^t_i &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}^*_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t, \end{aligned} \tag{19}$$

for some $\mathbf{s}, \mathbf{e}$ and $b \in \{0,1\}$, and $\mathbf{m}_0 = \bar{\mathbf{m}}$ and $\mathbf{m}_1$ is the plaintext of the challenge ciphertext $\bar{c}$. Similarly, for updated ciphertexts in any epoch $j \in \{i+1, \cdots, \mathsf{fwr}\}$ under the public key pk_j defined in step (3), we use the pre-generated invertible matrix $\mathbf{H}^*_{\mu,j}$ in the update algorithm. And the challenge ciphertext is in the form $c_j = (\mathbf{H}^*_{\mu,j}, \mathbf{b}_j)$ where

$$\begin{aligned}\mathbf{b}^t_j &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid (-\mathbf{A}_{0,j}\mathbf{R}_{1,j} - \mathbf{H}^*_{\mu,j}\mathbf{G}) + \mathbf{H}^*_{\mu,j}\mathbf{G} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid -\mathbf{A}_{0,j}\mathbf{R}_{1,j} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t, \end{aligned} \tag{20}$$

for some $\mathbf{s}, \mathbf{e}$.

Token. For challenge-equal tokens in epoch $i+1$, we set

$$\begin{bmatrix}\mathbf{X}_{01,i+1} \\ \mathbf{X}_{11,i+1}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1}\end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{21}$$

and

$$\mathbf{M}_{i+1} = \begin{bmatrix} \mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{22}$$

and $\mathbf{b}_{0,i+1}$ is the ciphertext of message $\mathbf{0}$ under pk_{i+1} with the invertible matrix $\mathbf{H}^*_{\mu,i+1}$, i.e., $\mathbf{b}_{0,i+1} = \mathbf{s}^t\mathbf{A}_{\mu,i+1} + \mathbf{e}^t$ for some $\mathbf{s}$ and $\mathbf{e}$. Based on Eq. (16), (17), and (18), we know that $\mathbf{M}_{i+1}$ is the transition matrix from $\mathbf{A}_{\mu,i}$ to $\mathbf{A}_{\mu,i+1}$: $\mathbf{A}_{\mu,i}\mathbf{M}_{i+1} = \mathbf{A}_{\mu,i+1}$, and moreover has the distribution $\mathsf{negl}(n)$-far from the distribution of the token in **Game 0** by Remark 6 and the distribution of $\mathbf{X}_{00,i+1}, \mathbf{X}_{10,i+1}, \mathbf{X}_{02,i+1}, \mathbf{X}_{12,i+1}$ we chose in step (3). Then we have

$$\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1}) \tag{23}$$

is a valid challenge-equal token in epoch i+1. Similarly, we generate challenge-equal tokens in epochs from $i+2$ to fwr.

(5) **Simulating non-challenge queries.** An overview of the oracles that the adversary has access to on non-challenge ciphertexts is summarised in Table 4. We should respond the queries on the encryption, update, as well as the decryption in all epochs. Since keys outside the insulated region are truly generated in step (2), we focus on the simulation inside the region.

Simulation	Non-challenge	
	i	$i+1$
Public Key	$\begin{matrix}\mathbf{A}_{0,i} \\ -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G} \\ -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\end{matrix}$	$\begin{matrix}\mathbf{A}_{0,i+1} \\ -\mathbf{A}_{0,i+1}\mathbf{R}_{2,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} \\ \mathbf{A}_{2,i+1}\end{matrix}$
$\mathsf{Enc}/\mathbf{A}_\mu$	$\begin{matrix}\mathbf{A}_{0,i} \\ -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G} + \mathbf{H}_{\mu,i}\mathbf{G} \\ -\mathbf{A}_{0,i}\mathbf{R}_{2,i}\end{matrix}$	$\begin{matrix}\mathbf{A}_{0,i+1} \\ -\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G} + \mathbf{H}_{\mu,i+1}\mathbf{G} \\ \mathbf{A}_{2,i+1}\end{matrix}$
Dec	SimDec	
$\mathsf{TokenGen}$	$\Delta_{i+1} = \mathsf{TokenGen}(\ )$	
Update	$\mathbf{b}^t_{i+1} = \mathbf{b}^t_i\Delta_{i+1} + \mathbf{b}^t_0$	

Table 4. Simulation to the queries on non-challenge ciphertexts. The algorithm SimDec is defined below. Game 1 correctly simulates all the responses if the random generated matrix $\mathbf{H}_{\mu,j} \neq \mathbf{H}^*_{\mu,j}$ for $j \in \{i+1, \mathsf{fwr}\}$.

