CCA-1 Secure Updatable Encryption with Adaptive Security

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. Updatable encryption (UE) enables a cloud server to update ciphertexts using client-generated tokens. There are two types of UE: *ciphertext-independent (c-i)* and *ciphertext-dependent (c-d)*. In terms of construction and efficiency, c-i UE utilizes a single token to update all ciphertexts. The update mechanism relies mainly on the homomorphic properties of exponentiation, which limits the efficiency of encryption and updating. Although c-d UE may seem inconvenient as it requires downloading parts of the ciphertexts during token generation, it allows for easy implementation of the Dec-then-Enc structure. This methodology significantly simplifies the construction of the update mechanism. Notably, the c-d UE scheme proposed by Boneh et al. (ASIACRYPT'20) has been reported to be 200 times faster than prior UE schemes based on DDH hardness, which is the case for most existing c-i UE schemes. Furthermore, c-d UE ensures a high level of security as the token does not reveal any information about the key, which is difficult for c-i UE to achieve. Moreover, previous security studies on c-d UE only addressed selective security, while the studies for adaptive security remain an open problem.

In this study, we make three significant contributions to ciphertext-dependent updatable encryption (c-d UE). Firstly, we provide stronger security notions compared to previous work, which encompass adaptive security and also consider the adversary's decryption capabilities. Secondly, we propose a new c-d UE scheme that achieves the proposed security notions. The token generation technique significantly differs from the previous Dec-then-Enc structure, while still preventing key leakages. At last, we introduce a packing technique that enables the simultaneous encryption and updating of multiple messages within a single ciphertext. This technique helps alleviate the cost of c-d UE by reducing the need to download partial ciphertexts during token generation. Our schemes are based on the Learning with Error assumption, resulting in faster performance compared to previous UE schemes that rely on the expensive group operations of the DDH assumption.

Keywords: Updatable Encryption · Adaptive Security · Lattice

1 Introduction

Regularly changing encryption keys is widely recognized as an effective approach to mitigate the risk of key compromise, especially when entrusting encrypted data to a semi-honest cloud server. Updatable encryption (UE), introduced by Boneh et al. [7], offers a practical solution to this challenge. In UE schemes, in addition to the usual KG, Enc, Dec algorithms, two core algorithms, TokenGen and Update, are employed. Essentially, TokenGen takes the old and new encryption keys, along with *possibly* a small fraction of the ciphertext, and generates an update token on the client side. This token is then sent to the cloud server, which utilizes the Update algorithm to convert ciphertexts from the old keys to the new keys.

c-d/c-i UE. Depending on if a part of ciphertext (called ciphertext header) is needed in the token generation algorithm TokenGen, UE schemes have two variants: *ciphertext-independent* (c-i) UE [8,17,19,20,22,25] and *ciphertext-dependent* (c-d) UE [6,7,11,13]. In the former, tokens are independent of ciphertexts, and a single update token is used to update all old ciphertexts. In the latter, update tokens depend on the specific ciphertext to be updated and a tiny part of the ciphertexts is downloaded by the client when generating the update tokens.

In this paper, we specifically focus on ciphertext-dependent UE (c-d UE) due to its notable advantages in terms of efficiency and security. First of all, c-d UE schemes have been reported to be more efficient than ciphertext-independent (c-i) constructions. For instance, the nested c-d UE construction presented in [6], which relies solely on symmetric cryptographic primitives, approaches the performance of AES. In contrast, c-i UE schemes imply the use of public key encryption, as proven by Alamati et al. [3], and most c-i constructions require costly exponentiation operations to update ciphertexts. With regards to UE security, Jiang [17] demonstrated that c-i UE schemes with no-directional key updates, defined in Section 3, are significantly stronger than c-i schemes with other directional key updates. However, constructing such schemes remains an open problem, primarily due to the requirement that update tokens should not reveal any information about either the old key or the new key. Consequently, only two c-i UE schemes with no-directional key updates have been proposed thus far. One is presented by Slamanig [25], which is based on the SXDH assumption, thus necessitating expensive exponential operations. The other is introduced by Nishimaki [22], relying on the existence of indistinguishability obfuscation, but remains purely theoretical. On the other hand, for c-d UE, the construction of no-directional key update schemes is considerably easier and practical. In fact, the token generation algorithm, denoted as TokenGen, in all existing c-d UE constructions [13,6,7] benefits from a "Dec-then-Enc" process. This involves decrypting the ciphertext header using the old key to recover the secret information, and then computing the token by encrypting the secret information using the new key. As a result, the old key remains independent of the token, while the new key is safeguarded by the underlying encryption scheme. The update token does not divulge any information about the old and new keys.

Security Notions (c-d UE). The primary security objective of UE is to ensure the confidentiality of ciphertexts even when the keys are exposed. Extensive research on this topic has been conducted in [13,6]. Previous security models provide guarantees that adversaries cannot differentiate between a freshly generated ciphertext in the current epoch and an updated ciphertext rotated to the current epoch. In practical scenarios, this property safeguards the confidentiality of the *age* of ciphertext, i.e., the number of times it has been updated, from being leaked to an adversary. For instance, consider a situation where a client stores its encrypted medical records with a cloud provider. The existing security notions ensure that the adversary observing the records cannot determine which records are new and which ones are old, thereby preserving the sort of privacy.

Limitation. Unfortunately, prior work on c-d UE has the following limitations:

1. The existing security notions for c-d UE solely capture selective security. It is important to note that this security does not provide assurance of post-compromise (or forward) security when the provided selective keys represent all the keys used after (or before, respectively) the challenge phase. Moreover, these notions do not take into account the decryption capabilities of the adversary. This leads to a fact that the existing constructions can only provide CPA security. It is a natural need to propose a stronger security notion and the corresponding construction.
2. Prior notions for c-d UE only apply to randomized ciphertext updates, whereas the ciphertext update procedure can be also deterministic [3], which can be seen in our construction in Sect. 5.1. It is still an open problem that how could we capture confidentiality for both types of ciphertext udpates.
3. The current security notions for c-d UE are complex, requiring multiple simulations of oracles the adversary has access to in the security analysis. A simpler and more compact notion can help one simplify the proof.

1.1 Related Work

Constructions of UE. Since the introduction of updatable encryption by Boneh et al. [7], various constructions have been proposed. All c-d UE schemes in [7,13,6], either treated in a symmetric manner to deploy a double encryption or relying on *key-homomorphic PRFs*, benefiting from a *Dec-then-Enc* structure in token generation. As a consequence, tokens only contain the ciphertext under new key, avoiding the issue of leaking neither old nor new key.

By comparison, all c-i UE schemes in [20,19,8] are based on the DDH or SXDH assumption and rely on the homomorphic properties of exponentiation to rotate ciphertexts. Tokens are the division of new key and old key; therefore, one of the two of the two successive keys key can be inferred if the other is leaked.

[3] Note this case does not require the sever to generate randomness for ciphertext updates, which is required in the former case.

	Schemes	Dir. Key	s/a	Prob.	Enc.	Token Gen.	Update
c-i UE	*SHINE [8]	bi	a	DDH	1 exp.	1 division	1 exp.
	LWEUE [17]	bi	a	LWE	(n, m, l)	1 subtract.	(n, m, l)
	Nishimaki [22]	bk.	a	LWE	$(1, m, l)$	$(nk, m, n + l)$	$(1, m, n + l)$
	Nishimaki [22]	no	a	IO, OWF	–	–	–
	SS [25]	no	a	SXDH	–	–	–
c-d UE	KSS [13]	no	s	Symmetry	1 AE.Enc	1 AE.Enc	1 vect. addition
	ReCrypt [13]	no	s	KH-PRF	1 KH-PRF	1 KH-PRF	1 KH-PRF
	Nested [6]	no	s	Symmetry	1 AE.Enc	1 AE.Enc	1 AE.Enc
	BEKS [6]	no	s	KH-PRF	1 KH-PRF	1 AE.Enc	1 KH-PRF
	TDUE (5.1)	no	a	LWE	$(1, n, \bar{m} + 2l)$	$\bar{m} + 2l$ sampling	$(1, \bar{m} + 2l, \bar{m} + 2l)$
	Packing UE (5.4)	no	a	LWE	$(1, n, \bar{m} + l + Nl)$	$\bar{m} + l + Nl$ sampling	$(1, \bar{m} + l + Nl, \bar{m} + l + Nl)$

Fig. 1. A comparison of c-i UE which can avoid the leakage of "ciphertext age" and all existing c-d UE. The second column set states the direction of key updates, achieved security, and the underlying assumptions, where bk. stands for the backward directional key updates, s and a represent the selective and adaptive security, respectively, and KH-PRF, IO, OWF represent key-homomorphic PRF, indistinguishability obfuscation, and one-way function, respectively. The third column set shows the computational efficiency in terms of the most expensive cost of encryption, token generation, and update for one-block ciphertext ([22] and [25] are omitted here as the first is theoretical and the second is built on a totally different expiry model). For lattice-based schemes, (a, b, c) denotes the major computation cost by the multiplication of two matrices of size $a \times b$ and $b \times c$, and m, n denote the size of matrix generated on $\mathbb{Z}_q$ in the setup, for which $m = \mathsf{poly}(n)$, $k = \lceil q \rceil$, message bit length $l = nk$, and $\bar{m} = O(nk)$. In our UE schemes, token are generate with multiple calls to a preimage sampling oracle, and N is a power of 2 that defines the associated cyclotomic ring. AE represents authenticated encryption, and KH-PRFs are constructed from the Ring-LWE problem in [6].

Such a leakage limitation is also applied to the scheme proposed by Jiang [17], because tokens are the subtraction of new and old keys, even though this scheme avoids the expensive exponentiation but instead lattice-based.

Two promising c-i UE schemes have been proposed to overcome this leakage limitation. Nishimaki [22] presented a construction that utilizes *indistinguishability obfuscation (IO)* for an update circuit, which operates as a Dec-then-Enc process taking a ciphertext as input. This scheme relies on an assumption that there exists a practical IO. Slamanig and Striecks [25] gave a pairing-based scheme and defined an *expiry* model: each ciphertext is associated with an expiry key, after which the updated ciphertext cannot be decrypted any more. Their scheme consumes expensive group operations and moreover the key size increases linearly

to number of maximum number of update. In Fig. 1, we provide a comparison of UE schemes in terms of security and efficiency.

Relative Primitives. Proxy Re-encryption (PRE) and Homomorphic Encryption (HE) are two highly related primitives to updatable encryption.

Proxy Re-encryption (PRE) enables a ciphertext to be decryptable by the new key after re-encryption. Compared to UE, it does not necessarily require the update ciphertext to be indistinguishable with fresh encryption, thereby not covering the confidentiality requirement inherent in UE. However, PRE schemes have served as a source of inspiration for the construction of UE due to the similar ciphertext update process, for example, the ElGamal-based proxy re-encryption scheme is adapted to RISE [20] and Sakurai et al.[24] to SHINE [8].

The PRE scheme proposed by Kirshanova [18] is based on lattice and only uses the old secret key (serving as the trapdoor) to sample a matrix as the update token to rotate ciphertexts, which are LWE samples. Such a matrix leaks neither the old nor the new keys, since it does not involve any function of old and new key (recall the key leakage caused the division or subtraction of two keys in the token of c-i UE schemes). However, Fan and Liu [14] pointed out a mistake in the security proof of [18] that the simulated game in the proof is not indistinguishable to the real game. In this work, a new UE scheme that leverages a part of techniques in [18] is constructed with a detailed reduction proof.

Fully Homomorphic Encryption (FHE) develops a key-switching technique [10,9] that takes as input the old ciphertext and the encryption of old key under new key (called key-switching key) and outputs a new ciphertext that is decryptable by the new key. Such a technique has been used in [19,15] to construct UE schemes with so-called backward directional key updates. In our UE construction, the matrix in token is called a key-switching matrix as it achieves the same functionality as the key-switching key in FHE.

1.2 Our Approaches

We propose new UE schemes that achieve the new confidentiality notion. To achieve this, we first build a new PKE scheme inspired by [21] that utilizes lattice trapdoor techniques as the underlying encryption scheme. For the UE construction, we leverage the "re-encryption key generation" process in [18] to generate a key-switching matrix, which is used to update ciphertexts from the old to the new key. However, the key-switching matrix alone is not sufficient to achieve our confidentiality notion, which will be discussed later in this section. A detailed proof of our construction is presented in Sect. 5.3.

A New PKE Scheme. This scheme is based on lattice trapdoor techniques. The public key is a 1×3 block matrix $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_1] \in \mathbb{Z}^{\bar{m}+2nk}$ where $\mathbf{A}_0$ and $\mathbf{A}_1$ are two random matrices, and $\mathbf{H}_\mu$ is an invertible matrix. The secret key is the trapdoor $\mathbf{R}$ for the first two block matrices of $\mathbf{A}_u$, which allows for an efficient algorithm for inverting LWE samples related to $\mathbf{A}_\mu$ (see Sect. 4.1

for more details). The ciphertext is a tuple $c = (\mathbf{H}_\mu, \mathbf{b})$ where $\mathbf{b}$ is a LWE sample as follows:

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

for a proper encoding algorithm encode, error items $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, and integer q. To decrypt a ciphertext, one first recovers $\mathbf{s}$ and $(\mathbf{e}_0, \mathbf{e}_1)$ from the first two blocks of $\mathbf{b}$ using the trapdoor $\mathbf{R}$ and an inversion algorithm. Then $\mathbf{m}$ and $\mathbf{e}_2$ are recovered from the last block of $\mathbf{b}$ with the recovered $\mathbf{s}$ and the inverse of encode.

