Fast and Secure Updatable Encryption[†]

Colin Boyd[1] Gareth T. Davies[2] Kristian Gjøsteen[1] and Yao Jiang[1]

[1]Norwegian University of Science and Technology, NTNU, Norway.
{colin.boyd,kristian.gjosteen,yao.jiang}@ntnu.no
[2]Bergische Universität Wuppertal, Wuppertal, Germany.
davies@uni-wuppertal.de

February 25, 2022

Abstract

Updatable encryption allows a client to outsource ciphertexts to some untrusted server and periodically rotate the encryption key. The server can update ciphertexts from an old key to a new key with the help of an update token, received from the client, which should not reveal anything about plaintexts to an adversary.

We provide a new and highly efficient suite of updatable encryption schemes that we collectively call SHINE. In the variant designed for short messages, ciphertext generation consists of applying one permutation and one exponentiation (per message block), while updating ciphertexts requires just one exponentiation. Variants for longer messages provide much stronger security guarantees than prior work that has comparable efficiency. We present a new confidentiality notion for updatable encryption schemes that implies prior notions. We prove that SHINE is secure under our new confidentiality definition while also providing ciphertext integrity.

† © IACR 2020: An extended abstract of this paper appears in the proceedings of IACR CRYPTO 2020, with DOI: 10.1007/978-3-030-56784-2_16. This is the full version.
An earlier version of this document was titled 'RISE and SHINE: Fast and Secure Updatable Encryption'. This version subsumes that work and includes new results; we summarize the changes in Section 1.4.

Contents

1 Introduction

The past decades have demonstrated clearly that key compromise is a real threat for deployed systems. The standard technique for mitigating key compromise is to regularly *rotate* the encryption keys – generate new ones and switch the ciphertexts to encryption under the new keys. Key rotation is a well-established technique in applications such as payment cards [Cou18] and cloud storage [KRS+03].

For a local drive or server, key rotation is feasible by decrypting and re-encrypting with a new key, since symmetric encryption operations are fast and parallelizable and bandwidth is often plentiful. When ciphertext storage has been outsourced to some (untrusted) cloud storage provider, bandwidth is often considerably more expensive than computation, and even for small volumes of data it may be prohibitively expensive to download, re-encrypt and upload the entire database even once. This means that key rotation by downloading, decrypting, re-encrypting and reuploading is practically infeasible.

An alternative approach to solving this problem is to use *updatable encryption* (UE), first defined by Boneh et al. [BLMR13] (henceforth BLMR). The user computes a *token* and sends it to the storage server. The token allows the server to update the ciphertexts so that they become encryptions under some new key. Although the token clearly depends on both the old and new encryption keys, knowledge of the token alone should not allow the server to obtain either key. In a typical usage of UE, the cloud storage provider will receive a new token on a periodic basis, and the provider then updates every stored ciphertext. The time period for which a given key is valid for is called an *epoch*.

In the past few years there has been considerable interest in extending the understanding of UE. A series of prominent papers [BLMR13, EPRS17a, LT18a, KLR19a] have provided both new (typically stronger) security definitions and concrete or generic constructions to meet their definitions. (We make a detailed comparison of related work in Section 1.1.1 next.) An important distinction between earlier schemes is whether or not the token (and in particular its size) depends on the ciphertexts to be updated (and in particular the number of ciphertexts). Schemes for which a token is assigned to each ciphertext are *ciphertext-dependent* and were studied by Everspaugh et al. [EPRS17a] (henceforth EPRS). If the token is independent of the ciphertexts to be updated, such as in BLMR [BLMR15], we have a *ciphertext-independent*[1] scheme. A clear and important goal is to limit the bandwidth required and so, in general, one should prefer ciphertext-independent schemes. Thus, as with the most recent work [LT18a, KLR19a], we focus on such schemes in this paper. The ciphertext update procedure, performed by the server, may be *deterministic* or *randomized* – note that in the latter case the server is burdened with producing (good) randomness and using it correctly.

Despite the considerable advances of the past few years, there remain some important open questions regarding basic properties of UE. In terms of security, various features have been added to protect against stronger adversaries. Yet it is not obvious what are the realistic and optimal security goals of UE and whether they have been achieved. In terms of efficiency, we only have a few concrete schemes to compare. As may be expected, schemes with stronger security are generally more expensive but it remains unclear whether this cost is necessary. In this paper we make contributions to both of these fundamental questions by defining **new and stronger security properties** and showing that these can be achieved with **more efficient concrete UE schemes**.

Security. The main security properties that one would expect from updatable encryption are by now well studied; however the breadth of information that is possible to protect in this context is more subtle than at first glance. Consider, for example, a journalist who stores a contact list with a cloud storage provider. At some point, the storage is compromised and an adversary recovers the ciphertexts. At this point, it may be important that the cryptography does not reveal which of the contacts are recent, and which are old. That is, it must be hard to decide if some ciphertext was recently created, or if it has been updated from a ciphertext stored in an earlier epoch.

So how do we define realistic adversaries in this environment? A natural first step for security in updatable encryption is *confidentiality* of ciphertexts – given a single ciphertext, the adversarial server should not be able to determine anything about the underlying plaintext. The security model here must take into account that this adversary could be in possession of a number of prior keys or update tokens, and snapshot access to the storage

[1] Note that Boneh et al. [[BLMR15], § Definition 7.6] use ciphertext-independence to mean that the updated ciphertext should have the same distribution as a fresh ciphertext (i.e. independent of the ciphertext in the previous epoch) – we follow the nomenclature of Lehmann and Tackmann [LT18a].

database in different epochs. The next step is to consider *unlinkability* between different epochs arising from the ciphertext update procedure: given a ciphertext for the current epoch, the adversary should not be able to tell which ciphertext (that existed in the previous epoch) a current ciphertext was updated from. Both of these properties can be naturally extended to chosen-ciphertext (CCA) security via provision of a decryption oracle.

These steps have been taken by prior work, but unfortunately even a combination of these properties is not enough to defend against our motivating example. Previous security definitions cannot guarantee that the adversary is unable to distinguish between a ciphertext new in the current epoch and an updated ciphertext from an earlier epoch. We give a single new security property that captures this requirement and implies the notions given in prior work. Therefore we believe that this definition is *the natural confidentiality property* that is required for updatable encryption.

An additional factor to consider is integrity: the user should be confident that their ciphertexts have not been modified by the adversarial server. While prior work has shown how to define and achieve integrity in the context of updatable encryption, a composition result of the style given by Bellare and Namprempre for symmetric encryption [BN08] – the combination of CPA security and integrity of ciphertexts gives CCA security – has been missing. We close this gap.

Efficiency and Functionality. Although UE is by definition a form of symmetric key cryptography, techniques from asymmetric cryptography appear to be needed to achieve the required functionality in a sensible fashion. All of the previous known schemes with security proofs use exponentiation in both the encryption and update functions, even for those with limited security properties. Since a modern database may contain large numbers of files, efficiency is critical both for clients who will have to encrypt plaintexts initially and for servers who will have to update ciphertexts for all of their users.

To bridge the gap between the academic literature and deployments of encrypted outsourced storage, *it is crucial to design fast schemes*. We present three novel UE schemes that not only satisfy our strong security definitions (CCA and ciphertext integrity), but in the vast majority of application scenarios are also at least twice as fast (in terms of computation each message block) as any previous scheme with comparable security level.

The *ciphertext expansion* of a scheme says how much the size of a ciphertext grows compared to the size of the message. For a cloud server that stores vast numbers of files, it is naturally crucial to minimize the ciphertext expansion rate. It is also desirable to construct UE schemes that can encrypt *arbitrarily large files*, since a client might want to upload media files such as images or videos. Prior schemes that have achieved these two properties have only been secure in comparatively weak models. Our construction suitable for long messages – enabling encryption of arbitrarily large files with almost no ciphertext expansion – is secure in our strong sense and is thus the first to bridge this gap.

1.1 Related Work

1.1.1 Security Models for UE.

We regard the sequential, epoch-based corruption model of Lehmann and Tackmann [LT18a] (LT18) as the most suitable execution environment to capture the threats in updatable encryption. In this model, the adversary advances to the next epoch via an oracle query. It can choose to submit its (single) challenge when it pleases, and it can later update the challenge ciphertext to the 'current' epoch. Further, the adversary is allowed to adaptively corrupt epoch (i.e. file encryption) keys and update tokens at any point in the game: only at the end of the adversary's execution does the challenger determine whether a trivial win has been made possible by some combination of the corruption queries and the challenge.

LT18 introduced two notions: IND-ENC asks the adversary to submit two plaintexts and distinguish the resulting ciphertext, while possibly having corrupted tokens (but of course not keys) linking this challenge ciphertext to prior or later epochs. Further, they introduced IND-UPD: the adversary provides two ciphertexts that it received via regular encryption-oracle queries in the previous epoch, and has to work out which one has been updated. They observed[2] that plaintext information can be leaked not only through the encryption

[2]The proceedings and full versions of LT18 stated that "IND-ENC *security cannot guarantee anything about the security of updates. In fact, a scheme where the update algorithm* UE.Upd *includes all the old ciphertexts* $C_0, \ldots, C_e$ *in the updated ciphertext* C_{e+1} *could be considered* IND-ENC *secure, but clearly lose all security if a single old key gets compromised.*" This line of argument is flawed,

procedure, but also via updates. For schemes with deterministic updates, the adversary would trivially win if it could acquire the update token that takes the adversarially-provided ciphertexts into the challenge epoch, hence the definition for this setting, named detIND-UPD, is different from that for the randomized setting, named randIND-UPD.

LT18's IND-UPD definition was not the first approach to formalizing the desirable property of *unlinkability* of ciphertexts, which attempts to specify that given two already-updated ciphertexts, the adversary cannot tell if the plaintext is the same. Indeed EPRS (UP-REENC) and later KLR19 (UP-REENC-CCA) also considered this problem, in the ciphertext-dependent update and CCA-secure setting respectively. The work of KLR19 [[KLR19a], § Appendix A] stated that "an even stronger notion [than IND-UPD] might be desirable: namely that fresh and re-encrypted ciphertexts are indistinguishable... which is not guaranteed by UP-REENC" – we will answer this open question later on in our paper.

In the full version of their work [BLMR15], BLMR introduced a security definition for UE denoted update – an extension of a model of symmetric proxy re-encryption. This non-sequential definition is considerably less adaptive than the later work of LT18, since the adversary's key/token corruption queries and ciphertext update queries are very limited. Further, they only considered schemes with deterministic update algorithms.

EPRS [EPRS17a] provided (non-sequential) definitions for updatable authenticated encryption, in the ciphertext-dependent setting. Their work (inherently) covered CCA security and ciphertext integrity (CTXT). These definitions were ambiguous regarding adaptivity: these issues have since been fixed in the full version [EPRS17b].

KLR19 attempted to provide stronger security guarantees for ciphertext-independent UE than LT18, concentrating on chosen-ciphertext security (and the weaker replayable CCA) in addition to integrity of plaintexts and ciphertexts. We revisit these definitions later on, and show how a small modification to their INT-CTXT game gives rise to natural composition results.

In practice, LT18's randIND-UPD definition insists that the ciphertext update procedure Upd requires the server to generate randomness for updating each ciphertext. Further, a scheme meeting both IND-ENC and IND-UPD can still leak the epoch in which the file was uploaded (the 'age' of the ciphertext). While it is arguable that metadata is inherent in outsourced storage, the use of updatable encryption is for high-security applications, and it would not be infeasible to design a system that does not reveal meta-data, which is clearly impossible if the underlying cryptosystem reveals the meta-data.

Recent work by Jarecki et al. [JKR19] considers the key wrapping entity as a separate entity from the data owner or the storage server. While this approach seems promising, their security model is considerably weaker than those considered in our work or the papers already mentioned in this section: the adversary must choose whether to corrupt the key management server (and get the epoch key) or the storage server (and get the update token) for each epoch, and thus it cannot dynamically corrupt earlier keys or tokens at a later stage.

1.1.2 Constructions of Ciphertext-Independent UE

The initial description of updatable encryption by Boneh et al. [BLMR13] was motivated by providing a symmetric-key version of proxy re-encryption (see below). BLMR imagined doing this in a symmetric manner, where each epoch is simply one period in which re-encryption (rotation) has occurred. Their resulting scheme, denoted BLMR, deploys a key-homomorphic PRF, yet the nonce attached to a ciphertext ensures that IND-UPD cannot be met (the scheme pre-dates the IND-UPD notion).

The symmetric-Elgamal-based scheme of LT18, named RISE, uses a randomized update algorithm and is proven to meet IND-ENC and randIND-UPD under DDH. These proofs entail a seemingly unavoidable loss – a cubic term in the total number of epochs – our results also have this factor. LT18 also presented an extended version of the scheme by BLMR, denoted BLMR+, where the nonce is encrypted: they showed that this scheme meets a weak version of IND-UPD called weakIND-UPD, in which if the adversary corrupts the token that links the challenge epoch to the epoch immediately after then a trivial win condition is triggered.

The aim of KLR19 was to achieve stronger security than BLMR, EPRS and LT18 in the ciphertext-independent setting: in particular CCA security and integrity protection. They observed that the structure of RISE ensures that ciphertext integrity cannot be achieved: access to just one update token allows the storage

and in fact IND-ENC rules out schemes of this form: encryptions were always fresh at some point. This claim was corrected and clarified in a June 2019 presentation by the first author [Leh19], and further elaborated on in an update to the full version in December 2019 [LT18b].

provider to construct ciphertexts of messages of its choice. Their generic constructions, based on encrypt-and-MAC and the Naor-Yung paradigm, are strictly less efficient than RISE. We show how to achieve CCA security and integrity protections with novel schemes that are comparably efficient with RISE.

1.1.3 Related Primitives

Proxy re-encryption (PRE) allows a ciphertext that is decryptable by some secret key to be re-encrypted such that it can be decrypted by some other key. Security models for PRE are closer to those for encryption than the strictly sequential outsourced-storage-centric models for UE, and as observed by Lehmann and Tackmann [LT18a] the concepts of allowable corruptions and trivial wins for UE need considerable care when translating to the (more general) PRE setting. Unlinkability is not necessarily desired in PRE – updating the entire ciphertext may not be essential for a PRE scheme to be deemed secure – thus even after conversion to the symmetric setting, prior schemes [AFGH05, CH07] cannot meet the indistinguishability requirements that we ask of UE schemes. Recent works by Lee [Lee17] and Davidson et al. [DDLM19] have highlighted the links between the work of BLMR and EPRS and PRE, and in particular the second work gives a public-key variant of the (sequential) IND-UPD definition of LT18. Myers and Shull [MS18] presented security models for hybrid proxy re-encryption, and gave a single-challenge version of the UP-IND notion of EPRS. While the models are subtly different, the techniques for achieving secure UE and PRE are often similar: in particular rotating keys via exponentiation to some simple function of old and new key (RISE is essentially a combination of Blaze et al.'s symmetric version of ElGamal [BBS98] and ciphertext randomization). Further, the symmetric-key PRE scheme of Sakurai et al. [SNS17] is at a high level similar to SHINE (their all-or-nothing-transform as an inner layer essentially serves the same purpose as the ideal cipher in SHINE), but in a security model that does not allow dynamic corruptions. Their approach includes – this natural approach is somewhat similar to the schemes that we introduce later in the paper.

Tokenization schemes aim to protect short secrets, such as credit card numbers, using deterministic encryption and deterministic updates: this line of work reflects the PCI DSS standard [Cou18] for the payment card industry. Provable security of such schemes was initially explored by Diaz-Santiago et al. [DRC14] and extended to the updatable setting by Cachin et al. [CCFL17]. While much of the formalism in the model of Cachin et al. has been used in recent works on UE (in particular the epoch-based corruption model), the requirements on ciphertext indistinguishability are stronger in the UE setting, where we expect probabilistic encryption of (potentially large) files.

1.2 Contributions

Our first major contribution is defining the xxIND-UE-atk security notion for updatable encryption, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, and comprehensively analyzing its relation to other, existing[3] security notions (xxIND-ENC-atk, xxIND-UPD-atk). Our single definition requires that ciphertexts output by the encryption algorithm are indistinguishable from ciphertexts output by the update algorithm. We show that our new notion is strictly stronger even than combinations of prior notions, both in the randomized- and deterministic-update settings under chosen-plaintext attack and chosen-ciphertext attack. This not only gives us the unlinkability desired by prior works, but also answers the open question posed by KLR19 mentioned on page 5. Fig. 18 describes the relationship between our new notion xxIND-UE-atk and prior notions.

After a slight tweak to KLR19's definitions for CTXT and CCA, we show the following generic composition result: detIND-yy-CPA + INT-CTXT $\Rightarrow$ detIND-yy-CCA for $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$. Combining this result with the relations from detIND-UE-atk above, we thus show that the combination of detIND-UE-CPA and INT-CTXT yields detIND-yy-CCA *for all* $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$.

Our second major contribution is in designing a new, fast updatable encryption scheme SHINE. Our scheme is based on a random-looking permutation combined with the exponentiation map in a cyclic group, and comes in a number of variants: SHINE0, MirrorSHINE and OCBSHINE, for small messages, medium-sized messages and arbitrarily large messages respectively. In Fig. 1, we provide a comparison of security, ciphertext expansion and efficiency between our new schemes and those from prior literature. We also further the understanding of

[3] The notions IND-ENC, randIND-UPD and detIND-UPD (which we denote as IND-ENC-CPA, randIND-UPD-CPA and detIND-UPD-CPA, resp.) are from LT18. The notions UP-IND-CCA and UP-REENC-CCA (detIND-ENC-CCA and detIND-UPD-CCA, resp.) are from KLR19. LT18 and KLR19 both build upon the definitions given by EPRS.

	IND	INT	$\lvert M\rvert$	$\lvert C\rvert$	Enc (Upd)
BLMR [BLMR13]	(det, ENC, CPA)	✘	$l\lvert\mathbb{G}\rvert$	$(l{+}1)\lvert\mathbb{G}\rvert$	$l\mathbf{E}$
BLMR+ [BLMR13, LT18a]	(weak, UE, CPA)	✘	$l\lvert\mathbb{G}\rvert$	$(l{+}1)\lvert\mathbb{G}\rvert$	$l\mathbf{E}$
RISE [LT18a]	(rand, UE, CPA)	✘	$1\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$	$2\mathbf{E}$
SHINE0[CPA] § 5.1.1	(det, UE, CPA)	✘	$(1\text{-}\gamma)\lvert\mathbb{G}\rvert$	$1\lvert\mathbb{G}\rvert$	$1\mathbf{E}$
NYUE [KLR19a]	(rand, ENC, RCCA) (rand, UPD, RCCA)	✘	$1\lvert\mathbb{G}_1\rvert$	$(34\lvert\mathbb{G}_1\rvert, 34\lvert\mathbb{G}_2\rvert)$	$(60\mathbf{E}, 70\mathbf{E})$
NYUAE [KLR19a]	(rand, ENC, RCCA) (rand, UPD, RCCA)	PTXT	$1\lvert\mathbb{G}_1\rvert$	$(58\lvert\mathbb{G}_1\rvert, 44\lvert\mathbb{G}_2\rvert)$	$(110\mathbf{E}, 90\mathbf{E})$
E&M [KLR19a]	(det, ENC, CCA) (det, UPD, CCA)	CTXT	$1\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$	$3\mathbf{E}$
SHINE0 § 5.1.1	(det, UE, CCA)	CTXT	$(1\text{-}2\gamma)\lvert\mathbb{G}\rvert$	$1\lvert\mathbb{G}\rvert$	$1\mathbf{E}$
MirrorSHINE § 5.1.2	(det, UE, CCA)	CTXT	$(1\text{-}\gamma)\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$	$2\mathbf{E}$
OCBSHINE § 5.1.3	(det, UE, CCA)	CTXT	$l\lvert\mathbb{G}\rvert$	$(l{+}2)\lvert\mathbb{G}\rvert$	$(l{+}2)\mathbf{E}$

Figure 1: Comparison of security, ciphertext expansion and efficiency for updatable encryption schemes. $(\mathsf{xx}, \mathsf{yy}, \mathsf{atk})$ represents the best possible xxIND-yy-atk notion that each scheme can achieve. $\mathbf{E}$ represents the cost of an exponentiation, for encryption Enc and ciphertext update Upd. γ represents the bit-size of the used nonce as a proportion of the group element bit-size. For NYUE and NYUAE, size/cost is in pairing groups $\mathbb{G}_1, \mathbb{G}_2$. SHINE0[CPA] is SHINE0 with a zero-length integrity tag. BLMR, BLMR+ and OCBSHINE support encryption of arbitrary size messages (of l blocks), with $\lvert M\rvert \approx l\lvert\mathbb{G}\rvert$.

schemes with deterministic update mechanisms. In particular, we identify the properties that are necessary of such schemes to meet a generalized version of our detIND-UE-atk notion. Another important contribution is that we further improve on the existing epoch insulation techniques that have been used to create proofs of security in the strong corruption environment we pursue. These have been shown to be very useful for studying updatable encryption schemes, and we expect our new techniques to be useful in the future.

1.3 Further Discussion

We have had to make a number of practical design decisions for our new UE scheme SHINE. The main idea is to permute the (combination of nonce and) message and then exponentiate the resulting value, with different mechanisms for enforcing ciphertext integrity depending on the flavor that is being used (which is in turn defined by the desired message length). In this subsection we give some motivation for why we believe that these choices are reasonable.

Deterministic updates. Since we will require indistinguishability of ciphertexts, we know that the UE encryption algorithm should be randomized. The update algorithm may or may not be randomized, however. All known schemes indicate that randomized updates are more expensive than deterministic updates, but there is a small, well-understood security loss in moving to deterministic updates: an adversary with an update token in an appropriate epoch can trivially distinguish between an update of a known ciphertext and other ciphertexts in the next epoch. As a result, in the detIND-UE-CPA case the adversary is only forbidden from obtaining one token compared to randIND-UE-CPA. Furthermore, UE schemes with randomized updates cannot achieve CTXT and CCA security, which is possible for the deterministic-update setting. We believe that the minor CPA security loss is a small price to pay for stronger security (CTXT and CCA) and efficiency gain, in particular to reduce computations in the UE encryption and update algorithms and also improve ciphertext expansion.

Bi-directional key updates. In principle, the token used to update ciphertexts need not be sufficient to derive the new key from the old key. But for every known practical scheme, this derivation is indeed easy. Moreover, for every known practical scheme including ours, the *old* key can be derived from the *new* key (and token). While *uni-directional* update algorithms are desirable, constructing efficient protocols has so far been elusive: this has technical implications for how security notions are defined.

Limited number of epochs. In many applications that we would like to consider, the user of the storage service will control when updates occur (perhaps when an employee with access to key material leaves the organisation, or if an employee loses a key-holding device): this indicates that the total number of key rotations in the lifetime of a storage system might be numbered in the thousands, and in particular could be considerably smaller than the number of outsourced files.

1.4 Document History

This is the fourth version of this document. The first version[4] contained two versions of SHINE that were only detIND-UE-CPA secure, with one flavor for long messages and the other for short messages. The second version[5] introduced a number of technical changes, including: three new schemes that all meet the stronger notion of detIND-UE-CCA security via differing integrity mechanisms; the composition result in Theorem 3; the formulation of INT-CTXT (a tweak of that given by KLR19), plus the description of the relationship between possible integrity notions (Lemma 1 and Remark 1). The third[6] and fourth versions are more incremental. Version 3 incorporated a discussion of implementation of SHINE in Section 5.6, and this version fixes the counterexample used in Theorem 2.7 and adds an explicit length check to queries in the IND-UE game.

1.5 Organization

After introducing syntax and preliminaries in Section 2, we detail the necessary formalism for security modeling in updatable encryption in Section 3. In Section 4 we define our new confidentiality property IND-UE and show how it implies prior notions; in Section 5 we give our new scheme, SHINE, including intuition behind its security analysis and implementation options. We show the security properties that can be met, in our new framework, by prior work schemes BLMR and RISE (LT18) in Section B and C respectively.

2 Preliminaries

Pseudocode $\mathbf{return}\ b' \stackrel{?}{=} b$ is used as shorthand for $\mathbf{if}\ b' = b\ \mathbf{then\ return}\ 1\ \mathbf{//\ else\ return}\ 0$, with an output of 1 indicating adversarial success. We use the concrete security framework, defining adversarial advantage as probability of success in the security game, and avoid statements of security with respect to security notions. In the cases where we wish to indicate that notion A implies notion B (for some fixed primitive), i.e. an adversary's advantage against B carries over to an advantage against A, we show this by bounding these probabilities.

2.1 Hardness Assumptions

For the definition of DDH, CDH and later on, we assume the existence of a group-generation algorithm that is parameterized by λ and outputs a cyclic group $\mathbb{G}$ of order q (where q is of length λ bits) and a generator g. We adapt the definition of pseudorandom functions from Boneh et al. [BLMR13].

Definition 1 (DDH)**.** Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Decision Diffie-Hellman (*DDH*)* problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{DDH\text{-}1}}_{\mathbb{G},\mathcal{A}}(\lambda) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{DDH\text{-}0}}_{\mathbb{G},\mathcal{A}}(\lambda) = 1]\right|$$

where the experiment $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\mathcal{A}}$ is given in Fig. 2.

Definition 2 (CDH)**.** Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Computational Diffie-Hellman (*CDH*)* problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = 1]$$

where the experiment $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}$ is given in Fig. 3.

[4] https://eprint.iacr.org/2019/1457/20191218:195141, 17th December 2019
[5] https://eprint.iacr.org/2019/1457/20200221:133540, 21st February 2020
[6] https://eprint.iacr.org/2019/1457/20200729:120307, 29th July 2020

$\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\mathcal{A}}(\lambda)$

1 : $x, y, r \xleftarrow{\$} \mathbb{Z}_q$
2 : $X \leftarrow g^x; Y \leftarrow g^y$
3 : **if** $\mathrm{b} = 0$
4 : $\quad Z \leftarrow g^{xy}$
5 : **else**
6 : $\quad Z \leftarrow g^r$
7 : $\mathrm{b}' \leftarrow \mathcal{A}(g, X, Y, Z)$
8 : **return** b'

Figure 2: DDH experiment $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\mathcal{A}}$

$\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda)$

1 : $x, y \xleftarrow{\$} \mathbb{Z}_q$
2 : $X \leftarrow g^x$
3 : $Y \leftarrow g^y$
4 : $Z \leftarrow \mathcal{A}(g, X, Y)$
5 : **if** $Z = g^{xy}$
6 : $\quad$ **return** 1
7 : **else**
8 : $\quad$ **return** 0

Figure 3: CDH experiment $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\mathcal{A}}$

Definition 3 (PRF)**.** Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be an efficiently computable function, where $\mathcal{K}$ is called the key space, $\mathcal{X}$ is the domain, and $\mathcal{Y}$ is the range. The PRF advantage for $\mathcal{A}$ against F is given by

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

where the experiment $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\mathcal{A}}$ is given in Fig. 4.