Ciphertext. For non-challenge ciphertexts in epoch i, we perform the encryption and update algorithm as Game 0 by generating random invertible

matrices $\mathbf{H}_{i,\mu}$. The non-challenge ciphertexts in epoch i are in the form $c_i = (\mathbf{H}_{\mu,i}, \mathbf{b}_i)$ where

$$\begin{aligned}\mathbf{b}_i^t = \mathbf{s}^t[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] \\ + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t,\end{aligned} \tag{24}$$

for some $\mathbf{s}, \mathbf{e}$ and $b \in \{0,1\}$, and $\mathbf{m}_0 = \bar{\mathbf{m}}$ and $\mathbf{m}_1$ is the plaintext of the challenge ciphertext $\bar{c}$. Similarly, for non-challenge ciphertexts in any epoch $j \in \{i+1, \cdots, \mathsf{fwr}\}$, we randomly generate invertible matrices $\mathbf{H}^*_{\mu,j}$ in the update algorithm and the encryption algorithm. The challenge ciphertext is in the form $c_j = (\mathbf{H}_{\mu,j}, \mathbf{b}_j)$ where

$$\begin{aligned}\mathbf{b}_j^t = \mathbf{s}^t[\mathbf{A}_{0,j} \mid (-\mathbf{A}_{0,j}\mathbf{R}_{1,j} - \mathbf{H}^*_{\mu,j}\mathbf{G}) + \mathbf{H}_{\mu,j}\mathbf{G} \mid \mathbf{A}_{2,j}] \\ + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_b))^t,\end{aligned} \tag{25}$$

for some $\mathbf{s}, \mathbf{e}$ and $b \in \{0,1\}$.

Decryption. To aid with update and decryption queries, we choose an arbitrary (not necessarily short) $\hat{\mathbf{R}}_i \in \mathbb{Z}_q^{\bar{m} \times nk}$ such that $-\mathbf{A}_{0,i}\hat{\mathbf{R}}_i = -\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}$. Then we know $\hat{\mathbf{R}}_i$ is a trapdoor for $\mathbf{A}_{\mu,i,01} = [\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$. We use the algorithm SimDec described below to simulate the decryption algorithm, and the simulated decryption algorithm can also be applied in epoch from $i+1$ to fwr.

- $\mathsf{SimDec}(\mathsf{sk} = \mathbf{R}_{1,i}, c = (\mathbf{H}_{\mu,i}, \mathbf{b}))$:
  1. If c or $\mathbf{b}$ does not parse, or $\mathbf{H}_{\mu,i} = \mathbf{0}$ or $\mathbf{H}^*_{\mu,i}$, output $\perp$. Otherwise parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)^t$. If $(\mathbf{b}_0, \mathbf{b}_1) \notin \Lambda(\mathbf{A}^t_{\mu,i,01})$, output $\perp$.
  2. Call $\mathsf{Invert}^{\mathcal{O}}(\hat{\mathbf{R}}_i, \mathbf{A}_{\mu,i,01}, (\mathbf{b}_0, \mathbf{b}_1) \bmod q, \mathbf{H}_{\mu,i})$ to get $\mathbf{z}$ and $\mathbf{e}^t = (\mathbf{e}_0, \mathbf{e}_1)^t$ such that $(\mathbf{b}_0, \mathbf{b}_1)^t = \mathbf{z}^t\mathbf{A}_{\mu,i,01} + (\mathbf{e}_0, \mathbf{e}_1)^t \bmod q$ (Lemma 1). If $\mathsf{Invert}^{\mathcal{O}}$ fails, output $\perp$.
  3. Let $\mathbf{u}, \mathbf{e}_2$ be the unique solution to the equation

$$\mathbf{b}_2^t - \mathbf{z}^t\mathbf{A}_{2,i} = \mathbf{u}^t\mathbf{G} + \mathbf{e}_2^t \mod q,$$

     if they exist; otherwise output $\perp$. Let $\mathbf{m} = \mathsf{encode}^{-1}(\mathbf{u}^t\mathbf{G} \bmod q)$.
  4. If $\|\mathbf{e}_0\| \geq \alpha q\sqrt{\bar{m}}$ or $\|\mathbf{e}_j\| \geq \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $j = 1, 2$, output $\perp$. Otherwise output $\mathbf{m}$.