Key-switching Matrix. The key-switching matrix enables the transition of a ciphertext in Eq. (1) to a new ciphertext with the same form, denoted as $c' = (\mathbf{H}_u, \mathbf{b}')$, where

$$\mathbf{b}'^t = \mathbf{s}^t \mathbf{A}'_\mu + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{2}$$

for new public matrix $\mathbf{A}'_\mu$ and new error items $\mathbf{e}'$. This matrix is essentially the transition matrix from $\mathbf{A}_u$ to $\mathbf{A}'_u$, with the last row block matrix $[\mathbf{0} \ \ \mathbf{0} \ \ \mathbf{I}]$, i.e., $\mathbf{A}_u \cdot \mathbf{M} = \mathbf{A}'_u$. The old ciphertext c is updated by by multiplying $\mathbf{b}^t$ and $\mathbf{M}$, that is

$$\begin{aligned} \mathbf{b}^t \mathbf{M} &= \mathbf{s}^t \mathbf{A}_\mu \cdot \mathbf{M} + \mathbf{e}^t \cdot \mathbf{M} + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \cdot \mathbf{M} \\ &= \mathbf{s}^t \mathbf{A}'_\mu + \mathbf{e}'^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \end{aligned}$$

which matches the desired form in Eq. (2). The matrix $\mathbf{M}$ can be efficiently generated by the trapdoor (secret key) $\mathbf{R}$ and the preimage sampling algorithm, as presented in Sect 2.2.

Challenges and a New UE Scheme. We state that there are two technical challenges on directly using the key-switching matrix as the update token to construct a secure UE scheme satisfying our confidentiality notion, which requires the indistinguishability between "fresh" and updated ciphertexts. The first observation is that $\mathbf{H}_\mu$, as part of the ciphertext, is never rotated in the update process. The adversary can distinguish the challenge ciphertexts by comparing $\mathbf{H}_\mu$ extracted from the challenge output and input. Beyond that, $\mathbf{s}$ is also never changed during update process. With the known $\mathbf{s}$ used in the challenge input ciphertext, the adversary may attempt to decrypt the challenge output (note that the last step in the decryption algorithm in PKE only requires $\mathbf{s}$). If it fails, then the adversary knows the challenge output is a fresh encryption of the challenge input message. Otherwise, that is an update of the challenge input ciphertext.

Our solution to address the challenges is to change the invertible matrix $\mathbf{H}_\mu$ and the variable $\mathbf{s}$ in each update. Specifically, a new invertible matrix $\mathbf{H}'_\mu$ and a fresh encryption of message $\mathbf{0}$ under the new key with $\mathbf{H}'_\mu$, denoted by $\mathbf{b}_0$, is generated in the token generation algorithm to improve the randomness. In summary, the update token is a triple $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$, and the update of ciphertext $c = (\mathbf{H}_\mu, \mathbf{b})$ works by multiplying $\mathbf{b}$ by $\mathbf{M}$ and then adding $\mathbf{b}_0$. That

is, $c' = (\mathbf{H}'_u, \mathbf{b}')$, where

$$\begin{aligned}(\mathbf{b}')^t &= \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ &= \left[\mathbf{s}^t \mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right] \mathbf{M} + (\mathbf{s}')^t \mathbf{A}'_\mu + (\mathbf{e}')^t \\ &= (\mathbf{s}+\mathbf{s}')^t \mathbf{A}'_\mu + \left(\mathbf{e}^t \mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q. \end{aligned} \tag{3}$$

Thus, the updated ciphertext shares the same form with old ciphertext, but has new independent invertible matrix and new random factor $\mathbf{s}+\mathbf{s}'$, thereby avoiding the two problems mentioned above. Note that even if an adversary corrupts the update token and the old key (or new key), it can only recover $\mathbf{A}'_\mu$ (or $\mathbf{A}_\mu$, resp.) that is actually public. Therefore, the UE scheme does not leak any information about secret keys, and its token generation process is different from the previously commonly used Dec-then-Enc method.

Regarding the CCA-1 security, at a high level, the decryption procedure allows the adversary to recover at most $\mathbf{A}_\mu$ before the challenge phase, while ensuring that the secret key $\mathbf{R}$ (i.e., the trapdoor) remains statistically hidden from the adversary. We state that the scheme cannot achieve CCA-2 security as the decryption of a challenge ciphertext with extra small noise, which is also a valid ciphertext, reveals the information of the challenge plaintext. We note that the construction of a CCA-2 secure UE remains open.

A Packing UE. Our packing UE scheme allows for the simultaneous encryption and updating of multiple messages in a single ciphertext. It is based on our UE scheme with main difference in the encoding algorithm as follows:

$$\mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) = \mathsf{encode}(\mathbf{m}_0) + \mathsf{encode}(\mathbf{m}_1) X + \cdots + \mathsf{encode}(\mathbf{m}_{N-1}) X^{N-1},$$

for messages $\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}$, where the encode in the right side is the same as that in the PKE scheme. Multiple messages blocks are encrypted into one single ciphertext, which can then be recovered degree by degree. This packing scheme enhances efficiency by requiring only one ciphertext header to be downloaded during the update process.

1.3 Summary of Contributions

We strengthen the confidentiality notions for c-d UE to address the above limitations 1-3 of existing work and provide efficient UE schemes that achieve the confidentiality we define. First, we simplify the description of confidentiality model with less oracles available to the adversary while maintaining the same security, which facilitates the security analysis of UE schemes. Our new definition "maximizes" the capability of the adversary, including the ability to corrupt keys in an adaptive manner and gain access to the decryption oracle, thus providing a stronger security than prior work. We then propose a new construction that is the first c-d UE to achieve CCA-1 security under the LWE assumption. It is built on our lattice-based PKE scheme, and rotates ciphertext with a key-switching matrix, which differs from the Dec-then-Enc structure used in existing

c-d UE schemes. We also propose a new packing method to further enhance the efficiency of c-d UE. Our approach enables multiple messages to be encrypted and updated simultaneously, reducing the overhead associated with downloading ciphertext headers during the update process.

2 Preliminaries

We use upper-case and lower-case bold letters to denote matrices and column vectors, respectively. For a vector $\mathbf{x}$, we denote the 2-norm of $\mathbf{x}$ by $\|\mathbf{x}\|$ and the infinity norm by $\|\mathbf{x}\|_\infty$. The largest singular value of a matrix $\mathbf{B}$ is denoted by $s_1(\mathbf{B}) := \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\|$, where the maxima is taken over all unit vectors $\mathbf{u}$ and $\mathbf{B}^t$ is the transposition of $\mathbf{B}$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, $[\mathbf{A} \mid \mathbf{B}]$ denotes the concatenation of the columns of $\mathbf{A}$ and $\mathbf{B}$. We also use standard asymptotical notations such as ω, Ω and O.

2.1 Updatable Encryption

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

Definition 1 ([7,13,6]). *A ciphertext-dependent UE scheme includes a tuple of PPT algorithms* $\{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update}\}$ *that operate in epochs starting from 0.*

- $\mathsf{KG}(1^\lambda)$*: the key generation algorithm outputs an epoch key* $\mathsf{k_e}$*.*
- $\mathsf{Enc}(\mathsf{k_e}, \mathsf{m})$*: 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.,* $\mathsf{ct} = (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$*.*
- $\mathsf{Dec}(\mathsf{k_e}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: the decryption algorithms takes as input an epoch key* $\mathsf{k_e}$ *and a ciphertext* $(\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e})$ *and outputs a message* m' *or* $\perp$*.*
- $\mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$*: 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* $\perp$*.*
- $\mathsf{Update}(\Delta_{\mathsf{e+1},\hat{\mathsf{ct}}_\mathsf{e}}, (\hat{\mathsf{ct}}_\mathsf{e}, \mathsf{ct_e}))$*: 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 via the update algorithm to produce an updated ciphertext. The *correctness* of a UE scheme requires both types of ciphertexts to decrypt correctly to the underlying message, except with a low failure probability.

Prior Notions of Confidentiality. To capture the security under key leakage, the challenger in prior confidentiality games [13,6] provides the adversary some

selective keys in the setup phase. In the query phase, the adversary is given access to query the algorithms involved in UE schemes, including {Enc, TokenGen, Update}, to obtain the encryption of messages, update tokens, and updates of ciphertexts, respectively. The adversary then submits two challenge inputs 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 which challenge input the challenge output is related to (encrypted or updated from). The adversary can continue querying those oracles as long as the combination of queries would not lead to a trivial win, and eventually submits a guess bit.

Prior confidentiality notions have three variants with the only difference in challenge inputs: UP-IND [13] has inputs of two messages $(\bar{m}_0, \bar{m}_1)$ to capture the security of fresh encryptions, UP-REENC [13] uses inputs of two ciphertexts $(\bar{c}_0, \bar{c}_1)$ to protect the confidentiality after updating, and Confidentiality [6], which is stronger the former two, takes one message and one ciphertext as input $(\bar{m}_0, \bar{c}_1)$ to protect against the leakage of the *age* of ciphertext, i.e., the number of update times, to the adversary. We rewrite the confidentiality game of Confidentiality in Fig. 2 with two modifications.

First, we describe oracles that operate in consecutive epochs $\{\ldots, e-1, e, e+1, \ldots\}$, which is more consistent with the practical periodic updating of ciphertexts and differs from the node-based oracles originating from proxy re-encryption in prior work. Second, we introduce a new lookup table in the game to track non-challenge ciphertexts (as defined in Definition 2) to address the insufficient analysis of trivial win conditions for deterministic UE schemes in prior work [7,13,6]. Our main observation is that for UE schemes with deterministic updates, the adversary should be prevented from querying $\mathcal{O}_{\mathsf{Update}}$ and $\mathcal{O}_{\mathsf{TokenGen}}$ on the challenge input ciphertext in the challenge epoch before querying the challenge oracle, as this would enable the adversary to know one of the possible challenge output ciphertexts in advance due to the determinism of the update. Such conditions are not analyzed in prior notions, which are therefore only applicable to UE with randomized updates; however, the update algorithm can be deterministic as in our construction, even though the encryption algorithm must be randomized.

Definition 2. *A ciphertext is called challenge-equal ciphertext, if adversary learns it via querying the challenge oracle $\mathcal{O}_{\mathsf{Chall}}$, or obtains it by updating the challenge ciphertext using $\mathcal{O}_{\mathsf{Update}}$ or tokens acquired from $\mathcal{O}_{\mathsf{TokenGen}}$. Any ciphertext that is not obtained through these methods is referred to as a non-challenge ciphertext.*

The functionalities and restrictions of oracles used in the Confidentiality game in Fig. 2 are as follows.

- $\mathcal{O}_{\mathsf{Enc}}$: returns an encryption of a message.
- $\mathcal{O}_{\mathsf{Update}}$: returns an update of a valid (line 1-3) ciphertext, recorded by $\mathsf{TC}_{\mathsf{chall}}$ (line 8) or $\mathsf{TC}_{\mathsf{non}}$ (line 9) according to the input. But the update of challenge-equal ciphertexts in epochs with known epoch keys is not allowed (line 7).
- $\mathcal{O}_{\mathsf{TokenGen}}$: returns a token related to a valid ciphertext, and updates $\mathsf{TC}_{\mathsf{chall}}$ (line 6) or $\mathsf{TC}_{\mathsf{non}}$ (line 8). But tokens related to challenge-equal ciphertexts in epochs with known epoch keys is not allowed to acquire (line 1-2).

$\mathrm{Expt}^{\mathsf{Confidentiality}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, b)$:

1 : $\mathsf{k}_1, \ldots, \mathsf{k}_l \leftarrow \mathsf{KG}(1^\lambda)$
2 : $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathcal{K})$
3 : **return** $b' = b$

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

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

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

1 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ and
2 : $\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ **then**
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
5 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$ **then**
6 : $\quad$ **if** $\mathsf{e} \in \mathcal{K}$ **return** $\perp$
7 : $\quad$ **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
8 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
9 : $\quad\quad \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
10 : **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
11 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
12 : $\quad\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
13 : **return** $(\hat{\mathsf{ct}}', \mathsf{ct}')$

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

1 : **if** $\mathsf{e} \in \mathcal{K}$ and $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$
2 : $\quad$ **return** $\perp$
3 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
4 : **if** $\mathsf{e} \notin \mathcal{K}$ and $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$
5 : $\quad \mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
6 : $\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
7 : $\quad \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
8 : **elseif** $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$
9 : $\quad \mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
10 : $\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
11 : $\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
12 : **else return** $\perp$
13 : **return** $\Delta_{\mathsf{e},\hat{\mathsf{ct}}}$

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

1 : **if** $\mathsf{e} \in \mathcal{K}$ **return** $\perp$
2 : $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}(\mathsf{k}_\mathsf{e}, \mathsf{m})$
3 : **if** $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \mathsf{ct}$
4 : $\quad$ **return** $\perp$
5 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}\left(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}}\right)$
6 : $(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
7 : **if** $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $\left|\mathsf{ct}'_0\right| \neq \left|\mathsf{ct}'_1\right|$
8 : $\quad$ **return** $\perp$
9 : **if** $(\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1)$
10 : $\quad$ **return** $\perp$
11 : $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$
12 : **return** $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Fig. 2. Security game for Confidentiality. The adversary in the startup are provided with selective keys whose epochs are recorded by the set $\mathcal{K}$, and the other keys are kept private from the adversary. Initially set to be empty, the table $\mathsf{T}_{\mathsf{chall}}$ (or $\mathsf{T}_{\mathsf{non}}$) maps an epoch and challenge-equal (or non-challenge, respectively) ciphertext header pair to the corresponding challenge-equal (or non-challenge, respectively) ciphertext body. $\mathsf{xx} = \mathsf{det}$ means the update algorithm is deterministic.