$\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\mathcal{A}}(\lambda)$

1 : **if** $\mathrm{b} = 0$
2 : $\quad \mathrm{k} \xleftarrow{\$} \mathcal{K}$
3 : $\quad f(\cdot) \leftarrow F(\mathrm{k}, \cdot)$
4 : **else**
5 : $\quad f(\cdot) \xleftarrow{\$} \{f : \mathcal{X} \to \mathcal{Y}\}$
6 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.f}()$
7 : **return** b'

$\mathcal{O}.f(x)$

8 : **if** $x \notin \mathcal{X}$
9 : $\quad$ **return** $\perp$
10 : **else**
11 : $\quad$ **return** $f(x)$

Figure 4: PRF experiment $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\mathcal{A}}$

2.2 Updatable Encryption

We follow the syntax of prior work [KLR19a], defining an Updatable Encryption (UE) scheme as a tuple of algorithms $\{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ that operate in epochs, these algorithms are described in Fig. 5. A scheme is defined over some plaintext space $\mathcal{MS}$, ciphertext space $\mathcal{CS}$, key space $\mathcal{KS}$ and token space $\mathcal{TS}$. We specify integer $n + 1$ as the (total) number of epochs over which a UE scheme can operate, though this is only for proof purposes. Correctness [KLR19a] is defined as expected: fresh encryptions and updated ciphertexts should decrypt to the correct message under the appropriate epoch key. In contrast to prior work, we only consider deterministic token generation algorithms – all schemes in prior literature and our schemes allow the token to be produced deterministically from epoch keys alone.

In addition to enabling ciphertext updates, in many schemes the token allows ciphertexts to be 'downgraded': performing some analog of the $\mathsf{UE.Upd}$ operation on a ciphertext C created in (or updated to) epoch e yields a valid ciphertext in epoch e-1. Such a scheme is said to have *bi-directional ciphertext updates*[7]. Furthermore, for many constructions, the token additionally enables key derivation, given one adjacent key. If this can be done in both directions – i.e. knowledge of k_{e} and $\Delta_{\mathsf{e}+1}$ allows derivation of $\mathrm{k}_{\mathsf{e}+1}$ AND knowledge of $\mathrm{k}_{\mathsf{e}+1}$ and $\Delta_{\mathsf{e}+1}$ allows derivation of k_{e} – then such schemes are referred to by LT18 as having *bi-directional key updates*. If such derivation is only possible in one 'direction' then the scheme is said to have *uni-directional key updates*. Much of the prior literature on updatable encryption has distinguished these notions:

[7]For example if the Upd procedure exponentiates all ciphertext components using the token, as done in SHINE, then Upd itself is sufficient to demonstrate this property.

Algorithm		Rand/Det	Input	Output	Syntax
UE.KG	Key Gen	Rand	λ	$\mathrm{k_e}$	$\mathrm{k_e} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
UE.TG	Token Gen	Det	$\mathrm{k_e}, \mathrm{k_{e+1}}$	Δ_{e+1}	$\Delta_{e+1} \leftarrow \mathsf{UE.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$
UE.Enc	Encryption	Rand	$\mathrm{M}, \mathrm{k_e}$	$\mathrm{C_e}$	$\mathrm{C_e} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M})$
UE.Dec	Decryption	Det	$\mathrm{C_e}, \mathrm{k_e}$	$\mathrm{M'}$ or $\perp$	$\{\mathrm{M'} / \perp\} \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C_e})$
UE.Upd	Update Ctxt	Rand/det	$\mathrm{C_e}, \Delta_{e+1}$	$\mathrm{C_{e+1}}$	$\mathrm{C_{e+1}} \xleftarrow{\$} \mathsf{UE.Upd}(\Delta_{e+1}, \mathrm{C_e})$

Figure 5: Syntax of algorithms defining an Updatable Encryption scheme UE.

we stress that all schemes and definitions of security considered in this paper have bi-directional key updates and bi-directional ciphertext updates.

3 Security Models for Updatable Encryption

We consider a number of indistinguishability-based confidentiality games and integrity games for assessing security of updatable encryption schemes. The environment provided by the challenger attempts to give as much power as possible to adversary $\mathcal{A}$. The adversary may call for a number of oracles, and after $\mathcal{A}$ has finished running the challenger computes whether or not any of the actions enabled a trivial win. The available oracles are described in Fig. 6. An overview of the oracles $\mathcal{A}$ has access to in each security game is provided in Fig. 7.

Setup(λ)

$\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda)$
$\Delta_0 \leftarrow \perp$
$\mathsf{e}, \mathsf{c} \leftarrow 0$
$\mathsf{phase}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

$\mathrm{C} \leftarrow \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M})$
$\mathsf{c} \leftarrow \mathsf{c} + 1$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}, \mathsf{e})\}$
return C

$\mathcal{O}.\mathsf{Dec}(\mathrm{C})$

$\mathsf{twf} \leftarrow 1$ **if**
  $\mathsf{phase} = 1$ **and** $\mathrm{C} \in \tilde{\mathcal{L}}$
$\mathrm{M'}$ **or** $\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C})$
return $\mathrm{M'}$ **or** $\perp$

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

$\mathsf{e} \leftarrow \mathsf{e} + 1$
$\mathrm{k_e} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
$\Delta_\mathsf{e} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k_{e\text{-}1}}, \mathrm{k_e})$
if $\mathsf{phase} = 1$
  $\tilde{\mathrm{C}}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \tilde{\mathrm{C}}_{\mathsf{e}\text{-}1})$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{e-1})$

if $(j, \mathrm{C}_{e-1}, \mathsf{e}-1) \notin \mathcal{L}$
  return $\perp$
$\mathrm{C_e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathrm{C}_{e-1})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \mathrm{C_e}, \mathsf{e})\}$
return $\mathrm{C_e}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

if $\hat{\mathsf{e}} > \mathsf{e}$
  return $\perp$
if $\mathsf{inp} = \mathsf{key}$
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
  return $\mathrm{k}_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
  return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
$\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e})\}$
return $\tilde{\mathrm{C}}_\mathsf{e}$

$\mathcal{O}.\mathsf{Try}(\tilde{\mathrm{C}})$

if $\mathsf{phase} = 1$
  return $\perp$
$\mathsf{phase} \leftarrow 1$
$\mathsf{twf} \leftarrow 1$ **if**
  $\mathsf{e} \in \mathcal{K}^*$ **or** $\tilde{\mathrm{C}} \in \mathcal{L}^*$
$\mathrm{M'}$ **or** $\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \tilde{\mathrm{C}})$
if $\mathrm{M'} \neq \perp$
  $\mathsf{win} \leftarrow 1$

Figure 6: Oracles in security games for updatable encryption. The boxed lines in $\mathcal{O}.\mathsf{Try}$ only apply to $\mathsf{INT\text{-}CTXT}^\mathsf{s}$: in this game the adversary is allowed to query the $\mathcal{O}.\mathsf{Try}$ oracle only once. Computing $\tilde{\mathcal{L}}^*$ is discussed in Section 3.2.

Notion	$\mathcal{O}$.Enc	$\mathcal{O}$.Dec	$\mathcal{O}$.Next	$\mathcal{O}$.Upd	$\mathcal{O}$.Corr	$\mathcal{O}$.Upd$\tilde{\mathrm{C}}$	$\mathcal{O}$.Try
detIND-yy-CPA	✓	×	✓	✓	✓	✓	×
randIND-yy-CPA	✓	×	✓	✓	✓	✓	×
detIND-yy-CCA	✓	✓	✓	✓	✓	✓	×
INT-CTXT	✓	×	✓	✓	✓	×	✓

Figure 7: Oracles the adversary is allowed to query in different security games, where $\mathsf{yy} \in \{\mathsf{ENC}, \mathsf{UPD}, \mathsf{UE}\}$. $\times$ indicates the adversary does not have access to the corresponding oracle,✓ indicates the adversary has access to the corresponding oracle.

Confidentiality. A generic representation of all confidentiality games described in this paper is detailed in Fig. 8. The current epoch is advanced by an adversarial call to $\mathcal{O}$.Next – simulating UE.KG and UE.TG – and keys and tokens (for the current or any prior epoch) can be corrupted via $\mathcal{O}$.Corr. The adversary can encrypt arbitrary messages via $\mathcal{O}$.Enc, and update these 'non-challenge' ciphertexts via $\mathcal{O}$.Upd. In CCA games, the adversary can additionally call decryption oracle $\mathcal{O}$.Dec (with some natural restrictions to prevent trivial wins). At some point $\mathcal{A}$ makes its challenge by providing two inputs, and receives the challenge ciphertext – and in later epochs can receive an updated version by calling $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$ (computing this value is actually done by $\mathcal{O}$.Next, a call to $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$ returns it). $\mathcal{A}$ can then interact with its other oracles again, and eventually outputs its guess bit. The flag phase tracks whether or not $\mathcal{A}$ has made its challenge, and we always give the epoch in which the challenge happens a special identifier $\tilde{\mathsf{e}}$. If $\mathcal{A}$ makes any action that would lead to a trivial win, the flag twf is set as 1 and $\mathcal{A}$'s output is discarded and replaced by a random bit. We follow the bookkeeping techniques of LT18 and KLR19, using the following sets to track ciphertexts and their updates that can be known to the adversary.

- $\mathcal{L}$: List of non-challenge ciphertexts (from $\mathcal{O}$.Enc or $\mathcal{O}$.Upd) with entries of form $(\mathsf{c}, \mathrm{C}, \mathsf{e})$, where query identifier c is a counter incremented with each new $\mathcal{O}$.Enc query.
- $\tilde{\mathcal{L}}$: List of updated versions of challenge ciphertext (created via $\mathcal{O}$.Next, received by adversary via $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$), with entries of form $(\tilde{\mathrm{C}}, \mathsf{e})$.

Further, we use the following lists that track epochs only.

- $\mathcal{C}$: List of epochs in which adversary learned updated version of challenge ciphertext (via CHALL or $\mathcal{O}$.Upd$\tilde{\mathrm{C}}$).
- $\mathcal{K}$: List of epochs in which the adversary corrupted the encryption key.
- $\mathcal{T}$: List of epochs in which the adversary corrupted the update token.

All experiments necessarily maintain some state, but we omit this for readability reasons. The challenger's state is $\mathbf{S} \leftarrow \{\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}\}$, and the system state in the current epoch is given by $\mathsf{st} \leftarrow (\mathrm{k}_\mathsf{e}, \Delta_\mathsf{e}, \mathbf{S}, \mathsf{e})$.

An at-a-glance overview of CHALL for various security definitions is given in Fig. 9. For security games such as LT18's IND-UPD notion, where the adversary must submit as its challenge two ciphertexts (that it received from $\mathcal{O}$.Enc) and one is updated, the game must also track in which epochs the adversary has updates of these ciphertexts. We will later specify a version of our new xxIND-UE-atk notion that allows the adversary to submit a ciphertext that existed in any epoch prior to the challenge epoch, not just the one immediately before: this introduces some additional bookkeeping (discussed further in Section 3.2).

	CHALL	Output of " Create $\tilde{\mathrm{C}}$ with CHALL" (in $\tilde{\mathsf{e}}$)		
xxIND-ENC-atk	$\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1$	UE.Enc$(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_0)$	**or**	UE.Enc$(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_1)$
xxIND-UPD-atk	$\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$	UE.Upd$(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}}_0)$	**or**	UE.Upd$(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}}_1)$
xxIND-UE-atk	$\bar{\mathrm{M}}, \bar{\mathrm{C}}$	UE.Enc$(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$	**or**	UE.Upd$(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$

Figure 9: Intuitive description of challenge inputs and outputs in confidentiality games for updatable encryption schemes, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$. Full definitions are given in Section 3.1 and 4.1.

$\mathbf{Exp}^{\mathsf{xxIND\text{-}yy\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : **do Setup**
2 : $\mathsf{CHALL} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 : $\mathsf{phase} \leftarrow 1; \tilde{\mathsf{e}} \leftarrow \mathsf{e}$
4 : Create $\tilde{\mathrm{C}}$ with CHALL; $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e})\}$
5 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
6 : $\mathsf{twf} \leftarrow 1$ **if**
7 : $\quad \mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ **or**
8 : $\quad \mathsf{xx} = \mathsf{det}$ **and** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
9 : **if** $\mathsf{twf} = 1$
10 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
11 : **return** b'

Figure 8: Generic description of confidentiality experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}yy\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ for scheme UE and adversary $\mathcal{A}$, for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$, $\mathsf{yy} \in \{\mathsf{ENC}, \mathsf{UPD}, \mathsf{UE}\}$ and $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. We do not consider (and thus do not formally define) randIND-yy-CCA; only in detIND-yy-CCA games does $\mathcal{A}$ have access to $\mathcal{O}$.Dec. CHALL is the challenge input provided by $\mathcal{A}$: how to perform Create $\tilde{\mathrm{C}}$ with CHALL is shown in Fig. 11, Fig. 12 and Fig. 17. Trivial win conditions, i.e. deciding the value of twf and computing $\mathcal{K}^*, \mathcal{C}^*, \mathcal{I}^*$, are discussed in Section 3.2.

A note on nomenclature: the adversary can make its challenge query to receive *the challenge ciphertext*, and then acquire *updates of the challenge ciphertext* via calls to $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, and additionally it can calculate *challenge-equal ciphertexts* via applying tokens it gets via $\mathcal{O}$.Corr queries.

When appropriate, we will restrict our experiments to provide definitions of security that are more suitable for assessing schemes with deterministic update mechanisms. For such schemes, access to the update token for the challenge epoch ($\Delta_{\tilde{\mathsf{e}}}$) allows the adversary to trivially win detIND-UPD-atk and detIND-UE-atk for $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. Note however that the definitions are not restricted to schemes with deterministic updates: such schemes are simply insecure in terms of randIND-UPD-CPA and randIND-UE-CPA.

Ciphertext Integrity. In ciphertext integrity (CTXT) game, the adversary is allowed to make calls to oracles $\mathcal{O}$.Enc, $\mathcal{O}$.Next, $\mathcal{O}$.Upd and $\mathcal{O}$.Corr. At some point $\mathcal{A}$ attempts to provide a forgery via $\mathcal{O}$.Try; as part of this query the challenger will assess if it is valid. We distinguish between the single-$\mathcal{O}$.Try case (INT-CTXT$^{\mathsf{s}}$) and the multi-$\mathcal{O}$.Try case (INT-CTXT). Here, "valid" means decryption outputs a message (i.e. not $\perp$). In the single-$\mathcal{O}$.Try case, $\mathcal{A}$ can continue making oracle queries after its $\mathcal{O}$.Try query, however this is of no benefit since it has already won or lost. In the multi-$\mathcal{O}$.Try case, $\mathcal{A}$ can make any number of $\mathcal{O}$.Try queries: as long as it wins once, it wins the ciphertext integrity game. Formally, the definition of ciphertext integrity is given in Definition 4.

Definition 4. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the INT-CTXT advantage of an adversary $\mathcal{A}$ against UE is defined as

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}} = 1]$$

where the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}}$ is given in Fig. 6 and Fig. 10. Particularly, if $\mathcal{A}$ is allowed to ask only one $\mathcal{O}$.Try query, denote such notion as INT-CTXT$^{\mathsf{s}}$.

$\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : **do Setup**
2 : $\mathsf{win} \leftarrow 0$
3 : $\mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Try}}(\lambda)$
4 : **if** $\mathsf{twf} = 1$
5 : $\quad \mathsf{win} \leftarrow 0$
6 : **return** win

Figure 10: INT-CTXT security notion for updatable encryption scheme UE and adversary $\mathcal{A}$. Deciding twf and computing $\mathcal{L}^*$ are discussed in Section 3.2.

Note that INT-CTXT trivially implies INT-CTXT$^{\mathsf{s}}$. We can prove that INT-CTXT$^{\mathsf{s}}$ implies INT-CTXT too, with loss upper-bounded by the number of $\mathcal{O}$.Try queries. We prove this result in Lemma 1. KLR19 defined ciphertext integrity with one $\mathcal{O}$.Try query plus access to $\mathcal{O}$.Dec, and the game ends when the $\mathcal{O}$.Try query happens. It is hard to prove the generic relation among CPA, CTXT and CCA using this formulation. Notice that decryption oracles give the adversary power to win the CTXT game even it only has one $\mathcal{O}$.Try query. The adversary can send its forgery to the decryption oracle to test if it is valid (if $\mathcal{O}$.Dec outputs a message and not $\perp$) – thus $\mathcal{A}$ can continue to send forgeries to $\mathcal{O}$.Dec until a valid one is found, and then send this as a $\mathcal{O}$.Try query (and win the game). So intuitively, a decryption oracle is equivalent to multiple $\mathcal{O}$.Try queries. Proving that all these variants of CTXT definitions are equivalent to each other is straightforward, with the loss upper-bounded by the sum of $\mathcal{O}$.Try queries and decryption queries.

Remark 1. The definition of INT-CTXT is more natural for defining ciphertext integrity, however, it is easier to use INT-CTXT$^{\mathsf{s}}$ notion to prove ciphertext integrity for specific UE schemes. As INT-CTXT $\iff$ INT-CTXT$^{\mathsf{s}}$, we use both definitions in this paper.

Lemma 1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any INT-CTXT adversary $\mathcal{A}$ against UE that queries as most Q_T $\mathcal{O}$.Try queries, there exists an INT-CTXT$^{\mathsf{s}}$ adversary $\mathcal{B}_1$ against UE such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq Q_T \cdot \mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{B}_1}(\lambda).$$

Proof. As definition 4, we have

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}} = 1].$$

We define Q_T games, for the i-th game $\mathcal{G}_i$, it is identical to INT-CTXT game except for the challenger only responses the i-th $\mathcal{O}$.Try query and returns $\perp$ to the rest of $\mathcal{O}$.Try queries. Then we have

$$\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\mathcal{A}} = 1] \leq \sum_{i=1}^{Q_T} \mathbf{Pr}[\mathcal{G}_i = 1].$$

Then we claim that for any $i \in \{1, \ldots, Q_T\}$ there exists an adversary

$$\mathbf{Pr}[\mathcal{G}_i = 1] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{B}_{1,i}}(\lambda).$$

We can construct the reduction $\mathcal{B}_{1,i}$ playing INT-CTXT$^{\mathsf{s}}$ game and simulating the responses of $\mathcal{G}_i$ by submitting the i-th $\mathcal{O}$.Try query to its INT-CTXT$^{\mathsf{s}}$ challenger and returns $\perp$ for the rest $\mathcal{O}$.Try queries. Other queries and the final result can be passed from INT-CTXT$^{\mathsf{s}}$ game to $\mathcal{G}_i$. Then we have the desired result.

□

3.1 Existing Definitions of Confidentiality

Here we describe existing confidentiality notions given by LT18 and KLR19, including formal definitions for their IND-yy-CPA and IND-yy-CCA notions, respectively. (Note that KLR19 used UP-REENC to refer to the the unlinkability notion that we and LT18 call IND-UPD). We will define our new security notion in Section 4.1 and compare the relationship between all notions in Section 4.2.

Definition 5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the xxIND-ENC-atk advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ is given in Fig. 6, Fig. 8 and Fig. 11.

Definition 6. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the xxIND-UPD-atk advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiments $\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ are given in Fig. 6, Fig. 8 and Fig. 12.

$\mathbf{Exp}^{\mathsf{xxIND\text{-}ENC\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$
3 : **if** $|\bar{\mathrm{M}}_0| \neq |\bar{\mathrm{M}}_1|$
4 : **return** $\perp$
5 : $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathrm{e}}}, \bar{\mathrm{M}}_{\mathrm{b}})$
6 : **return** $\tilde{\mathrm{C}}$

Figure 11: Challenge call definition for xxIND-ENC-atk security experiment; the full experiment is given in combination with Fig. 6 and Fig. 8.

$\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1)$
3 : **if** $|\bar{\mathrm{C}}_0| \neq |\bar{\mathrm{C}}_1|$
4 : **return** $\perp$
5 : **if** $(\bar{\mathrm{C}}_0, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$ **or** $(\bar{\mathrm{C}}_1, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$
6 : **return** $\perp$
7 : $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Upd}(\Delta_{\tilde{\mathrm{e}}}, \bar{\mathrm{C}}_{\mathrm{b}})$
8 : **return** $\tilde{\mathrm{C}}$

Figure 12: Challenge call definition for xxIND-UPD-atk security experiment; the full experiment is given in combination with Fig. 6 and Fig. 8.

We do not define randIND-ENC-CCA or randIND-UPD-CCA – these notions were formalized by KLR19. Note that trivial win via direct update (see Section 3.2) is never triggered in the detIND-ENC-CPA game. Thus, randIND-ENC-CPA is equivalent to detIND-ENC-CPA. For simplicity, we will often denote the notion xxIND-ENC-CPA as IND-ENC-CPA.

Remark 2. LT18 defined weakIND-ENC-CPA and weakIND-UPD-CPA for analyzing BLMR+, a modification of BLMR's scheme where the nonce is encrypted using symmetric encryption. In this notion, the adversary trivially loses if it obtains an update token linking the challenge epoch to the epoch before or after. In Section B we show that BLMR+ is weakIND-UE-CPA secure.

3.2 Trivial Win Conditions

3.2.1 Trivial Win Conditions in Confidentiality Games

Trivial wins via keys and ciphertexts. The following is for analyzing all confidentiality games. We again follow LT18 in defining the epoch identification sets $\mathcal{C}^*$, $\mathcal{K}^*$ and $\mathcal{T}^*$ as the extended sets of $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{T}$ in which the adversary has learned or inferred information via its acquired tokens. These extended sets are used to exclude cases in which the adversary trivially wins, i.e. if $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$, then there exists an epoch in which the adversary knows the epoch key and a valid update of the challenge ciphertext. Note that the challenger computes these sets once the adversary has finished running. We employ the following algorithms of LT18 (for bi-directional updates):

$$\mathcal{K}^* \leftarrow \{\mathsf{e} \in \{0,\ldots,n\} | \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\}$$
$$\mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \iff (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}\text{+}1) \wedge \mathsf{e}\text{+}1 \in \mathcal{T})$$
$$\mathcal{T}^* \leftarrow \{\mathsf{e} \in \{0,\ldots,n\} | (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e}\text{-}1 \in \mathcal{K}^*)\}$$
$$\mathcal{C}^* \leftarrow \{\mathsf{e} \in \{0,\ldots,n\} | \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\}$$
$$\mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \iff$$
$$(\mathsf{e} = \tilde{\mathsf{e}}) \vee (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*) \vee (\mathsf{ChallEq}(\mathsf{e}\text{+}1) \wedge \mathsf{e}\text{+}1 \in \mathcal{T}^*)$$

Trivial wins via direct updates. The following is for analyzing detIND-yy-atk security notions, for yy $\in \{\mathsf{UE}, \mathsf{UPD}\}$ and atk $\in \{\mathsf{CPA}, \mathsf{CCA}\}$, where the adversary provides as its challenge one or two ciphertexts that it received from $\mathcal{O}$.Enc. The challenger needs to use $\mathcal{L}$ to track the information the adversary has about these challenge input values.

Define a new list $\mathcal{I}$ as the list of epochs in which the adversary learned an updated version of the ciphertext(s) given as a challenge input. Furthermore, define $\mathcal{I}^*$ to be the extended set in which the adversary has learned or inferred information via token corruption. We will use this set to exclude cases which the adversary trivially wins, i.e. if $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$, then there exists an epoch in which the adversary knows the updated ciphertext of $\bar{\mathrm{C}}$ and a valid challenge-equal ciphertext. For deterministic updates, the adversary can simply compare these ciphertexts to win the game. In particular, if $\bar{\mathrm{C}}$ is restricted to come from $\tilde{\mathsf{e}} - 1$ (recall the challenge epoch is $\tilde{\mathsf{e}}$), then the condition $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ is equivalent to the win condition that LT18 used for IND-UPD: $\Delta_{\tilde{\mathsf{e}}} \in \mathcal{T}^*$ or $\mathcal{A}$ did $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ in $\tilde{\mathsf{e}}$. Our generalization is necessary for a variant of xxIND-UE-atk that we define later in which the challenge ciphertext input can come from any prior epoch, and not just the epoch immediately before the one in which the challenge is made.

To compute $\mathcal{I}$, find an entry in $\mathcal{L}$ that contains challenge input $\bar{\mathrm{C}}$. Then for that entry, note the query identifier c, scan $\mathcal{L}$ for other entries with this identifier, and add into list $\mathcal{I}$ all found indices:

$$\mathcal{I} \leftarrow \{\mathsf{e} \in \{0,\ldots,n\} | (\mathsf{c}, \cdot, \mathsf{e}) \in \mathcal{L}\}.$$

Then compute $\mathcal{I}^*$ as follows:

$$\mathcal{I}^* \leftarrow \{\mathsf{e} \in \{0,\ldots,n\} | \mathsf{ChallinputEq}(\mathsf{e}) = \mathsf{true}\}$$
$$\mathsf{true} \leftarrow \mathsf{ChallinputEq}(\mathsf{e}) \iff$$
$$(\mathsf{e} \in \mathcal{I}) \vee (\mathsf{ChallinputEq}(\mathsf{e}\text{-}1) \wedge \mathsf{e} \in \mathcal{T}^*) \vee (\mathsf{ChallinputEq}(\mathsf{e}\text{+}1) \wedge \mathsf{e}\text{+}1 \in \mathcal{T}^*)$$

Additionally, if the adversary submits two ciphertexts $\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$ as challenge (as in xxIND-UPD-atk), we compute $\mathcal{I}_i, \mathcal{I}_i^*, i \in \{0,1\}$ first and then use $\mathcal{I}^* = \mathcal{I}_0^* \cup \mathcal{I}_1^*$ to check the trivial win condition. An example of trivial win conditions $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ is shown in Fig. 16.

We do not consider this trivial win condition for the ENC notion, as there is no ciphertext in the challenge input value, i.e. $\mathcal{I}^* = \emptyset$. Thus, the assessment $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ in experiment $\mathbf{Exp}^{\mathsf{detIND\text{-}ENC\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ (see Fig. 8) will never be true.

Trivial wins via decryptions. The following is for analyzing detIND-yy-CCA for yy $\in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$, where the adversary has access to $\mathcal{O}$.Dec. We follow the trivial win analysis in KLR19: suppose the adversary knows a challenge ciphertext $(\tilde{\mathrm{C}}, \mathsf{e}_0) \in \tilde{\mathcal{L}}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can update the challenge ciphertext from epoch e_0 to epoch e. If $\mathcal{A}$ sends the updated ciphertext to $\mathcal{O}$.Dec this will reveal the underlying message, and $\mathcal{A}$ trivially wins the game: we shall exclude this type of attack.

Define $\tilde{\mathcal{L}}^*$ to be the extended set of $\tilde{\mathcal{L}}$ in which the adversary has learned or inferred information via token corruption. Whenever $\mathcal{O}$.Dec receives a ciphertext located in $\tilde{\mathcal{L}}^*$, the challenger will set the trivial win flag twf to be 1. The list $\tilde{\mathcal{L}}^*$ is updated while the security game is running. After the challenge query happens, the challenger updates $\tilde{\mathcal{L}}^*$ whenever an element is added to list $\tilde{\mathcal{L}}$ or a token is corrupted. In Fig. 13 we show how list $\tilde{\mathcal{L}}^*$ is updated.