Whenever $\mathbf{H}_{\mu,i} \neq \mathbf{H}^*_{\mu,i}$ which is the case with probability $2^{-\Omega(n)}$, the call to the invert algorithm returns $\mathbf{z}$ and $\mathbf{e}$ properly if they exist, and SimDec has the same decryption ability as Dec.

Token. By construction, the matrix $\hat{\mathbf{R}}_i$ is the trapdoor for the $[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$. We can use the real token generation algorithm to generate matrices $\{\mathbf{X}_{i,00}, \mathbf{X}_{i,01}, \mathbf{X}_{i,02}, \mathbf{X}_{i,10}, \mathbf{X}_{i,11}, \mathbf{X}_{i,12}\}$ with the same distributions as in Game 0 by calling the invert algorithm on $[\mathbf{A}_{0,i} \mid$

$(-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G}]$ with the trapdoor $\hat{\mathbf{R}}_i$ such that

$$[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}] \begin{bmatrix} \mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$
$$= [\mathbf{A}_{0,i+1} \mid (-\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G}) + \mathbf{H}_{\mu,i+1}G \mid \mathbf{A}_{2,i+1}]. \quad (26)$$

Let $\mathbf{b}_{0,i+1}$ be the ciphertext of message $\mathbf{0}$ under pk_{i+1} with the random invertible matrix $\mathbf{H}_{\mu,i}$, $\mathbf{M}_{i+1}$ be the transition matrix from $\mathbf{A}_{\mu,i}$ to $\mathbf{A}_{\mu,i+1}$ in Eq. (26) and $\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1})$. Based on Eq. (24), (25) and (26), we know that Δ_{i+1} is a valid non-challenge token in epoch $i+1$. Since this process only uses the property that $\hat{\mathbf{R}}_i$ is a trapdoor, it can be applied to any epoch $j \in \{i+1 \cdots \mathsf{fwr}\}$ by choosing an matrix $\mathbf{R}_j$ with the same property, which works as long as $\mathbf{H}_{\mu,j} \neq \mathbf{H}^*_{\mu,j}$.

Overall, we conclude that Game 1 and Game 0 are indistinguishable, which follows from

$$\begin{aligned} |\Pr[E_1] - \Pr[E_0]| &\leq (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}}) \cdot \Pr[\mathbf{H} = \mathbf{H}^*] + \mathsf{negl}(\lambda) \cdot (l+1) \\ &= (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}}) \cdot 2^{-\Omega(n)} + \mathsf{negl}(\lambda) \cdot (l+1), \end{aligned}$$

where n_{Dec} and n_{sUpd} are the number of queries to decryption and update, respectively; $\Pr[\mathbf{H} = \mathbf{H}^*]$ is the probability that two random invertible matrices are equal, and $l+1$ is the maximal length of the firewall.

Game 2:

Compared to Game 1, we only change challenge-equal ciphertexts in epoch from i to fwr, while keeping the other simulation, especially the challenge-equal token, the same. In epoch i, we change the last two nk-coordinates of the challenge ciphertext and keep the first $\bar{m}$ coordinates unchanged. That is,

$$\begin{aligned} \mathbf{b}_0^t &= \mathbf{s}^t\mathbf{A}_{0,i} + \widehat{\mathbf{e}}_0^t, \\ \mathbf{b}_1^t &= -\mathbf{b}_0^t \cdot \mathbf{R}_{1,i} + \widehat{\mathbf{e}}_1^t, \\ \mathbf{b}_2^t &= -\mathbf{b}_0^t \cdot \mathbf{R}_{2,i} + \widehat{\mathbf{e}}_2^t + \mathsf{encode}(\mathbf{m}_{\mathrm{b}})^t, \end{aligned}$$