- $\mathcal{O}_{\mathsf{Chall}}$: returns the challenge output, either a fresh encryption of input message or an update of input valid ciphertext (line 2-4). However, this oracle

should not be queried in epochs with known epoch keys (line 1), and for deterministic UE, the input ciphertext should not be updated in advance (line 9-10).

In fact, the adversary may infer more ciphertexts, tokens and keys from corrupted information, aside from the recorded sets, and the extended leakages cannot be tracked (but can be computed) by look-up tables. For example, a token can be inferred if two successive epoch keys are known. We will show in lemmas 3 to 5 that trivial win conditions on recorded leakages and extended leakages are actually the same for no-directional UE (Def. 5). Therefore, it is sufficient to check the above restrictions on recorded look-up tables to avoid trivial win.

2.2 Gaussians and Lattices

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we first review the Learning With Errors (LWE) and Short Integer Solution (SIS) problems as follows:

- $\mathsf{LWE}_{q,\alpha}$: for arbitrary $\mathbf{s} \in \mathbb{Z}_q^n$ and error $\mathbf{e}$ from the discrete Gaussian distribution $D_{\mathbb{Z}^m,\alpha q}$ (Def. 4), 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 $\mathbf{s}$ and $\mathbf{e}$ from $(\mathbf{A}, \mathbf{b})$; the *decision*-$\mathsf{LWE}_{q,\alpha}$ is to distinguish between $\mathbf{b}$ and a uniformly random sample from $\mathbb{Z}_q^m$.
- $\mathsf{SIS}_{q,\beta}$: find a nonzero $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\| \le \beta$.

When $\mathbf{A}$ is a uniformly random matrix, solving the above two problems is computationally intractable under some parameter settings [2,23]. However, for a random matrix $\mathbf{A}$ with a $\mathbf{G}$-trapdoor (Def. 3), those two problems can be solved immediately (Lemma 1 and Lemma 2).

For the rest of the paper, let $q \ge 2$ be an integer modulus with $k = \lceil \log_2 q \rceil$, and $\mathbf{G}$ is defined as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n\times nk}$, i.e.,

$$\mathbf{G} = diag(\mathbf{g}^t, \ldots, \mathbf{g}^t),$$

where $\mathbf{g}^t = [1\ 2\ 4\ \ldots\ 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$ and integer $n \ge 1$.

Definition 3 (G-trapdoor). *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for some $m \ge nk \ge n$. A $\mathbf{G}$-trapdoor for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}_q^{(m-nk)\times nk}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$.*

As an example in [21], $\mathbf{R}$ is a $\mathbf{G}$-trapdoor for a random matrix $\mathbf{A} = [\mathbf{A}_0| - \mathbf{A}_0\mathbf{R} + \mathbf{HG}]$, where $\mathbf{A}_0$ is a uniform matrix in $\mathbb{Z}_q^{n\times m}$, $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ is an invertible matrix and $\mathbf{R}$ is chosen from a distribution over $\mathbb{Z}_q^{m\times nk}$.

Lemma 1 ([21], Theorem 5.4). *Given 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 $\|[\mathbf{R}^t\ \mathbf{I}] \cdot \mathbf{e}\|_\infty \le q/4$, 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 $\mathbf{b}^t = \mathbf{s}^t\mathbf{A}+\mathbf{e}^t$.*

Lemma 2 ([21], Theorem 5.5). *Given a* $\mathbf{G}$*-trapdoor* $\mathbf{R}$ *for* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *with invertible matrix* $\mathbf{H}$ *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* $D_{\mathbb{Z}^m,s}$ *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).*

Definition 4 ([1]). *For a positive real* s*, the discrete Gaussian distribution over a countable set* A *is defined by the density function*

$$D_{A,s}(x) := \frac{\rho_s(x)}{\sum_{y\in A}\rho_s(y)},$$

where $\rho_s(x) = \exp\left(-\pi\|x\|^2/s^2\right)$.

Lemma 1 and Lemma 2 work for $\mathbf{G}$ as well. More conclusions related to Gaussians and lattices are provided in Supplementary Material B.

3 New Confidentiality Notions for Updatable Encryption

To simplify the security notion given in [6], we define a new confidentiality notion called $\mathsf{sConfidentiality}$, where we replace $\mathcal{O}_{\mathsf{TokenGen}}$ and $\mathcal{O}_{\mathsf{Update}}$ in the security game with a single $\mathcal{O}_{\mathsf{sUpd}}$ that returns both the update token and updated ciphertext to the adversary simultaneously. We prove in Theorem 1 that $\mathsf{sConfidentiality}$ and $\mathsf{Confidentiality}$ are equal for UE schemes with no-directional key updates.

Meanwhile, in order to provide the adversary with maximum power, we introduce a new stronger confidentiality notion than $\mathsf{sConfidentiality}$, called $\mathsf{xxIND\text{-}UE\text{-}atk}$[4], where the adversary is given extra access to $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{Corr}}$, which enables it to corrupt epoch keys at any time during the game. To avoid making the security game trivial, we fully analyze the conditions for any trivial win in this game model. A brief comparison of the proposed notions with those of prior work is presented in Fig. 3.

3.1 UE Schemes with No-Directional Key Updates

In c-i UE schemes, update tokens are generated by two successive epoch keys: $\Delta = \mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}})$, e.g., $\Delta = \mathsf{k_{e+1}}/\mathsf{k_e}$ in [8] or $\Delta = \mathsf{k_{e+1}} - \mathsf{k_e}$ in [17]); therefore, one key may be derived by the other if the token is known by the adversary. However, in c-d UE schemes, tokens are also determined by the ciphertext header: $\Delta = \mathsf{TokenGen}(\mathsf{k_e}, \mathsf{k_{e+1}}, \hat{\mathsf{ct}}_\mathsf{e})$, so keys may not be derived via corrupted tokens. We generalise the definition of no-directional key updates from c-i UE to c-d UE as follows.

[4] The same notion for c-i UE scheme was proposed in [8]. We aim to unify the notions for c-i/c-d UE that both capture adaptive security and prevent the leakage of ciphertext age. Note that, as analysed in the introduction, there are intrinsic differences between c-i UE and c-d UE. The disparity is evident in the confidentiality notion, specifically in the approach to recording leakage sets.

Notions	Oracles	Compromised Key	Challenge Input	Update
UP-IND [13]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{m}}_1)$	rand
UP-REENC [13]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{c}}_0, \bar{\mathsf{c}}_1)$	rand
Confidentiality [6]	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{TokenGen}}, \mathcal{O}_{\mathsf{Update}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	rand
sConfidentiality Sect. 3.2	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}$	Selective	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	rand
xxIND-UE-CPA Sect. 3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}},$	Adaptive	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	xx
xxIND-UE-CCA Sect. 3.3	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Corr}}, \mathcal{O}_{\mathsf{Dec}}$	Adaptive	$(\bar{\mathsf{m}}_0, \bar{\mathsf{c}}_1)$	xx

Fig. 3. 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 in [6], and $\mathcal{O}_{\mathsf{sUpd}}$ is defined in Sect. 3.2.

Definition 5. *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.*

No-directional UE is stronger than other variants [17]. All known c-d UE schemes in [7,13,6] (as well as our construction in Sect. 5), have no-directional key updates, which benefits from a Dec-then-Enc process as discussed in the introduction, even though there are only two c-i UE schemes with no-directional key update: one is not practical [19] and the other is less efficient [25]. In the following, we focus on c-d UE schemes with no-directional keys updates.

3.2 A Simplified Confidentiality Notion

Based on our refinement on Confidentiality, we now define a new simplified confidentiality notion by substituting the oracles $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{Upd}}$ in the Confidentiality game with a single $\mathcal{O}_{\mathsf{sUpd}}$ that returns both the token and update simultaneously. We call this new notion sConfidentiality. In Theorem 1, we prove sConfidentiality is equivalent to Confidentiality for UE schemes with no-directional key updates, as defined in [6].

Definition 6 (sConfientiality). *Let* $\mathsf{UE} = \{\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update}\}$ *be an updatable encryption scheme. For a security parameter* λ*, an integer* l*, an adversary* $\mathcal{A}$*, and a binary bit* $b \in \{0, 1\}$*, we define the confidentiality experiment* $\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, b)$ *and oracles* $\mathcal{O} = \{\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \mathcal{O}_{\mathsf{Chall}}\}$ *as described in Fig. 4. The experiment maintains two look-up tables* $\mathsf{TC}_{\mathsf{non}}$ *and* $\mathsf{TC}_{\mathsf{chall}}$ *that record non-challenge and challenge-equal ciphertexts known to the adversary, respectively, and an epoch set* $\mathcal{K}$ *in which epoch keys are provided to the adversary in setup.*

We say that an updatable encryption scheme UE *satisfies* sConfidentiality *if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $\mathcal{K} \subseteq [0, \ldots, l]$ *and*

$\mathrm{Expt}^{\mathsf{sConf}}_{\mathsf{UE}}(\lambda, l, \mathcal{A}, b)$:

1 : $\mathsf{k}_1, \ldots, \mathsf{k}_l \leftarrow \mathsf{KG}(1^\lambda)$
2 : $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathcal{K})$
3 : **return** $b' = b$

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

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

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

1 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ and
2 : $\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ **then**
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
5 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$ **then**
6 : $\quad$ **if** $\mathsf{e} \in \mathcal{K}$ **return** $\perp$
7 : $\quad$ **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
8 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
9 : $\quad\quad \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
10 : **else** $\mathsf{ct} \leftarrow \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}]$
11 : $\quad\quad (\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
12 : $\quad\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
13 : **return** $(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}', \mathsf{ct}'))$

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

1 : **if** $\mathsf{e} \in \mathcal{K}$ **return** $\perp$
2 : $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}(\mathsf{k}_\mathsf{e}, \mathsf{m})$
3 : **if** $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \mathsf{ct}$
4 : $\quad$ **return** $\perp$
5 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e}, \hat{\mathsf{ct}})$
6 : $(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
7 : **if** $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $|\mathsf{ct}'_0| \neq |\mathsf{ct}'_1|$
8 : $\quad$ **return** $\perp$
9 : **if** $(\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1)$
10 : $\quad$ **return** $\perp$
11 : $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$
12 : **return** $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Fig. 4. Security game for sConfidentiality in Definition 6.

efficient adversaries $\mathcal{A}$, we have

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

Theorem 1. *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update})$ *be an updatable encryption scheme with no-directional key updates. For any* sConfidentiality *adversary $\mathcal{A}$ against* UE*, there is a* Confidentiality *adversary $\mathcal{B}$ against* UE *such that*

$$\mathsf{Adv}^{\mathsf{sConf}}_{\mathsf{UE},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{Conf}}_{\mathsf{UE},\mathcal{B}}(1^\lambda). \tag{4}$$

In addition, 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. In general, we construct a reduction that runs the Confidentiality (or sConfidentiality) game and simulates all responses to the queries of the adversary in the sConfidentiality (or Confidentiality game, respectively), as shown in Fig. 5. The details are presented in Supplementary Material A. □

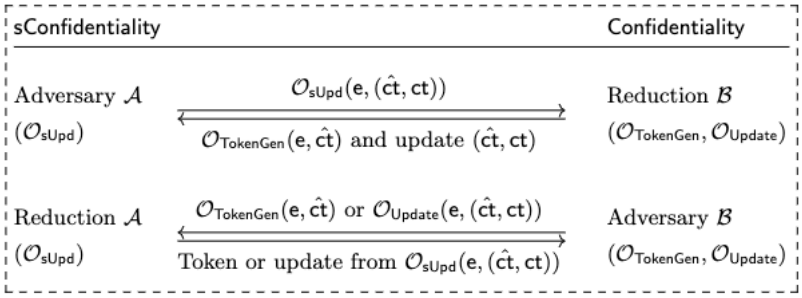

Fig. 5. Reductions in the proof of Theorem 1. When the adversary makes queries to specific oracles, indicated above the arrow, the reduction forward to the adversary the corresponding responses from its own challenger, marked below the arrow.

3.3 A Stronger Confidentiality Notion

We now provide a stronger confidentiality notion, called xxIND-UE-atk for c-d UE in Definition 7, which provides the adversary with more power than the notion of sConfidentiality in Sect. 3.2. All available oracles that the adversary has access to are described in Fig. 8. The stronger notion allows the adversary to corrupt keys at any time during the game by querying $\mathcal{O}_{\mathsf{Corr}}$, instead of selecting the compromised keys in the setup phrase. In addition, the adversary is provided with an extra ability to query the decryption oracle compared with sConfidentiality. Prior to define xxIND-UE-atk, we first analyze the conditions that lead the adversary to trivially win the game through a combination of queries, which therefore should be excluded from the game.

Leakage Information. To track the information leaked to the adversary, we similarly record two look-up tables $\mathsf{TC}_{\mathsf{non}}$ and $\mathsf{TC}_{\mathsf{chall}}$ as defined in Sect. 3.2, and an epoch set $\mathcal{K}$ in which the epoch key is corrupted via $\mathcal{O}_{\mathsf{Corr}}$. We define $\mathsf{TC}_{\mathsf{chall}}[0]$ as the set of epochs in which the adversary learns a challenge-equal ciphertext, and $\mathcal{T}$ as the set of epochs in which the adversary learns a token corresponding to a challenge-equal ciphertext, which are exactly the epochs stored in $\mathsf{TC}_{\mathsf{chall}}$ and $\Delta_{\mathsf{e},\hat{\mathsf{ct}}}$, respectively. A summary of notations is shown in Table 1.