Update $\tilde{\mathcal{L}}^*$

1 : **if** challenge query **or** $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ happens
2 : $\quad\tilde{\mathcal{L}}^* \leftarrow \tilde{\mathcal{L}}^* \cup \{(\tilde{\mathrm{C}}, \cdot)\}$
3 : **if** $\mathsf{phase} = 1$ **and** $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, \cdot)$ happens
4 : $\quad$**for** $i \in \mathcal{T}^*$ **and** $(\tilde{\mathrm{C}}_{i-1}, i-1) \in \tilde{\mathcal{L}}^*$ **do**
5 : $\quad\quad\tilde{\mathcal{L}}^* \leftarrow \tilde{\mathcal{L}}^* \cup \{(\tilde{\mathrm{C}}_i, i)\}$

Figure 13: Update procedure for list $\tilde{\mathcal{L}}^*$

Update $\mathcal{L}^*$

1 : **if** $\mathcal{O}.\mathsf{Enc}$ **or** $\mathcal{O}.\mathsf{Upd}$ happens
2 : $\quad\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\cdot, \mathrm{C}, \cdot)\}$
3 : **if** $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, \cdot)$ happens
4 : $\quad$**for** $i \in \mathcal{T}^*$ **do**
5 : $\quad\quad$**for** $(j, \mathrm{C}_{i-1}, i-1) \in \mathcal{L}^*$ **do**
6 : $\quad\quad\quad\mathrm{C}_i \leftarrow \mathsf{UE.Upd}(\Delta_i, \mathrm{C}_{i-1})$
7 : $\quad\quad\quad\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C}_i, i)\}$

Figure 14: Update procedure for list $\mathcal{L}^*$.

3.2.2 Trivial Win Conditions in Ciphertext Integrity Games

We again follow the trivial win analysis in KLR19. In ciphertext integrity games for updatable encryption, we do not consider the randomized update setting as the adversary can update an old ciphertext via a corrupted token to provide any number of new valid forgeries to the Try query to trivially win this game.

Trivial wins via keys. If an epoch key is corrupted, then the adversary can use this key to forge ciphertexts in this epoch. We exclude this trivial win: if the adversary provides a forgery in an epoch in list $\mathcal{K}^*$, the challenger sets twf to 1.

Trivial wins via ciphertexts. Suppose the adversary knows a ciphertext $(\mathrm{C}, \mathsf{e}_0) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_0 + 1$ to epoch e, then the adversary can provide a forgery by updating C to epoch e. We shall exclude this type of forgeries.

Define $\mathcal{L}^*$ to be the extended set of $\mathcal{L}$ in which the adversary has learned or inferred information via token corruption. If $\mathcal{O}.\mathsf{Try}$ receives a ciphertext located in $\mathcal{L}^*$, the challenger will set twf to 1. The list $\mathcal{L}^*$ is updated while the security game is running. Ciphertexts output by $\mathcal{O}.\mathsf{Enc}$ and $\mathcal{O}.\mathsf{Upd}$ are known to the adversary. Furthermore, whenever a token is corrupted, the challenger may update list $\mathcal{L}^*$ as well. In Fig. 14 we show how list $\mathcal{L}^*$ is updated.

3.3 Firewall Technique

In order to prove security for updatable encryption in the epoch-based model with strong corruption capabilities, cryptographic separation is required between the epochs in which the adversary knows key material, and those in which it knows challenge-equal ciphertexts (acquired/calculated via queries to $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ and $\mathcal{O}.\mathsf{Corr}(\Delta)$). To ensure this, we follow prior work in explicitly defining the 'safe' or *insulated* regions, as we explain below. These regions insulate epoch keys, tokens and ciphertexts: outside of an insulated region a reduction in a security proof can generate keys and tokens itself, but within these regions it must embed its challenge while still providing the underlying adversary with access to the appropriate oracles. A thorough discussion of how we leverage these insulated regions in proofs is given in Section 5.3.

To understand the idea of firewalls, consider any security game (for bi-directional schemes) in which the trivial win conditions are *not* triggered. If the adversary $\mathcal{A}$ corrupts all tokens then either it never corrupts any keys or it never asks for a challenge ciphertext. Suppose that $\mathcal{A}$ does ask for a challenge ciphertext in epoch $\tilde{\mathsf{e}}$ [8]. Then there exists an (unique) epoch continuum around $\tilde{\mathsf{e}}$ such that no keys in this epoch continuum, and no tokens in the boundaries of this epoch continuum are corrupted. Moreover, we can assume that all tokens within this epoch continuum are corrupted, because once the adversary has finished corrupting keys, it can corrupt any remaining tokens that do not 'touch' those corrupted keys. This observation is first used in the IND-UPD proof of RISE provided by Lehmann and Tackmann [LT18a], and Klooß et al. [KLR19a] provided an extended description of this 'key insulation' technique. We name these epoch ranges *insulated regions* and their boundaries to be *firewalls*.

[8] In the situation that the adversary does not corrupt any keys to the left or the right (or both) of the challenge epoch, the insulated region thus extends to the boundary (or boundaries) of the epoch continuum.

Definition 7. An *insulated region* with *firewalls* fwl and fwr is a consecutive sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ for which:

- no key in the sequence of epochs $(\mathsf{fwl}, \ldots, \mathsf{fwr})$ is corrupted;
- the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are not corrupted (if they exist);
- all tokens $(\Delta_{\mathsf{fwl}+1}, \ldots, \Delta_{\mathsf{fwr}})$ are corrupted (if any exist).

We denote the firewalls bordering the special insulated region that contains $\tilde{\mathsf{e}}$ as $\hat{\mathsf{fwl}}$ and $\hat{\mathsf{fwr}}$ – though note that there could be (many, distinct) insulated regions elsewhere in the epoch continuum. Specifically, when the adversary asks for updated versions of the challenge ciphertext, the epoch in which this query occurs must also fall within (what the challenger later calculates as) an insulated region. In Fig. 15 we give an algorithm FW-Find for computing firewall locations. The list $\mathcal{FW}$ tracks, and appends a label to, each insulated region and its firewalls. Observe that if an epoch is a left firewall, then neither the key nor the token for that epoch are corrupted. From the left firewall, since we assume that all tokens are corrupted, track to the right until either a token is not corrupted or a key is.

FW-Find

$\mathcal{FW} \leftarrow \emptyset$
$j = 0$
for $\mathsf{e} \in \{0, \ldots, n\}$ **do**
  if $\mathsf{e} \in \neg(\mathcal{T}^* \cup \mathcal{K}^*)$
    $j \leftarrow j + 1$
    $\mathsf{fwl}_j \leftarrow \mathsf{e}$
  if $(\mathsf{e}+1 \notin \mathcal{T}^*)$ **and** $(\mathsf{e} \notin \mathcal{K}^*)$
    $\mathsf{fwr}_j \leftarrow \mathsf{e}$
    $\mathcal{FW} \leftarrow \{(j, \mathsf{fwl}_j, \mathsf{fwr}_j)\}$

Figure 15: Algorithm FW-Find for computing all firewalls.

3.3.1 Example of Epoch Corruption and Trivial Wins

In Fig. 16 we indicate the trivial win conditions and insulated regions for a particular adversarial corruption strategy, in the experiment for detIND-UE*-CPA (this notion chosen here to demonstrate how the challenger populates its lists). Suppose challenge epoch $\tilde{\mathsf{e}} = 8$, and further assume $\mathcal{K}^* = \{1, 6, 9\}$, and $\mathcal{T}^* = \{3, 4, 8\}$, meaning that $\mathcal{C}^* = \{7, 8\}$. Suppose $\bar{\mathrm{C}}$ is in epoch 1 and the adversary has asked $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ in epoch 2, so $\bar{\mathrm{C}}_2, \bar{\mathrm{C}}_3, \bar{\mathrm{C}}_4$ are updated ciphertexts of $\bar{\mathrm{C}}$, therefore $\mathcal{I}^* = \{1, 2, 3, 4\}$. So $\mathcal{C}^* \cap \mathcal{K}^* = \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* = \emptyset$, the trivial win conditions have not occurred. Then we see insulated regions: $\{0\}$ is the first insulated region, $\{2, 3, 4\}$ is the second insulated region, etc. We compute $\mathcal{T}^* \cup \mathcal{K}^* = \{1, 3, 4, 6, 8, 9\}$, so $\neg(\mathcal{T}^* \cup \mathcal{K}^*) = \{0, 2, 5, 7\}$: using FW-Find we know this is the set of left firewalls, and the right firewalls are $\{0, 4, 5, 8\}$.

Epoch	$\{0\}$	1	$\{2$	3	4$\}$	$\{5\}$	6	$\{7$	$\tilde{\mathsf{e}}\}$	9
Key	×	k_1	×	×	×	×	k_6	×	×	k_9
Token	×	×	Δ_3	Δ_4	×	×	×	Δ_8	×	
Challenge ciphertexts	×	×	×	×	×	×	×	$\tilde{\mathrm{C}}_7$	$\tilde{\mathrm{C}}_8$	×
Challenge input	×	$\bar{\mathrm{C}}$	$\bar{\mathrm{C}}_2$	$\bar{\mathrm{C}}_3$	$\bar{\mathrm{C}}_4$	×	×	×	×	×

Figure 16: An example of trivial win conditions and insulated regions incurred by an adversary playing detIND-UE*-CPA, where × indicates the keys/tokens/ciphertexts not revealed to the adversary, and $\{\}$ indicates insulated regions.

4 On the Security of Updates

In this section we present a new notion of security for updatable encryption schemes, which we denote xxIND-UE-atk. This notion captures both security of fresh encryptions (i.e. implies xxIND-ENC-atk) and unlinkability (i.e. implies xxIND-UPD-atk). We first explain the new notion and then describe its relation to previous notions. Then, we prove a generic relationship among CPA, CTXT and CCA to complete the picture for security notions for UE schemes.

4.1 A New Definition of Confidentiality

In the security game for xxIND-UE-atk, the adversary submits one message and a ciphertext from an earlier epoch that the adversary received via a call to $\mathcal{O}.\mathsf{Enc}$. The challenger responds with either an encryption of that message or an update of that earlier ciphertext, in the challenge (current) epoch $\tilde{\mathsf{e}}$.

Definition 8 (xxIND-UE-atk)**.** Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. Then the xxIND-UE-atk advantage, for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$, of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}$ is given in Fig. 6, Fig. 8 and Fig. 17.

Note that randIND-UE-CPA is strictly stronger than detIND-UE-CPA, since the adversary has strictly more capabilities. A generalized version of xxIND-UE-atk, denoted xxIND-UE*-atk, is also given in Fig. 17. In this game the input challenge ciphertext can come from (i.e. be known to $\mathcal{A}$ in) any prior epoch, not just the epoch immediately before $\tilde{\mathsf{e}}$. Note that xxIND-UE-atk is a special case of xxIND-UE*-atk. Under some fairly weak requirements (that all schemes discussed in this paper satisfy) we can prove that xxIND-UE-atk implies xxIND-UE*-atk – we prove this result in Section. 4.2.1.

$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : $(\bar{\mathrm{M}}, \bar{\mathrm{C}}) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$
3 : **if** $(\bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1) \notin \mathcal{L}$
4 : **return** $\perp$
5 : $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},0} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
6 : $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},1} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$
7 : **if** $|\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},0}| \neq |\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},1}|$
8 : **return** $\perp$
9 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},\mathrm{b}}$

$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE^*\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1 : $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}')) \leftarrow \mathcal{A}$
2 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$
3 : **if** $(\bar{\mathrm{C}}, \mathsf{e}') \notin \mathcal{L}$
4 : **return** $\perp$
5 : $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},0} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
6 : $\tilde{\mathrm{C}}_{\mathsf{e}',1} \leftarrow \bar{\mathrm{C}}$
7 : **for** $j \in \{\mathsf{e}'+1, \ldots, \tilde{\mathsf{e}}\}$ **do**
8 : $\tilde{\mathrm{C}}_{j,1} \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{\mathrm{C}}_{j-1,1})$
9 : **if** $|\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},0}| \neq |\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},1}|$
10 : **return** $\perp$
11 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}},\mathrm{b}}$

Figure 17: Challenge call definition for xxIND-UE-atk and xxIND-UE*-atk security experiments; the full experiment is defined in Fig. 6 and Fig. 8.

Remark 3. The definition of xxIND-UE-atk is more concise and intuitively easier to understand than that of xxIND-UE*-atk, however in Theorem 2.1 and Theorem 2.2 in Section 4.2.1 we show that xxIND-UE-atk $\iff$ xxIND-UE*-atk. This result and our generic proof techniques mean that all results in this paper that hold for xxIND-UE-atk, also hold for xxIND-UE*-atk, and vice versa.

Remark 4. In Section C.1 we show that the RISE scheme presented by LT18 is randIND-UE-CPA secure under DDH. While this result is perhaps unsurprising[9], the proof techniques we use are novel and may be of

[9] LT18 had already shown that RISE is IND-ENC-CPA and randIND-UPD-CPA, so our result shows that it is stronger than was previously known.

independent interest. We give an Oracle-DDH-like game that inherits the epoch-based nature of the updatable encryption security model, and then use this as a bridge to prove security.

4.2 Relations among Security Notions

In Fig. 18 we show the relationship between the new and existing UE security notions. Note that our new notion is strictly stronger than the xxIND-ENC-atk and xxIND-UPD-atk notions presented in prior work, and is in fact stronger than the combination of the prior notions. Further, we show that the generic relation among CPA, CTXT and CCA, that CPA security coupled with ciphertext integrity implies CCA security, also holds for updatable encryption schemes. The relationships are proven via Theorem 2.2 to 2.8, and Theorem 3, which follow next.

Theorem 2 (Informal Theorem)**.** The relationship among the security notions xxIND-UE-atk, xxIND-ENC-atk and xxIND-UPD-atk are as in Fig. 18. This is proven via Theorem 2.2 to 2.8, and Theorem 3.

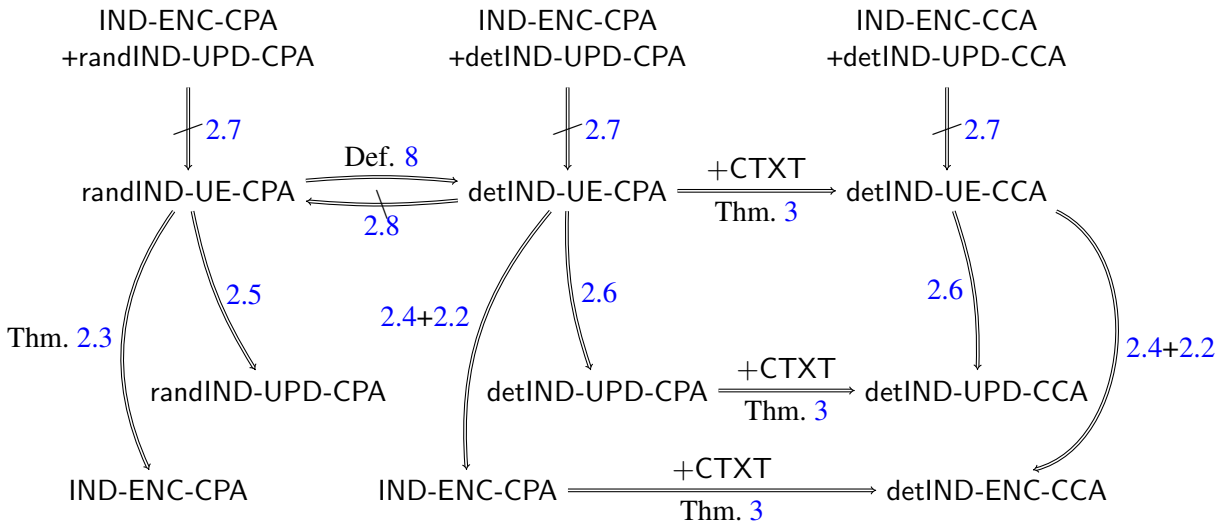

Figure 18: Relations among confidentiality notions xxIND-yy-atk for $\mathsf{xx} \in \{\mathsf{det}, \mathsf{rand}\}$, $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$, $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, and ciphertext integrity (INT-CTXT). Arrow labelling refers to Theorem numbering except where otherwise specified.

4.2.1 Relations between IND-UE and IND-UE*

Properties of Deterministic Updates Here we will use an an alternative representation of UE.Enc that specifies a deterministic algorithm with randomness as input, i.e. $\mathrm{C_e} \leftarrow \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M}; r)$.

One of our main contributions is a scheme with a deterministic update mechanism – we now discuss some of the properties of such schemes. The first two properties, simulatable token generation and randomness-preserving updates, were introduced by Klooß et al. [KLR19a]. Simulatable token generation states that the real token looks like a token generated from a token simulation algorithm, as we consider bi-directional updates we omit the generation of the reverse token. Randomness-preserving states that the update of a ciphertext looks like an encryption of the same message, with the same randomness, under the new key.

Definition 9 (Simulatable token [KLR19a])**.** Let UE be an updatable encryption scheme. We say that UE has simulatable token generation if it has the following property: There is a PPT algorithm $\mathsf{SimTG}(\lambda)$ which samples a token Δ. Furthermore, for arbitrary (fixed) $\mathrm{k_{old}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$ following distributions of Δ are identical:

- $\{\Delta \mid \Delta \xleftarrow{\$} \mathsf{SimTG}(\lambda)\}$
- $\{\Delta \mid \mathrm{k_{new}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda), \Delta \leftarrow \mathsf{UE.TG}(\mathrm{k_{old}}, \mathrm{k_{new}})\}$

Notice that BLMR, BLMR+, RISE, SHINE all have simulatable token generation. Furthermore, the simulatable token generation algorithms of these UE schemes generates a token by randomly picking a token from the token space, i.e. $\mathsf{SimTG} : \Delta \xleftarrow{\$} \mathcal{TS}$.

Definition 10 (Randomness-preserving [KLR19a])**.** Let UE be an updatable encryption scheme. We say that UE.Upd (for UE) is randomness-preserving if the following holds: First, as usually assumed, UE encrypts with uniformly chosen randomness. Second, all keys $(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}}) \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, tokens $\Delta^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}})$, plaintext m and randomness r, we have

$$\mathsf{UE.Upd}(\Delta^{\mathsf{new}}, \mathrm{C}^{\mathsf{old}}) = \mathsf{UE.Enc}(\mathrm{k}^{\mathsf{new}}, m; r),$$

where $\mathrm{C}^{\mathsf{old}} = \mathsf{UE.Enc}(\mathrm{k}^{\mathsf{old}}, m; r)$.

Suppose $\mathrm{C}_i = \mathsf{UE.Enc}(\mathrm{k}_i, m; r)$, and C_j is an update of C_i from epoch i to epoch j. Randomness-preserving property makes sure that $\mathrm{C}_j = \mathsf{UE.Enc}(\mathrm{k}_j, m; r)$, which means a (updated) ciphertext under some epoch key is uniquely decided by the message and randomness.

We define an even weaker property which we call update-preserving: any update sequence starting and ending at the same key that starts with the same ciphertext will result in the same ciphertext.

Definition 11 (Update-preserving)**.** Let UE be an updatable encryption scheme with deterministic update algorithm. We say that UE.Upd (for UE) is update-preserving if the following holds: for any two sequence of key pairs with the same start key and end key $(\mathrm{k}_i, \mathrm{k}_{i+1}, \ldots, \mathrm{k}_j)$ and $(\mathrm{k}'_i, \mathrm{k}'_{i+1}, \ldots, \mathrm{k}'_j)$, where $\mathrm{k}_i (= \mathrm{k}'_i), \mathrm{k}_{i+1}, \mathrm{k}'_{i+1}, \ldots, \mathrm{k}_{j-1}, \mathrm{k}'_{j-1}, \mathrm{k}_j (= \mathrm{k}'_j) \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, and tokens $\Delta_l \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}_{l-1}, \mathrm{k}_l)$, $\Delta'_l \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}'_{l-1}, \mathrm{k}'_l)$, for any ciphertext C_i in epoch i, we have $\mathrm{C}_j = \mathrm{C}'_j$ where $\mathrm{C}'_i = \mathrm{C}_i, \mathrm{C}_l = \mathsf{UE.Upd}(\Delta_l, \mathrm{C}_{l-1})$, $\mathrm{C}'_l = \mathsf{UE.Upd}(\Delta'_l, \mathrm{C}'_{l-1})$ for $l = i+1, \ldots, j$.

The diagrams in Fig. 19 show how the property works, with the left-hand side indicating keys and tokens and the right-hand side showing ciphertexts.

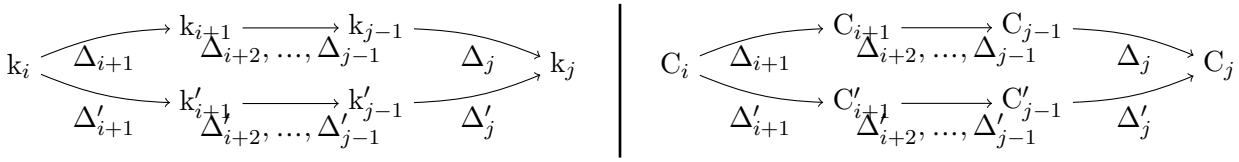

Figure 19: Keys and updated ciphertexts in Definition 11

Update-preserving property implies that the updated ciphertext is uniquely determined by $(\mathrm{C}_i, k_j, j\text{-}i)$, where C_i is the beginning ciphertext for updating, j-i decides how many updates have occurred, and k_j decides the value of the ending epoch's epoch key.

We now define another property which states that ciphertexts encrypted under one key can be simulated by ciphertexts encrypted under another key. All schemes in this paper meet this property.

Definition 12 (Simulatable Encryption)**.** Let UE be an updatable encryption scheme. For all keys $\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$, tokens $\Delta^{\mathsf{new}} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}})$, plaintext m, define X^{old}, X^{new} to be the statistical distribution of the ciphertexts output by $\mathsf{UE.Enc}(\mathrm{k}^{\mathsf{old}}, m)$, $\mathsf{UE.Enc}(\mathrm{k}^{\mathsf{new}}, m)$, resp.. We say that UE has simulatable encryption if update algorithm keeps the ciphertext distribution, i.e. $\mathsf{UE.Upd}(\Delta^{\mathsf{new}}, X^{\mathsf{old}}) \overset{dist}{=} X^{\mathsf{new}}$.

Note that we do not restrict that the update algorithm is probabilistic. This means when the update algorithm is deterministic, it will not add randomness to the updated ciphertext, and it maintains the ciphertext distribution. For example, suppose $U(\mathbb{Z})$ is a uniform distribution over $\mathbb{Z}$, and for any integer Δ, let $\mathsf{UE.Upd}(\Delta, x) = x + \Delta$, then $\mathsf{UE.Upd}(\Delta, U(\mathbb{Z})) = U(\mathbb{Z})$. This definition looks similar to the definition of perfect re-encryption provided by Klooß et al. [KLR19a], which mandates that update has the same distribution as decrypt-then-encrypt. Perfect re-encryption requires the update algorithm is probabilistic, which makes it possible for any updated ciphertext looks like a fresh encryption. The simulatable encryption property is to make sure that the update algorithm can keep the distribution of encryption – however it is not necessary to require that the update algorithm is probabilistic.

All schemes discussed in this paper satisfy all of the above properties. Note that randomness-preserving property is strictly stronger than the update-preserving property and simulatable encryption. Obviously, if a scheme is randomness-preserving then it is also update-preserving and has simulatable encryption. However, the update-preserving property does not imply randomness-preserving property, even with simulatable encryption. To see this, construct a deterministic update variant of the RISE scheme (Section C) such that the randomness r updates to $r+2$: this scheme has the update-preserving property and simulatable encryption, but not the randomness-preserving property.

xxIND-UE-atk implies xxIND-UE*-atk. We prove xxIND-UE-atk implies xxIND-UE*-atk in this section, and consequently xxIND-UE-atk and xxIND-UE*-atk are equivalent.

randIND-UE-CPA implies randIND-UE*-CPA.

Theorem 2.1. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any randIND-UE*-CPA adversary $\mathcal{A}$ against UE, there exists an randIND-UE-CPA adversary $\mathcal{B}_{2.1}$ against UE such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE^*\text{-}CPA}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.1}}(\lambda).$$

Proof. We construct a reduction $\mathcal{B}_{2.1}$: before the epoch counter is incremented, every ciphertext is updated using the available update oracles. This needs to happen when the adversary moves to the next epoch, so that it is always possible to provide a valid challenge input to the reducton's own randIND-UE-CPA challenger and respond with a valid challenge output to the adversary.

More precisely, when the adversary makes the randIND-UE*-CPA challenge query, the reduction make its own randIND-UE-CPA query, submitting the ciphertext provided by the adversary but updated to the epoch one before the challenge epoch that both algorithms are in. This should give the exact same result as updating the older ciphertext. Consequently, and since all other oracle queries can just be forwarded, the reduction perfectly simulates the randIND-UE*-CPA game. We have the required result. □

detIND-UE-atk implies detIND-UE*-atk.

Proof technique of Theorem 2.2. The proof uses the firewall technique, where the reduction will 'pause' its own epoch continuum while responding to the adversary's queries. The main left firewall in the detIND-UE*-atk game is an epoch in which the detIND-UE-atk reduction can possibly ask for a valid challenge query. Before the left firewall, the reduction sends the queries received from the adversary $\mathcal{A}$ to its own detIND-UE-atk challenger, and forwards responses to $\mathcal{A}$. Within the firewalls, the reduction stops asking any $\mathcal{O}$.Next queries, and instead simulates the responses of each query to provide answers to $\mathcal{A}$. Because of this action, the detIND-UE-atk challenger will stay in epoch $\hat{\mathsf{fwl}}$. When $\mathcal{A}$ makes the detIND-UE*-atk challenge queries (if the trivial win conditions of the detIND-UE*-atk game are not satisfied then the trivial win conditions of the detIND-UE-atk game will be not satisfied as well), the reduction makes its detIND-UE-atk query using the old ciphertext (in $\hat{\mathsf{fwl}} - 1$) instead. After receiving the response, the reduction updates its challenge ciphertext to the challenge epoch to reply to $\mathcal{A}$. After the right firewall, the query responses are calculated and forwarded, in the same manner as before the left firewall.

Theorem 2.2. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme has simulatable token generation, update-preserving property and simulatable encryption property. For any detIND-UE*-atk adversary $\mathcal{A}$ against UE, where $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, there exists an detIND-UE-atk adversary $\mathcal{B}_{2.2}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE^*\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq (n+1)^2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.2}}(\lambda),$$

Proof. We use three steps to prove this result.

(Step 1.) Consider a modified version of detIND-UE*-atk. For $\mathrm{b} \in \{0,1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ to be the same as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE^*\text{-}atk\text{-}b}}$ except that the experiments randomly pick $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$, and if $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are not the firewalls around challenge epoch $\tilde{\mathrm{e}}$, then the experiment returns a random bit b'. More formally, the

values $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are the desired firewalls if the challenge is made inside – i.e. $\tilde{\mathsf{e}} \in [\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}]$ – and they actually consitute an insulate region, i.e. $(, \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}) \in \mathcal{FW}$)).