where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\widehat{\mathbf{e}}_0 \leftarrow D^{\bar{m}}_{\mathbb{Z},\alpha q}$ and $\widehat{\mathbf{e}}_1$, $\widehat{\mathbf{e}}_2 \leftarrow D^{nk}_{\mathbb{Z},\alpha q\sqrt{m}\cdot\omega(\sqrt{\log n})}$. By substitution, we have $\mathbf{b}_1^t = \mathbf{s}^t\mathbf{A}_{0,i} \cdot \mathbf{R}_{1,i} + \widehat{\mathbf{e}}_0^t \cdot \mathbf{R}_{1,i} + \widehat{\mathbf{e}}_1^t$. According to [[17], Corollary 3.10], the distribution of $\widehat{\mathbf{e}}_0^t \cdot \mathbf{R}_{1,i} + \widehat{\mathbf{e}}_1^t$ is $\mathsf{negl}(n)$-far from $D^{nk}_{\mathbb{Z},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m} \cdot \alpha q) \cdot \omega(\sqrt{\log n})^2$ that is the error distribution of $\mathbf{b}_1$ in Game 1. Therefore, the distribution of $\mathbf{b}_1$ is within $\mathsf{negl}(n)$-distance from that in Game 1. The same applies to $\mathbf{b}_2$. Then we change the ciphertext in epoch $i+1$ by updating $(\mathbf{H}^*_{\mu,i}, (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2))$ using the challenge-equal token in Eq. (23), which is therefore a valid token, and similarly change ciphertexts in epoch from $i+2$ to fwr by updating the modified ciphertext using challenge-equal tokens in Game

1. And thus, we have $|\Pr[E_2] - \Pr[E_1]| \leq \mathsf{negl}(\lambda)$.

Game 3:

We consider a modified game that is the same as Game 2, except that we change the first $\bar{m}$ coordinates of the challenge ciphertext in epoch i, letting it be the challenge ciphertext: uniformly random or $\mathbf{s}^t\mathbf{A}_{0,i} + \widehat{\mathbf{e}}_0^t$, which are hard to distinguish under the $\mathsf{LWE}_{n,q,\alpha}$ problem. Then we also update this modified challenge-equal ciphertext and change the ciphertext in epochs from $i+1$ to fwr as in Game 2 to make sure that the challenge-equal tokens still sever as valid tokens. Thus, we have $|\Pr[E_3] - \Pr[E_2]| \leq \mathsf{Adv}^{\mathsf{LWE}}_{n,q,\alpha}$.

The rest we need to prove now is that $|\Pr[E_3]| = 1/2$. It follows from $(\mathbf{A}_{0,i}, \mathbf{b}_0, \mathbf{b}_0 \cdot \mathbf{R}_{1,i}, \mathbf{b}_0 \cdot \mathbf{R}_{2,i})$ is $\mathsf{negl}(\lambda)$-from uniform by the leftover hash lemma for random $\mathbf{A}_{0,i} \in \mathbb{Z}_q^{n \times \bar{m}}$, $\mathbf{b}_0 \in \mathbb{Z}_q^{\bar{m}}$, and $\mathbf{R}_{1,i}$, $\mathbf{R}_{2,i} \in \mathcal{D}$.

We thus complete the proof.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: STOC. pp. 99–108 (1996). https://doi.org/10.1145/237814.237838
2. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory Comput. Syst. **48**(3), 535–553 (2011). https://doi.org/10.1007/s00224-010-9278-3
3. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Math. Ann. **296**, 625–635 (1993). https://doi.org/10.1007/BF01445125
4. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 559–589. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_19
5. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). https://doi.org/978-3-642-40041-4_23
6. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 464–493. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_16
7. Chen, L., Li, Y., Tang, Q.: CCA updatable encryption against malicious re-encryption attacks. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 590–620. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_20
8. Everspaugh, A., Paterson, K., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 98–129. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63697-9_4
9. Fan, X., Liu, F.: Proxy re-encryption and re-signatures from lattices. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 363–382. Springer (2019). https://doi.org/10.1007/978-3-030-21568-2_18

10. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press (1991). https://doi.org/10.1017/CBO9780511840371
11. Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 529–558. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_18
12. Kirshanova, E.: Proxy re-encryption from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 77–94. Springer (2014). https://doi.org/10.1007/978-3-642-54631-0_5
13. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 68–99. Springer (2019). https://doi.org/10.1007/978-3-030-17653-2_3
14. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 685–716. Springer (2018). https://doi.org/10.1007/978-3-319-78372-7_22
15. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_41
16. Nishimaki, R.: The direction of updatable encryption does matter. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 194–224. Springer (2022). https://doi.org/10.1007/978-3-030-97131-1_7
17. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) ACM 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603
18. Slamanig, D., Striecks, C.: Puncture 'em all: Stronger updatable encryption with no-directional key updates. IACR Cryptol. ePrint Arch. p. 268 (2021), https://eprint.iacr.org/2021/268