Table 1. Summary of leakage set notations

Notations	Descriptions
$\mathsf{TC}_{\mathsf{non}}$	Look-up table recording leaked non-challenge ciphertexts
$\mathsf{TC}_{\mathsf{chall}}$	Look-up table recording leaked challenge-equal ciphertexts
$\mathsf{TC}_{\mathsf{chall}}[0]$	Set of epochs in which the a challenge-equal ciphertext is learned
$\mathcal{K}$	Set of epochs in which the adversary learned the epoch key
$\mathcal{T}$	Set of epochs in which a token w.r.t. a challenge-equal ct is learned

Leakage Extension. Note that the adversary possibly extends its corrupted information $\mathsf{TC}_{\mathsf{non}}, \mathsf{TC}_{\mathsf{chall}}, \mathcal{K}$ via corrupted tokens, and the former leakages may also in turn help to corrupt more tokens. We denote $\mathsf{TC}^*_{\mathsf{chall}}[0], \mathcal{K}^*, \mathcal{T}^*$ as the extended sets of $\mathsf{TC}_{\mathsf{chall}}[0], \mathcal{K}, \mathcal{T}$, respectively. Following the analysis in [20], the extended leakage sets are computed as follows:

$$\mathcal{K}^* = \mathcal{K} \text{ (no-directional key updates)}, \tag{5}$$

$$\mathcal{T}^* = \{\mathsf{e} \in \{0, \ldots, l\} \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e} - 1 \in \mathcal{K}^*\}, \tag{6}$$

$$\begin{aligned} \mathsf{TC}^*_{\mathsf{chall}}[0] = \{\mathsf{e} \in \{0, \ldots, l\} \mid (\mathsf{e} \in \mathsf{TC}_{\mathsf{chall}}[0]) \vee (\mathsf{e} - 1 \in \mathsf{TC}_{\mathsf{chall}}[0] \wedge \mathsf{e} \in \mathcal{T}^*) \vee \\ (\mathsf{e} + 1 \in \mathsf{TC}_{\mathsf{chall}}[0] \wedge \mathsf{e} + 1 \in \mathcal{T}^*)\}. \end{aligned} \tag{7}$$

An example is shown in Fig. 6. Assume the adversary queries $\mathcal{O}_{\mathsf{sUpd}}$ only in epoch $\mathsf{e} - 5$ and corrupts epoch keys in epochs $\mathsf{e} - 5$ and $\mathsf{e} - 4$. Even though it cannot learn the token in epoch $\mathsf{e} - 4$ by $\mathcal{O}_{\mathsf{sUpd}}$, it can infer that token via corrupted keys in $\mathsf{e} - 5$ and $\mathsf{e} - 4$, which further infers the ciphertexts in $\mathsf{e} - 4$.

Epoch	...	$\mathsf{e}-5$		$\mathsf{e}-4$	...
$\mathsf{TC}_{\mathsf{chall}}[0]$		✓		×	
$\mathcal{K}$		✓		✓	
$\mathcal{T}$	✓		×		
$\mathsf{TC}^*_{\mathsf{chall}}[0]$		✓		✓	
$\mathcal{K}^*$		✓		✓	
$\mathcal{T}^*$	✓		✓		

Fig. 6. Example of leakage sets. Marks ✓ and × indicate if an epoch key/token is corrupted. The green mark ✓ indicates an epoch key/token can be inferred from other corrupted keys and tokens.

Trivial Win Conditions. We follow the analysis of trivial win conditions for c-i UE in [20,19,8,17], as shown in Fig. 7. Our analysis for c-d UE in lemmas 3 to 5 shows that it is sufficient to check trivial win conditions on recorded leakages $\mathcal{K}, \mathsf{TC}_{\mathsf{chall}}, \mathcal{T}$, eliminating the need to calculate extended leakages $\mathcal{K}^*, \mathsf{TC}^*_{\mathsf{chall}}, \mathcal{T}^*$ and check trivial win conditions on them.

I. Trivial win by keys and ciphertexts

If the adversary knows the epoch key and a valid challenge-equal ciphertext in the same epoch, it can recover the underlying message by a direct decryption with its corrupted key and therefore win the game. Namely, we should ensure $\mathcal{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] = \emptyset$. The following lemma shows this condition is equal to $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$ for c-d UE with no-directional key updates.

Lemma 3. *For c-d UE schemes with no-directional key updates, we have* $\mathcal{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] = \emptyset \iff \mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$.

Abilities	Trivial Win Conditions
Keys and ciphertexts	$\mathcal{K}^* \cap \mathsf{TC}^*_{\mathsf{chall}}[0] \neq \emptyset$
Updates	rand-UE : $-$ det-UE : $\bar{\mathsf{e}} \in \mathcal{T}^*$ or $\mathcal{O}_{\mathsf{sUpd}}(\bar{\mathsf{e}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ is queried (line 8-9)
Decryptions	rand-UE : $\mathsf{e} \in \mathsf{TC}^*_{\mathsf{chall}}[0]$ and $(\mathsf{m}' = \mathsf{m}$ or $\mathsf{m}_1)$ (line 3-4) det-UE : $\mathsf{TC}^*_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}] = \mathsf{ct}$ (line 2)

Fig. 7. A summary of trivial win conditions, where $\bar{\mathsf{e}}$ is the challenge epoch, $(\hat{\mathsf{ct}}, \mathsf{ct})$ is the challenge input ciphertext whose underlying message is m_1, m is the challenge input message, and m' is returned message of decryption algorithm. Oracles are given in Fig. 8.

Proof. By the definition of no-directional key updates, we have $\mathcal{K}^* = \mathcal{K}$. In addition, we have $\mathsf{TC}_{\mathsf{chall}}[0] \subseteq \mathsf{TC}^*_{\mathsf{chall}}[0]$. Therefore, we only need to prove $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$ when $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$.

Suppose $\mathsf{TC}_{\mathsf{chall}}[0] = \cup\{\mathsf{e}_{start}, \ldots, \mathsf{e}_{end}\}$. We prove that the adversary cannot learn a challenge-equal ciphertext in epoch e_{end+1} either by querying or inferring. First, the adversary cannot learn a challenge-equal ciphertext in epoch e_{end+1} via querying $\mathcal{O}_{\mathsf{sUpd}}$, since e_{end} is the last epoch in the epoch continuum; otherwise 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 token as Eq. (7). 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} (see Eq. (6)).

The proof is the same for the challenge-equal ciphertext in epoch e_{start}. Therefore, the adversary cannot learn a challenge-equal ciphertext in any epoch outside of the set $\mathsf{TC}_{\mathsf{chall}}[0]$, which implies that $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$. □

Remark 1. Lemma 3 shows that the adversary cannot infer a challenge-equal ciphertext in an epoch that is not recorded in the look-up table, i.e., $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^*_{\mathsf{chall}}[0]$. But that does not mean all the ciphertexts known to the adversary are stored in the table $\mathsf{TC}_{\mathsf{chall}}$, or equally $\mathsf{TC}_{\mathsf{chall}} = \mathsf{TC}^*_{\mathsf{chall}}$, which is only true for deterministic UE. 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.

II. Trivial win by updates

For UE schemes with randomized updates, there is no restrictions on the update oracle. However, for UE schemes with deterministic updates, the adversary can learn one of the possible challenge outputs by querying the oracle $\mathcal{O}_{\mathsf{sUpd}}$ on the challenge input $(\hat{\mathsf{ct}}, \mathsf{ct})$, or infer the update of $(\hat{\mathsf{ct}}, \mathsf{ct})$ if $\bar{\mathsf{e}} \in \mathcal{T}^*$, in advance before the challenge phase. In the first case, all known ciphertext leakages before

the challenge are recorded by $\mathsf{TC}_{\mathsf{non}}$, so that we can set line 8-9 in challenge oracle to check for this, as shown Fig. 8. In the second case, if $\bar{\mathsf{e}} \in \mathcal{T}(\subseteq \mathcal{T}^*)$, i.e., the token is learned by querying $\mathcal{O}_{\mathsf{sUpd}}$, it goes back to the first case ($\mathcal{O}_{\mathsf{sUpd}}$ also returns the update ciphertext, which is recorded in $\mathsf{TC}_{\mathsf{non}}$). If $\bar{\mathsf{e}} \in \mathcal{T}^* \backslash \mathcal{T}$, the following lemma shows the impossibility.

Lemma 4. *For c-d UE schemes with no-directional key updates, if* $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$*, then the challenge epoch* $\bar{\mathsf{e}} \notin \mathcal{T}^* \backslash \mathcal{T}$.

Proof. Note that since the adversary queries the challenge oracle in $\bar{\mathsf{e}}$, then $\bar{\mathsf{e}} \in \mathsf{TC}_{\mathsf{chall}}[0]$. Due to $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$, we know the epoch key $\mathsf{k}_{\bar{\mathsf{e}}}$ is unknown to the adversary, which is necessary to infer $\Delta_{\bar{\mathsf{e}},\hat{\mathsf{ct}}}$ (see Eq. (6)). □

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

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

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

1 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ and
2 : $\quad \mathsf{TC}_{\mathsf{non}}[\mathsf{e}-1, \hat{\mathsf{ct}}] = \perp$ **then**
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\mathsf{e},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_{\mathsf{e}}, \hat{\mathsf{ct}})$
5 : $(\hat{\mathsf{ct}}', \mathsf{ct}') \leftarrow \mathsf{Update}(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
6 : **if** $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}-1, \hat{\mathsf{ct}}] \neq \perp$
7 : $\quad \mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
8 : **else** $\mathsf{TC}_{\mathsf{non}}[\mathsf{e}, \hat{\mathsf{ct}}'] \leftarrow \mathsf{ct}'$
9 : **return** $(\Delta_{\mathsf{e},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}', \mathsf{ct}'))$

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

1 : $\mathcal{K} = \mathcal{K} \cup \{\mathsf{e}\}$
2 : **return** k_{e}

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

1 : m' or $\perp \leftarrow \mathsf{Dec}(\mathsf{k}_{\mathsf{e}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
2 : **if** $(\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{chall}}[\mathsf{e}, \hat{\mathsf{ct}}] = \mathsf{ct})$ or
3 : $\quad \big((\mathsf{xx} = \mathsf{rand}$ and $\mathsf{e} \in \mathsf{TC}_{\mathsf{chall}}[0])$ and
4 : $\quad (\mathsf{m}' = \mathsf{m}$ or $\mathsf{m}_1)\big)$ **then**
5 : $\quad$ **return** $\perp$
6 : **return** $\mathsf{Dec}\left(\mathsf{k}_{\mathsf{e}}, (\hat{\mathsf{ct}}, \mathsf{ct})\right)$

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

1 : $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) \leftarrow \mathsf{Enc}\left(\mathsf{k}_{\bar{\mathsf{e}}}, \mathsf{m}\right)$
2 : **if** $(\hat{\mathsf{ct}}'_0, \mathsf{ct}'_0) = \perp$ or $\mathsf{TC}_{\mathsf{non}}[\bar{\mathsf{e}}-1, \hat{\mathsf{ct}}] \neq \mathsf{ct}$
3 : $\quad$ **return** $\perp$
4 : $\Delta_{\bar{\mathsf{e}},\hat{\mathsf{ct}}} \leftarrow \mathsf{TokenGen}\left(\mathsf{k}_{\bar{\mathsf{e}}-1}, \mathsf{k}_{\bar{\mathsf{e}}}, \hat{\mathsf{ct}}\right)$
5 : $(\hat{\mathsf{ct}}'_1, \mathsf{ct}'_1) \leftarrow \mathsf{Update}(\Delta_{\bar{\mathsf{e}},\hat{\mathsf{ct}}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$
6 : **if** $|\hat{\mathsf{ct}}'_0| \neq |\hat{\mathsf{ct}}'_1|$ or $\left|\mathsf{ct}'_0\right| \neq \left|\mathsf{ct}'_1\right|$
7 : $\quad$ **return** $\perp$
8 : **if** $(\mathsf{xx} = \mathsf{det}$ and $\mathsf{TC}_{\mathsf{non}}[\bar{\mathsf{e}}, \hat{\mathsf{ct}}'_1] = \mathsf{ct}'_1)$
9 : $\quad$ **return** $\perp$
10 : $\mathsf{TC}_{\mathsf{chall}}[\bar{\mathsf{e}}, \hat{\mathsf{ct}}'_b] \leftarrow \mathsf{ct}'_b$
11 : **return** $(\hat{\mathsf{ct}}'_b, \mathsf{ct}'_b)$

Fig. 8. An overview of the oracles that the adversary has access to in Definition 7. In the decryption oracle, m is the challenge input message and m_1 is the underlying message of the challenge input ciphertext.

III. Trivial win by decryptions

Table $\mathsf{TC}^*_{\mathsf{chall}}$ records all the challenge-equal ciphertexts known to the adversary in the game. By remark 1, we first have the following lemma.

Lemma 5. *For c-d UE schemes with no-directional key updates, if* $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$*, then* $\mathsf{TC}^*_{\mathsf{chall}} = \mathsf{TC}_{\mathsf{chall}}$ *for deterministic UE, and* $\mathsf{TC}^*_{\mathsf{chall}}[0] = \mathsf{TC}_{\mathsf{chall}}[0]$ *for randomized UE.*

For UE schemes with deterministic ciphertext updates, table $\mathsf{TC}_{\mathsf{chall}}$ records all leaked challenge-equal ciphertexts in the game. The adversary can trivially win the game by querying the decryption oracle on the challenge-equal ciphertexts recorded on the table $\mathsf{TC}_{\mathsf{chall}}$ (line 2 in $\mathcal{O}_{\mathsf{Dec}}$, Fig. 8).

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 (line 3-4).