These firewalls $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ could take any value in $\{0, \ldots, n\}$, so this loss is upper bounded by $(n+1)^2$. We have

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE^*atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq (n+1)^2 \mathbf{Adv}^{\mathsf{INT}_1}_{\mathsf{UE},\mathcal{A}}.$$

(Step 2.) Then we consider experiments $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$, which is the same as $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$ except for: in the insulated region all encryptions are updated ciphertexts of ciphertexts encrypted in left firewall $\hat{\mathsf{fwl}}$ By this we mean that if the adversary asks for any $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Dec}$ and challenge query, the responses work as follows:

- $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$: if called in an epoch $\hat{\mathsf{fwl}} < \mathsf{e} \leq \hat{\mathsf{fwr}}$, encrypt the message in left firewall $\hat{\mathsf{fwl}}$, then update the ciphertext to epoch e, and return the updated ciphertext.
- $\mathcal{O}.\mathsf{Dec}(\mathrm{C})$: if called in an epoch $\hat{\mathsf{fwl}} < \mathsf{e} \leq \hat{\mathsf{fwr}}$, reverse update the ciphertext from e back to $\hat{\mathsf{fwl}}$, then decrypt the updated ciphertext, and return the decrypted value.
- challenge query, on input $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$: if $\mathrm{b} = 0$, encrypt the message $\bar{\mathrm{M}}$ in left firewall $\hat{\mathsf{fwl}}$, then update the ciphertext to the challenge epoch $\tilde{\mathsf{e}}$; if $\mathrm{b} = 1$, update ciphertext $\bar{\mathrm{C}}$ from epoch e' to epoch $\tilde{\mathsf{e}}$. Return the challenge ciphertext.

Since we assume that UE has the simulatable encryption property, both operations are possible so we have

$$\mathbf{Adv}^{\mathsf{INT}_1}_{\mathsf{UE},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE},\mathcal{A}}(\lambda).$$

(Step 3.) We construct a reduction $\mathcal{B}_{2.2}$, detailed in Fig. 20 and Fig. 21, that is playing the $\mathsf{detIND\text{-}UE\text{-}atk}$ game and runs $\mathcal{A}$. We claim that

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\mathcal{B}_{2.2}}(\lambda).$$

If $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}$ are the desired firewalls, then $[\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] \subseteq \mathcal{C}^*$. If the trivial win conditions in $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$ are not set (the same result as the trivial win conditions in $\mathbf{Exp}^{\mathsf{detIND\text{-}UE^*\text{-}atk\text{-}b}}$), i.e. $\mathcal{I}^* \cap \mathcal{C}^* = \emptyset$, then $\mathcal{I}^* \cap [\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] = \emptyset$. That means $\mathcal{A}$ never asks $\mathcal{O}.\mathsf{Upd}(\bar{\mathrm{C}})$ and $\mathcal{O}.\mathsf{Corr}(\mathsf{token})$ in $\hat{\mathsf{fwl}}$. So the reduction uses the relevant challenge input to ask a challenge query to its own $\mathsf{detIND\text{-}UE\text{-}atk}$ challenger in epoch $\hat{\mathsf{fwl}}$, and it will not trivially lose.

Before the epoch counter is incremented, every ciphertext is updated using the available update oracles. This needs to happen when the adversary moves to the next epoch, so that it is always possible to provide a valid challenge input to the reducton's own $\mathsf{detIND\text{-}UE\text{-}atk}$ challenger and respond with a valid challenge output to the adversary.

Within the firewalls, the reduction simulates all ciphertexts and uses the list $\mathcal{FL}$ and the list $\tilde{\mathcal{FL}}$ to track non-challenge ciphertexts and challenge-equal ciphertexts, respectively. When the challenge query happens with input $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$, the reduction can find all updated versions of $\bar{\mathrm{C}}$ by checking the first entry of the list $\mathcal{L}$. The reduction uses the ciphertext in epoch $\hat{\mathsf{fwl}}$-1 with the same query identifier c as a challenge input, sending to its own $\mathsf{detIND\text{-}UE\text{-}atk}$ challenger. (Note that $\mathsf{e}' < \hat{\mathsf{fwl}}$, otherwise, $[\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}] \subseteq \mathcal{I}^*$ and the trivial win condition is triggered.) After receiving the response from the $\mathsf{detIND\text{-}UE\text{-}atk}$ challenger, $\mathcal{B}_{2.2}$ updates the received ciphertext to the challenge epoch to reply $\mathcal{A}$.

Eventually $\mathcal{B}_{2.2}$ receives b' from $\mathcal{A}$, and simply outputs b' to its $\mathsf{detIND\text{-}UE\text{-}atk}$ challenger. When $\mathcal{B}_{2.2}$ interacts with $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk\text{-}b}}_{\mathsf{UE},\mathcal{B}_{2.2}}$, $\mathcal{B}_{2.2}$ can perfectly simulate $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}_{\mathsf{UE},\mathcal{A}}$ to $\mathcal{A}$. Then we have the required result.

□

4.2.2 Relations among IND-ENC, IND-UPD, and IND-UE

In this section, we analyze the relations among notions with different challenge input. Since similar proof techniques are used in Theorems 2.3, 2.4, 2.5 and 2.6, of these we give full proof details only for Theorem 2.3.

Theorem 2.3. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any $\mathsf{IND\text{-}ENC\text{-}CPA}$ adversary $\mathcal{A}$ against UE, there exists an $\mathsf{randIND\text{-}UE\text{-}CPA}$ adversary $\mathcal{B}_{2.3}$ against UE such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{UE},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\mathcal{B}_{2.3}}(\lambda).$$

Reduction $\mathcal{B}_{2.2}$ playing $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk}\text{-}b}_{\mathsf{UE},\,\mathcal{B}_{2.2}}$

$\mathbf{do\ Setup}; \mathcal{FL}, \tilde{\mathcal{FL}}, \mathcal{L}, \tilde{\mathcal{L}} \leftarrow \emptyset$
$\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \ldots, n\}$
$\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}') \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
$\mathsf{phase} \leftarrow 1$
Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$, $\mathbf{get}\ \tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
$\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},(\mathcal{O}.\mathsf{Dec}),\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
$\mathsf{twf} \leftarrow 1\ \mathbf{if}$
  $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ $\mathbf{or}$ $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
$\mathbf{if}$ ABORT occurred $\mathbf{or}$ $(\cdot, \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}) \notin \mathcal{FW}$ $\mathbf{or}$ $\mathsf{twf} = 1$
  $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
$\mathbf{return}\ \mathrm{b}'$

Figure 20: Reduction $\mathcal{B}_{2.2}$ for proof of Theorem 2.2. The adversary may call for $\mathcal{O}$.Dec in the CCA game. The oracles in Fig. 21 show how $\mathcal{B}_{2.2}$ responds to $\mathcal{A}$, including calls to oracles in its own detIND-UE-atk game.

Proof. We construct a reduction $\mathcal{B}_{2.3}$ running the randIND-UE-CPA experiment which will simulate the responses of queries made by the IND-ENC-CPA adversary $\mathcal{A}$. To provide a valid non-challenge ciphertext to its own challenger, $\mathcal{B}_{2.3}$ must run $\mathcal{A}$ out of step with its own game, so epoch 0 as far as $\mathcal{A}$ is concerned is actually epoch 1 for $\mathcal{B}_{2.3}$, and so on.

1. $\mathcal{B}_{2.3}$ chooses $\mathrm{b} \xleftarrow{\$} \{0, 1\}$.

2. $\mathcal{B}_{2.3}$ receives the setup parameters from its randIND-UE-CPA challenger, chooses $\mathrm{M} \xleftarrow{\$} \mathcal{MS}$ and calls $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ which returns some C_0. Then $\mathcal{B}_{2.3}$ calls $\mathcal{O}$.Next once and sends the setup parameters to $\mathcal{A}$.

3. (a) Whenever $\mathcal{B}_{2.3}$ receives the queries $\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}$ from $\mathcal{A}$, $\mathcal{B}_{2.3}$ sends these queries to its randIND-UE-CPA challenger, and forwards the responses to $\mathcal{A}$.

   (b) Whenever $\mathcal{O}$.Next is called by $\mathcal{A}$, $\mathcal{B}_{2.3}$ randomly chooses a message $\mathrm{M} \xleftarrow{\$} \mathcal{MS}$ and calls $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ to receive some C_{e}, and then calls $\mathcal{O}$.Next.

4. At some point, in epoch $\tilde{\mathsf{e}}$ (for its game), $\mathcal{B}_{2.3}$ receives the challenge query $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$ from $\mathcal{A}$. Then $\mathcal{B}_{2.3}$ sends $(\bar{\mathrm{M}}_{\mathrm{b}}, \mathrm{C}_{\tilde{\mathsf{e}}-1})$ as challenge to its own randIND-UE-CPA challenger. After receiving the challenge ciphertext, $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$, from its challenger, $\mathcal{B}_{2.3}$ sends $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ to $\mathcal{A}$.

5. $\mathcal{B}_{2.3}$ continues to answer $\mathcal{A}$'s queries using its own oracles, now including $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$.

6. Finally $\mathcal{B}_{2.3}$ receives the output bit b' from $\mathcal{A}$. If $\mathrm{b} = \mathrm{b}'$ then $\mathcal{B}_{2.3}$ returns 0. Otherwise $\mathcal{B}_{2.3}$ returns 1.

We now bound the advantage of $\mathcal{B}_{2.3}$. The point is that whenever $\mathcal{B}_{2.3}$ returns a random encryption to $\mathcal{A}$, $\mathcal{B}_{2.3}$'s probability of winning is exactly 1/2 because the bit b' from $\mathcal{A}$ is independent of its choice of b. This happens with probability 1/2. However, when $\mathcal{B}_{2.3}$ returns a "correct" value to $\mathcal{A}$ (an encryption of $\bar{\mathrm{M}}_0$ or $\bar{\mathrm{M}}_1$), then $\mathcal{B}_{2.3}$'s probability of winning is the same as the probability that $\mathcal{A}$ wins.

First note that, as usual,

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\mathcal{B}_{2.3}} = |\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1]|.$$

We claim that $\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1] = 1/2$ because in this case $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ is independent of b and so b' must

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

1 : $\mathsf{c} \leftarrow \mathsf{c}+1$
2 : **if** $\mathsf{e} \notin \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
3 : **call** $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, **get** $\mathrm{C_e}$
4 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
5 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
6 : **call** $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, **get** $\mathrm{C}_{\hat{\mathsf{fwl}}}$
7 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\hat{\mathsf{fwl}}}, \hat{\mathsf{fwl}})\}$
8 : **for** $j \in \{\hat{\mathsf{fwl}}+1, \ldots, \mathsf{e}\}$ **do**
9 : $\mathrm{C}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathrm{C}_{j-1})$
10 : $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
11 : **return** $\mathrm{C_e}$

$\mathcal{O}.\mathsf{Next}$

12 : **if** $\mathsf{e} \in \{1, \ldots, \hat{\mathsf{fwl}}\text{-}1\}$ **then**
13 : **for** $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \in \mathcal{L}$ **do**
14 : **call** $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$, **get** $\mathrm{C_e}$
15 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
16 : **call** $\mathcal{O}.\mathsf{Next}$
17 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwr}}+1, \ldots, n\}$ **then**
18 : **call** $\mathcal{O}.\mathsf{Next}$
19 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\text{-}1\}$ **then**
20 : $\mathsf{e} \leftarrow \mathsf{e}+1$
21 : $\Delta_\mathsf{e} \xleftarrow{\$} \mathsf{SimTG}(\lambda)$
22 : **if** $\mathsf{e} = \hat{\mathsf{fwr}}$ **then**
23 : **for** $j \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}+1\}$ **do**
24 : **call** $\mathcal{O}.\mathsf{Next}$
25 : **for** $(\mathsf{c}, \mathrm{C}_{j\text{-}1}, j\text{-}1) \in \mathcal{L}$ **do**
26 : **call** $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{j\text{-}1})$
27 : **get** C_j
28 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_j, j)\}$
29 : **for** $(\tilde{\mathrm{C}}_{j\text{-}1}, j\text{-}1) \in \tilde{\mathcal{L}}$ **do**
30 : **call** $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}(\tilde{\mathrm{C}}_{j\text{-}1})$
31 : **get** $\tilde{\mathrm{C}}_j$
32 : $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_j, j)\}$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$

33 : **if** $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \notin \mathcal{L} \cup \mathcal{FL}$
34 : **return** $\perp$
35 : **if** $\mathsf{e} \in \{1, \ldots, \hat{\mathsf{fwl}}\}$
36 : **call** $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$, **get** $\mathrm{C_e}$
37 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
38 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwr}}+2, \ldots, n\}$
39 : **call** $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$, **get** $\mathrm{C_e}$
40 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
41 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
42 : $\mathrm{C_e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathrm{C}_{\mathsf{e}-1})$
43 : $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e})\}$
44 : **if** $\mathsf{e} = \hat{\mathsf{fwr}}+1$
45 : **find** $(\mathsf{c}, \mathrm{C_e}, \mathsf{e}) \in \mathcal{L}$
46 : **return** $\mathrm{C_e}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

47 : **do** $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}})$
48 : **if** $\mathsf{inp} = \mathsf{key}$
49 : $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
50 : **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
51 : **if** $\mathsf{inp} = \mathsf{token}$
52 : $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
53 : **if** $\hat{\mathsf{e}} \in \{1, \ldots, \hat{\mathsf{fwl}}\text{-}1\}$
54 : **call** $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
55 : **get** $\Delta_{\hat{\mathsf{e}}}$
56 : **if** $\{\hat{\mathsf{fwr}}+2, \ldots, n\}$
57 : **call** $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$
58 : **get** $\Delta_{\hat{\mathsf{e}}}$
59 : **if** $\hat{\mathsf{e}} \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
60 : **find** $\Delta_{\hat{\mathsf{e}}}$
61 : **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (\bar{\mathrm{C}}, \mathsf{e}'))$

62 : **if** $\tilde{\mathsf{e}} \notin \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\}$
63 : ABORT
64 : **if** $(\mathsf{c}, \bar{\mathrm{C}}, \mathsf{e}') \notin \mathcal{L}$
65 : ABORT
66 : **find** $(\mathsf{c}, \mathrm{C}_{\hat{\mathsf{fwl}}-1}, \hat{\mathsf{fwl}}-1) \in \mathcal{L}$
67 : **call cq** $(\bar{\mathrm{M}}, \mathrm{C}_{\hat{\mathsf{fwl}}-1})$
68 : **get** $\tilde{\mathrm{C}}_{\hat{\mathsf{fwl}}}$
69 : $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\hat{\mathsf{fwl}}}, \hat{\mathsf{fwl}})\}$
70 : **for** $j \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$ **do**
71 : $\tilde{\mathrm{C}}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{\mathrm{C}}_{j-1})$
72 : $\tilde{\mathcal{FL}} \leftarrow \tilde{\mathcal{FL}} \cup \{(\tilde{\mathrm{C}}_j, j)\}$
73 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

74 : **if** $\mathsf{e} \in \{1, \ldots, \hat{\mathsf{fwl}}\text{-}1\}$
75 : **return** $\perp$
76 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
77 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}\}$
78 : **find** $(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}$
79 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
80 : **find** $(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{FL}}$
81 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwr}}+1, \ldots, n\}$
82 : **call** $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, **get** $\tilde{\mathrm{C}}_\mathsf{e}$
83 : **return** $\tilde{\mathrm{C}}_\mathsf{e}$

$\mathcal{O}.\mathsf{Dec}(\mathrm{C})$

84 : **if** $\mathsf{phase} \leftarrow 1$ **and** $\mathrm{C} \in \tilde{\mathcal{L}}^*$
85 : $\mathsf{twf} \leftarrow 1$
86 : **return** $\perp$
87 : **if** $\mathsf{e} \notin \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
88 : **call** $\mathcal{O}.\mathsf{Dec}(\mathrm{C})$, **get** $\mathrm{M}'/\perp$
89 : **if** $\mathsf{e} \in \{\hat{\mathsf{fwl}}+1, \ldots, \hat{\mathsf{fwr}}\}$
90 : **for** $j \in \{\mathsf{e}, \ldots, \hat{\mathsf{fwl}}+1\}$ **do**
91 : $\mathrm{C}_{j-1} \leftarrow \mathsf{UE.Upd}^{-1}(\Delta_j, \mathrm{C}_j)$
92 : **call** $\mathcal{O}.\mathsf{Dec}(\mathrm{C}_{\hat{\mathsf{fwl}}})$, **get** $\mathrm{M}'/\perp$
93 : **return** M' **or** $\perp$

Figure 21: Oracles used in the proof of Theorem 2.2. In line 67 of the Create $\tilde{\mathrm{C}}$ description, **call cq** means that the reduction makes its own challenge query with these values.

also be independent of b. Then we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\mathcal{B}_{2.3}} &= \left|\frac{1}{2} - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{B}_{2.3}} = 1]\right| \\
&= \left|\frac{1}{2} - \left(\frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1] + \frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 0]\right)\right| \\
&= \left|\frac{1}{2} - \frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1] - \frac{1}{2}\left(1 - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 1]\right)\right| \\
&= \left|\frac{1}{2}\cdot\left(\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}0}}_{\mathsf{UE},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}1}}_{\mathsf{UE},\,\mathcal{A}} = 1]\right)\right| \\
&= \frac{1}{2}\cdot\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{UE},\,\mathcal{A}}.
\end{aligned}$$

□

A remark on Theorem 2.4. Directly proving that detIND-UE-atk implies detIND-ENC-atk is very challenging, and in fact the difficulty is the same as proving that detIND-UE-atk implies detIND-UE*-atk. Since we have proved detIND-UE-atk implies detIND-UE*-atk in Theorem 2.2 in Section 4.2.1, we do not repeat the similar proof approach here. We can just prove detIND-UE*-atk implies detIND-ENC-atk, which is easy. We follow a very similar approach to the proof of Theorem 2.3. The detIND-UE*-atk (reduction) adversary can ask for tokens almost as freely as the detIND-ENC-atk adversary without incurring the trivial win conditions. But since there should at least one token, in an epoch before (include) challenge epoch $\tilde{\mathrm{e}}$, is unknown to the adversary, we again have to run our reduction out of step with the detIND-ENC-atk adversary (essentially creating an artificial epoch).

Theorem 2.4. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any detIND-ENC-atk adversary $\mathcal{A}$ against UE, where $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, there exists a detIND-UE*-atk adversary $\mathcal{B}_{2.4}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}ENC\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2\cdot\mathbf{Adv}^{\mathsf{detIND\text{-}UE^*\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.4}}(\lambda).$$

Proof. Similar to the proof strategy in Theorem 2.3, we construct a detIND-UE*-atk adversary $\mathcal{B}_{2.4}$ against UE to simulate the responses to queries made by detIND-ENC-atk adversary $\mathcal{A}$. $\mathcal{B}_{2.4}$ chooses $\mathrm{b} \xleftarrow{\$} \{0,1\}$. Since $\mathcal{B}_{2.4}$ is allowed to takes its challenge ciphertext from any epoch in its detIND-UE*-atk experiment, it can in particular uses the random ciphertext created in epoch 0, C_0. Note that, in contrast to Step 3 (b) of the simulation in the proof of Theorem 2.3, there is now no need for $\mathcal{B}_{2.4}$ to generate a random ciphertext for each epoch. Also $\mathcal{B}_{2.4}$ will never ask for $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_0)$ to its own detIND-UE*-atk challenger.

When receiving the challenge query $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$ from $\mathcal{A}$, the reduction $\mathcal{B}_{2.4}$ sends $(\bar{\mathrm{M}}_\mathrm{b}, \mathrm{C}_0)$ as challenge to its own detIND-UE*-atk challenger. Since $\mathcal{A}$ has no view of epoch 0, $\mathcal{A}$ cannot ask for Δ_1. In addition, $\mathcal{A}$ will never ask for $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_0)$.

Thus the trivial win condition $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ will not be satisfied in the detIND-UE*-atk game. The other trivial win(s), i.e. trivial win via keys and ciphertexts (and trivial wins via decryptions), will also pass from detIND-ENC-atk game to detIND-UE*-atk game. The result follows using the same calculation as in the proof of Theorem 2.3.

□

Theorem 2.5. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any randIND-UPD-CPA adversary $\mathcal{A}$ against UE, there exists an randIND-UE-CPA adversary $\mathcal{B}_{2.5}$ against UE such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UPD\text{-}CPA}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2\cdot\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.5}}(\lambda).$$

Proof. Similarly to the proof of Theorem 2.3, we construct a randIND-UE-CPA reduction $\mathcal{B}_{2.5}$ against UE to simulate the responses of queries made by randIND-UPD-CPA adversary $\mathcal{A}$. However in this case it is not necessary for the reduction to be out-of-step with the adversary. $\mathcal{B}_{2.5}$ first chooses $\mathrm{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}_{2.5}$ forwards all queries from $\mathcal{A}$ to its own oracles, and when it receives challenge query $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1)$ from $\mathcal{A}$, $\mathcal{B}_{2.5}$ samples a

random message M, and sends $(\mathrm{M}, \bar{\mathrm{C}}_\mathrm{b})$ to the randIND-UE-CPA challenger. The result follows using a similar calculation to that in the proof of Theorem 2.3.

□

Theorem 2.6. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme For any detIND-UPD-atk adversary $\mathcal{A}$ against UE, where $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, there exists an detIND-UE-atk adversary $\mathcal{B}_{2.6}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UPD\text{-}atk}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\,\mathcal{B}_{2.6}}(\lambda).$$

Proof. The proof follows exactly the same steps as that of Theorem 2.5, in addition to the observation that in the det versions and CCA versions of the games, the trivial win flag twf is either triggered for both adversary and reduction or for neither when $\mathcal{O}$.Corr queries are made by the underlying adversary. □

Theorem 2.7. Each of the following hold for $(\mathsf{xx}, \mathsf{atk}) \in \{(\mathsf{det}, \mathsf{CPA}), (\mathsf{rand}, \mathsf{CPA}), (\mathsf{det}, \mathsf{CCA})\}$.
(i). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{ENC} be the xxIND-ENC-atk advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE′ such that $\mathcal{A}$'s xxIND-ENC-atk advantage against UE′ is (still) α_{ENC}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE′ with advantage 1.
(ii). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{UPD} be the xxIND-UPD-atk advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE′ such that $\mathcal{A}$'s xxIND-UPD-atk advantage against UE′ is (still) α_{UPD}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE′ with advantage 1.
(iii). Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme and let α_{ENC} be the xxIND-ENC-atk advantage of an adversary $\mathcal{A}_{\mathsf{ENC}}$ against UE and α_{UPD} be the xxIND-UPD-atk advantage of an adversary . Then there exists a modified scheme UE′ such that $\mathcal{A}_{\mathsf{ENC}}$'s xxIND-ENC-atk advantage against UE′ is (still) α_{ENC}, $\mathcal{A}_{\mathsf{UPD}}$'s xxIND-UPD-atk advantage against UE′ is (still) α_{UPD}, and there exists an xxIND-UE-atk adversary $\mathcal{B}$ against UE′ with advantage 1.

Proof. All three are demonstrated using the same counterexample. All algorithms for UE′ are the same as for UE, except UE′.Enc is defined by modifying UE.Enc to append 0, and UE′.Upd is defined by modifying UE.Upd to append 1. Hence, freshly encrypted ciphertexts and updated ciphertexts are distinguishable. This does not affect an adversary's ability to win the xxIND-ENC-atk or xxIND-UPD-atk games but trivially breaks xxIND-UE-atk security. □

A remark on Theorem 2.8. We construct an updatable encryption scheme which is detIND-UE-CPA secure but not randIND-UE-CPA secure to prove that detIND-UE-CPA does not imply randIND-UE-CPA. Note that in Section 5 we will prove that SHINE is detIND-UE-CPA secure (yet it is trivially not randIND-UE-CPA secure), which provides an example to support this result. However the proof that SHINE is detIND-UE-CPA in the ideal cipher model (if DDH holds). Here we demonstrate the theorem based on a weaker assumption, namely the existence of pseudorandom functions.

Proof technique of Theorem 2.8. We use a xxIND-UE-CPA secure UE scheme to construct a new UE scheme $\mathsf{UE}^{\mathsf{new}}$, where we use a PRF to make a part of the new update algorithm deterministic. Because of this $\mathsf{UE}^{\mathsf{new}}$ will not be randIND-UE-CPA secure. In order to bound the detIND-UE-CPA security of $\mathsf{UE}^{\mathsf{new}}$, we need to make sure the newly added deterministic part of the updates will not make $\mathsf{Enc}(m)$ and $\mathsf{Upd}(\mathrm{C})$ distinguishable – this is where we need the PRF.

Theorem 2.8. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme, and define a new updatable encryption scheme $\mathsf{UE}^{\mathsf{new}}$ in Fig. 22, built using pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. Then, for any detIND-UE-CPA adversary $\mathcal{A}$ against $\mathsf{UE}^{\mathsf{new}}$ that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists a PRF adversary $\mathcal{B}^{\mathsf{PRF}}$ against F and a detIND-UE-CPA adversary $\mathcal{B}_{2.8}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}(\lambda) \leq (n+1) \cdot (\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.8}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{F,\,\mathcal{B}^{\mathsf{PRF}}} + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}),$$

and there exists a randIND-UE-CPA adversary $\mathcal{C}$ against $\mathsf{UE}^{\mathsf{new}}$ that wins with probability 1.

$\mathsf{UE}^{\mathsf{new}}.\mathsf{KG}(\lambda)$

1 : $\mathrm{k} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
2 : **return** k

$\mathsf{UE}^{\mathsf{new}}.\mathsf{TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$

3 : $\Delta'_{\mathsf{e}+1} \leftarrow \mathsf{UE.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$
4 : $\mathrm{fk_{e+1}} \xleftarrow{\$} \mathcal{K}$
5 : **return** $(\Delta'_{\mathsf{e}+1}, \mathrm{fk_{e+1}})$

$\mathsf{UE}^{\mathsf{new}}.\mathsf{Enc}(\mathrm{k_e}, \mathrm{M})$

6 : $r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
7 : $\mathrm{C}'_{\mathsf{e}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k_e}, \mathrm{M})$
8 : **return** $(r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$

$\mathsf{UE}^{\mathsf{new}}.\mathsf{Dec}(\mathrm{k_e}, \mathrm{C_e})$

9 : **parse** $\mathrm{C_e} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
10 : $\mathrm{M}'/\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C}'_{\mathsf{e}})$
11 : **return** M'

$\mathsf{UE}^{\mathsf{new}}.\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathrm{C_e})$

12 : **parse** $\Delta_{\mathsf{e}+1} = (\Delta'_{\mathsf{e}+1}, \mathrm{fk_{e+1}})$
13 : **parse** $\mathrm{C_e} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
14 : $r_{\mathsf{e}+1} \leftarrow F(\mathrm{fk_{e+1}}, r_{\mathsf{e}})$
15 : $\mathrm{C}'_{\mathsf{e}+1} \leftarrow \mathsf{UE.Upd}(\Delta'_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}})$
16 : **return** $(r_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}+1})$

Figure 22: Updatable encryption scheme $\mathsf{UE}^{\mathsf{new}}$ for proof of Theorem 2.8, built from PRF F and updatable encryption scheme UE.

Proof. $\mathsf{UE}^{\mathsf{new}}$ is not randIND-UE-CPA secure. If the token in the challenge epoch is corrupted then the adversary can compare r in the value it receives with the value it provided and therefore trivially win. So $\mathsf{UE}^{\mathsf{new}}$ is not randIND-UE-CPA secure.

$\mathsf{UE}^{\mathsf{new}}$ is detIND-UE-CPA secure. We proceed in three steps.