A Lattice Background

A.1 Lattices and Hard Problems

Given n linearly independent vectors $\mathbf{b}_1, \cdots, \mathbf{b}_n \in \mathbb{Z}^m$, the lattice Λ generated by them is the set of all $\mathbb{Z}$-linear combinations of $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\}$, i.e. $\Lambda := \{\mathbf{Bz} \mid \mathbf{z} \in \mathbb{Z}^n\}$. We call $\mathbf{B}$ a basis of Λ, n the rank of Λ and m the dimension of Λ. The dual lattice Λ^* of Λ is the set of all $v \in \mathrm{span}(\Lambda)$ such that $\langle v, x \rangle \in \mathbb{Z}$ for all $x \in \Lambda$.

For a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ where $m = \mathsf{poly}(n)$, there are two lattice-based hard problems associated with it:

- $\mathsf{SIS}_{q,\beta}$: find a nonzero $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\| \leq \beta$;
- $\mathsf{LWE}_{q,\alpha}$: for arbitrary $\mathbf{s} \in \mathbb{Z}_q^n$ and error $\mathbf{e}$ from the Gaussian distribution over $\mathbb{Z}^m$ with parameter α (defined in Sect. A.2), let $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$. The search-$\mathsf{LWE}_{q,\alpha}$ is to find $\mathbf{s}$ from $(\mathbf{A}, \mathbf{b})$; while the decision-$\mathsf{LWE}_{q,\alpha}$ is to distinguish between $\mathbf{b}$ and uniformly random samples from $\mathbb{Z}_q^m$ with noticeable probability.

When $q \geq \beta\sqrt{n} \cdot \omega(\sqrt{\log n})$, solving $\mathsf{SIS}_{q,\beta}$ is at least as hard as solving the approximating Shortest Independent Vector Problem (SIVP) on n-dimensional lattices

[1,17], which is believed to be a hard problem in the study of lattices. When $q \geq 2\sqrt{n}/\alpha$, the hardness also applies to the search-$\mathsf{LWE}_{q,\alpha}$ and decision-$\mathsf{LWE}_{q,\alpha}$ problem [17].

A.2 Gaussians on Lattices

Let $N(0, \sigma^2)$ denote the Gaussian distribution over $\mathbb{R}$ with mean 0 and variance σ^2. It is defined by the density function $(1/\sigma\sqrt{2\pi}) \cdot \exp\left(-x^2/2\sigma^2\right)$. For a positive real s, the n-dimensional Gaussian function is defined as $\rho_s(x) = \exp\left(-\pi\|x\|^2/s^2\right)$ for $x \in \mathbb{R}^n$. The discrete Gaussian distribution over a countable set A is defined by the density function $D_{A,s}(x) := \frac{\rho_s(x)}{\sum_{y\in A}\rho_s(y)}$. The following lemma shows a tail bound on discrete Gaussians.

Lemma 11 ([3], Lemma 1.5). *Let $c \geq 1, C = c \cdot \exp\left(\left(1-c^2\right)/2\right)$. For any real $s > 0$ and any integer $n \geq 1$, we have that*

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

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

In the construction of encryption schemes, the notion of *subgaussian* is used to analyze the behavior of error terms in [15,12] and this paper. For any $\delta \geq 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is *δ-subgaussian* with parameter $r > 0$, if the (scaled) moment-generating function satisfies

$$\mathbb{E}[\exp(2\pi tX)] \leq \exp(\delta) \cdot \exp\left(\pi r^2 t^2\right),$$

for all $r > 0$.

More generally, the notion of subgaussianity can be extended to vectors and matrices. A random vector $\mathbf{x}$ or its distribution (respectively, a random matrix $\mathbf{X}$) is δ-subgaussian with parameter $r > 0$ if all its one-dimensional marginals $\langle \mathbf{u}, \mathbf{v}\rangle$ (respectively, $\mathbf{u}^t\mathbf{X}\mathbf{v}$) for unit vectors $\mathbf{u}, \mathbf{v}$ is δ-subgaussian with parameter $r > 0$.