In summary, the above analysis shows trivial win conditions for c-d UE can be checked immediately based on the recorded leakages during the confidentiality game, without the need for extra calculations and further checks of the extended leaked sets of keys, tokens and ciphertext as in previous work for c-i UE in [20,8,17]. After all the queries, if $\perp$ is not returned, only one condition remains to be checked: $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] = \emptyset$. This advantage is due to both the no-directional key update setting and the proper ways of recording leakage information via look-up tables. Finally, we introduce the definition of xxIND-UE-atk.

Definition 7 (xxIND-UE-atk). *Let* $\mathsf{UE} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TokenGen}, \mathsf{Update})$ *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. 9 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

Firewall Technique. In c-i UE, the firewall technique was developed in [20,19] to facilitate the security proof by separating epochs into different regions. Inside an insulated region, 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.

In c-d UE, we similarly define the insulated region, inside which all tokens related to challenge-equal ciphertexts (called challenge-equal tokens) are corrupted but no epoch key is corrupted.

Definition 8 (Firewall). *In ciphertext-dependent UE schemes, an insulated region with firewalls* fwl *and* fwr*, denoted by* $\mathcal{FW}$*, is consecutive sequence of epochs* $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ *for which:*

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

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. 9. The confidentiality game $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ where $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$ indicates the type of UE scheme (deterministic or randomized) and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA\text{-}1}, \mathsf{CCA}\}$ indicates the type of attack model. In the game, the adversary is given access to a set of oracles, denoted by $\mathcal{O}_1$ and $\mathcal{O}_2$ which are shown in Fig. 8 and Fig. 10. During the setup phase, the adversary generates a challenge plaintext and a challenge ciphertext using the oracles in $\mathcal{O}_1$, and submits them to the challenger in the challenge phase. The adversary continues to query the oracles in $\mathcal{O}_2$ and eventually provides a guess bit. The only condition for the adversary to lose the game is $\mathcal{K} \cap \mathsf{TC}_{\mathsf{chall}}[0] \neq \emptyset$.

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	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \boxed{\mathcal{O}_{\mathsf{Dec}}}, \mathcal{O}_{\mathsf{Corr}}$	$\mathcal{O}_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{sUpd}}, \boxed{\mathcal{O}_{\mathsf{Dec}}}, \mathcal{O}_{\mathsf{Corr}}$

Fig. 10. 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.

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

Suppose a xxIND-UE-atk adversary $\mathcal{A}$ queries the challenge oracle in epoch $\bar{\mathsf{e}}$ and does not trigger trivial win conditions in the game, and $\mathsf{TC}_{\mathsf{chall}}[0] = \cup\{\mathsf{e}_{start}, \ldots, \mathsf{e}_{end}\}$. The proof of Lemma 3 shows $\mathcal{A}$ cannot update a ciphertext from the epoch e_{end} to the start epoch e'_{start} of the next continuum. Thus, we have $\mathsf{TC}_{\mathsf{chall}}[0] = \{\mathsf{e}_{start}, \ldots, \mathsf{e}_{end}\}$, meaning that the epoch set in which $\mathcal{A}$ knows a challenge-equal ciphertext is only a consecutive continuum starting from the challenge epoch ($\mathsf{e}_{start} = \bar{\mathsf{e}}$), and ending 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 epochs e_{start} and $\mathsf{e}_{end}+1$, following from the proof of Lemma 3;
– $\mathcal{A}$ knows all challenge-equal tokens in epochs in $\{\mathsf{e}_{start}+1,\ldots,\mathsf{e}_{end}\}$, 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 $\{\mathsf{e}_{start},\ldots,\mathsf{e}_{end}\}$, as $\mathcal{K}\cap\mathsf{TC}_{\mathsf{chall}}[0]=\emptyset$;

We thus have the Lemma 6, following from the discussion above, and Lemma 7, as a corollary of Lemma 6, both of which provide important tools in the confidentiality proof for c-d UE.

Lemma 6. *Let* $\mathsf{UE}=(\mathsf{KG},\mathsf{Enc},\mathsf{TokenGen},\mathsf{Update},\mathsf{Decrypt})$ *be a c-d UE 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 an insulated region (Def. 8), starting from the challenge epoch and ending at the last epoch in which the adversary queries the* $\mathcal{O}_{\mathsf{sUpd}}$.

Lemma 7. *For a c-d UE with no-direcitonal key updates, 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 propose a new PKE scheme called TDP, which is based on the lattice trapdoor techniques. We will use this scheme in Sect. 5 as the underlying encryption scheme to build our UE scheme.

4.1 A New PKE Scheme

Our overall idea is to construct a 1×3 block matrix $\mathbf{A}_\mu$ in the encryption algorithm, with the secret key serving as the trapdoor for the first two blocks of $\mathbf{A}_\mu$ to ensure the correctness of decryption.

We introduce some parameters involved in the construction in Fig. 11, where we use standard asymptotic notations of O,Ω,ω. Let λ be the security parameter, $\omega(\sqrt{\log n})$ is a fixed function that grows asymptotically faster than $\sqrt{\log n}$, and $\Lambda(\mathbf{G}^t)$ is the lattice generated by $\mathbf{G}^t$.

The PKE scheme TDP is described as follows. On a first reading, we suggest readers to neglect the error parameter settings that are used to control the error bound within the decryption capability, in order to have a simpler view at a high level.

– $\mathsf{TDP.KG}(1^\lambda)$: choose $\mathbf{A}_0\xleftarrow{\$}\mathbb{Z}_q^{n\times\bar{m}}$, $\mathbf{R}_1,\mathbf{R}_2\xleftarrow{\$}\mathcal{D}$ and let $\mathbf{A}=[\mathbf{A}_0\mid\mathbf{A}_1\mid\mathbf{A}_2]=[\mathbf{A}_0\mid-\mathbf{A}_0\mathbf{R}_1\mid-\mathbf{A}_0\mathbf{R}_2]\in\mathbb{Z}_q^{n\times m}$ where $m=\bar{m}+2nk$. The public key is $\mathsf{pk}=\mathbf{A}$ and the secret key is $\mathsf{sk}=\mathbf{R}_1$.

Notations	Functionalities
$\mathbf{G} = \mathbb{Z}_q^{n\times nk}$(Sect. 2.2) $k = \lceil \log_2 q \rceil = O(\log n)$, $q = \mathsf{poly}(\lambda)$	Make oracles $\mathsf{Invert}^{\mathcal{O}}$ and $\mathsf{SampleD}^{\mathcal{O}}$efficient for the random matrix with a $\mathbf{G}$-trapdoor
$\bar{m} = O(nk)$, $\mathcal{D} = D_{\mathbb{Z}^{\bar{m}\times nk},\omega(\sqrt{\log n})}$	Ensure $(\mathbf{A}, \mathbf{AR})$ is $\mathsf{negl}(\lambda)$-far from uniform for $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, due to leftover hash lemma
$\mathsf{encode} : \{0,1\}^{nk} \to \Lambda(\mathbf{G}^t)$ by $\mathsf{encode}(\mathbf{m}) = \mathbf{Bm} \in \mathbb{Z}^{nk}$, and $\mathbf{B}$ is any basis of $\Lambda(\mathbf{G}^t)$	Ensure an efficient decoding for decryption
LWE error rate α such that : $1/\alpha = 4 \cdot O(nk) \cdot \omega(\sqrt{\log n})$	Control the magnitude of error in ciphertext

Fig. 11. A summary of notations used in PKE construction and their functionalities.

– $\mathsf{TDP.Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m} \in \{0,1\}^{nk})$: 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 $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2) \in D_{\mathbb{Z}^{\bar{m},\alpha q}} \times D_{\mathbb{Z}^{nk},d} \times D_{\mathbb{Z}^{nk},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m}\cdot(\alpha q)^2)\cdot\omega(\sqrt{\log n})^2$. Let

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q, \tag{8}$$

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 $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. The decryption first recovers $\mathbf{s}$ from the first two blocks via the invert algorithm and then the message $\mathbf{m}$ from third block by decoding (when $\mathbf{s}$ is known):

$$\begin{aligned}(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)^t =& \mathbf{s}^t[\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2] \\ &+ (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \mod q.\end{aligned}$$

1. If c or $\mathbf{b}$ does not parse, or $\mathbf{H}_\mu = \mathbf{0}$, output $\perp$. Otherwise parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)^t$.
2. **Recover s.** 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 $\mathbf{s}$ and $(\mathbf{e}_0, \mathbf{e}_1)$ such that

$$(\mathbf{b}_0, \mathbf{b}_1)^t = \mathbf{s}^t[\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G}] + (\mathbf{e}_0, \mathbf{e}_1)^t \mod q.$$

   If $\mathsf{Invert}^{\mathcal{O}}$ fails, output $\perp$. Invert $\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2$ again and find the unique solution $\mathbf{u}, \mathbf{e}_2$ to the equation

$$\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2 = \mathbf{u}^t\mathbf{G} + \mathbf{e}_2^t \mod q,$$

3. 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 12).

4. **Recover the plaintext.** Output the following result

$$\mathsf{encode}^{-1}\left(\mathbf{b}_2^t - \mathbf{s}^t\mathbf{A}_2 - \mathbf{e}_2^t\right) \in \mathbb{Z}_2^{nk},$$

if it exists, otherwise output $\perp$.

4.2 Correctness and Security

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

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

Proof. The proof is the same as that of Lemma 10, except the bound for the error vectors. The secret key $\mathbf{R}$ serves as the trapdoor the first two blocks of $\mathbf{A}_\mu$, which ensures the proper recovery of $\mathbf{s}$ in Step 2 as long as the error bound is within the capability of Invert. That is $\|\mathbf{e}^t(\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix})\| \leq q/4$ by Lemma 1. By Corollary 14, we have $s_1(\mathbf{R}) = \omega(\sqrt{\log n}) \cdot O(\sqrt{nk})$. By Lemma 12, 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 $q/4$ since $1/\alpha = 4 \cdot O(nk) \cdot \omega(\sqrt{\log n})$, and $\|\mathbf{e}_2\|_\infty \leq q/4$ for the same reason, which ensures the correct recovery of $\mathbf{s}$, $\mathbf{u}$ and $\mathbf{m}$. □

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

Proof. We provide a detailed CCA-1 proof for our UE scheme in Theorem 2. Note that, if the adversary is disallowed to query the token generation and update algorithm, the CCA-1 game for UE is exactly the standard CCA-1 game for the underlying PKE. Therefore, CCA-1 security of TDP follows from Theorem 2. □

5 A CCA-1 Secure Updatable Encryption Scheme

Based on our PKE 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

Our UE scheme uses the same encryption and decryption algorithm in TDP, i.e., the ciphertext of a plaintext $\mathbf{m}$ is of the form $(\hat{\mathsf{ct}}, \mathsf{ct}) = (\mathbf{H}_\mu, \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m})^t)$. To update a ciphertext, at a high level, the update algorithm first generates a key-switching matrix $\mathbf{M}$ with the last row block matrix $[\mathbf{0} \quad \mathbf{0} \quad \mathbf{I}]$, such that $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ for the aimed $\mathbf{A}'_\mu$ in the new ciphertext. This step is

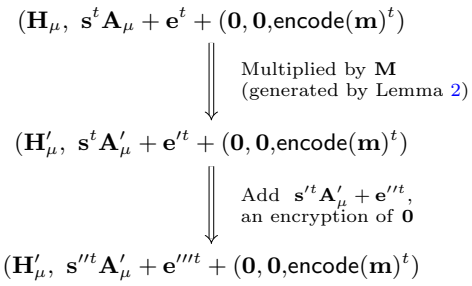

Fig. 12. An overview of ciphertext update in our UE construction. The first step mainly updates $\mathbf{A}_\mu$ to $\mathbf{A}'_\mu$, and the second step refreshes the randomness $\mathbf{s}$.

feasible since the secret key is the trapdoor for the first two blocks of $\mathbf{A}_\mu$, ensuring an efficient preimage sampling algorithm (Lemma 2). To increase the randomness of $\mathbf{s}$, then we add a fresh encryption of message $\mathbf{0}$ to the ciphertext. Fig. 12 shows an overview of the ciphertext update.

We use the same parameters as in Sect. 4.1 except the followings. We also suggest readers on first reading to neglect the parameter setting for error items which are used to control the updated error bound.

- $1/\alpha = 4l\cdot\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}$ 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 $\mathbf{A}'_\mu = [\mathbf{A}'_0 \mid \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G} \mid \mathbf{A}'_2]$. 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 $\mathsf{Sample}^{\mathcal{O}}(\mathbf{R}_1, [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}], \mathbf{H}_\mu, \mathbf{A}'_0, \tau)$ (Lemma 2 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

$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix}\mathbf{X}_{00}\\ \mathbf{X}_{10}\end{bmatrix} = \mathbf{A}'_0. \tag{9}$$

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

$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix}\mathbf{X}_{01}\\ \mathbf{X}_{11}\end{bmatrix} = \mathbf{A}'_1 + \mathbf{H}'_\mu\mathbf{G}. \tag{10}$$

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

$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu\mathbf{G}] \begin{bmatrix}\mathbf{X}_{02}\\ \mathbf{X}_{12}\end{bmatrix} = \mathbf{A}'_2 - \mathbf{A}_2. \tag{11}$$

   Let $\mathbf{M}$ be the key-switching matrix as follows:

$$\mathbf{M} = \begin{bmatrix}\mathbf{X}_{00} & \mathbf{X}_{01} & \mathbf{X}_{02}\\ \mathbf{X}_{10} & \mathbf{X}_{11} & \mathbf{X}_{12}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}. \tag{12}$$

   Note that $\mathbf{A}_\mu = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_\mu\mathbf{G} \mid \mathbf{A}_2]$. Then we have $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ from Equations (9) to (11).
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 the update token $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}'_\mu)$.