(Step 1.) Consider a modified version of detIND-UE-CPA. For $\mathrm{b} \in \{0,1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ to be the same as $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}$ except that the experiments randomly pick $\mathsf{e}^* \leftarrow \{0, \ldots, n\}$, and if $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ the experiments return a random bit for b'. The loss is upper bounded by $n+1$. Then:

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}(\lambda) \leq (n+1) \cdot \mathbf{Adv}^{\mathsf{INT_1}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}(\lambda).$$

(Step 2.) Then we consider modified experiments $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$, which are the same as $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ except that the first element of ciphertexts in the guessed epoch e^* is a uniformly random element. We show that the ability to notice this change is upper bounded by PRF advantage. More precisely, experiment $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$ tracks randomness in the set X (initialized as empty), and when adversary asks an $\mathcal{O}.\mathsf{Upd}$ or challenge query:

- For $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}^*\text{-}1})$: parse $\Delta_{\mathsf{e}^*} = (\Delta'_{\mathsf{e}^*},)$, parse $\mathrm{C}_{\mathsf{e}^*\text{-}1} = (r_{\mathsf{e}^*\text{-}1}, \mathrm{C}'_{\mathsf{e}^*\text{-}1})$, if $r_{\mathsf{e}^*\text{-}1} \in X$, then the experiment aborts; otherwise, set $X \leftarrow X \cup \{r_{\mathsf{e}^*\text{-}1}\}$, randomly choose $r_{\mathsf{e}^*} \xleftarrow{\$} \mathcal{X}$. Return $(r_{\mathsf{e}^*}, \mathsf{UE.Upd}(\Delta'_{\mathsf{e}^*}, \mathrm{C}'_{\mathsf{e}^*\text{-}1}))$.
- For challenge input $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$: parse $\Delta_{\mathsf{e}^*} = (\Delta'_{\mathsf{e}^*},)$, if $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ or $r \in X$, then the experiment aborts; otherwise, set $X \leftarrow X \cup \{r\}$. If $\mathrm{b} = 0$, return $\mathsf{UE}^{\mathsf{new}}.\mathsf{Enc}(\mathrm{k}_{\mathsf{e}^*}, \bar{\mathrm{M}})$. If $\mathrm{b} = 1$, randomly choose $r_{\mathsf{e}^*} \xleftarrow{\$} \mathcal{X}$, return $(r_{\mathsf{e}^*}, \mathsf{UE.Upd}(\Delta'_{\mathsf{e}^*}, \bar{\mathrm{C}}))$.

Note that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{INT_1}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}(\lambda) &= \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_1\text{-}1}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_1\text{-}0}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1]\right| \\
&\leq \mathbf{Adv}^{\mathsf{INT_2}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}(\lambda) + \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_2\text{-}1}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_1\text{-}1}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1]\right| \\
&\quad + \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_2\text{-}0}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_1\text{-}0}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1]\right|
\end{aligned}$$

For $\mathrm{b} \in \{0,1\}$, we wish to prove that

$$\left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}} = 1]\right| \leq \mathbf{Adv}^{\mathsf{PRF}}_{F} + \frac{\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}$ is an adversary trying to distinguish $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}$ from $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}_{\mathsf{UE}^{\mathsf{new}},\mathcal{A}}$. We construct a PRF reduction $\mathcal{B}^{\mathsf{PRF}}$, detailed in Fig. 23, against F to simulate the responses of queries made by $\mathcal{A}$. $\mathcal{B}^{\mathsf{PRF}}$ first guesses when $\mathcal{A}$ is going to ask a challenge query (assume e^*) and in that epoch $\mathcal{B}^{\mathsf{PRF}}$ does bookkeeping for the randomness in X (initialized as empty set). Note that the reduction generates all keys and tokens except for $\mathrm{fk}_{\mathsf{e}^*}$. Update randomness in epoch e^* is simulated by sending the randomness to the PRF challenger and forwarding the response to $\mathcal{A}$.

Reduction $\mathcal{B}^{\mathsf{PRF}}$ playing $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{F,\,\mathcal{B}^{\mathsf{PRF}}}$

$\mathbf{do\ Setup}$
$\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
$\mathsf{phase} \leftarrow 1$
Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$, $\mathbf{get}\ \tilde{\mathrm{C}}_{\mathsf{e}^*}$
$\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\mathsf{e}^*})$
$\mathsf{twf} \leftarrow 1$ **if**
  $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
if ABORT occurred **or** $\mathsf{twf} = 1$
  $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
  return b'
if $\mathrm{b}' = \mathrm{b}$
  return 0
else
  return 1

Setup(λ)

$\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda)$
$\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
$\mathsf{e}^* \xleftarrow{\$} \{0, \ldots, n\}$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Next}^{\dagger}$

$\mathcal{O}.\mathsf{Upd}(r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1})$

if $(\cdot, (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}), \mathsf{e}-1) \notin \mathcal{L}$
  return $\perp$
if $\mathsf{e} \neq \mathsf{e}^*$
  $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{UE}^{\mathsf{new}}.\mathsf{Upd}((\Delta'_{\mathsf{e}}, \mathrm{fk}_{\mathsf{e}}), (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}))$
if $\mathsf{e} = \mathsf{e}^*$
  if $r_{\mathsf{e}-1} \in X$
    return ABORT
  $X \leftarrow X \cup \{r_{\mathsf{e}-1}\}$
  $r_{\mathsf{e}} \leftarrow \mathcal{O}.f(r_{\mathsf{e}-1})$ // embed
  $\mathrm{C}'_{\mathsf{e}} \leftarrow \mathsf{UE.Upd}(\Delta'_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}-1})$
  $\mathrm{C}_{\mathsf{e}} \leftarrow (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
return C_{e}

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$

if $(\cdot, (r, \bar{\mathrm{C}}), \tilde{\mathsf{e}}-1) \notin \mathcal{L}$ **or** $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ **or** $r \in X$
  return $\perp$
$X \leftarrow X \cup \{r\}$
if $\mathrm{b} = 0$
  $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE}^{\mathsf{new}}.\mathsf{Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}})$
else
  $r_{\tilde{\mathsf{e}}} \leftarrow \mathcal{O}.f(r)$ // embed
  $\tilde{\mathrm{C}}'_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta'_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$
  $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow (r_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}'_{\tilde{\mathsf{e}}})$
return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

Figure 23: Reduction $\mathcal{B}^{\mathsf{PRF}}$ for proof of Theorem 2.8, $\mathcal{B}^{\mathsf{PRF}}$ simulates $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Corr}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ queries as described in Fig. 6. Recall that the PRF advantage in Definition 3, $\mathcal{O}.f$ replies with $\mathsf{F}(\mathrm{k}, r)$ or a random value. † indicates $\mathrm{fk}_{\mathsf{e}^*}$ are skipped in the generation.

Eventually $\mathcal{B}^{\mathsf{PRF}}$ receives the guess from $\mathcal{A}$, and outputs 0 if $\mathcal{A}$ guesses that it is in $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$, and 1 if $\mathcal{A}$ guesses that it is in $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$.

When $\mathcal{B}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}^{\mathsf{PRF\text{-}0}}$, it can simulate $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$ perfectly except with a negligible probability. The negligible term is due to $\mathcal{B}^{\mathsf{PRF}}$ aborts the game. Since the number of existed randomnesses is small compared to the number of possible random elements, the probability that $\mathcal{B}^{\mathsf{PRF}}$ aborts the game is upper bounded by $\frac{\mathrm{Q_E}^2}{|\mathcal{X}|}$. When $\mathcal{B}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}^{\mathsf{PRF\text{-}1}}$, it can perfectly simulate $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$, thus we have the required result.

(Step 3.) Now we conclude that the advantage of winning INT_2 is upper bounded by detIND-UE-CPA advantage (against UE). Suppose an INT_2 adversary $\mathcal{A}$ is trying to attack $\mathsf{UE}^{\mathsf{new}}$. We construct a detIND-UE-CPA reduction $\mathcal{B}_{2.8}$, detailed in Fig. 24, attacking UE and runs $\mathcal{A}$. $\mathcal{B}_{2.8}$ first guesses when $\mathcal{A}$ is going to ask a challenge query (assume e^*) and in that epoch $\mathcal{B}_{2.8}$ does bookkeeping for the randomness in X (initialized as empty set).

Eventually $\mathcal{B}_{2.8}$ receives b' from $\mathcal{A}$, and simply outputs b'. Then $\mathcal{B}_{2.8}$ perfectly simulates $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}$ to $\mathcal{A}$. We have the required result

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.8}}(\lambda).$$

Reduction $\mathcal{B}_{2.8}$ playing $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{2.8}}$

1 : **do Setup**
2 : $\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 : $\mathsf{phase} \leftarrow 1$
4 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$, **get** $\tilde{\mathrm{C}}_{\mathsf{e}^*}$
5 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\mathsf{e}^*})$
6 : $\mathsf{twf} \leftarrow 1$ **if**
7 : $\quad \mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
8 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
9 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
10 : **return** b'

Setup(λ)

11 : $\mathrm{k}_0 \leftarrow \mathsf{UE.KG}(\lambda)$
12 : $\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
13 : $\mathsf{e}^* \xleftarrow{\$} \{0, \ldots, n\}$
14 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

15 : **call** $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$, **get** C'
16 : $r \xleftarrow{\$} \mathcal{X}$
17 : **return** (r, C')

$\mathcal{O}.\mathsf{Next}$

18 : **call** $\mathcal{O}.\mathsf{Next}$
19 : $\mathrm{fk}_{\mathsf{e}+1} \xleftarrow{\$} \mathcal{K}$
20 : **if** $\mathsf{phase} = 1$
21 : $\quad \tilde{r}_{\mathsf{e}+1} = F(\mathrm{fk}_{\mathsf{e}+1}, \tilde{r}_{\mathsf{e}})$
22 : $\quad \tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\mathsf{e}+1}, \cdot), \mathsf{e}+1)\}$

$\mathcal{O}.\mathsf{Upd}((r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}))$

23 : **if** $(\cdot, (r_{\mathsf{e}-1}, \mathrm{C}'_{\mathsf{e}-1}), \mathsf{e}-1) \notin \mathcal{L}$
24 : $\quad$ **return** $\perp$
25 : **call** $\mathcal{O}.\mathsf{Upd}(\mathrm{C}'_{\mathsf{e}-1})$, **get** C'_{e}
26 : **if** $\mathsf{e} \neq \mathsf{e}^*$
27 : $\quad r_{\mathsf{e}} = F(\mathrm{fk}_{\mathsf{e}}, r)$
28 : **if** $\mathsf{e} = \mathsf{e}^*$
29 : $\quad$ **if** $r_{\mathsf{e}-1} \in X$
30 : $\quad\quad$ **return** ABORT
31 : $\quad X \leftarrow X \cup \{r_{\mathsf{e}-1}\}$
32 : $\quad r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
33 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}}), \mathsf{e})\}$
34 : **return** $(r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

35 : **call** $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$, **get** $\perp$ **or** $\mathrm{k}_{\hat{\mathsf{e}}}$ **or** $\Delta'_{\hat{\mathsf{e}}}$
36 : **return** $\perp$ **or** $\mathrm{k}_{\hat{\mathsf{e}}}$ **or** $(\Delta'_{\hat{\mathsf{e}}}, \mathrm{fk}_{\hat{\mathsf{e}}})$

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, (r, \bar{\mathrm{C}}))$

37 : **if** $(\cdot, (r, \bar{\mathrm{C}}), \tilde{\mathsf{e}}-1) \notin \mathcal{L}$ **or** $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ **or** $r \in X$
38 : $\quad$ **return** $\perp$
39 : $X \leftarrow X \cup \{r\}$
40 : **call cq with** $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, **get** $\tilde{\mathrm{C}}'$ // embed
41 : $\tilde{r}_{\tilde{\mathsf{e}}} \xleftarrow{\$} \mathcal{X}$
42 : $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}'), \tilde{\mathsf{e}})\}$
43 : **return** $(\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{\mathrm{C}}')$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

44 : **call** $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, **get** $\tilde{\mathrm{C}}'_{\mathsf{e}}$ // embed
45 : $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\mathsf{e}}, \tilde{\mathrm{C}}'_{\mathsf{e}}), \mathsf{e})\}$
46 : **return** $(\tilde{r}_{\mathsf{e}}, \tilde{\mathrm{C}}'_{\mathsf{e}})$

Figure 24: Reduction $\mathcal{B}_{2.8}$ for proof of Theorem 2.8. In line 40 of the Create $\tilde{\mathrm{C}}$ description, **call cq** means that the reduction makes its own challenge query with these values.

□

4.2.3 Relation among CPA, CTXT and CCA Security

Theorem 3. Let $\mathsf{UE} = \{\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ be an updatable encryption scheme. For any detIND-yy-CCA adversary $\mathcal{A}$ against UE, there exists an INT-CTXT adversary $\mathcal{B}_{3a}$ and an detIND-yy-CPA adversary $\mathcal{B}_{3b}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2\mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{B}_{3a}}(\lambda) + \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{3b}}(\lambda)$$

where $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$.

We now prove Theorem 3, which states that the combination of detIND-yy-CPA security and INT-CTXT security yields detIND-yy-CCA, for $\mathsf{yy} \in \{\mathsf{UE}, \mathsf{ENC}, \mathsf{UPD}\}$. The proof proceeds via a single game hop.

Proof. **Game 0**

The first game is the experiment $\mathbf{Exp}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\,\mathcal{A}}$, given in Fig. 11 (or Fig. 12 or Fig. 17). From Def. 5 (or Def. 6 or Def. 8) we have

$$\mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CCA}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) = 2\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - 1/2\Big|.$$

Game 1

In this game we introduce an event bad that is triggered if the adversary asks its decryption oracle for something that would count as a forgery, and then show that the success probability of a distinguisher between the two games is bounded by INT-CTXT. Then, we bound the success probability in this modified game by an adversary against detIND-yy-CCA. We modify $\mathcal{O}$.Dec, such that the boxed statements only run in Game 1:

$\mathcal{O}.\mathsf{Dec}(\mathrm{C})$

1 : **if** phase $\leftarrow 1$ **and** $\mathrm{C} \in \tilde{\mathcal{L}}^*$
2 : **return** $\perp$
3 : $\mathrm{M}/\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C})$
4 : $\boxed{\textbf{if } \mathrm{C} \notin \mathcal{L}^* \textbf{ and } \mathrm{M}' \neq \perp}$
5 : $\boxed{\mathsf{bad} \leftarrow \mathsf{true}}$
6 : $\boxed{\textbf{return } \perp}$

We have

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - 1/2\Big| \leq \Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| + \Big|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\Big|$$

We claim that there exists an INT-CTXT adversary $\mathcal{B}_{3a}$ against UE such that

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| \leq \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{B}_{3a}}(\lambda),$$

and there exists an detIND-yy-CPA adversary $\mathcal{B}_{3b}$ such that

$$2\Big|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\Big| \leq \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{3b}}(\lambda).$$

Claim 1.

$$\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| = \mathbf{Pr}[\mathsf{bad} = \mathsf{true} \text{ in } \mathcal{G}_1] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{B}_{3a}}(\lambda).$$

We construct a reduction $\mathcal{B}_{3a}$ playing INT-CTXT that simulates the environment of $\mathcal{G}_1$ to $\mathcal{A}$. $\mathcal{B}_{3a}$ starts by picking a random bit b, then runs $\mathcal{A}$ answering its queries as follows. For a challenge query with input ($\tilde{\mathrm{C}}_0$ or $\bar{\mathrm{M}}_0, \tilde{\mathrm{C}}_1$ or $\bar{\mathrm{M}}_1$). If b =0, $\mathcal{B}_{3a}$ sends $\tilde{\mathrm{C}}_0$ (or $\bar{\mathrm{M}}_0$) to its $\mathcal{O}$.Upd (or $\mathcal{O}$.Enc); b =1, $\mathcal{B}_{3a}$ sends $\tilde{\mathrm{C}}_1$ (or $\bar{\mathrm{M}}_1$) to its $\mathcal{O}$.Upd (or $\mathcal{O}$.Enc); eventually, returns the response to $\mathcal{A}$. Furthermore, $\mathcal{B}_{3a}$ simulates $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ by sending the challenge ciphertext to $\mathcal{O}$.Upd, and forwards the response to $\mathcal{A}$.

For a decryption query $\mathcal{O}$.Dec with input C: If $\mathrm{C} \in \mathcal{L}^*$, $\mathcal{B}_{3a}$ checks the corresponding message in list $\mathcal{L}^*$ (adversary $\mathcal{B}_{3a}$ does bookkeeping for this list and additionally stores message in this list, list is updated by $\mathcal{O}$.Enc, $\mathcal{O}$.Upd, $\mathcal{O}$.Corr. Hence this simulation is feasible by an INT-CTXT adversary.), and returns it to $\mathcal{A}$. If $\mathrm{C} \notin \mathcal{L}^*$, $\mathcal{B}_{3a}$ returns $\perp$ to $\mathcal{A}$, and sends C to its $\mathcal{O}$.Try oracle.

$\mathcal{B}_{3a}$ perfectly simulates $\mathcal{G}_1$. Notice that $\mathcal{G}_0$ and $\mathcal{G}_1$ are identical until $\mathsf{UE.Dec}(\mathrm{k_e}, \mathrm{C}) \neq \perp$ and $\mathrm{C} \notin \mathcal{L}^*$ happens (which causes $\mathsf{bad} = \mathsf{true}$ in $\mathcal{G}_1$): denote this event to be E. Thus, we have $\Big|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]\Big| = \mathbf{Pr}[\mathsf{bad} = \mathsf{true} \text{ in } \mathcal{G}_1] = \mathbf{Pr}[E]$. If event E happens, which results in $\mathsf{win} = 1$ in the INT-CTXT game, that means C is a valid forgery in INT-CTXT game. So $\mathbf{Pr}[E] = \mathbf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{UE},\,\mathcal{B}_{3a}}(\lambda)$.

Claim 2.

$$2\Big|\mathbf{Pr}[\mathcal{G}_1 = 1] - 1/2\Big| \leq \mathbf{Adv}^{\mathsf{detIND\text{-}yy\text{-}CPA}}_{\mathsf{UE},\,\mathcal{B}_{3b}}(\lambda).$$

We only need to consider how the detIND-yy-CPA adversary $\mathcal{B}_{3b}$ simulates the $\mathcal{O}$.Dec oracle. Since $\mathcal{B}_{3b}$ knows $\tilde{\mathcal{L}}^*$ and $\mathcal{L}^*$, whenever $\mathcal{A}$ asks $\mathcal{O}$.Dec with C, $\mathcal{B}_{3b}$ checks if $\mathrm{C} \in \tilde{\mathcal{L}}^*$ or $\mathrm{C} \notin \mathcal{L}^*$, and if so, responds $\perp$.

Otherwise, $\mathcal{B}_{3b}$ checks the corresponding message in list $\mathcal{L}^*$ (adversary $\mathcal{B}_{3b}$ does bookkeeping for this list and additionally stores message in this list as well, list is updated by $\mathcal{O}$.Enc, $\mathcal{O}$.Upd, $\mathcal{O}$.Corr. Hence this simulation is feasible by an detIND-yy-CPA adversary), and returns it to $\mathcal{A}$.

□

5 The SHINE Schemes

We now describe our new UE scheme SHINE (Secure Homomorphic Ideal-cipher Nonce-based Encryption). The encryption algorithm uses a permutation to obfuscate the input to the exponentiation function. Updating a ciphertext simply requires exponentiation once by the update token, which itself is the quotient of the current epoch key and the previous epoch key. The scheme comes in three flavors: SHINE0 is presented in Fig. 25 and takes in short messages and only uses a single permutation. The second flavor, MirrorSHINE, is provided in Fig. 26 and runs two different permutations with the same input. The third flavor OCBSHINE is given in Fig. 28 and is for applications with arbitrarily long messages, using a family of permutations.

We discuss implementation details of the SHINE schemes in Section 5.6. In particular, for each scheme in the SHINE suite, it is necessary to embed the output of the permutation (a regular block cipher) into an appropriate DDH-hard group.

Our proofs of security, given as Theorem 4 for confidentiality and Theorem 5 for integrity, bound an adversary's detIND-UE-CPA (INT-CTXT$^{\mathsf{s}}$) advantage by DDH (CDH), and are provided in the ideal cipher model. Furthermore, combining the results of Theorem 3, Theorem 4 and Theorem 5, we have that the suite of SHINE schemes (i.e. SHINE0, MirrorSHINE and OCBSHINE) are detIND-UE-CCA secure.

5.1 Construction of SHINE Schemes

5.1.1 SHINE via Zero Block: SHINE0.

Suppose a message space of $\mathcal{MS} = \{0,1\}^{\mathrm{m}}$ and random nonce space $\mathcal{N} = \{0,1\}^{\mathrm{v}}$. The encryption algorithm feeds as input to the permutation a nonce, the message, and a zero string. The decryption algorithm will return $\perp$ if the decrypted value does not end with 0^{t}. The SHINE0 scheme is defined in Fig. 25. If ciphertext integrity is not required (or file/ciphertext integrity is performed in some other manner), then SHINE0 without the zero block results in a scheme (denoted SHINE0[CPA]) that is still detIND-UE-CPA secure.

SHINE0.KG(λ)

$\mathrm{k} \xleftarrow{\$} \mathbb{Z}_q^*$

return k

SHINE0.TG($\mathrm{k_e}, \mathrm{k_{e+1}}$)

$\Delta_{\mathrm{e+1}} \leftarrow \dfrac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$

return $\Delta_{\mathrm{e+1}}$

SHINE0.Enc($\mathrm{k_e}, \mathrm{M}$)

$\mathrm{N} \xleftarrow{\$} \mathcal{N}$

$\mathrm{C_e} \leftarrow (\pi(\mathrm{N}\|\mathrm{M}\|0^{\mathrm{t}}))^{\mathrm{k_e}}$

return $\mathrm{C_e}$

SHINE0.Dec($\mathrm{k_e}, \mathrm{C_e}$)

$a \leftarrow \pi^{-1}(\mathrm{C_e}^{1/\mathrm{k_e}})$

parse† a **as** $\mathrm{N}'\|\mathrm{M}'\|\mathrm{Z}$

if $\mathrm{Z} = 0^{\mathrm{t}}$

  return M'

else

  return $\perp$

SHINE0.Upd($\Delta_{\mathrm{e+1}}, \mathrm{C_e}$)

$\mathrm{C_{e+1}} \leftarrow (\mathrm{C_e})^{\Delta_{\mathrm{e+1}}}$

return $\mathrm{C_{e+1}}$

Figure 25: Updatable encryption scheme SHINE0. Note that there may be an additional embedding step after the permutation π, as discussed in Section 5.6. †: $\|\mathrm{N}'\| = \mathrm{v}, \|\mathrm{M}'\| = \mathrm{m}, \|\mathrm{Z}\| = \mathrm{t}$.

5.1.2 SHINE via Double Encryption: MirrorSHINE

The construction of MirrorSHINE is similar to SHINE0, except that instead of padding a zero block after the message, the encryption algorithm processes each message and nonce twice using two different random permutations. For authentication, we compare if two ciphertext blocks have the same underlying message and nonce. The idea of the authentication is similar to SHINE0, however here we use the difference of the underlying message and nonce as the "zero block". Compared to SHINE0, MirrorSHINE does one more exponentiation in both encryption and update and requires two ciphertext elements per message, but larger messages are supported (when using the same – for example standardized – group).

$\underline{\mathsf{MirrorSHINE.KG}(\lambda)}$

$\mathrm{k} \xleftarrow{\$} \mathbb{Z}_q^*$

return k

$\underline{\mathsf{MirrorSHINE.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})}$

$\Delta_{\mathrm{e+1}} \leftarrow \dfrac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$

return $\Delta_{\mathrm{e+1}}$

$\underline{\mathsf{MirrorSHINE.Enc}(\mathrm{k_e}, \mathrm{M})}$

$\mathrm{N} \xleftarrow{\$} \mathcal{N}$

$\mathrm{C_e^1} \leftarrow (\pi_1(\mathrm{N}||\mathrm{M}))^{\mathrm{k_e}}$

$\mathrm{C_e^2} \leftarrow (\pi_2(\mathrm{N}||\mathrm{M}))^{\mathrm{k_e}}$

$\mathrm{C_e} \leftarrow (\mathrm{C_e^1}, \mathrm{C_e^2})$

return $\mathrm{C_e}$

$\underline{\mathsf{MirrorSHINE.Dec}(\mathrm{k_e}, \mathrm{C_e})}$

parse $\mathrm{C_e} = (\mathrm{C_e^1}, \mathrm{C_e^2})$

$a_1 \leftarrow \pi_1^{-1}((\mathrm{C_e^1})^{1/\mathrm{k_e}})$

$a_2 \leftarrow \pi_2^{-1}((\mathrm{C_e^2})^{1/\mathrm{k_e}})$

parse$^{\ddagger} a_1$ **as** $\mathrm{N}'||\mathrm{M}'$

if $a_1 = a_2$

  return M'

else

  return $\perp$

$\underline{\mathsf{MirrorSHINE.Upd}(\Delta_{\mathrm{e+1}}, \mathrm{C_e})}$

parse $\mathrm{C_e} = (\mathrm{C_e^1}, \mathrm{C_e^2})$

$\mathrm{C_{e+1}^1} \leftarrow (\mathrm{C_e^1})^{\Delta_{\mathrm{e+1}}}$

$\mathrm{C_{e+1}^2} \leftarrow (\mathrm{C_e^2})^{\Delta_{\mathrm{e+1}}}$

return $\mathrm{C_{e+1}^1}, \mathrm{C_{e+1}^2}$

Figure 26: Updatable encryption scheme MirrorSHINE, where π_1, π_2 are two different random permutations. Note that there may be an additional embedding step after the permutations π_1 and π_2, as discussed in Section 5.6. ‡: $\|\mathrm{N}'\| = \mathrm{v}, \|\mathrm{M}'\| = \mathrm{m}$.

5.1.3 SHINE for Long Messages via Checksum: OCBSHINE.

The schemes SHINE0 and MirrorSHINE both require that the message space be smaller than the size of an element of the exponentiation group. This ciphertext expansion is undesirable in many practical scenarios, and so we wish to construct a SHINE scheme which gives us (almost) no ciphertext expansion and can be applied to arbitrarily long messages. We build a new SHINE scheme, OCBSHINE, with these properties.

The construction of OCBSHINE is inspired by the authenticated encryption scheme OCB [RBBK01]. Different from OCB mode, the nonce is encrypted inside the ciphertext instead of sending it along with the ciphertext. In order to determine the length of the last message block, the encryption algorithm of OCB mode removes some bits of the last ciphertext block to reveal this information. However in our setting, the output of the permutations are (mapped to) the input of the exponentiation function: thus all bits of permutation outputs must be included. Therefore, OCBSHINE includes the length of the last message block in the first ciphertext component. If ciphertext integrity is not required, then OCBSHINE can be improved by removing the last ciphertext block.

OCBSHINE is formally defined in Fig. 28 and the encryption process is pictorially represented in Fig. 27; we give an intuitive description here. Suppose the blocksize is m, and assume the encryption algorithm OCBSHINE.Enc has input message M. By "partition M into $\mathrm{M}_1, \ldots, \mathrm{M}_l$" we mean setting $l \leftarrow \max\{\lceil |\mathrm{M}|/m \rceil, 1\}$ and dividing M into l blocks, i.e. $\mathrm{M}_1, \ldots, \mathrm{M}_l$, where $|\mathrm{M}_1| = \ldots = |\mathrm{M}_{l-1}| = m$. The last message block M_l is padded with zeros to make it length m before computing the permutation output and the checksum, i.e $\mathrm{M}_l\|0^*$ with $|\mathrm{M}_l\|0^*| = m$. Let $a = \lceil \log(m) \rceil$, so the length of M_l ($|\mathrm{M}_l| \leq m$) can be written as an a-bit representation.