Lemma 12 ([15], Lemma 2.8). *Let $\Lambda \subset \mathbb{R}^n$ be a lattice. For any $s > 0$, $D_{\Lambda,s}$ is 0-subgaussian with parameter s.*

Lemma 13 ([15], Lemma 2.9). *Let $\mathrm{X} \in \mathbb{R}^{n\times m}$ be a δ-subgaussian random matrix with parameter s. There exists a universal constant $C > 0$ such that for any $t \geq 0$, we have $s_1(\mathbf{X}) \leq C \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2\exp(\delta)\exp\left(-\pi t^2\right)$.*

Micciancio and Peikert [15] showed that the concatenation of independent δ_i-subgaussian vectors with common parameter s, interpreted as either a vector or matrix, is $\sum \delta_i$-subgaussian, which follows directly from the definition. For any $m, n > 0$, $D^n_{\mathbb{Z},s}$ and $D^{n\times m}_{\mathbb{Z},s}$ are 0-subgaussian with parameter s, since $D_{\mathbb{Z},s}$ is 0-subgaussian (with s) by Lemma 12. This leads to the following result by Lemma 13.

Corollary 1. *For any $m, n, s > 0$, let $\mathbf{R} \in D_{\mathbb{Z},s}^{n\times m}$, we have $s_1(\mathbf{R}) \le s \cdot O(\sqrt{n} + \sqrt{m})$, except with probability $2^{-\Omega(n+m)}$.*

The following lemma bounds the maximal singular value of the product of two matrices, which follows directly from the definition. More details about singular values can be seen in Theorem 3.3.16 [10]. We will also use the Lemma 15 and 16 in our proof.

Lemma 14. *Let $\mathbf{A} \in \mathbb{R}^{m,n}$, $\mathbf{B} \in \mathbb{R}^{n,m}$, then $s_1(\mathbf{AB}) \le s_1(\mathbf{A})s_1(\mathbf{B})$ and $s_1(\mathbf{A}+\mathbf{B}) \le s_1(\mathbf{A}) + s_1(\mathbf{B})$.*

Lemma 15 (Leftover Hash Lemma). *Let $\mathcal{P}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \le (k - 2\log(1/\epsilon) - O(1))/\log(q)$, the joint distribution of $(\boldsymbol{C}, \boldsymbol{Cs})$ is ϵ-close to the uniform distribution over $\mathbb{Z}_q^{l\times n} \times \mathbb{Z}_q^l$, where $\boldsymbol{C} \xleftarrow{\$} \mathbb{Z}_q^{l\times n}$ and $s \leftarrow \mathcal{P}$.*

Lemma 16 ([2], Fact 2.2). *Let $X_1, \cdots, X_n$ be independent mean-zero subgaussian random variables with parameter s, and let $u \in \mathbb{R}^n$ be arbitrary. Then $\sum_k (a_k X_k)$ is subgaussian with parameter $s\|u\|$.*

B Proof of Inequality (15)

We start from the error $\mathbf{e}_0$ which is generated in the fresh encryption in epoch $i = 0$ and estimate the bound on $\mathbf{e}_0^t \cdot \prod_{j=1}^{l} \mathbf{M}_j$. Errors generated in the update algorithm in later epochs have the same distribution as $\mathbf{e}_0$, but are multiplied less times than $\mathbf{e}_0$ by the transition matrix $\{\mathbf{M}_j\}$.

Step 1. Let $\mathbf{e}_0 = (\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{20})$ and $\mathbf{M}^{(s)}$ be the s products of $\{\mathbf{M}_j\}_{j=1}^s$ that are denoted as follows:

$$\mathbf{M}_j = \begin{bmatrix} \mathbf{X}_{00,j} & \mathbf{X}_{01,j} & \mathbf{X}_{02,j} \\ \mathbf{X}_{10,j} & \mathbf{X}_{11,j} & \mathbf{X}_{12,j} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \text{ and } \mathbf{M}^{(s)} = \prod_{t=1}^{s} \mathbf{M}_t = \begin{bmatrix} \mathbf{X}_{00}^{(s)} & \mathbf{X}_{01}^{(s)} & \mathbf{X}_{02}^{(s)} \\ \mathbf{X}_{10}^{(s)} & \mathbf{X}_{11}^{(s)} & \mathbf{X}_{12}^{(s)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix},$$