– $\mathsf{TDUE.Update}(\Delta, c = (\mathbf{H}_\mu, \mathbf{b}))$: 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 $c' = (\mathbf{H}'_\mu, \mathbf{b}')$.

No-directional Key Updates. TDUE has no-directional key updates since one can only learn from the update token about the value of $\mathbf{A}'_\mu$ (or $\mathbf{A}_\mu$) through $\mathbf{A}_\mu\mathbf{M} = \mathbf{A}'_\mu$ even if sk (or sk', resp.) is corrupted, whereas $\mathbf{A}'_\mu$ and $\mathbf{A}_\mu$ are random due to the leftover hash lemma and the distribution of secret key. Therefore, the adversary cannot infer any information about the secret key from the update tokens.

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:

$$\begin{aligned}(\mathbf{b}')^t &= \mathbf{b}^t \cdot \mathbf{M} + \mathbf{b}_0^t \\ &= \left[\mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t\right]\mathbf{M} + (\mathbf{s}')^t\mathbf{A}'_\mu + (\mathbf{e}')^t \\ &= (\mathbf{s}+\mathbf{s}')^t\mathbf{A}'_\mu + \left(\mathbf{e}^t\mathbf{M} + (\mathbf{e}')^t\right) + (\mathbf{0},\, \mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod q. \end{aligned} \tag{13}$$

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 cause the failure in the invert algorithm $\mathsf{Invert}^{\mathcal{O}}$ and further influence the correctness of the decryption algorithm. In the following, we show that the decryption algorithm can tolerate the accumulated errors in the updated ciphertexts by choosing an appropriate value for the 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 Update), we choose to prove that the decryption algorithm can output a correct plaintext after performing l updates from epoch 0, where l is the maximal update number.

Let $(\mathsf{pk}_\mathrm{e}, \mathsf{sk}_\mathrm{e} = \mathbf{R}_\mathrm{e})_{0\le \mathrm{e}\le l} \leftarrow \mathsf{KG}(1^n)$ 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, which is updated from $c_0 = \mathsf{Enc}(\mathbf{m}) = (\mathbf{H}_{\mu,0}, \mathbf{s}_0^t\mathbf{A}_{\mu,0} + \mathbf{e}_0^t + (\mathbf{0},\, \mathbf{0},\, \mathsf{encode}(\mathbf{m}))^t)$. 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., $\mathbf{b}_{0,i}^t = \mathbf{s}_i^t\mathbf{A}_{\mu,i} + \mathbf{e}_i^t$ in which $\mathbf{A}_{\mu,i} = [\mathbf{A}_{0,i} \mid \mathbf{A}_{1,i} + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}]$. Iteratively by Eq. (13), we know the updated ciphertext of $\mathbf{m}$ in epoch l is $c_l = (\mathbf{H}_{\mu,l}, \mathbf{b}_l)$ where

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

Let $\sum_{i=0}^{l}(\mathbf{e}_i^t\prod_{j=i+1}^{l}\mathbf{M}_j) = (\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)}, \mathbf{e}_2^{(l)})^t = (\mathbf{e}^{(l)})^t$. We provide in Supplementary Material C the upper bound for the error $\mathbf{e}^{(l)}$ that

$$\left\|(\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)}, \mathbf{e}_2^{(l)})^t \cdot \begin{bmatrix}\mathbf{R}_l \\ \mathbf{I} \\ \mathbf{0}\end{bmatrix}\right\|_\infty < q/4 \ \text{ and } \ \left\|\mathbf{e}_2^{(l)}\right\|_\infty < q/4, \tag{14}$$

except with probability $2^{-\Omega(n)}$ via the appropriate parameter selection for the scheme. Let $\mathbf{b}_l^t = (\mathbf{b}_0^{(l)}, \mathbf{b}_1^{(l)}, \mathbf{b}_2^{(l)})^t$. Then by Lemma 1, the call to $\mathsf{Invert}^{\mathcal{O}}$ made by $\mathsf{Dec}(\mathsf{sk}_l, (\mathbf{H}_{\mu,l}, \mathbf{b}_l))$ returns $\mathbf{s}$ $(=\sum_{i=0}^{l}\mathbf{s}_i)$ and $(\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)})$ correctly, for which

$$(\mathbf{b}_0^{(l)}, \mathbf{b}_1^{(l)})^t = \mathbf{s}^t[\mathbf{A}_{0,l} \mid \mathbf{A}_{1,l} + \mathbf{H}_{\mu,l}\mathbf{G}] + (\mathbf{e}_0^{(l)}, \mathbf{e}_1^{(l)})^t \mod q.$$

It follows that

$$(\mathbf{b}_2^{(l)})^t - \mathbf{s}^t \mathbf{A}_{2,l} = (\mathbf{e}_2^{(l)})^t + \mathsf{encode}(\mathbf{m})^t, \tag{15}$$

where $\left\|\mathbf{e}_2^{(l)}\right\| < q/4$ by Inequality (14) 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 $(\mathbf{b}_2^{(l)})^t - \mathbf{s}^t\mathbf{A}_{2,l}$, we can find the unique solution $\mathbf{e}_2^{(l)}$ and $\mathbf{u}$ to Eq. (15). 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 IND-UE-CCA-1 secure under the hardness assumption of LWE.

Theorem 2. *For any* IND-UE-CCA-1 *adversary* $\mathcal{A}$ *against* TDUE*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{LWE}_{n,q,\alpha}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1}}_{\mathsf{TDUE},\mathcal{A}}(1^\lambda) \le 2(l+1)^3 \cdot \Big[&(l+2)\cdot \mathsf{negl}(\lambda) \\ &+ (n_{\mathsf{Dec}} + n_{\mathsf{sUpd}})\cdot 2^{-\Omega(n)} + \mathsf{Adv}^{\mathsf{LWE}}_{n,q,\alpha}(\mathcal{B})\Big],\end{aligned}$$

where l *is the maximal number of ciphertext updates that the scheme* TDUE *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.*

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

Fig. 13. Steps in the security proof of TDUE. Within an insulated region, the reduction should appropriately respond to all the queries made by the adversary. Outside the region, the reduction can generate epoch keys and tokens freely. ct is the abbreviation of ciphertext.

Proof. In general, we take three steps, see Fig. 13, to bound the advantage of the adversary. In the first step, we build a hybrid game H_i for each epoch i,

following [17,22]. To the left of i, the game H_i returns the real challenge-equal ciphertexts and real generated tokens to respond to $\mathcal{O}_{\mathsf{Chall}}$ and $\mathcal{O}_{\mathsf{sUpd}}$ queries; while, to the right of i, H_i returns random ciphertexts and tokens as responses. To distinguish games H_i and H_{i+1}, we assume the adversary queries a challenge-equal ciphertext in epoch i, otherwise the response of both games will be the same. Therefore, the epoch i must be in an insulated region by Lemma 7. In Step 2, we then set up a modified game of H_i, called $\mathcal{G}_i$ that is the same as H_i except for the two randomly chosen epochs $\mathsf{fwr}, \mathsf{fwl}$ to simulate the insulated region around epoch i: if the adversary queries keys inside the region $[\mathsf{fwr}, \cdots, \mathsf{fwl}]$ or challenge-equal tokens in epochs fwr or $\mathsf{fwl}+1$, $\mathcal{G}_i$ aborts. In the last step, we play three games to bound the advantage of distinguishing games $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$. In Game 1, we simulate keys inside the insulated region, which are unknown to the adversary, and show how to simulate response to queries on challenge-equal and non-challenge ciphertexts with the simulated keys. We then simulate the challenge-equal ciphertext in the second game, which allows for the construction of a reduction that solves the LWE by simulating the second game to the adversary. We provide the proof details in Supplementary Material D. □

5.4 A Packing UE

We now introduce a packing method to further improve the efficiency of c-d UE, which allows us to encrypt multiple messages into one ciphertext and execute ciphertext updates simultaneously.

Let N be a power of 2, $\mathcal{R} = \mathbb{Z}[X]/(X^N+1)$, and $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$ be the residue ring of $\mathcal{R}$ modulo q. Any polynomial $p(X)$ in $\mathcal{R}$ can be represented by $p(X) = \sum_{i=0}^{N-1} p_i X^i$ with degree less than N, which is associated with to its coefficient vector $\{p_0, \ldots, p_{N-1}\} \in \mathbb{Z}^N$. For a distribution $\mathcal{X}$, when we say $p(X) \xleftarrow{\$} \mathcal{X}$, we mean the coefficient of $p(X)$ is chosen from $\mathcal{X}$. We the same notations as in Sect. 5.1

Encoding. Prior to the packing construction, we first present an efficient encoding algorithm that encodes multiple messages $\mathbf{m}_0, \ldots, \mathbf{m}_{N-1} \in \mathbb{Z}_2^{nk}$ as an element in $\mathcal{R}_q$ with coefficients in $\Lambda(\mathbf{G}^t)$ as follows:

$$\mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) = \mathbf{B} \cdot \left(\mathbf{m}_0 + \mathbf{m}_1 x + \cdots + \mathbf{m}_{N-1} x^{N-1}\right),$$

where $\mathbf{B} \in \mathbb{Z}^{nk \times nk}$ is any basis of $\Lambda(\mathbf{G}^t)$. Note that it can be efficiently decoded.

At a high level, multiple message blocks are encrypted in the following form:

$$\begin{aligned}(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2(x))^t = \mathbf{s}^t [\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu \mathbf{G} \mid \mathbf{A}_2(x)] \\ + (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2(x))^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}))^t \mod q. \quad (16)\end{aligned}$$

Compared to TDUE, the major modification in this approach is in the third block that uses polynomial matrices and vectors. The secret key $\mathbf{R}$ is still the trapdoor for $[\mathbf{A}_0 \mid \mathbf{A}_0\mathbf{R} + \mathbf{H}_\mu\mathbf{G}]$. Therefore, the decryption procedure is able to properly recover $\mathbf{s}$ from the first two block in Eq. (16) as TDUE.Dec, and then call $\mathsf{Invert}^{\mathcal{O}}$

over $\mathbf{b}_2(x) - \mathbf{s}^t\mathbf{A}_2(x)$ degree by degree to recover every message. Moreover, the token generation is feasible due to a generalized preimage sampling algorithm in Lemma 11.

Lemma 11. *Given a* $\mathbf{G}$*-trapdoor* $\mathbf{R}$ *for* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *with invertible matrix* $\mathbf{H}$ *and any polynomial vector* $\mathbf{u}(X) \in \mathcal{R}_q^n$*, there is an efficient algorithm called* $\mathsf{GSampleD}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}, \mathbf{H}, \mathbf{u}(X), s)$ *that samples a Gaussian polynomial vector* $\mathbf{p}(X) \in \mathcal{R}_q^m$ *with coefficients from* $D_{\mathbb{Z},s}$ *such that* $\mathbf{A}\cdot\mathbf{p}(X) = \mathbf{u}(X)$*, where* s *is the smallest Gaussian parameter defined in Lemma 2.*

Proof. Calling the oracle $\mathsf{SampleD}^{\mathcal{O}}$ on each coefficient vector of $\mathbf{u}(X)$ returns a vector $\mathbf{p}_i$ such that

$$\mathbf{A}\mathbf{p}_i = \mathbf{u}_i,$$

for $0 \le i \le N$ and $\mathbf{u}(X) = \sum_{i=0}^{N-1} \mathbf{u}_i X^i$. Denote $\mathbf{p}(X) = \sum_{i=0}^{N-1} \mathbf{p}_i X^i$, then we know $\mathbf{A}\mathbf{p}(X) = \mathbf{u}(X)$. □

Packing UE. Our packing UE scheme is described as follows.

- $\mathsf{KG}(1^\lambda)$: choose $\mathbf{A}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$, $\mathbf{R}_1$, $\mathbf{R}_2(X) \xleftarrow{\$} \mathcal{D}$ and let $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1 \mid \mathbf{A}_2] = [\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 \mid -\mathbf{A}_0\mathbf{R}_2(X)] \in \mathcal{R}_q^{n\times m}$ where $m = \bar{m} + 2nk$. The public key is $\mathsf{pk} = \mathbf{A}$ and the secret key is $\mathsf{sk} = \mathbf{R}_1$.

- $\mathsf{Enc}(\mathsf{pk} = \mathbf{A}, \mathbf{m}_0, \ldots, \mathbf{m}_{N-1} \in \{0,1\}^{nk})$: 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 $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2(X)) \in D_{\mathbb{Z}^{\bar{m}},\alpha q} \times D_{\mathbb{Z}^{nk},d} \times D_{\mathbb{Z}^{nk},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m}\cdot(\alpha q)^2)\cdot\omega(\sqrt{\log n})^2$. Let

$$\mathbf{b}^t = \mathbf{s}^t\mathbf{A}_\mu + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1})^t \mod q, \tag{17}$$

  where $\mathsf{encode}(\mathbf{m}_0, \ldots, \mathbf{m}_{N-1}) = \mathbf{B}\cdot(\mathbf{m}_0 + \mathbf{m}_1 X + \cdots + \mathbf{m}_{N-1}X^{N-1})$.

- $\mathsf{Dec}(\mathsf{sk} = \mathbf{R}_1, c = (\mathbf{H}_\mu, \mathbf{b}))$: Recover $\mathbf{s}$ as the steps 1 to 3 in the decryption algorithm of TDP. Parse $\mathbf{b}^t = (\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2(X))^t$, invert $\mathbf{b}_2(X)^t - \mathbf{s}^t\mathbf{A}_2$ degree by degree, and find the unique solution $\mathbf{u}_i, \mathbf{e}_{2,i}$ to the equation

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

  by Lemma 1 if they exist, where $\mathbf{b}_2(X) = \sum \mathbf{b}_{2,i}X^i$ and $\mathbf{A}_2 = \sum \mathbf{A}_{2,i}X^i$. Output the following result as $\mathbf{m}_i$ if it exists,

$$\mathsf{encode}^{-1}\left((\mathbf{u}_i^t\mathbf{G})^t\right) \in \mathbb{Z}_2^{nk},$$

  for $0 \le i \le N-1$, otherwise output $\perp$.