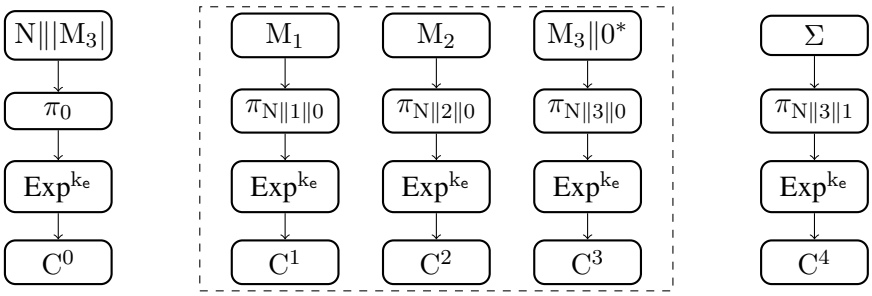

Figure 27: Diagram describing how the OCBSHINE encryption algorithm works on message $\mathrm{M} = (\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3)$. $\Sigma = \mathrm{M}_1 \oplus \mathrm{M}_2 \oplus \mathrm{M}_3 \| 0^*$. There may be an additional embedding step after the permutations, as discussed in Section 5.6.

Let $\mathrm{Perm}(m)$ be the set of all permutations on $\{0,1\}^m$. Randomly choose $\pi_0 \xleftarrow{\$} \mathrm{Perm}(m)$, and use this permutation to randomize the concatenation of the nonce N and an a-bit representation of the last message block length. Then, index the (random) permutations used to encrypt message blocks by the nonce and a counter. Let $\mathrm{Perm}(S, m)$ be the set of all mappings from S to permutations on $\{0,1\}^m$. Suppose the nonce space is $\mathcal{N} = \{0,1\}^{m-a}$, $S = \mathcal{N} \times \mathbb{N}^* \times \{0,1\}$, for each $(\mathrm{N} \in \mathcal{N}, i \in \mathbb{N}^*, b \in \{0,1\})$, set $\pi_{\mathrm{N}\|i\|b} \xleftarrow{\$} \mathrm{Perm}(\mathcal{N} \times \mathbb{N}^* \times \{0,1\}, m)$, which form a random permutation family: we use these permutations to randomize message blocks and the checksum.

OCBSHINE.KG(λ)

1 : $\mathrm{k} \xleftarrow{\$} \mathbb{Z}_q^*$
2 : **return** k

OCBSHINE.TG($\mathrm{k_e}, \mathrm{k_{e+1}}$)

3 : $\Delta_{\mathrm{e}+1} \leftarrow \dfrac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$
4 : **return** $\Delta_{\mathrm{e}+1}$

OCBSHINE.Enc($\mathrm{k_e}, \mathrm{M}$)

5 : partition M into $\mathrm{M}_1, ..., \mathrm{M}_l$
6 : $\Sigma \leftarrow \oplus_{i=1}^{l-1} \mathrm{M}_i \oplus \mathrm{M}_l \| 0^*$
7 : $\mathrm{N} \xleftarrow{\$} \mathcal{N}$
8 : $\mathrm{C}^0 \leftarrow \left(\pi_0(\mathrm{N}\|\|\mathrm{M}_l|)\right)^{\mathrm{k_e}}$
9 : $\mathrm{C}^{l+1} \leftarrow \left(\pi_{\mathrm{N}\|l\|1}(\Sigma)\right)^{\mathrm{k_e}}$
10 : **for** $i = 1, ..., l$-1 **do**
11 : $\quad \mathrm{C}^i \leftarrow \left(\pi_{\mathrm{N}\|i\|0}(\mathrm{M}_i)\right)^{\mathrm{k_e}}$
12 : $\mathrm{C}^l \leftarrow \left(\pi_{\mathrm{N}\|l\|0}(\mathrm{M}_l\|0^*)\right)^{\mathrm{k_e}}$
13 : $\mathrm{C_e} \leftarrow (\mathrm{C}^0, ..., \mathrm{C}^l, \mathrm{C}^{l+1})$
14 : **return** $\mathrm{C_e}$

OCBSHINE.Dec($\mathrm{k_e}, \mathrm{C_e}$)

15 : **parse** $\mathrm{C_e} = (\mathrm{C}^0, ..., \mathrm{C}^l, \mathrm{C}^{l+1})$
16 : $\mathrm{N}'\|A' \leftarrow \pi_0^{-1}\left(\left(\mathrm{C}^0\right)^{1/\mathrm{k_e}}\right)$
17 : $\Sigma' \leftarrow \pi_{\mathrm{N}'\|l\|1}^{-1}\left(\left(\mathrm{C}^{l+1}\right)^{1/\mathrm{k_e}}\right)$
18 : **for** $i = 1, ..., l$ **do**
19 : $\quad \mathrm{M}'_i \leftarrow \pi_{\mathrm{N}'\|i\|0}^{-1}\left(\left(\mathrm{C}^i\right)^{1/\mathrm{k_e}}\right)$
20 : **if** $\Sigma' = \oplus_{i=1}^{l} \mathrm{M}'_i$
21 : $\quad \mathrm{M}' \leftarrow (\mathrm{M}'_1, ..., \mathrm{M}'_l[x])$
22 : $\quad$ **return** M'
23 : **else**
24 : $\quad$ **return** $\perp$

OCBSHINE.Upd($\Delta_{\mathrm{e}+1}, \mathrm{C_e}$)

25 : **parse** $\mathrm{C_e} = (\mathrm{C}^0, ..., \mathrm{C}^l, \mathrm{C}^{l+1})$
26 : **for** $i = 0, ..., l+1$ **do**
27 : $\quad \mathrm{C}^i_{\mathrm{e}+1} \leftarrow (\mathrm{C}^i_{\mathrm{e}})^{\Delta_{\mathrm{e}+1}}$
28 : **return** $\mathrm{C_{e+1}}$

Figure 28: Updatable encryption scheme OCBSHINE. Note that there may be an additional embedding step after the permutations, as discussed in Section 5.6. In line 21, $\mathrm{M}_l[x]$ represents the first A' bits of M_l.

5.2 Security - SHINE is detIND-UE-CPA, INT-CTXT, detIND-UE-CCA Secure

All three SHINE schemes, i.e. SHINE0, MirrorSHINE and OCBSHINE, have the same security properties, and the proofs are very similar for each flavor. We refer to SHINE to mean the family containing all these three schemes. In Theorem 4, we show that SHINE is detIND-UE-CPA in the ideal cipher model, if DDH holds. In Theorem 5, we show that SHINE is INT-CTXT$^{\mathsf{s}}$, and therefore INT-CTXT (INT-CTXT and INT-CTXT$^{\mathsf{s}}$ are equivalent, recall Section 3), in the ideal cipher model, if CDH holds. The loss incurred by this proof is the normal $(n+1)^3$ (or $(n+1)^2$ for INT-CTXT) and also the number of encryption queries the adversary makes before it makes its challenge: to avoid the issues described in Section 5.3 we not only need to guess the locations of the challenge firewalls but also the ciphertext that the adversary will submit as its challenge.

The ideal cipher model, a version of which was initially given by Shannon [Sha49] and shown to be equivalent to the random oracle model by Coron et al. [CPS08], gives all parties access to a permutation chosen randomly from all possible key-permutation possibilities of appropriate length. The SHINE schemes exponentiate the output of the permutation by the epoch key to encrypt, so our reduction can 'program' the transformation from permutation outputs to group elements.

In the following two Theorems we detail the security properties met by SHINE, i.e. detIND-UE-CPA, INT-CTXT and thus detIND-UE-CCA. Note that this is the strongest known security property for updatable encryption schemes with deterministic updates. In Section 5.3 we discuss the challenges that arise in the proofs of these two theorems, and in Section 5.4 and Section 5.5 we describe the novel techniques and methods used in the proofs.

Theorem 4 (SHINE is detIND-UE-CPA)**.** Let SHINE $\in \{\mathsf{SHINE0}, \mathsf{MirrorSHINE}, \mathsf{OCBSHINE}\}$ be the UE scheme described in Fig. 25 (or Fig. 26 or Fig. 28). For any ideal cipher model adversary $\mathcal{A}$ (that makes max $\mathrm{Q_E}$ encryption queries before its challenge), there exists an adversary $\mathcal{B}_4$ such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{SHINE},\,\mathcal{A}}(\lambda) \leq O(1)(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_4}(\lambda).$$

This is proven via Theorem 4.1, and Theorem 4.3 and Theorem 4.4 in Section 5.4.

Theorem 5 (SHINE is INT-CTXT$^{\mathsf{s}}$)**.** Let SHINE $\in \{\mathsf{SHINE0}, \mathsf{MirrorSHINE}, \mathsf{OCBSHINE}\}$ be the UE scheme described in Fig. 25 (or Fig. 26 or Fig. 28).For any ideal cipher model adversary $\mathcal{A}$ (that makes max $\mathrm{Q_E}$ encryption queries before calling $\mathcal{O}.\mathsf{Try}$), there exists an adversary $\mathcal{B}_5$ such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE},\,\mathcal{A}}(\lambda) \leq O(1)(n+1)^2 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_5} + \text{negligible terms}$$

This is proven via Theorem 5.1, and Theorem 5.3 and Theorem 5.4 in Section 5.5.

Remark 5. Combining the results of Theorem 3, Theorem 4 and Theorem 5, we have that SHINE is detIND-UE-CCA.

Remark 6. Proofs for SHINE0 are extendable to MirrorSHINE and OCBSHINE, so we only show full proof details of Theorem 4.1 and Theorem 5.1 in Section 5.4 and Section 5.5, respectively.

5.3 Proof Challenges in Schemes with Deterministic Updates

In each variant of SHINE all ciphertext components are raised to the epoch key, so the update mechanism transforms a ciphertext for epoch e to one for $\mathsf{e}+1$ by raising this value to $\frac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$. We now highlight the difficulties in creating security proofs for such 'single-component' updatable encryption schemes. Randomness is used in creation of the initial ciphertext (via N) but updates are completely deterministic, and thus in any reduction it is necessary to provide consistent ciphertexts to the adversary (i.e. the N value must be consistent). The (cryptographic) separation gained by using the firewall technique (see Section 3.3 for discussion and definition) assists with producing (updates of) non-challenge ciphertexts, but embedding any challenge value while also providing answers to the $\mathcal{O}.\mathsf{Corr}$ queries of the underlying adversary is very challenging.

The regular key insulation technique as introduced by LT18 – where the reduction constructs one hybrid for each epoch – does not work. Specifically, in any reduction to a DDH-like assumption, it is not possible to provide a challenge ciphertext in a left or right sense (to the left of this challenge ciphertext are of some form, and to the right of this challenge ciphertext are of some other form) if the underlying adversary asks for tokens

around the challenge epoch: deterministic updates mean that tokens will make these ciphertexts of the same form and this gap will be easily distinguishable.

We counteract this problem by constructing a *hybrid argument across insulated regions*. This means that in each hybrid, we can embed at one firewall of the insulated region, and simulate all tokens within that insulated region to enable answering queries to both $\mathcal{O}.\mathsf{Upd}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$. The reduction's distinguishing task is thus ensured to be at the boundaries of the insulated regions, the firewalls, so any (non-trivial) win for the underlying adversary is ensured to carry through directly to the reduction.

5.4 SHINE is detIND-UE-CPA

We now explain how we bound the advantage of any adversary playing the detIND-UE-CPA game for SHINE by the advantage of a reduction playing DDH.

Proof Method for Confidentiality: Constructing a Hybrid Argument across Insulated Regions. Notice that the non-corrupted key space is the union set of all insulated regions, i.e. $\{0, 1, ..., n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j, ..., \mathsf{fwr}_j\}$. If the trivial win conditions are not triggered and the adversary knows a challenge-equal ciphertext in some epoch within an insulated region, then since the adversary knows all tokens in that insulated region, the adversary will know all challenge-equal ciphertexts in that insulated region. Then we have

$$\mathcal{C}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in S\subseteq\mathcal{FW}}\{\mathsf{fwl}_j, ..., \mathsf{fwr}_j\},$$

where S is a subset of firewall list $\mathcal{FW}$.

We apply the firewall technique to set up hybrid games such that in hybrid i, we embed within the i-th insulated region: this means that to the left of the i-th insulated region the game responds with the $\mathrm{b} = 1$ case of the detIND-UE-CPA experiment, and to the right of the i-th insulated region it gives an encryption of the challenge input message as output, i.e. $\mathrm{b} = 0$. This means we have one hybrid for each insulated region, moving left-to-right across the epoch space.

We construct a reduction $\mathcal{B}$ playing the DDH experiment in hybrid i. Initially, $\mathcal{B}$ guesses the location of the i-th insulated region. If the underlying adversary has performed a corrupt query within this insulated region that would lead to the reduction failing, the reduction aborts the game. We use the algorithm Check described in Fig. 29 to check if this event happens.

Check(inp, ê; e; fwl, fwr)

```
1 : if ê > e
2 :     return ⊥
3 : if inp = key and ê ∈ {fwl, ..., fwr}
4 :     return ABORT
5 : if inp = token and ê ∈ {fwl, fwr+1}
6 :     return ABORT
```

Figure 29: Algorithm Check, used in proofs in this section. In Check, ê is the epoch in the adversary's request, and e is the current epoch.

In particular, within the insulated region, the reduction can simulate challenge ciphertexts and non-challenge ciphertexts using its DDH tuple. Furthermore, ciphertexts can be moved around within the insulated region by tokens.

Remark 7. We note that the problem of challenge insulation in schemes with deterministic updates was also observed independently by Klooß et al. [[KLR19b], § B.2]. Their solution (though in the different context of CCA security of UE with certain properties) is to form a hybrid argument with a hybrid for each epoch, and essentially guess an epoch r which is the first token 'after' the hybrid index that the adversary has not corrupted, and use the inherent 'gap' in the adversary's knowledge continuum to replace challenge updates across this gap with encryptions of just one of the challenge messages. It is not clear if this approach would work for showing detIND-UE-CPA (or IND-ENC-CPA) of SHINE. We conjecture that even if it were possible to construct a

reduction in this vein, our approach enables a more direct proof: in particular we do not need to assume specific additional properties of the UE scheme in question for it to work.

SHINE0 is detIND-UE-CPA.

Theorem 4.1 (SHINE0 is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE0 be the updatable encryption scheme described in Fig. 25. For any detIND-UE-CPA adversary $\mathcal{A}$ against SHINE0 that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.1}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.1}}(\lambda).$$

Proof. Play hybrid games. We begin by partitioning non-corrupted key space as follows: $\{0,1,\ldots,n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j,\ldots,\mathsf{fwr}_j\}$, where fwr_i and fwr_i are firewalls of the i-th insulated region. Recall the definition from Section 3.3 and firewall computing algorithm FW-Find in Fig. 15: $\mathsf{fwl}_i, \mathsf{fwr}_i$ are firewalls of the i-th insulated region if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \in \mathcal{FW}$.

For $\mathrm{b} \in \{0,1\}$, define game $\mathcal{G}^{\mathrm{b}}_i$ as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{SHINE0},\,\mathcal{A}}$ except for:

- The game randomly picks an integer h, and if the challenge input $\bar{\mathrm{C}}$ is not an updated ciphertext of the h-th $\mathcal{O}$.Enc query, it (aborts and) returns a random bit for b'. This loss is upper-bounded by $\mathrm{Q_E}$.
- The game randomly picks $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0,\ldots,n\}$ and if $\mathsf{fwl}_i, \mathsf{fwr}_i$ are not the i-th firewalls, returns a random bit for b'. This loss is upper-bounded by $(n+1)^2$.
- For the challenge (made in epoch $\tilde{\mathrm{e}}$, input $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$): If $\tilde{\mathrm{e}} < \mathsf{fwl}_i$ then return a ciphertext with respect to $\bar{\mathrm{C}}$, if $\tilde{\mathrm{e}} > \mathsf{fwr}_i$ return a ciphertext with $\bar{\mathrm{M}}$, and if $\mathsf{fwl}_i \leq \tilde{\mathrm{e}} \leq \mathsf{fwr}_i$ then return a ciphertext with $\bar{\mathrm{M}}$ when $\mathrm{b} = 0$, return a ciphertext with respect to $\bar{\mathrm{C}}$ when $\mathrm{b} = 1$.
- After $\mathcal{A}$ outputs b', returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition triggers.

If $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$ are the desired values, then $\mathcal{G}^0_1$ is $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{SHINE0},\,\mathcal{A}}$, i.e. all challenges are encryptions of $\bar{\mathrm{M}}$. And there exists some l (the total number of insulated regions, bounded by $n+1$), game $\mathcal{G}^1_l$ is $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{SHINE0},\,\mathcal{A}}$, i.e. all challenges are updates of $\bar{\mathrm{C}}$. Let E to be the event that $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$ are the desired values, notice that $\mathbf{Pr}[\mathcal{G}^{\mathrm{b}}_i = 1 | \neg\mathrm{E}] = \frac{1}{2}$ for any $1 \leq i \leq n+1$ and $\mathrm{b} \in \{0,1\}$. Then

$$\begin{aligned}
\mathbf{Pr}[\mathcal{G}^1_l = 1] &= \mathbf{Pr}[\mathcal{G}^1_l = 1|\mathrm{E}] \cdot \mathbf{Pr}[\mathrm{E}] + \mathbf{Pr}[\mathcal{G}^1_l = 1|\neg\mathrm{E}] \cdot \mathbf{Pr}[\neg\mathrm{E}] \\
&= \mathbf{Pr}[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{SHINE0},\,\mathcal{A}} = 1] \cdot \frac{1}{(n+1)^2\mathrm{Q_E}} + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2\mathrm{Q_E}}), \text{ and} \\
\mathbf{Pr}[\mathcal{G}^0_1 = 1] &= \mathbf{Pr}[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{SHINE0},\,\mathcal{A}} = 1] \cdot \frac{1}{(n+1)^2\mathrm{Q_E}} + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2\mathrm{Q_E}})
\end{aligned}$$

Thus we have that

$$\begin{aligned}
\left|\mathbf{Pr}[\mathcal{G}^1_l = 1] - \mathbf{Pr}[\mathcal{G}^0_1 = 1]\right| &= \frac{1}{(n+1)^2\mathrm{Q_E}} \cdot \left|\mathbf{Pr}[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{SHINE0},\,\mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{SHINE0},\,\mathcal{A}} = 1]\right| \\
&= \frac{1}{(n+1)^2\mathrm{Q_E}} \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda)
\end{aligned}$$

Notice that all queries in $\mathcal{G}^1_{i-1}$ and $\mathcal{G}^0_i$ have the equal responses: for the challenge query and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, if called in epoch in first $i-1$ insulated regions, the reduction returns a ciphertext with respect to $\bar{\mathrm{C}}$, otherwise returns an encryption of $\bar{\mathrm{M}}$. Therefore, for any $l(\leq n+1)$, $|\mathbf{Pr}[\mathcal{G}^1_l = 1] - \mathbf{Pr}[\mathcal{G}^0_1 = 1]| \leq \sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]|$. We prove that for any $1 \leq i \leq l$, $|\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]| \leq 2\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},}(\lambda)$. We only prove one of these l hybrids, the rest can be similarly proven.

In hybrid i. Suppose that $\mathcal{A}_i$ is an adversary attempting to distinguish $\mathcal{G}^0_i$ from $\mathcal{G}^1_i$. For all queries concerning epochs outside of the i-th insulated region the responses will be equal in either game, so we assume that $\mathcal{A}_i$ asks for at least one challenge ciphertext in an epoch within the i-th insulated region (then $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{C}^*$) and this is where we will embed DDH tuples in our reduction.

We construct a reduction $\mathcal{B}_{4.1}$, detailed in Fig. 30, that is playing the standard DDH game and will simulate the responses of queries made by adversary $\mathcal{A}_i$.

Reduction $\mathcal{B}_{4.1}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.1}}$

1 : **receive** (g, X, Y, Z)
2 : **do Setup**
3 : $\bar{\mathrm{M}}, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathtt{ors}}(\lambda)$
4 : $\mathsf{phase} \leftarrow 1$
5 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, **get** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
6 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathtt{ors},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
7 : $\mathsf{twf} \leftarrow 1$ **if**
8 : $\quad \mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or**
9 : $\quad \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
10 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
11 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
12 : $\quad$ **return** b'
13 : **if** $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
14 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
15 : $\quad$ **return** b'
16 : **if** $\mathrm{b}' = \mathrm{b}$
17 : $\quad$ **return** 0
18 : **else**
19 : $\quad$ **return** 1

$\mathbf{Setup}(\lambda)$

20 : $\mathrm{b} \xleftarrow{\$} \{0,1\}$
21 : $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
22 : $\Delta_0 \leftarrow \perp$
23 : $\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
24 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
25 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
26 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
27 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\}$ **do**
28 : $\quad \mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}^{\bowtie}$
29 : $\quad \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$

$\mathbf{Setup}(\lambda)$ cont.

33 : **for** $j \in \{\mathsf{fwr}_i\text{+}1, \ldots, n\}$ **do**
34 : $\quad \mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}^{\bowtie}$
35 : $\quad \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
36 : **if** $\mathrm{b} = 0$
37 : $\quad \mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y; \mathrm{C} \xleftarrow{\$} \mathbb{G}$
38 : **else**
39 : $\quad \mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}; \mathrm{C} \leftarrow X$
40 : **for** $j \in \{\mathsf{fwl}_i\text{+}1, \ldots, \mathsf{fwr}_i\}$ **do**
41 : $\quad \Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

39 : $\mathsf{c} \leftarrow \mathsf{c} + 1$
40 : **if** $\mathsf{c} = \mathrm{h}$
41 : $\quad \mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}; \mathsf{inf} \leftarrow \mathsf{e}$
42 : **else**
43 : $\quad \mathsf{inf} \xleftarrow{\$} \mathbb{Z}_q^*$
44 : $\quad \pi(\mathrm{N}||\mathrm{M}) \leftarrow g^{\mathsf{inf}}$
45 : $\quad \mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
46 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
47 : **return** C_{e}

$\mathcal{O}.\mathsf{Next}$

48 : $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$

49 : **if** $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}) \notin \mathcal{L}$
50 : $\quad$ **return** $\perp$
51 : **if** $\mathsf{c} = \mathrm{h}$
52 : $\quad \mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
53 : **else**
54 : $\quad \mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
55 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
56 : **return** C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

57 : **do** $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
58 : **if** $\mathsf{inp} = \mathsf{key}$
59 : $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
60 : $\quad$ **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
61 : **if** $\mathsf{inp} = \mathsf{token}$
62 : $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
63 : $\quad$ **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$

64 : **if** $(\mathrm{h}, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$
65 : $\quad$ **return** ABORT
66 : **if** $\mathrm{b} = 0$
67 : $\quad \pi(\mathrm{N}||\bar{\mathrm{M}}) \leftarrow X; \tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
68 : **else**
69 : $\quad \pi(\mathrm{N}||\bar{\mathrm{M}}) \xleftarrow{\$} \mathbb{G}$
70 : $\quad \tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z^{\prod_{j=\mathsf{inf}+1}^{\mathsf{fwl}_i - 1} \Delta_j}$
71 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
72 : $\quad \tilde{\mathrm{C}}_j \leftarrow \bar{\mathrm{C}}^{(\prod_{k=0}^{j} \Delta_k)/(\prod_{k=0}^{\tilde{\mathsf{e}}-1} \Delta_k)}$ // left
73 : **for** $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
74 : $\quad \tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$ // embed
75 : **for** $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
76 : $\quad \tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}))^{k_j}$ // right
77 : $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
78 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

79 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
80 : **find**$(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
81 : **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$

Figure 30: Reduction $\mathcal{B}_{4.1}$ for proof of Theorem 4.1, in hybrid i. Moving left-to-right through embedding DDH tuples in the i-th insulated region: when $\mathrm{b} = 1$, embedding DDH tuples to token values to move left to random; when $\mathrm{b} = 0$, embedding DDH tuples to key values to move right to random. inf encodes fixed programming information: it marks an epoch if $\mathsf{c} = \mathrm{h}$, otherwise it is the random encryption exponent. On lines 3 and 6, ors refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. On lines 28 and 34, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

The reduction $\mathcal{B}_{4.1}$ receives DDH tuples (X, Y, Z), flips a coin b and simulates game $\mathcal{G}_i^{\mathrm{b}}$. Whenever the

reduction needs to provide an output of $\pi(\cdot)$ to $\mathcal{A}$, it chooses ('programs') some random value r such that $\pi(\cdot) = g^r$. Then, we use the fact that $(g^r)^{\mathrm{k_e}} = (g^{\mathrm{k_e}})^r$ and use the $g^{\mathrm{k_e}}$ values as 'public keys' to allow simulation. In this setting, decryption (i.e. π^{-1}) is simply a lookup to this mapping of the ideal cipher π. A summary of the technical simulations follows:

Initially,

1. $\mathcal{B}_{4.1}$ guesses the values of $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$.
2. $\mathcal{B}_{4.1}$ generates all keys and tokens except for $\mathrm{k}_{\mathsf{fwl}_i}, ..., \mathrm{k}_{\mathsf{fwr}_i}, \Delta_{\mathsf{fwl}_i}, \Delta_{\mathsf{fwr}_i+1}$. If $\mathcal{A}_i$ ever corrupts these keys and tokens – which indicates the firewall guess is wrong – the reduction aborts the game.
3. $\mathcal{B}_{4.1}$ computes the public values of keys in an epoch:
   - $\mathsf{e} \notin \{\mathsf{fwl}_i, ..., \mathsf{fwr}_i\}$: $\mathcal{B}_{4.1}$ computes $\mathsf{PK}_\mathsf{e} = g^{\mathrm{k_e}}$;
   - $\mathsf{e} \in \{\mathsf{fwl}_i, ..., \mathsf{fwr}_i\}$: $\mathcal{B}_{4.1}$ embeds DDH value Y as $\mathsf{PK}_{\mathsf{fwl}_i}$. More precisely, if $\mathrm{b} = 0, \mathsf{PK}_{\mathsf{fwl}_i} = Y$, otherwise $\mathsf{PK}_{\mathsf{fwl}_i} = Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}$ (since $g^{\Delta_{\mathsf{fwl}_i}} = Y$). Then $\mathcal{B}_{4.1}$ uses tokens $\Delta_{\mathsf{fwl}_i+1}, ..., \Delta_{\mathsf{fwr}_i}$ to compute the remaining public key values PK_e in the insulated region.

To simulate a non-challenge ciphertext that is:

- not the h-th query to $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_{4.1}$ generates a random value r for each encryption (so that the randomness will be consistent for calls that $\mathcal{A}_i$ makes to $\mathcal{O}.\mathsf{Upd}$) and programs the ideal cipher with $\mathrm{C}_\mathsf{e} = \mathsf{PK}_\mathsf{e}^r$. To respond to $\mathcal{O}.\mathsf{Upd}$ queries, the reduction computes $\mathrm{C}_{\mathsf{e}'} = \mathsf{PK}_{\mathsf{e}'}^r$ to update a non-challenge ciphertext to epoch e'.
- the h-th query to $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_{4.1}$ embeds either a random ciphertext ($\mathrm{b} = 0$) or a DDH value ($\mathrm{b} = 1$) to the encryption ($\mathrm{C}_{\mathsf{e}_\mathrm{h}}$). Furthermore, the reduction uses tokens $\Delta_0, ..., \Delta_{\mathsf{fwl}_i - 1}$ to update the h-th encryption. Note that $\bar{\mathrm{C}}$ is an update of the h-th encryption. The adversary can not ask for update of the h-th encryption in an epoch $\mathsf{e} \geq \mathsf{fwl}_i$, as this would trigger the trivial win condition $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$.