for $s \in \{1, \cdots, l\}$. We first estimate the bound on the maximal singular values of $\mathbf{M}_j$ and $\mathbf{M}^{(s)}$. For any $j \in \{1, \cdots, l\}$, we have $s_1(\mathrm{X}_{kt,j}) \le \tau \cdot O(\sqrt{nk})$ for $kt \in \{00, 10\}$ and $s_1(\mathrm{X}_{kt,j}) \le \tau\sqrt{\bar{m}/2} \cdot O(\sqrt{nk})$ for $kt \in \{01, 11, 02, 12\}$ by Corollary 1. Let $\mu = \sqrt{\bar{m}/2}$. For $s = 2$, we obtain $\mathbf{X}_{00}^{(2)} = \mathbf{X}_{00,1} \cdot \mathbf{X}_{00,2} + \mathbf{X}_{01,1} \cdot \mathbf{X}_{10,2}$. Then by Lemma 14, we have $s_1(\mathbf{X}_{00}^{(2)}) \le s_1(\mathbf{X}_{00,1}) \cdot s_1(\mathbf{X}_{00,2}) + s_1(\mathbf{X}_{01,1}) \cdot s_1(\mathbf{X}_{10,2}) \le \tau^2(1+\mu) \cdot O(\sqrt{nk})^2$. Similarly, we give the bound on maximal singular value of other block matrices in $\mathbf{M}^{(2)}$ (except the last nk rows, which equal $[\mathbf{0}, \mathbf{0}, \mathbf{I}]$ for all $\mathbf{M}$) as follows (element-wise comparison):

$$\begin{bmatrix} s_1(\mathbf{X}_{00}^{(2)}) & s_1(\mathbf{X}_{01}^{(2)}) & s_1(\mathbf{X}_{02}^{(2)}) \\ s_1(\mathbf{X}_{10}^{(2)}) & s_1(\mathbf{X}_{11}^{(2)}) & s_1(\mathbf{X}_{12}^{(2)}) \end{bmatrix} \le \begin{bmatrix} \tau^2(1+\mu) & \tau^2\mu(1+\mu) & \tau^2\mu[(1+\mu)+1] \\ \tau^2(1+\mu) & \tau^2\mu(1+\mu) & \tau^2\mu[(1+\mu)+1] \end{bmatrix} \cdot O(\sqrt{nk})^2.$$

Iteratively, we have the bound on maximal singular value of each block matrix in $\mathbf{M}^{(l)}$, that is

$$\begin{bmatrix} s_1(\mathbf{X}_{00}^{(l)}) & s_1(\mathbf{X}_{01}^{(l)}) & s_1(\mathbf{X}_{02}^{(l)}) \\ s_1(\mathbf{X}_{10}^{(l)}) & s_1(\mathbf{X}_{11}^{(l)}) & s_1(\mathbf{X}_{12}^{(l)}) \end{bmatrix} \leq \begin{bmatrix} \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \\ \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \end{bmatrix} \cdot O(\sqrt{nk})^l \,. \tag{27}$$

Now we can estimate the bound on the updated $\mathbf{e}_0$ in the last epoch. Notice that

$$\begin{aligned} \mathbf{e}_0^t \cdot \mathbf{M}^{(l)} \cdot \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} &= (\mathbf{e}_{00}, \mathbf{e}_{10}, \mathbf{e}_{20})^t \begin{bmatrix} \mathbf{X}_{00}^{(l)} & \mathbf{X}_{01}^{(l)} & \mathbf{X}_{02}^{(l)} \\ \mathbf{X}_{10}^{(l)} & \mathbf{X}_{11}^{(l)} & \mathbf{X}_{12}^{(l)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \\ &= (\mathbf{e}_{00}^t\mathbf{X}_{00}^{(l)} + \mathbf{e}_{10}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l} + \mathbf{e}_{00}^t\mathbf{X}_{01}^{(l)} + \mathbf{e}_{10}^t\mathbf{X}_{11}^{(l)}. \end{aligned} \tag{28}$$