- $\mathsf{TokenGen}(\mathsf{pk}, \mathsf{sk}, \mathsf{pk}', \mathbf{H}_\mu)$: Generate the first two row block matrices $\mathbf{M}_{00}$, $\mathbf{M}_{01}$, $\mathbf{M}_{10}$, $\mathbf{M}_{11}$ of $\mathbf{M}$ as in step 2 and 3 of $\mathsf{TDUE.TokenGen}$, and call the

algorithm $\mathsf{GSampleD}^{\mathcal{O}}$ in Lemma 11 to find $\mathbf{M}_{02}(X) \in \mathcal{R}_q^{\bar{n} \times nk}, \mathbf{M}_{12}(X) \in \mathcal{R}_q^{nk \times nk}$ such that

$$[\mathbf{A}_0 \mid -\mathbf{A}_0\mathbf{R}_1 + \mathbf{H}_\mu \mathbf{G}] \begin{bmatrix} \mathbf{M}_{02}(X) \\ \mathbf{M}_{12}(X) \end{bmatrix} = \mathbf{A}_2' - \mathbf{A}_2.$$

Generate $\mathbf{b}_0$ a fresh encryption of message 0. Output the update token $\Delta = (\mathbf{M}, \mathbf{b}_0, \mathbf{H}_\mu')$.

– $\mathsf{TDUE.Update}(\Delta, c = (\mathbf{H}_\mu, \mathbf{b}))$: 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 $c' = (\mathbf{H}_\mu', \mathbf{b}')$.

Remark. The correctness and IND-UE-CCA-1 security of packing UE is analogous to those of TDUE (as shown in Lemma 10 and Theorem 2). We omit the details. For a message of bit length Nnk, packing UE, compared to TDUE, reduces the number of ciphertexts by a factor of N, and only one ciphertext header is required to be downloaded in token generation procedure.

6 Conclusion and Future Work

In this paper, we propose a stronger confidentiality notion than prior work for ciphertext-dependent updatable encryption, which captures adaptive security and is applied to both types of UE schemes: deterministic and randomized updates. We also provide a new public key encryption scheme, based on which we construct our updatable encryption scheme. Moreover, we propose a cost-effective packing UE scheme that is able to execute ciphertext updates simultaneously.

Future Work. The first FHE scheme introduced by Gentry [16] and all its subsequent work require a "circular security" assumption, namely that it is safe to encrypt old secret key with new key. Such an idea has inspired the UE construction with backward directional key updates. In turn, we suggest an open problem that if no-directional updatable encryption, which is able to update ciphertext without revealing old and new keys, can be used to construct FHE that does not rely on the assumption.

References

1. Agrawal, S., Gentry, C., Halevi, S., Sahai, A.: Discrete gaussian leftover hash lemma over infinite domains. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 97–116. Springer (2013). https://doi.org/10.1007/978-3-642-42033-7_6

2. Ajtai, M.: Generating hard instances of lattice problems. In: STOC. pp. 99–108 (1996). https://doi.org/10.1145/237814.237838
3. Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 650–679. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_23
4. 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
5. 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
6. 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
7. 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
8. 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
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITIC 2012. pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262, https://doi.org/10.1145/2090236.2090262
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) FOCS, 2011. pp. 97–106. IEEE Computer Society (2011). https://doi.org/10.1109/FOCS.2011.12
11. 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
12. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 335–352. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2_19
13. 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
14. 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
15. Galteland, Y.J., Pan, J.: Backward-leak uni-directional updatable encryption from public key encryption. IACR Cryptol. ePrint Arch. p. 324 (2022), https://eprint.iacr.org/2022/324
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC 2009. pp. 169–178. ACM (2009). https://doi.org/10.1145/1536414.1536440

17. 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
18. 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
19. 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
20. 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
21. 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
22. 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
23. 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
24. Sakurai, K., Nishide, T., Syalim, A.: Improved proxy re-encryption scheme for symmetric key cryptography. In: IWBIS, 2017. pp. 105–111. IEEE (2017). https://doi.org/10.1109/IWBIS.2017.8275110
25. 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

Supplementary Material

A Proof of Theorem 1

We provide reductions for both $\mathsf{sConfidentiality}$ and $\mathsf{Confidentiality}$ games. For any $\mathsf{sConfidentiality}$ adversary $\mathcal{A}$, we construct a reduction $\mathcal{B}$ that runs the $\mathsf{Confidentiality}$ game and simulates all responses to the queries of $\mathcal{A}$ as the first line shown in Fig. 5. 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 oracle $\mathcal{O}_{\mathsf{sUpd}}$ on some input, the reduction $\mathcal{B}$ submits the same input to $\mathcal{O}_{\mathsf{TokeGen}}$. 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, $\mathcal{B}$ forwards $\mathcal{A}$'s guess to its challenger.

We check the trivial win conditions of $\mathcal{A}$ and $\mathcal{B}$ one by one. Suppose $\mathcal{A}$ does not query an update of challenge-equal ciphertext in an epoch in which it knows the key (via $\mathcal{O}_{\mathsf{sUpd}}$). From the proof of Lemma 3, then we know $\mathsf{TC}_{\mathsf{chall}}[0] = \mathsf{TC}^{*}_{\mathsf{chall}}[0]$, and therefore the epochs in which $\mathcal{A}$ knows a challenge-equal ciphertext are the same as $\mathcal{B}$. Therefore, $\mathcal{B}$ will not query the token related to challenge-equal ciphertexts in epochs in which it knows the epoch key (since $\mathcal{A}$ and $\mathcal{B}$ have the same known keys). There is no additional restrictions for randomized UE. For deterministic UE, the adversary should not learn the update of challenge input ciphertext in advance before the challenge. Note that the look-up table $\mathsf{TC}_{\mathsf{non}}$ for $\mathcal{A}$ and $\mathcal{B}$ is the same. If $\mathcal{A}$ does not query $\mathcal{O}_{\mathsf{sUpd}}(\bar{\mathsf{e}}, (\hat{\mathsf{ct}}, \mathsf{ct}))$ before querying the challenge oracle, the same applies to $\mathcal{B}$ (recall the ciphertexts acquired before the challenge are all recorded in $\mathsf{TC}_{\mathsf{non}}$). Therefore, $\perp$ will also not be returned from the challenger of $\mathcal{B}$ when $\mathcal{A}$ does not trigger trivial win conditions. Thus $\mathcal{B}$ has at least the same advantage as $\mathcal{A}$, i.e., the Inequality (4).

Similarly, for any $\mathsf{Confidentiality}$ adversary $\mathcal{B}$, we construct a reduction $\mathcal{A}$ that runs the $\mathsf{sConfidentiality}$ game and simulates all the responses to the queries of the given $\mathcal{B}$ as shown in the second line of Fig. 5. 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. Suppose $\mathcal{B}$ does not query an update of challenge-equal ciphertext in an epoch in which it knows the epoch key. By proof of Lemma 3 again, we know $\mathcal{A}$ does not query $\mathsf{O}_{\mathsf{sUpd}}$ in an epoch in which it knows the epoch key. In addition, the analysis for deterministic is almost the same as above. We omit the details. 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 (4), we conclude the advantage of $\mathcal{A}$ is equal to that of $\mathcal{B}$.

B Gaussians and Lattices

Lemma 12 ([5], 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*

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

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

Lemma 13 ([21], 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 $2\exp(\delta)\exp\left(-\pi t^2\right)$.*

Lemma 14 ([12], Fact 6). *For any $m, n, s > 0$, let $\mathbf{R} \in D_{\mathbb{Z},s}^{n \times m}$, we have $s_1(\mathbf{R}) \leq s \cdot O(\sqrt{n} + \sqrt{m})$, except with probability $2^{-\Omega(n+m)}$.*

The following lemma bounds the maximal singular value of the product and addition of two matrices, which follows directly from the definition.

Lemma 15. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, then $s_1(\mathbf{AB}) \leq s_1(\mathbf{A})s_1(\mathbf{B})$ and $s_1(\mathbf{A}+\mathbf{B}) \leq s_1(\mathbf{A}) + s_1(\mathbf{B})$.*

Lemma 16 (Leftover Hash Lemma). *Let $\mathcal{P}$ be a distribution over $\mathbb{Z}_q^n$ with min-entropy k. For any $\epsilon > 0$ and $l \leq (k - 2\log(1/\epsilon) - O(1))/\log(q)$, the joint distribution of $(\boldsymbol{C}, \boldsymbol{Cs})$ 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 17 ([4], Fact 2.2). *Let $X_1, \ldots, X_n$ 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 $s\|u\|$.*

C Proof of Inequality (14)

We start from the error $\mathbf{e}_0$ generated in the fresh encryption in epoch 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 fewer times than $\mathbf{e}_0$ by the transition matrix $\{\mathbf{M}_j\}$.

Step 1. Let $\mathbf{e}_0^t = (\mathbf{e}_{0,0}, \mathbf{e}_{1,0}, \mathbf{e}_{2,0})^t$ and $\mathbf{M}^{(s)}$ be the s products of $\{\mathbf{M}_j\}_{j=1}^s$, denoted as follows:

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

for $s \in \{1, \ldots, l\}$.

We first estimate the bound on the maximal singular values of $\mathbf{M}_j$ and $\mathbf{M}^{(s)}$. For any $j \in \{1, \ldots, l\}$, we have $s_1(\mathrm{X}_{kt,j}) \leq \tau \cdot O(\sqrt{nk})$ for $kt \in \{00, 10\}$, and $s_1(\mathrm{X}_{kt,j}) \leq \tau\sqrt{\bar{m}/2} \cdot O(\sqrt{nk})$ for $kt \in \{01, 11, 02, 12\}$ by Lemma 14. Let $\mu = \sqrt{\bar{m}/2}$. For $s = 2$, since $\mathbf{X}_{00}^{(2)} = \mathbf{X}_{00,1} \cdot \mathbf{X}_{00,2} + \mathbf{X}_{01,1} \cdot \mathbf{X}_{10,2}$, we have

$$\begin{aligned} s_1(\mathbf{X}_{00}^{(2)}) &\leq s_1(\mathbf{X}_{00,1}) \cdot s_1(\mathbf{X}_{00,2}) + s_1(\mathbf{X}_{01,1}) \cdot s_1(\mathbf{X}_{10,2}) \\ &\leq \tau^2(1+\mu) \cdot O(\sqrt{nk})^2, \end{aligned}$$

by Lemma 15. Similarly, we give the bound on the 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):

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

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

$$\begin{aligned} &\begin{bmatrix} s_1(\mathbf{X}_{00}^{(l)}) & s_1(\mathbf{X}_{01}^{(l)}) & s_1(\mathbf{X}_{02}^{(l)}) \\ s_1(\mathbf{X}_{10}^{(l)}) & s_1(\mathbf{X}_{11}^{(l)}) & s_1(\mathbf{X}_{12}^{(l)}) \end{bmatrix} \\ &\quad \leq \begin{bmatrix} \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \\ \tau^l(1+\mu)^{l-1} & \tau^l\mu(1+\mu)^{l-1} & \tau^l\mu\sum_{k=0}^{l-1}(1+\mu)^k \end{bmatrix} \cdot O(\sqrt{nk})^l. \end{aligned} \tag{18}$$

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

$$\begin{aligned} \mathbf{e}_0^t \cdot \mathbf{M}^{(l)} \cdot \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} &= (\mathbf{e}_{0,0}, \mathbf{e}_{1,0}, \mathbf{e}_{2,0})^t \begin{bmatrix} \mathbf{X}_{00}^{(l)} & \mathbf{X}_{01}^{(l)} & \mathbf{X}_{02}^{(l)} \\ \mathbf{X}_{10}^{(l)} & \mathbf{X}_{11}^{(l)} & \mathbf{X}_{12}^{(l)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{1,l} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \\ &= (\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l} + \mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}. \end{aligned} \tag{19}$$

By Lemma 12, we have $\|\mathbf{e}_{0,0}\| < \alpha q\sqrt{\bar{m}}$ and $\|\mathbf{e}_i\| < \alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$ for $\mathbf{e}_i \in \{\mathbf{e}_{1,0}, \mathbf{e}_{2,0}\}$, except with probability $2^{-\Omega(n)}$. Combining this conclusion with Eq. (18), we obtain

$$\begin{cases} \left\|\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)}\mathbf{R}_{1,l}\right\| \leq \|\mathbf{e}_{0,0}\| \cdot s_1(\mathbf{X}_{00}^{(l)}) \cdot s_1(\mathbf{R}_{1,l}) < \alpha q\sqrt{\bar{m}} \cdot \tau^l(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)}\mathbf{R}_{1,l}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l \cdot \sigma, \\ \left\|\mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)}\right\| < \alpha q\sqrt{\bar{m}} \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l, \\ \left\|\mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}\right\| < (\alpha q\sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})) \cdot \tau^l\mu(1+\mu)^{l-1}O(\sqrt{nk})^l, \end{cases} \tag{20}$$

where σ is the upper bound for $s_1(\mathbf{R}_{1,l})$ by Corollary 14, i.e., $\sigma := \omega(\sqrt{\log n}) \cdot O(\sqrt{nk})$.