To simulate challenge-equal ciphertext in an epoch that is:

- to the left of the i-th insulated region: $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Upd}(\bar{\mathrm{C}})$ using the tokens that it created itself.
- within the i-th insulated region: $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Upd}(\bar{\mathrm{C}})$ if $\mathrm{b} = 1$, and simulates $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}})$ if $\mathrm{b} = 0$. More precisely, $\mathcal{B}_{4.1}$ embeds DDH value X to ciphertext information of challenge input, embeds DDH value Z to the challenge ciphertext. Which means if $\mathrm{b} = 1$, the reduction will give the value X to $\mathrm{C}_{\mathsf{e}_\mathrm{h}}$, $Z^{\prod_{j=\mathsf{e}_\mathrm{h}+1}^{\mathsf{fwl}_i - 1} \Delta_j}$ to $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ (recall $g^{\Delta_{\mathsf{fwl}_i}} = Y$) since

  $$\tilde{\mathrm{C}}_{\mathsf{fwl}_i} = \mathsf{SHINE0.Upd}(\bar{\mathrm{C}}) = \bar{\mathrm{C}}_{\mathsf{fwl}_i - 1}^{\Delta_{\mathsf{fwl}_i}} = (\mathrm{C}_{\mathsf{e}_\mathrm{h}}^{\Delta_{\mathsf{fwl}_i}})^{\prod_{j=\mathsf{e}_\mathrm{h}+1}^{\mathsf{fwl}_i - 1} \Delta_j}.$$

  If $\mathrm{b} = 0$, the reduction will give X to $\pi(\mathrm{N}||\bar{\mathrm{M}})$, and Z to $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ (recall $\mathsf{PK}_{\mathsf{fwl}_i} = Y$) since $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} = \mathsf{SHINE0.Enc}(\bar{\mathrm{M}}) = \pi(\mathrm{N}||\bar{\mathrm{M}})^{\mathrm{k}_{\mathsf{fwl}_i}}$. Furthermore, the reduction uses tokens $\Delta_{\mathsf{fwl}_i+1}, ..., \Delta_{\mathsf{fwr}_i}$ to update $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ to simulate all challenge ciphertext in epochs within the insulated region.
- to the right of the i-th insulated region: $\mathcal{B}_{4.1}$ simulates $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}})$ using the keys that it created itself.

Eventually, $\mathcal{B}_{4.1}$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{4.1}$ guesses that it has seen real DDH tuples (returns 0 to its DDH challenger), otherwise, $\mathcal{B}_{4.1}$ guesses that it has seen random DDH tuples (returns 1).

If $\mathcal{B}_{4.1}$ receives a real DDH tuple, then $\mathcal{B}_{4.1}$ perfectly simulates the environment of $\mathcal{A}_i$ in $\mathcal{G}_i^{\mathrm{b}}$. If $\mathcal{B}_{4.1}$ receives a random DDH tuple, then $\mathcal{B}_{4.1}$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.1}}(\lambda) = \frac{1}{2}|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$.

□

SHINE0 **is** IND-ENC-CPA **Secure.** As a corollary of Theorem 2.4 and Theorem 4.1, SHINE0 is IND-ENC-CPA – however in Appendix A we give a tighter proof – eliminating the $\mathrm{Q_E}$ term – by directly proving the IND-ENC-CPA security of SHINE0. The proof follows a similar strategy to that of Theorem 4.1, with one hybrid for each insulated region.

MirrorSHINE **is** detIND-UE-CPA**.**

Theorem 4.3 (MirrorSHINE is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let MirrorSHINE be the updatable encryption scheme described in Fig. 26. For any detIND-UE-CPA adversary $\mathcal{A}$ against MirrorSHINE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.3}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{MirrorSHINE},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.3}}(\lambda).$$

Proof. The proof is similar to the security proof of Theorem 4.1, except that the reduction will embed two random group elements to the image of π_1, π_2 while it simulates the output of the $\mathcal{O}$.Enc oracle, instead of embedding one random group element to the image of π. We will not give a detailed construction of the reduction in this proof, since we will give a more general construction in the proof of Theorem 4.4. □

OCBSHINE **is** detIND-UE-CPA**.** For convenience, we denote a vector with $l+2$ elements to be $\vec{C} = (C^0, \ldots, C^{l+1})$. We also use the shorthand $\vec{A} \leftarrow \vec{B}^c$ (component-wise exponentiation) and $\vec{A} \leftarrow B^{\vec{c}}$ (common-base exponentiation) to mean the following:

$\vec{A} \leftarrow \vec{B}^c$	$\vec{A} \leftarrow B^{\vec{c}}$
1 : **for** $j = 0, \ldots, l+1$ **do**	1 : **for** $j = 0, \ldots, l+1$ **do**
2 : $A^j \leftarrow (B^j)^c$	2 : $A^j \leftarrow B^{c_j}$

Theorem 4.4 (OCBSHINE is detIND-UE-CPA)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime, $\lambda = m$) with generator g, and let OCBSHINE be the updatable encryption scheme described in Fig. 28. For any detIND-UE-CPA adversary $\mathcal{A}$ against OCBSHINE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_{4.4}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{OCBSHINE},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.4}}(\lambda).$$

Proof. The proof is similar to the security proof of Theorem 4.1, except that here we are dealing with vector of ciphertexts. To simulate the output of $\mathcal{O}$.Enc, the reduction will embed a vector of random group elements to the image of the random permutation family. The reduction is detailed in Fig. 31.

□

5.5 SHINE **is** INT-CTXT$^{\mathsf{s}}$

We then explain how we bound the advantage of any adversary playing the INT-CTXT$^{\mathsf{s}}$ game for SHINE by the advantage of a reduction playing CDH.

Proof Method for ciphertext integrity. In the INT-CTXT$^{\mathsf{s}}$ game, the challenger will keep a list of consistent values for ciphertexts (i.e. the underlying permutation output). Suppose $\tilde{\mathrm{C}}$ is a forgery attempt sent to the $\mathcal{O}$.Try query in epoch $\tilde{\mathrm{e}}$. Let $\tilde{c} = (\tilde{\mathrm{C}})^{1/\mathrm{k}_{\tilde{\mathrm{e}}}}$ be the underlying permutation output.

If $\tilde{c}$ is a new value, since we have that π (or (π_1, π_2) or $(\pi_0, \{\pi_{\mathrm{N}\|i\|b}\})$) is a random permutation then the INT-CTXT$^{\mathsf{s}}$ challenger simulates the preimage of $\tilde{c}$ under the corresponding random permutation(s) to be a random string(s). So the probability that this (these) random string(s) matches the integrity tag is negligible, and this carries over to the probability that the adversary wins the INT-CTXT$^{\mathsf{s}}$ game.

If $\tilde{c}$ is an already-existing value, and suppose this event happens with probability p. We construct a reduction playing the CDH game such that it wins CDH game with probability $p \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$. Similar to the proof method of Theorem 4.1, we construct a reduction playing the CDH experiment by guessing the location of the firewalls around the challenge epoch. The reduction embeds the CDH value and simulates the INT-CTXT$^{\mathsf{s}}$ game, using any successfully-forged ciphertext to compute the CDH output to its CDH challenger.

Reduction $\mathcal{B}_{4.4}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.4}}$

1 : **receive** (g, X, Y, Z)
2 : **do Setup**
3 : $\vec{\mathrm{M}}, \vec{\mathrm{C}} \leftarrow \mathcal{A}^{\mathtt{ors}}(\lambda)$
4 : $\mathsf{phase} \leftarrow 1$
5 : Create $\tilde{\mathrm{C}}$ with $(\vec{\mathrm{M}}, \vec{\mathrm{C}})$, **get** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
6 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathtt{ors},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
7 : $\mathsf{twf} \leftarrow 1$ **if**
8 : $\quad \mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or**
9 : $\quad \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
10 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
11 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
12 : $\quad$ **return** b'
13 : **if** $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
14 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
15 : $\quad$ **return** b'
16 : **if** $\mathrm{b}' = \mathrm{b}$
17 : $\quad$ **return** 0
18 : **else**
19 : $\quad$ **return** 1

Setup(λ)

20 : $\mathrm{b} \xleftarrow{\$} \{0,1\}$
21 : $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
22 : $\Delta_0 \leftarrow \perp$
23 : $\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
24 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
25 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
26 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
27 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\}$ **do**
28 : $\quad \mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ $\bowtie$
29 : $\quad \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
30 : **for** $j \in \{\mathsf{fwr}_i\text{+}1, \ldots, n\}$ **do**
31 : $\quad \mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ $\bowtie$
32 : $\quad \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$

Setup(λ) cont.

33 : **if** $\mathrm{b} = 0$
34 : $\quad \mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y$
35 : $\quad \vec{\mathrm{C}} \xleftarrow{\$} \mathbb{G}^{l+2}$
36 : **else**
37 : $\quad \mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}$
38 : $\quad \vec{r} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
39 : $\quad \vec{\mathrm{C}} \xleftarrow{\$} X^{\vec{r}}$
40 : **for** $j \in \{\mathsf{fwl}_i\text{+}1, \ldots, \mathsf{fwr}_i\}$ **do**
41 : $\quad \Delta_j \xleftarrow{\$} \mathbb{Z}_q^*$
42 : $\quad \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$

$\mathcal{O}.\mathsf{Enc}(\vec{\mathrm{M}})$

43 : $\mathsf{c} \leftarrow \mathsf{c} + 1$
44 : **if** $\mathsf{c} = \mathrm{h}$
45 : $\quad \vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \vec{\mathrm{C}}; \mathsf{inf} \leftarrow \mathsf{e}$
46 : **else**
47 : $\quad \vec{\mathsf{inf}} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
48 : $\quad \Pi(\vec{\mathrm{M}}) \xleftarrow{\$} g^{\vec{\mathsf{inf}}}$ / see caption
49 : $\quad \vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\vec{\mathsf{inf}}}$
50 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \vec{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf}$ **or** $\vec{\mathsf{inf}})\}$
51 : **return** $\vec{\mathrm{C}}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$

52 : $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\vec{\mathrm{C}}_{\mathsf{e}-1})$

53 : **if** $(j, \vec{\mathrm{C}}_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}$ **or** $\vec{\mathsf{inf}}) \notin \mathcal{L}$
54 : $\quad$ **return** $\perp$
55 : **if** $j = \mathrm{h}$
56 : $\quad \vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \vec{\mathrm{C}}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
57 : **else**
58 : $\quad \vec{\mathrm{C}}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\vec{\mathsf{inf}}}$
59 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \vec{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf}$ **or** $\vec{\mathsf{inf}})\}$
60 : **return** $\vec{\mathrm{C}}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

61 : **do** $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
62 : **if** $\mathsf{inp} = \mathsf{key}$
63 : $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
64 : $\quad$ **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
65 : **if** $\mathsf{inp} = \mathsf{token}$
66 : $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
67 : $\quad$ **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\vec{\mathrm{M}}, \vec{\mathrm{C}})$

68 : **if** $(\mathrm{h}, \vec{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$
69 : $\quad$ **return** ABORT
70 : **if** $\mathrm{b} = 0$
71 : $\quad \vec{s} \xleftarrow{\$} (\mathbb{Z}_q^*)^{l+2}$
72 : $\quad \Pi(\vec{\mathrm{M}}) \xleftarrow{\$} X^{\vec{s}}$
73 : $\quad \vec{\tilde{\mathrm{C}}}_{\mathsf{fwl}_i} \leftarrow Z^{\vec{s}}$
74 : **else**
75 : $\quad \Pi(\vec{\mathrm{M}}) \xleftarrow{\$} \mathbb{G}^{l+2}$
76 : $\quad \vec{\tilde{\mathrm{C}}}_{\mathsf{fwl}_i} \leftarrow (Z^{\Pi_{j=\mathsf{inf}+1}^{\mathsf{fwl}_i - 1} \Delta_j})^{\vec{r}}$
77 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
78 : $\quad \vec{\tilde{\mathrm{C}}}_j \leftarrow \vec{\tilde{\mathrm{C}}}^{(\Pi_{k=0}^{j} \Delta_k)/(\Pi_{k=0}^{\tilde{\mathsf{e}}-1} \Delta_k)}$ / left
79 : **for** $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
80 : $\quad \vec{\tilde{\mathrm{C}}}_j \leftarrow \vec{\tilde{\mathrm{C}}}_{j-1}^{\Delta_j}$ / embed
81 : **for** $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
82 : $\quad \vec{\tilde{\mathrm{C}}}_j \leftarrow \{\Pi(\vec{\mathrm{M}})\}^{k_j}$ / right
83 : $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\vec{\tilde{\mathrm{C}}}_j, j)\}$
84 : **return** $\vec{\tilde{\mathrm{C}}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

85 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
86 : **find**$(\vec{\tilde{\mathrm{C}}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
87 : **return** $\vec{\tilde{\mathrm{C}}}_{\mathsf{e}}$

Figure 31: Reduction $\mathcal{B}_{4.4}$ for proof of Theorem 4.4, in hybrid i. On lines 3 and 6, ors refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. On lines 28 and 31, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation. On line 48 of $\mathcal{O}.\mathsf{Enc}$, vector $\Pi(\vec{\mathrm{M}}) = (\pi_0(\mathrm{N}), \pi_{\mathrm{N}\|1\|0}(\mathrm{M}_1), \ldots, \pi_{\mathrm{N}\|l\|0}(\mathrm{M}_l), \pi_{\mathrm{N}\|l\|1}(\Sigma))$.

SHINE0 is INT-CTXT$^{\mathsf{s}}$.

Theorem 5.1 (SHINE0 is INT-CTXT$^{\mathsf{s}}$)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime, where $\lambda = v + m + l$) with generator g, and let SHINE0 be the updatable encryption scheme described in Fig. 25. For any INT-CTXT$^{\mathsf{s}}$

adversary $\mathcal{A}$ against SHINE0 that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it asks its $\mathcal{O}$.Try query, there exists an adversary $\mathcal{B}_{5.1}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) \leq 1/2^l + \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.1}}.$$

Proof. Note that the probability of a random string ends by 0^l is $1/2^l$.

In the INT-CTXT$^\mathsf{s}$ game, whenever, the adversary sends a forgery, suppose C^*, to the $\mathcal{O}$.Try oracle. If the trivial win conditions does not trigger, then C^* is a new ciphertext to the challenger and there exists an insulated region around the challenge epoch. We split the proof into two parts based on if $\mathrm{C}^{*1/\mathrm{k_e}}$ is a new value to the challenger:

1. If $\mathrm{C}^{*1/\mathrm{k_e}}$ is a new value, the random permutation π^{-1} will pick a random string a as the output of $\pi^{-1}(\mathrm{C}^{*1/\mathrm{k_e}})$. So the probability of a is valid (a $(v+m+l)$-bit string with a l-bit zero string in the end) is upper bounded by $1/2^l$.
2. If $\mathrm{C}^{*1/\mathrm{k_e}}$ is an existed value, denote this event as E_3, we claim that the probability of E_3 happens is very low. Which means it is hard to provide a valid forgery with a known permutation value. In other words, without the knowledge of the encryption key, it is difficult to provide a correct exponentiation. Formally, we prove the following inequality in Lemma 5.2.

   $\mathbf{Pr}[E_3] = \mathbf{Pr}[\mathrm{C}^{*1/\mathrm{k_e}} \text{ exists}, \mathrm{C}^* \text{ is new}] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}$.

In order to analyze the security, we define some events:

- $E_1 = \{\mathrm{C}^* \text{ is new}\}$,
- $E_2 = \{\mathrm{C}^{*1/\mathrm{k_e}} \text{ is new}, \mathrm{C}^* \text{ is new}\}$,
- Recall $E_3 = \{\mathrm{C}^{*1/\mathrm{k_e}} \text{ exists}, \mathrm{C}^* \text{ is new}\}$.

Denote the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}$ to be $\mathbf{Exp}$. Then we have the following results:

- $\mathbf{Pr}[\mathbf{Exp} = 1 \mid \neg E_1] = 0$.
- We have proved $\mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \leq 1/2^l$ in part 1.
- Events $\neg E_1, E_2, E_3$ are disjoint from each other, so $\mathbf{Pr}[\neg E_1] + \mathbf{Pr}[E_2] + \mathbf{Pr}[E_3] = 1$.
- We have proved $\mathbf{Pr}[E_3] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}$ in Lemma 5.2.

Applying the above properties, we can compute the INT-CTXT$^\mathsf{s}$ advantage

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) &= \mathbf{Pr}[\mathbf{Exp} = 1] \\
&= \mathbf{Pr}[\mathbf{Exp} = 1 \mid \neg E_1] \cdot \mathbf{Pr}[\neg E_1] + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \cdot \mathbf{Pr}[E_2] + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_3] \cdot \mathbf{Pr}[E_3] \\
&= \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] \cdot \mathbf{Pr}[E_2] + \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_3] \cdot \mathbf{Pr}[E_3] \\
&\leq \mathbf{Pr}[\mathbf{Exp} = 1 \mid E_2] + \mathbf{Pr}[E_3] \\
&\leq 1/2^l + \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}.
\end{aligned}$$

Lemma 5.2. Let $\mathcal{A}$ be an INT-CTXT$^\mathsf{s}$ adversary against SHINE0 that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it asks its $\mathcal{O}$.Try query. Suppose $\tilde{\mathrm{C}}$ is a forgery attempt provided by $\mathcal{A}$ and the corresponding permutation value is $\tilde{c}$. Define E to be the event that $\tilde{c}$ is an existed value but $\tilde{\mathrm{C}}$ is a new value. Then there exists an adversary $\mathcal{B}_{5.2}$ against CDH such that

$$\mathbf{Pr}[E] \leq \mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.2}}$$

Proof. Suppose $\mathcal{A}$ is an adversary against INT-CTXT$^{\mathsf{s}}$ game, and it can provide a forgery such that $\tilde{\mathrm{C}}$ is a new ciphertext but the underlying permutation value is an existed one with probability $\mathbf{Pr}[E]$. We claim that there exists a reduction $\mathcal{B}_{5.2}$, detailed in Figure 32, such that it wins CDH game with probability $\mathbf{Pr}[E] \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$.

Reduction $\mathcal{B}_{5.2}$ playing $\mathbf{Exp}^{\mathsf{CDH}}_{\mathbb{G},\,\mathcal{B}_{5.2}}$

1 : **receive** (g, X, Y)
2 : **do Setup**
3 : $\mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Try}}(\lambda)$
4 : **if** twf $= 1$ **or** ABORT occurred
5 : win $\leftarrow 0$
6 : **else**
7 : **return** win

Setup(λ)

8 : $\mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
9 : $\Delta_0 \leftarrow \perp$; $\mathsf{e}, \mathsf{c} \leftarrow 0$; win $\leftarrow 0$
10 : $\mathcal{L}^*, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
11 : $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \ldots, n\}$
12 : $\mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
13 : **for** $j \in \{0, \ldots, \hat{\mathsf{fwl}}\text{-}1\}$ **do**
14 : $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*$
15 : $\Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ // except Δ_0
16 : $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
17 : **for** $\{\hat{\mathsf{fwr}}\text{+}1, \ldots, n\}$ **do**
18 : $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*$
19 : $\Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}$ // except $\Delta_{\hat{\mathsf{fwr}}+1}$
20 : $\mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
21 : $\mathsf{PK}_{\hat{\mathsf{fwl}}} \leftarrow Y$
22 : **for** $j \in \{\hat{\mathsf{fwl}}\text{+}1, \ldots, \hat{\mathsf{fwr}}\}$ **do**
23 : $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*$; $\mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

24 : $\mathsf{c} \leftarrow \mathsf{c} + 1$
25 : **if** $\mathsf{c} = \mathrm{h}$
26 : **if** $\mathsf{e} < \hat{\mathsf{fwl}}$
27 : $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow X$
28 : $\mathrm{C_e} \leftarrow X^{\mathrm{k_e}}$
29 : **else**
30 : **return** ABORT
31 : **else**
32 : $r \xleftarrow{\$} \mathbb{Z}_q^*$;
33 : $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow g^r$
34 : $\mathrm{C_e} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
35 : $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\mathsf{c}, \mathrm{C_e}, \mathsf{e}; r)\}^{\gtrdot}$
36 : **return** $\mathrm{C_e}$

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

37 : $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C_{e-1}})$

38 : **if** $(j, \mathrm{C_{e-1}}, \mathsf{e} - 1; r) \notin \mathcal{L}^*$
39 : **return** $\perp$
40 : **if** $j = \mathrm{h}$
41 : **if** $\mathsf{e} < \hat{\mathsf{fwl}}$
42 : $\mathrm{C_e} \leftarrow \mathrm{C}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
43 : **else**
44 : **return** ABORT
45 : **else**
46 : $\mathrm{C_e} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
47 : $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C_e}, \mathsf{e}; r)\}$
48 : **return** $\mathrm{C_e}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

49 : **do** Check$(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}})$
50 : **if** inp $=$ key
51 : $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
52 : **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
53 : **if** inp $=$ token
54 : $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
55 : **for** $i \in \mathcal{T}^*$ **do**
56 : **for** $(j, \mathrm{C}_{i-1}, i-1; r) \in \mathcal{L}^*$ **do**
57 : $\mathrm{C}_i \leftarrow \mathcal{O}.\mathsf{Upd}(\mathrm{C}_{i-1})$
58 : $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, \mathrm{C}_i, i; r)\}$
59 : **return** $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Try}(\tilde{\mathrm{C}})$

60 : **if** phase $= 1$
61 : **return** $\perp$
62 : phase $\leftarrow 1$
63 : **if** $\tilde{\mathsf{e}} \in \mathcal{K}^*$ **or** $\tilde{\mathrm{C}} \in \mathcal{L}^*$
64 : twf $\leftarrow 1$
65 : **if** $\tilde{\mathsf{e}} \notin \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\}$
66 : twf $\leftarrow 1$
67 : $y \leftarrow \tilde{\mathrm{C}}^{1/\prod_{\mathsf{e}=\hat{\mathsf{fwl}}+1}^{\mathsf{e}=\tilde{\mathsf{e}}} \Delta_{\mathsf{e}}}$
68 : **call Chall with** y; **get** b
69 : win $\leftarrow$ b

Figure 32: Reduction $\mathcal{B}_{5.2}$ for proof of Lemma 5.2. On line 35, $\gtrdot$ indicates r is empty when $\mathsf{c} = \mathrm{h}$.

The reduction will guess the location of firewalls around the epoch when $\mathcal{O}.\mathsf{Try}$ query happens, furthermore, it guess which message (suppose the h-th encryption) might be the underlying message of the forgery. After it receives the CDH values g^a, g^b, it embeds g^a to the h-th encryption as $\pi(\mathrm{N}\|\mathrm{M}\|0^l) \leftarrow g^a$, embeds g^b to the public key value on the left firewall as $\mathsf{PK}_{\hat{\mathsf{fwl}}} = g^{\mathrm{k}_{\hat{\mathsf{fwl}}}} \leftarrow g^b$. Then $\tilde{\mathrm{C}} = g^{ab \prod_{\mathsf{e}=\hat{\mathsf{fwl}}+1}^{\mathsf{e}=\tilde{\mathsf{e}}} \Delta_{\mathsf{e}}}$ with probability $\mathbf{Pr}[E] \cdot \frac{1}{\mathrm{Q_E}(n+1)^2}$, which is the advantage of winning the CDH game. □

MirrorSHINE **is** INT-CTXT$^{\mathsf{s}}$**.**

Theorem 5.3 (MirrorSHINE is INT-CTXT$^{\mathsf{s}}$)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let MirrorSHINE be the updatable encryption scheme described in Fig. 26. For any INT-CTXT$^{\mathsf{s}}$ adversary

$\mathcal{A}$ against MirrorSHINE that asks at most queries at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it asks its $\mathcal{O}$.Try query, there exists an adversary $\mathcal{B}_{5.3}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{MirrorSHINE},\,\mathcal{A}}(\lambda) \leq 1/2^{|\mathcal{N}|} + 2\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.3}}$$

Proof. Similar to the proof in Theorem 5.1. Suppose $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^1, \tilde{\mathrm{C}}^2)$ is a new ciphertext sent as the $\mathcal{O}$.Try query. Then at least one block of the ciphertext, suppose $\tilde{\mathrm{C}}^i$, is new. We split the proof in two parts:

- If $(\tilde{\mathrm{C}}^i)^{1/\mathrm{k_e}}$ is new. The random permutation π_i^{-1} will pick a random string a_i as the output of $\pi_i^{-1}((\tilde{\mathrm{C}}^i)^{1/\mathrm{k_e}})$. So the probability of a_i is valid (satisfy $a_1 = a_2$) is upper bounded by $1/2^{|\mathcal{N}|}$.
- If $(\tilde{\mathrm{C}}^i)^{1/\mathrm{k_e}}$ is an existed value outputted by permutation π_i. Similar to the proof in Theorem 5.1, we can prove that $\mathbf{Pr}[(\tilde{\mathrm{C}}^i)^{1/\mathrm{k_e}}$ exists, $\tilde{\mathrm{C}}^i$ is new $] \leq 2\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.3}}$. □

OCBSHINE **is** INT-CTXT$^{\mathsf{s}}$**.**

Theorem 5.4 (OCBSHINE is INT-CTXT$^{\mathsf{s}}$)**.** Let $\mathbb{G}$ be a group of order q (a λ-bit prime, $\lambda = m$) with generator g, and let OCBSHINE be the updatable encryption scheme described in Fig. 28. For any INT-CTXT$^{\mathsf{s}}$ adversary $\mathcal{A}$ against OCBSHINE that asks at most $\mathrm{Q_E}$ queries, on messages with at most L message blocks, to $\mathcal{O}$.Enc before it asks its $\mathcal{O}$.Try query, there exists an adversary $\mathcal{B}_{5.4}$ against CDH such that

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\,\mathcal{A}}(\lambda) \leq \frac{{\mathrm{Q_E}}^2 + \mathrm{Q_E}}{2^{m-a}} + (\mathrm{L}+1)\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_{5.4}}.$$

Proof. **Game 0**

The first game is the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\,\mathcal{A}}$, given in Fig. 6 and Fig. 10. As definition 4 we have

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{OCBSHINE},\,\mathcal{A}}(\lambda) = \mathbf{Pr}[\mathcal{G}_0 = 1]$$

Game 1

Modify the response of the encryption oracle such that the game randomly picks a nonce and if the nonce repeats, it aborts the game and returns win $= 0$. This loss is upper-bounded by $\frac{{\mathrm{Q_E}}^2}{2^{m-a}}$, then we have

$$|\mathbf{Pr}[\mathcal{G}_0 = 1] - \mathbf{Pr}[\mathcal{G}_1 = 1]| \leq \frac{{\mathrm{Q_E}}^2}{2^{m-a}}$$

Finally, we claim $\mathbf{Pr}[\mathcal{G}_1 = 1] \leq \frac{\mathrm{Q_E}}{2^{m-a}} + (l+1)\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}$. Similar to the proof in Theorem 5.1. Suppose $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^0, \ldots, \tilde{\mathrm{C}}^{\tilde{l}+1})$ is a forgery attempt sent as the $\mathcal{O}$.Try query in epoch $\tilde{\mathrm{e}}$. Suppose $\tilde{c}_i = (\tilde{\mathrm{C}}^i)^{1/\mathrm{k}_{\tilde{\mathrm{e}}}}$, the corresponding message is $\tilde{\mathrm{M}}_i$. Let $\tilde{\mathrm{N}}$ be the underlying nonce and $\tilde{\Sigma}$ be the underlying checksum. We consider the following two situations.