By Lemma 11, we have $\|\mathbf{e}_{00}\| < \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_i\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $i \in \{10, 20\}$, except with probability $2^{-\Omega(n)}$. Combining this conclusion with Eq. (27), we obtain

$$\begin{cases} \left\|\mathbf{e}_0^t\mathbf{X}_{00}^{(l)}\mathbf{R}_{1,l}\right\| \leq \|\mathbf{e}_0\| \cdot s_1(\mathbf{X}_{00}^{(l)}) \cdot s_1(\mathbf{R}_{1,l}) < \alpha q\sqrt{\bar{m}} \cdot \tau^l(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_1^t\mathbf{X}_{10}^{(l)}\mathbf{R}_{1,l}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_0^t\mathbf{X}_{01}^{(l)}\right\| < \alpha q\sqrt{\bar{m}} \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l, \\ \left\|\mathbf{e}_1^t\mathbf{X}_{11}^{(l)}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l. \end{cases} \tag{29}$$

Substituting σ, τ and μ, we get the upper bound for the norm of the updated $\mathbf{e}_0^t$ in Eq.(28) by the triangle inequality as follows:

$$\left\|(\mathbf{e}_{00}^t\mathbf{X}_{00}^{(l)} + \mathbf{e}_{10}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l} + \mathbf{e}_{00}^t\mathbf{X}_{01}^{(l)} + \mathbf{e}_{10}^t\mathbf{X}_{11}^{(l)}\right\| < \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}. \tag{30}$$

Notice that $\mathbf{X}_{01}^{(l)}, \mathbf{X}_{11}^{(l)}$, $\mathbf{X}_{02}^{(l)}, \mathbf{X}_{12}^{(l)}$ have the same asymptotic bound for the maximal singular value by Inequality (27), and $\|\mathbf{e}_2\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$. We obtain the same bound for the norm of the last nk coordinates, that is

$$\left\|\mathbf{e}_{00}^t\mathbf{X}_{02}^{(l)} + \mathbf{e}_{10}^t\mathbf{X}_{12}^{(l)} + \mathbf{e}_2^t\right\| < \alpha q\omega(\sqrt{\log n})^{2l+1}O(\sqrt{nk})^{3l+2}. \tag{31}$$

Since the infinity norm is smaller than the 2-norm, we have

$$\left\|(\mathbf{e}_0^t \cdot \prod_{j=1}^{l}\mathbf{M}_j)\begin{bmatrix} \mathbf{R}_{1,l} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\right\|_\infty \leq \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}, \tag{32}$$

by combining the Inequality (30) and Inequality (31).

Step 2. Errors generated from epoch 1 to $l-1$ have the same distribution with $\mathbf{e}_0$, but are multiplied less time by the transition matrices $\{\mathbf{M}_j\}$, which therefore yields a lower bound than $\mathbf{e}_0$.

Step 3. For $\mathbf{e}_l$, it is not multiplied by any transition matrix. Let $\mathbf{e}_l^t = (\mathbf{e}_{0,l}, \mathbf{e}_{1,l}, \mathbf{e}_{2,l})$. Then we have

$$\mathbf{e}_l^t \cdot \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = (\mathbf{e}_{0,l}^t \mathbf{R}_1 + \mathbf{e}_{1,l}^t, \mathbf{e}_{0,l}^t \mathbf{R}_2 + \mathbf{e}_{2,l}^t).$$

Note that $\|\mathbf{e}_i\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $i \in \{1, 2\}$ by Corollary 1. Therefore, the bound on update $\mathbf{e}_0$ in Inequality (32) also holds for $\mathbf{e}_l$.

Finally, we conclude that the upper bound on the infinity norm of the sum of updated errors is $\alpha q l \omega(\sqrt{\log n})^{2l+2} O(\sqrt{nk})^{3l+3}$ by triangle inequality, except with probability $2^{-\Omega(n)}$. That is

$$\left\| \sum_{i=0}^{l} (\mathbf{e}_i^t \cdot \prod_{j=i+1}^{l} \mathbf{M}_j) \cdot \begin{bmatrix} \mathbf{R}_{1,l} & \mathbf{R}_{2,l} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \right\|_\infty \leq \alpha q l \omega(\sqrt{\log n})^{2l+2} O(\sqrt{nk})^{3l+3}.$$

Since $1/\alpha = 4l\omega(\sqrt{\log n})^{2l+2} O(\sqrt{nk})^{3l+3}$, we have the desired property of error vectors, i.e., the Inequality (15).