Substituting σ, τ and μ in Inequality (20), we get the upper bound for the norm of the updated e_0^t in Eq.(19) by the triangle inequality as follows:

$$\left\|(\mathbf{e}_{0,0}^t\mathbf{X}_{00}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{10}^{(l)})\mathbf{R}_{1,l} + \mathbf{e}_{0,0}^t\mathbf{X}_{01}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{11}^{(l)}\right\| < \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}, \tag{21}$$

and similarly, the bound for the norm of the last nk coordinates of $\mathbf{e}_0^t \cdot \mathbf{M}^{(l)}$ is

$$\left\|\mathbf{e}_{0,0}^t\mathbf{X}_{02}^{(l)} + \mathbf{e}_{1,0}^t\mathbf{X}_{12}^{(l)} + \mathbf{e}_{2,0}^t\right\| < \alpha q\omega(\sqrt{\log n})^{2l+1}O(\sqrt{nk})^{3l+2}. \tag{22}$$

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

$$\left\|(\mathbf{e}_0^t \cdot \prod_{j=1}^{l}\mathbf{M}_j)\begin{bmatrix}\mathbf{R}_{1,l} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\right\|_\infty \leq \alpha q\omega(\sqrt{\log n})^{2l+2}O(\sqrt{nk})^{3l+3}, \tag{23}$$

by combining the Inequality (21) and Inequality (22).

Step 2. Errors generated from epoch 1 to $l-1$ have the same distribution with $\mathbf{e}_0$, but are multiplied fewer time by the key-switching 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

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

The bound on updated $\mathbf{e}_0$ in Inequality (23) also holds for $\mathbf{e}_l$.

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

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

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 (14).

D Proof of Theorem 2

Denote the challenge input as $(\bar{\mathbf{m}}, \bar{c})$. We proceed via the following three steps.

Step 1. Consider a sequence of hybrid experiments $H_0^{\mathrm{b}}, \ldots, H_{l+1}^{\mathrm{b}}$ for $\mathrm{b} \in \{0,1\}$. The game H_i^{b} is the same as the IND-UE-CCA-1 game except when the adversary queries a challenge-equal ciphertext via $\mathcal{O}_{\mathsf{Chall}}$ or $\mathcal{O}_{\mathsf{sUpd}}$ in epoch j:

– if $j < i$, return an honestly generated challenge-equal ciphertext to $\mathcal{O}_{\mathsf{Chall}}$. That is, return the encryption of $\mathsf{Enc}(\mathsf{pk}_j, \bar{\mathbf{m}})$ if b = 0, and the update ciphertext of $\mathsf{Upd}(\Delta_j, \bar{\mathrm{c}})$ if b = 1. For the query to $\mathcal{O}_{\mathsf{sUpd}}$, return the real generated token and the truly update of challenge-equal ciphertexts.
– if $j \geq i$, return a random ciphertext to $\mathcal{O}_{\mathsf{Chall}}$. For the query to $\mathcal{O}_{\mathsf{sUpd}}$, return a randomly generated token and the update of ciphertext by $\mathsf{TDUE.Update}$.

We see that H^{b}_{l+1} is the same as $\mathsf{Exp}^{\mathsf{IND\text{-}UE\text{-}CCA\text{-}1\text{-}b}}_{\mathsf{UE},\mathcal{A}}$, and $H^0_0 = H^1_0$ since all challenge responses are the same. Then We have

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

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

Step 2. Since the responses in all epochs except i will be the same in both games H^{b}_{i+1} and H^{b}_i for $\mathrm{b} \in \{0,1\}$, we assume the adversary who tries to distinguish the two games asks for a challenge-equal ciphertext in epoch i. Therefore, there exists an insulated region $[\mathsf{fwl}, \mathsf{fwr}]$ around epoch i such that no epoch keys in $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ and no tokens related to challenge-equal ciphertexts in epochs fwl and $\mathsf{fwr}+1$ are corrupted by Lemma 7.

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

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

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

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

Game 0:

For a random number $d \xleftarrow{\$} \{0,1\}$, if $d = 0$ the game plays $\mathcal{G}^b_i$ to $\mathcal{A}_i$; otherwise it plays $\mathcal{G}^b_{i+1}$ to $\mathcal{A}_i$. Denote E_j be the event that the adversary succeeds in guessing d in the **Game** j for $j \in \{0,1,2,3\}$. Then we have

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

Game 1:

We consider a modified game same as **Game 0**, except the way that the epoch keys and tokens are generated in epochs from i to fwr. The overall idea to change the 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 to generate token in epochs from $i+1$ to fwr. We will show how to simulate tokens in epoch from $i+1$ to fwr.

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 $\mathsf{pk}_i, \ldots, \mathsf{pk}_{\mathsf{fwr}}$.
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. (28) and further facilitate the simulation of challenge-equal ciphertext in epoch i in **Game 2**.

   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

$$\mathbf{A}_{0,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \cdot \begin{bmatrix}\mathbf{X}_{00,i+1}\\ \mathbf{X}_{10,i+1}\end{bmatrix}, \tag{24}$$

   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

$$\mathbf{A}_{1,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix}\mathbf{X}_{00,i+1}\\ \mathbf{X}_{10,i+1}\end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{25}$$

$$\mathbf{A}_{2,i+1} = [\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i}] \begin{bmatrix}\mathbf{X}_{02,i+1}\\ \mathbf{X}_{12,i+1}\end{bmatrix} - \mathbf{A}_{2,i}, \tag{26}$$

   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 secret key be $\mathsf{sk}_{i+1} = \mathbf{R}_{1,i+1}$, where $\mathbf{H}^*_{\mu,i+1}$ is generated in process 1.

Remark 2. By Lemma 16, 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 3. 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 17, i.e., the same distribution as the second-column block matrix of the token in **Game 0**, which is the same distribution as in the real game.

Remark 4. We do not require the last block matrix $\mathbf{A}_{2,i+1}$ 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. However, we will show this does not affect the responses to the queries.

Public key in epochs from $i+2$ to fwr. For any epoch j in $\{i+2,\ldots,\mathsf{fwr}\}$, we iteratively generate the public key pk_j and secret key sk_j in a way as generating 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 are summarised in Fig. 14. We should respond to the update of and token w.r.t challenge-equal ciphertexts in epochs from i to fwr, but do not need to answer decryption queries on challenge-equal ciphertexts.

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	–	–
$\mathsf{TokenGen}$	Eq. (31)	
Update	$\mathbf{b}^t_{i+1}=\mathbf{b}^t_i\Delta_{i+1}+\mathbf{b}^t_0$	

Fig. 14. 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 $\mathsf{TokenGen}$ and Update simultaneously. Public keys and ciphertexts (only $\mathbf{A}_\mu$ is shown) are represented by the block matrices.

Ciphertext. For challenge-equal ciphertexts in epoch i, notice that a random invertible matrix should be generated when updating ciphertexts and encrypting messages; here we use the special pre-generated invertible matrix $\mathbf{H}^*_{\mu,i}$ in both of the two algorithms, which is randomly generated in process 1 and is unknown to the adversary conditioned on the public keys we sampled in process 3. Then the challenge ciphertext in epoch i generated 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)

$$\begin{aligned}\mathbf{b}_i^t &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}^*_{\mu,i}\mathbf{G} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_\mathrm{b}))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{1,i} \mid -\mathbf{A}_{0,i}\mathbf{R}_{2,i}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_\mathrm{b}))^t, \end{aligned} \tag{27}$$

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

$$\begin{aligned}\mathbf{b}_j^t &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid (-\mathbf{A}_{0,j}\mathbf{R}_{1,j} - \mathbf{H}^*_{\mu,j}\mathbf{G}) + \mathbf{H}^*_{\mu,j}\mathbf{G} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_\mathrm{b}))^t \\ &= \mathbf{s}^t[\mathbf{A}_{0,j} \mid -\mathbf{A}_{0,j}\mathbf{R}_{1,j} \mid \mathbf{A}_{2,j}] + \mathbf{e}^t + (\mathbf{0}, \mathbf{0}, \mathsf{encode}(\mathbf{m}_\mathrm{b}))^t, \end{aligned} \tag{28}$$

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

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

$$\begin{bmatrix}\mathbf{X}_{01,i+1} \\ \mathbf{X}_{11,i+1}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_{00,i+1} \\ \mathbf{X}_{10,i+1}\end{bmatrix} \cdot \mathbf{R}_{1,i+1}, \tag{29}$$

and

$$\mathbf{M}_{i+1} = \begin{bmatrix}\mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}, \tag{30}$$

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 Equations (24) to (26), we know that $\mathbf{M}_{i+1}$ is the key-switching matrix from epoch i to $i+1$: $\mathbf{A}_{\mu,i}\mathbf{M}_{i+1} = \mathbf{A}_{\mu,i+1}$, and moreover has the distribution $\mathsf{negl}(n)$-far from the distribution of the token in **Game 0** by Remark 3. Then we have

$$\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1}), \tag{31}$$

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 Fig. 15. We should respond to the queries on the encryption, update, and decryption in all epochs. Since keys outside the insulated region are truly generated in process 2, we focus on the simulation inside the region.

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

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	$\mathbf{b}^t_{i+1}=\mathbf{b}^t_i\Delta_{i+1}+\mathbf{b}^t_0$	

Fig. 15. Simulation to the queries on non-challenge ciphertexts. The algorithm SimDec is defined below, which has the same decryption ability as $\mathsf{TDUE.Dec}$ if $\mathbf{H}_{\mu,j}\neq\mathbf{H}^*_{\mu,j}$ for $j\in\{i+1,\mathsf{fwr}\}$. When the adversary queries the oracle $\mathcal{O}_{\mathsf{sUpd}}$, the simulation returns the output of $\mathsf{TokenGen}$ and Update simultaneously.

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

$$\begin{aligned}\mathbf{b}^t_i=\mathbf{s}^t[\mathbf{A}_{0,i}\mid(-\mathbf{A}_{0,i}\mathbf{R}_{1,i}-\mathbf{H}^*_{\mu,i}\mathbf{G})+\mathbf{H}_{\mu,i}\mathbf{G}\mid-\mathbf{A}_{0,i}\mathbf{R}_{2,i}]\\ +\mathbf{e}^t+(\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}))^t,\end{aligned}\tag{32}$$

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

$$\begin{aligned}\mathbf{b}^t_j=\mathbf{s}^t[\mathbf{A}_{0,j}\mid(-\mathbf{A}_{0,j}\mathbf{R}_{1,j}-\mathbf{H}^*_{\mu,j}\mathbf{G})+\mathbf{H}_{\mu,j}\mathbf{G}\mid\mathbf{A}_{2,j}]\\ +\mathbf{e}^t+(\mathbf{0},\mathbf{0},\mathsf{encode}(\mathbf{m}))^t.\end{aligned}\tag{33}$$

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 for non-challenge ciphertexts in epoch i, and the simulated decryption algorithm can also be applied to non-challenge ciphertexts 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 $\mathbf{b}^t=(\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2)^t$.
  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 $\mathbf{u}$ properly if they exist, and SimDec has the same decryption ability as $\mathsf{TDUE.Dec}$.

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

$$\begin{aligned}
&[\mathbf{A}_{0,i} \mid (-\mathbf{A}_{0,i}\mathbf{R}_{1,i} - \mathbf{H}^*_{\mu,i}\mathbf{G}) + \mathbf{H}_{\mu,i}\mathbf{G} \mid \mathbf{A}_{2,i}] \begin{bmatrix} \mathbf{X}_{00,i+1} & \mathbf{X}_{01,i+1} & \mathbf{X}_{02,i+1} \\ \mathbf{X}_{10,i+1} & \mathbf{X}_{11,i+1} & \mathbf{X}_{12,i+1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \\
&= [\mathbf{A}_{0,i+1} \mid (-\mathbf{A}_{0,i+1}\mathbf{R}_{1,i+1} - \mathbf{H}^*_{\mu,i+1}\mathbf{G}) + \mathbf{H}_{\mu,i+1}G \mid \mathbf{A}_{2,i+1}]. \qquad (34)
\end{aligned}$$

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. (34), and $\Delta_{i+1} = (\mathbf{M}_{i+1}, \mathbf{b}_{0,i+1}, \mathbf{H}^*_{\mu,i+1})$. Based on Equations (32) to (34), we know that Δ_{i+1} is a valid non-challenge token in epoch $i+1$. Since this process only requires the property that $\hat{\mathbf{R}}_i$ is a trapdoor, it can be applied to any epoch $j \in \{i+1 \ldots \mathsf{fwr}\}$ by choosing an matrix $\hat{\mathbf{R}}_j$ 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 simulations the same, especially the challenge-equal token. In epoch i, we modify 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 $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\widehat{\mathbf{e}}_0 \leftarrow D_{\mathbb{Z}^{\bar{m}},\alpha q}$ and $\widehat{\mathbf{e}}_1$, $\widehat{\mathbf{e}}_2 \leftarrow D_{\mathbb{Z}^{nk},\alpha q\sqrt{m}\cdot\omega(\sqrt{\log n})}$. 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 the Corollary 3.10 in [23], the distribution of $-\widehat{\mathbf{e}}_0^t \cdot \mathbf{R}_{1,i} + \widehat{\mathbf{e}}_1^t$ is $\mathsf{negl}(n)$-far from $D_{\mathbb{Z}^{nk},d}$ where $d^2 = (\|\mathbf{e}_0\|^2 + \bar{m}\cdot\alpha q)\cdot\omega(\sqrt{\log n})^2$, which 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. (31), which is therefore a valid token, and similarly change challenge-equal ciphertexts in epochs from $i+2$ to fwr. 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 to the adversary $\mathcal{A}_i$: either uniformly random or $\mathbf{s}^t\mathbf{A}_{0,i} + \widehat{\mathbf{e}}_0^t$, which should be 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 $\mathsf{negl}(\lambda)$-far 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.