1. If $(\tilde{c}_0, \ldots, \tilde{c}_{\tilde{l}+1})$ is new, we claim that the probability of the adversary correctly guessing $\tilde{\Sigma}$ is upper-bounded by $\frac{\mathrm{Q_E}}{2^{m-a}}$.
   (a) If $\tilde{c}_0$ is new, then either $\tilde{\mathrm{N}}$ is new or $|\tilde{\mathrm{M}}_{\tilde{l}}|$ is new.
      i. If $\tilde{\mathrm{N}}$ is equal to some already existing nonce N, we claim that the probability that this event happens is very low. Since π_0 is a random permutation, the first $(m\text{-}a)$ bits of the preimage of $\tilde{c}_0$ under π_0, which is $\tilde{\mathrm{N}}$, is as likely as any $(m\text{-}a)$-bit string. So the probability that $\tilde{\mathrm{N}}$ collides with one of the existed nonces is upper bounded by $\mathrm{Q_E}/2^{m-a}$.
      ii. If $\tilde{\mathrm{N}}$ is new, therefore $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$ is a new random permutation. The adversary sees no image and preimage of $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$. The preimage of $\tilde{c}_{\tilde{l}+1}$ under $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$, which is $\tilde{\Sigma}$, is as likely as any other m-bit string. So the probability that the adversary correctly guesses $\tilde{\Sigma}$ (which means $\tilde{\Sigma} = \oplus_{i=1}^{l}\tilde{\mathrm{M}}_i$) is 2^{-m}.

(b) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, ..., c_{l+1})$ but $\tilde{l} \neq l$, similar to the above situation $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1}$ is a new random permutation. So the probability that the adversary correctly guesses $\tilde{\Sigma}$ is 2^{-m}.

(c) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, ..., c_{l+1})$ and $\tilde{l} = l$ but $\tilde{c}_j \neq c_j$ for some $j \in \{1, ..., \tilde{l}\}$. The adversary sees one image of $\pi_{\tilde{\mathrm{N}}\|j\|0}$, that is, c_j. The preimage of $\tilde{c}_j$ under $\pi_{\tilde{\mathrm{N}}\|j\|0}$, which is $\tilde{\mathrm{M}}_j$, is as likely as any m-bit string except for M_j (or $\mathrm{M}_l\|0^*$ when $j = l$). So the probability that the adversary correctly guesses $\tilde{\Sigma}$ is $\frac{1}{2^m-1}$.

(d) If $\tilde{c}_0 = c_0$ for some already existing $(c_0, ..., c_{l+1})$ and $\tilde{l} = l$ and $\tilde{c}_j = c_j$ for all $j \in \{1, ..., \tilde{l}\}$ but $\tilde{c}_{\tilde{l}+1} \neq c_{l+1}$. This event is impossible to happen, as $\tilde{\mathrm{N}} = \mathrm{N}, |\tilde{\mathrm{M}}_{\tilde{l}}| = |\mathrm{M}_l|$ and $\tilde{l} = l$ so $\pi_{\tilde{\mathrm{N}}\|\tilde{l}\|1} = \pi_{\mathrm{N}\|l\|1}$, furthermore $\tilde{\mathrm{M}}_j = \mathrm{M}_j$ for all $j \in \{1, ..., \tilde{l}\}$, so $\tilde{\Sigma} = \Sigma$ and then $\tilde{c}_{\tilde{l}+1} = c_{l+1}$.

2. If $(\tilde{c}_0, ..., \tilde{c}_{\tilde{l}+1})$ is an already existing value output by the permutations but $\tilde{\mathrm{C}} = (\tilde{\mathrm{C}}^0, ..., \tilde{\mathrm{C}}^{\tilde{l}+1})$ is a new ciphertext, then this is equivalent to $(\tilde{c}_0, ..., \tilde{c}_{\tilde{l}})$ exists but $(\tilde{\mathrm{C}}^0, ..., \tilde{\mathrm{C}}^{\tilde{l}})$ is new, as in the analysis of 1.(d). Similar to the proof in Theorem 5.1, we can prove that the probability of this event happening is upper-bounded by CDH advantage:

$$\mathbf{Pr}[\ (\tilde{c}_0, ..., \tilde{c}_{\tilde{l}}) \text{ exists but } (\tilde{\mathrm{C}}^0, ..., \tilde{\mathrm{C}}^{\tilde{l}}) \text{ is new }] \leq (\mathrm{L}+1)\mathrm{Q_E}(n+1)^2\mathbf{Adv}^{\mathsf{CDH}}. \qquad \square$$

5.6 Implementing the SHINE Schemes

In the proofs of Theorem 4 and Theorem 5, we require that π is a random (unkeyed) permutation which must be followed by a mapping to an appropriate group for exponentiation by the epoch key. For the permutation we do not need any specific and strong properties that are provided by modern constructions of block ciphers and sponges. As far as the proof goes, and in practice, the property that we want from this permutation is that given a ciphertext and the inverse of the epoch key $\mathrm{k_e}$, the only way to extract useful information about the message is to apply the inverse permutation π^{-1}. The random permutation model (or ideal cipher model) is thus the tool we need here to create a simple interface for this aspect of our proof.

The different members of the SHINE family are suited to different application scenarios. The variants SHINE0 and MirrorSHINE are best suited to cases where messages are of small, fixed size, such as customer credentials (or phone contact details, to return to the motivating example in the Introduction). For applications with longer messages (i.e. larger than the size of the exponentiation group), OCBSHINE is considerably faster and we will assume that these choices are made in our implementation suggestions. This removes any need for larger groups in order to encrypt longer messages. Using larger groups would not only carry a significant performance penalty, but also force us to construct custom large blocklength block ciphers. Although this can be done (and has been for RSA groups [GOR18], where our approach would not work), the analysis is tricky.

Instantiating the ideal permutation. The message block in SHINE0, MirrorSHINE and the final message block in OCBSHINE must be appropriately padded to allow application of the permutation. The permutation could be deployed using a variable-output-length sponge construction, a block cipher or an authenticated encryption scheme with a fixed key and suitably large nonce space. In practice, we suggest to instantiate the random permutation with a block cipher of a suitable block length. AES has only 128-bit blocks which does not match the minimum required size of the group, so we instead suggest block ciphers such as Threefish, or original Rijndael, allowing block lengths of 256 or 512 bits.

Mapping to elliptic curve group. We would like to instantiate our groups using elliptic curves. Using modern techniques it is always possible to find a suitable curve over a field with a size matching the block length of the ideal permutation, but using standard curves like NIST P-256 or P-521 seems desirable. A standard approach [Kob87] is to embed bit strings in the X-coordinate of a point as follows. Note that close to half the field elements are X-coordinates of points. Given a field of size q, we consider a t-bit block as an integer x_0 and find a small integer u such that $u2^t + x_0$ is the X-coordinate of a curve point. If $\log q - t$ is between 8 and 9, this will fail to terminate with probability around 2^{-256} under reasonable assumptions.

With this approach we could use Threefish with 512-bit blocks together with NIST P-521 curve. If we want to use 256-bit blocks from Threefish, or original Rijndael, together with NIST P-256 curve, we can use

a standard block cipher iteration trick [RSA78] to reduce the block length from 256 bits, so that embedding in the X-coordinate still works, as follows. With block length $t+\tau$, concatenate a t-bit block with τ leading zeros and apply the block cipher until the τ leading bits of the result are all zeros. Discard these zeros to get a t-bit block. This is fairly cheap as for our purposes 8 or 9 bits will do.

Note that we have constructed an injective embedding of a block into an elliptic curve, not a bijection as assumed in our proofs. When we sample group elements in our proof, we must take care to sample points in the image of our embedding, but this can be done cheaply.

6 Conclusions

In this work we provided a suite of new updatable encryption schemes, collectively called SHINE, and a new definition of security xxIND-UE-atk (that implies prior notions) in which we prove our schemes secure. In the process, we provided a greater understanding of the proof techniques that are inherent in the strong corruption model that is desirable for updatable encryption – in particular in the context of deterministic updates that is desirable in practice.

Acknowledgements. This research was funded by the Research Council of Norway under Project No. 248166. Gareth T. Davies is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement 802823. We would like to thank Frederik Armknecht, Håvard Raddum and Mohsen Toorani for fruitful discussions in the initial stages of this project, and anonymous reviewers for a number of useful suggestions for improvement. We would also like to thank Adam O'Neill for a number of useful suggestions for improvement. Parts of this work were conducted while Gareth T. Davies was employed by NTNU Trondheim and the University of Paderborn.

References

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*. The Internet Society, 2005. Cited on page 6.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998. Cited on page 6.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013. Cited on pages 3, 5, 7, 8, 50, and 51.

[BLMR15] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. *IACR Cryptology ePrint Archive, Report 2015/220*, 2015. Cited on pages 3 and 5.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008. Cited on page 4.

[CCFL17] Christian Cachin, Jan Camenisch, Eduarda Freire-Stögbuchner, and Anja Lehmann. Updatable tokenization: Formal definitions and provably secure constructions. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 59–75. Springer, 2017. Cited on page 6.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 185–194. ACM, 2007. Cited on page 6.

[Cou18] PCI Security Standards Council. Data security standard (PCI DSS v3.2.1), 2018. https://www.pcisecuritystandards.org/. Cited on pages 3 and 6.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008. Cited on page 34.

[DDLM19] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, volume 11547 of *Lecture Notes in Computer Science*, pages 58–77. Springer, 2019. Cited on page 6.

[DRC14] Sandra Diaz-Santiago, Lil María Rodríguez-Henríquez, and Debrup Chakraborty. A cryptographic study of tokenization systems. In Mohammad S. Obaidat, Andreas Holzinger, and Pierangela Samarati, editors, *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, pages 393–398. SciTePress, 2014. Cited on page 6.

[EPRS17a] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 98–129. Springer, 2017. Cited on pages 3 and 5.

[EPRS17b] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. *IACR Cryptology ePrint Archive, Report 2017/527*, 2017. Cited on page 5.

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. A unified framework for trapdoor-permutation-based sequential aggregate signatures. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 34–57, Cham, 2018. Springer International Publishing. Cited on page 44.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 379–393. ACM, 2019. Cited on page 5.

[KLR19a] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99. Springer, 2019. Cited on pages 3, 5, 7, 9, 16, 19, and 20.

[KLR19b] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. *IACR Cryptology ePrint Archive, Report 2019/222*, 2019. Cited on page 35.

[Kob87] Neil Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987. Cited on page 44.

[KRS+03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In Jeff Chase, editor, *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA*. USENIX, 2003. Cited on page 3.

[Lee17] Ela Lee. Improved security notions for proxy re-encryption to enforce access control. In Tanja Lange and Orr Dunkelman, editors, *Progress in Cryptology - LATINCRYPT 2017 - 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20-22, 2017, Revised Selected Papers*, volume 11368 of *Lecture Notes in Computer Science*, pages 66–85. Springer, 2017. Cited on page 6.

[Leh19] Anja Lehmann. Updatable encryption & key rotation (presentation slides), 2019. https://summerschool-croatia.cs.ru.nl/2019/slides/lehmann1.pdf. Cited on page 5.

[LT18a] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716. Springer, 2018. Cited on pages 3, 4, 6, 7, 16, 51, and 56.

[LT18b] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. *IACR Cryptology ePrint Archive, Report 2018/118*, 2018. Cited on pages 5 and 56.

[MS18] Steven Myers and Adam Shull. Practical revocation and key rotation. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 157–178. Springer, 2018. Cited on page 6.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999. Cited on page 50.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001*, pages 196–205. ACM, 2001. Cited on page 32.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978. Cited on page 45.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949. Cited on page 34.

[SNS17] Kouichi Sakurai, Takashi Nishide, and Amril Syalim. Improved proxy re-encryption scheme for symmetric key cryptography. In *International Workshop on Big Data and Information Security, IWBIS 2017, Jakarta, Indonesia, September 23-24, 2017*, pages 105–111. IEEE, 2017. Cited on page 6.

A SHINE0 is IND-ENC -CPA Secure

Theorem 4.2. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE0 be the updatable encryption scheme described in Fig. 25. For any IND-ENC -CPA adversary $\mathcal{A}$ against SHINE0, there exists an adversary $\mathcal{B}_{4.2}$ against DDH such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}(\lambda).$$

Proof. Similarly to the proof of Theorem 4.1, we use the firewall technique and construct hybrid games. For $\mathrm{b} \in \{0,1\}$, define game $\mathcal{G}^{\mathrm{b}}_i$ as $\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}CPA\text{-}b}}_{\mathsf{SHINE0},\,\mathcal{A}}$ except for:

- The game randomly picks two numbers $\mathsf{fwl}_i, \mathsf{fwr}_i$ and if $\mathsf{fwl}_i, \mathsf{fwr}_i$ are not the i-th firewalls, a random bit is returned for b'. This loss is upper bounded by $(n+1)^2$;
- For challenge made in epoch $\tilde{\mathsf{e}}$ with input $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$: If $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ then return a ciphertext of $\bar{\mathrm{M}}_1$, if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ return a ciphertext of $\bar{\mathrm{M}}_0$, and if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ return a ciphertext of $\bar{\mathrm{M}}_{\mathrm{b}}$;
- After $\mathcal{A}$ outputs b': returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition is triggered.

Similarly to the computation in Theorem 4.1, we have

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC\text{-}CPA}}_{\mathsf{SHINE0},\,\mathcal{A}}(\lambda) = (n+1)^2 \cdot \left(\sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]| \right),$$

for some l. Again we need to prove that $|\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]| \leq 2\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},}(\lambda)$.

We construct a reduction $\mathcal{B}_{4.2}$, detailed in Fig. 33, that is playing the standard DDH game and runs $\mathcal{A}_i$. The reduction $\mathcal{B}_{4.2}$ flips a coin b, and simulates $\mathcal{G}^{\mathrm{b}}_i$ by using DDH tuples (X, Y, Z) to output $\mathsf{SHINE0.Enc}(\bar{\mathrm{M}}_{\mathrm{b}})$ in the i-th insulated region. If $\mathcal{A}_i$ guess b correctly, then $\mathcal{B}_{4.2}$ guesses its real DDH tuples, otherwise, $\mathcal{B}_{4.2}$ guess its random DDH tuples. If $\mathcal{B}_{4.2}$ receives a real DDH tuple, then $\mathcal{B}_{4.2}$ perfectly simulates the input of $\mathcal{A}_i$ in $\mathcal{G}^{\mathrm{b}}_i$. If $\mathcal{B}_{4.2}$ receives a random DDH tuple, then $\mathcal{B}_{4.2}$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have that $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}(\lambda) = \frac{1}{2}|\mathbf{Pr}[\mathcal{G}^1_i = 1] - \mathbf{Pr}[\mathcal{G}^0_i = 1]|$.

Reduction $\mathcal{B}_{4.2}$ playing $\mathbf{Exp}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{4.2}}$ in hybrid i

1 : **receive** (g, X, Y, Z)
2 : **do Setup**
3 : $\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1 \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
4 : $\mathsf{phase} \leftarrow 1$
5 : Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$, **get** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
6 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
7 : $\mathsf{twf} \leftarrow 1$ **if**
8 : $\quad \mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
9 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
10 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
11 : $\quad$ **return** b'
12 : **if** $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
13 : $\quad \mathrm{b}' \xleftarrow{\$} \{0,1\}$
14 : $\quad$ **return** b'
15 : **if** $\mathrm{b}' = \mathrm{b}$
16 : $\quad$ **return** 0
17 : **else**
18 : $\quad$ **return** 1

Setup(λ)

19 : $\mathrm{b} \xleftarrow{\$} \{0,1\}; \mathrm{k}_0 \leftarrow \mathsf{SHINE0.KG}(\lambda)$
20 : $\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
21 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
22 : $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
23 : $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y$
24 : **for** $j \in \{\mathsf{fwl}_i{+}1, \ldots, \mathsf{fwr}_i\}$ **do**
25 : $\quad \Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$
26 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i{-}1\} \cup \{\mathsf{fwr}_i{+}1, \ldots, n\}$ **do**
27 : $\quad \mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \dfrac{\mathrm{k}_j}{\mathrm{k}_{j-1}} \overset{\bowtie}{;} \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

28 : $r \xleftarrow{\$} \mathbb{Z}_q^*; \pi(\mathrm{N}||\mathrm{M}) \leftarrow g^r; \mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
29 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; r)\}$
30 : **return** C_{e}

$\mathcal{O}.\mathsf{Next}$

31 : $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$

32 : **if** $(\cdot, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1; r) \notin \mathcal{L}$
33 : $\quad$ **return** $\perp$
34 : $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
35 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; r)\}$
36 : **return** C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

37 : **do** $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
38 : **if** $\mathsf{inp} = \mathsf{key}$
39 : $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
40 : $\quad$ **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
41 : **if** $\mathsf{inp} = \mathsf{token}$
42 : $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
43 : $\quad$ **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ with $(\bar{\mathrm{M}}_0, \bar{\mathrm{M}}_1)$

44 : $\pi(\mathrm{N}||\bar{\mathrm{M}}_{\mathrm{b}}) \leftarrow X$
45 : $\pi(\mathrm{N}||\bar{\mathrm{M}}_{\mathrm{b}\oplus 1}) \xleftarrow{\$} \mathbb{G}$
46 : $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
47 : **for** $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
48 : $\quad \tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}_1))^{k_j}$ // left
49 : **for** $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
50 : $\quad \tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$ // embed
51 : **for** $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
52 : $\quad \tilde{\mathrm{C}}_j \leftarrow (\pi(\mathrm{N}||\bar{\mathrm{M}}_0))^{k_j}$ // right
53 : $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
54 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

55 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
56 : **find**$(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
57 : **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$

Figure 33: Reduction $\mathcal{B}_{4.2}$ for proof of Theorem 4.2. Embedding DDH tuples to challenge ciphertexts requires faithful responses to queries within the i-th insulated region. On line 27, $\bowtie$ indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

□

B The BLMR Scheme of Boneh, Lewi, Montgomery, and Raghunathan

We present the original scheme given by Boneh et al. [BLMR13], which we denote by BLMR. The scheme is a direct application of the key-homomorphic PRFs defined in the same paper: the authors observed that the Naor-Reingold-Pinkas PRF [NPR99] is key homomorphic, and presented a number of other constructions based on DLIN and LWE. The updatable encryption scheme BLMR [BLMR13], which is ciphertext-independent and defined in Fig. 34, represented the first UE construction.

$\underline{\mathsf{BLMR.KG}(\lambda)}$

1 : $\mathrm{k_e} \xleftarrow{\$} \mathsf{F.KG}(\lambda)$
2 : **return** $\mathrm{k_e}$

$\underline{\mathsf{BLMR.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})}$

3 : $\Delta_{e+1} \leftarrow \mathrm{k_e} \oplus \mathrm{k_{e+1}}$
4 : **return** Δ_{e+1}

$\underline{\mathsf{BLMR.Enc}(\mathrm{k_e}, \mathrm{M})}$

5 : $\mathrm{N} \xleftarrow{\$} \chi$
6 : $\mathrm{C_e^1} \leftarrow \mathsf{F}(\mathrm{k_e}, \mathrm{N}) \otimes \mathrm{M}$
7 : $\mathrm{C_e} \leftarrow (\mathrm{C_e^1}, \mathrm{N})$
8 : **return** $\mathrm{C_e}$

$\underline{\mathsf{BLMR.Dec}(\mathrm{k_e}, \mathrm{C_e})}$

9 : **parse** $\mathrm{C_e} = (\mathrm{C_e^1}, \mathrm{N})$
10 : $\mathrm{M'} \leftarrow \mathrm{C_e^1} \otimes \mathsf{F}(\mathrm{k_e}, \mathrm{N})$
11 : **return** $\mathrm{M'}$

$\underline{\mathsf{BLMR.Upd}(\Delta_{e+1}, \mathrm{C_e})}$

12 : **parse** $\mathrm{C_e} = (\mathrm{C_e^1}, \mathrm{N})$
13 : $\mathrm{C_{e+1}} \leftarrow (\mathrm{C_e^1} \otimes \mathsf{F}(\Delta_{e+1}, \mathrm{N}), \mathrm{N})$
14 : **return** $\mathrm{C_{e+1}}$

Figure 34: Updatable encryption scheme BLMR [BLMR13] for key-homomorphic PRF F.

To present the schemes and the results in this section, we first need to introduce a definition of a key-homomorphic PRF, and also the regular (left-or-right) IND-CPA security definition for symmetric encryption.

Definition 13 (Key-homomorphic PRF [BLMR13])**.** Let $F : \mathcal{KS} \times \mathcal{X} \to \mathcal{Y}$ be some efficiently-computable function, where $(\mathcal{KS}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are groups. Then, $(F, \oplus, \otimes)$ is a key-homomorphic PRF if F is a PRF, and for every $\mathrm{k_1}, \mathrm{k_2} \in \mathcal{KS}$ and every $x \in \mathcal{X}$, $F(\mathrm{k_1}, x) \otimes F(\mathrm{k_2}, x) = F(\mathrm{k_1} \oplus \mathrm{k_2})$.

Definition 14. Let $\mathsf{SKE} = \{\mathsf{KG}, \mathsf{E}, \mathsf{D}\}$ be an symmetric encryption scheme. Then the IND-CPA advantage of an adversary $\mathcal{A}$ against SKE is defined as

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

where the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}b}}_{\mathsf{SKE},\mathcal{A}}$ is given in Fig. 35.

$\underline{\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}b}}_{\mathsf{SKE},\mathcal{A}}(\lambda)}$

1 : $\mathrm{k} \xleftarrow{\$} \mathsf{KG}$
2 : $(\mathrm{M_0}, \mathrm{M_1}, \mathrm{st}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\lambda)$
3 : **if** $|\mathrm{M_0}| \neq |\mathrm{M_1}|$
4 : **return** $\perp$
5 : $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathrm{k}, \mathrm{M_b})$
6 : $\mathrm{b'} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\tilde{\mathrm{C}})$
7 : **return** $\mathrm{b'}$

$\underline{\mathcal{O}.\mathsf{E}(\mathrm{M})}$

8 : $\mathrm{C} \leftarrow \mathsf{SKE.Enc}(\mathrm{k}, \mathrm{M})$
9 : **return** C

Figure 35: The experiments defining IND-CPA security for symmetric encryption schemes.

Note that BLMR is trivially insecure in terms of xxIND-UPD-atk (in any of its three flavors) since the adversary can gain the epoch key for the epoch preceding the challenge epoch, allowing decryption of the challenge input ciphertexts and consequently a direct comparison of nonce values between these input ciphertexts and the challenge ciphertext.

$\mathsf{BLMR}+.\mathsf{KG}(\lambda)$

$\mathrm{k}^1 \xleftarrow{\$} \mathsf{F.KG}(\lambda)$

$\mathrm{k}^2 \xleftarrow{\$} \mathsf{SKE.KG}(\lambda)$

$\mathrm{k_e} \leftarrow (\mathrm{k}^1, \mathrm{k}^2)$

return $\mathrm{k_e}$

$\mathsf{BLMR}+.\mathsf{TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$

parse $\mathrm{k_e} = (\mathrm{k}^1_{\mathrm{e}}, \mathrm{k}^2_{\mathrm{e}})$

parse $\mathrm{k_{e+1}} = (\mathrm{k}^1_{\mathrm{e+1}}, \mathrm{k}^2_{\mathrm{e+1}})$

$\Delta_{\mathrm{e+1}} \leftarrow (\mathrm{k}^1_{\mathrm{e}} \oplus \mathrm{k}^1_{\mathrm{e+1}}, (\mathrm{k}^2_{\mathrm{e}}, \mathrm{k}^2_{\mathrm{e+1}}))$

return $\Delta_{\mathrm{e+1}}$

$\mathsf{BLMR}+.\mathsf{Enc}(\mathrm{k_e}, \mathrm{M})$

parse $\mathrm{k_e} = (\mathrm{k}^1_{\mathrm{e}}, \mathrm{k}^2_{\mathrm{e}})$

$\mathrm{N} \xleftarrow{\$} \chi$

$\mathrm{C}^1_{\mathrm{e}} \leftarrow \mathsf{F}(\mathrm{k}^1_{\mathrm{e}}, \mathrm{N}) \otimes \mathrm{M}$

$\mathrm{C}^2_{\mathrm{e}} \leftarrow \mathsf{SKE.E}(\mathrm{k}^2_{\mathrm{e}}, \mathrm{N})$

$\mathrm{C_e} \leftarrow (\mathrm{C}^1_{\mathrm{e}}, \mathrm{C}^2_{\mathrm{e}})$

return $\mathrm{C_e}$

$\mathsf{BLMR}+.\mathsf{Dec}(\mathrm{k_e}, \mathrm{C_e}$

parse $\mathrm{k_e} = (\mathrm{k}^1_{\mathrm{e}}, \mathrm{k}^2_{\mathrm{e}})$

parse $\mathrm{C_e} = (\mathrm{C}^1_{\mathrm{e}}, \mathrm{C}^2_{\mathrm{e}})$

$\mathrm{N} \leftarrow \mathsf{SKE.D}(\mathrm{k}^2_{\mathrm{e}}, \mathrm{C}^2_{\mathrm{e}})$

$\mathrm{M}' \leftarrow \mathrm{C}^1_{\mathrm{e}} \otimes \mathsf{F}(\mathrm{k}^1_{\mathrm{e}}, \mathrm{N})$

return M'

$\mathsf{BLMR}+.\mathsf{Upd}(\Delta_{\mathrm{e+1}}, \mathrm{C_e})$

parse $\Delta_{\mathrm{e+1}} = (\Delta'_{\mathrm{e+1}}, (\mathrm{k}^2_{\mathrm{e}}, \mathrm{k}^2_{\mathrm{e+1}}))$

parse $\mathrm{C_e} = (\mathrm{C}^1_{\mathrm{e}}, \mathrm{C}^2_{\mathrm{e}})$

$\mathrm{N} \leftarrow \mathsf{SKE.D}(\mathrm{k}^2_{\mathrm{e}}, \mathrm{C}^2_{\mathrm{e}})$

$\mathrm{C}^1_{\mathrm{e+1}} \leftarrow \mathrm{C}^1_{\mathrm{e}} \otimes \mathsf{F}(\Delta'_{\mathrm{e+1}}, \mathrm{N})$

$\mathrm{C}^2_{\mathrm{e+1}} \leftarrow \mathsf{SKE.E}(\mathrm{k}^2_{\mathrm{e+1}}, \mathrm{N})$

$\mathrm{C_{e+1}} \leftarrow (\mathrm{C}^1_{\mathrm{e+1}}, \mathrm{C}^2_{\mathrm{e+1}})$

return $\mathrm{C_{e+1}}$

Figure 36: Updatable encryption scheme BLMR+ [BLMR13, LT18a] for key-homomorphic PRF F and symmetric key encryption scheme SKE.

LT18 detailed an extension of BLMR, denoted BLMR+, where the nonce is encrypted: this scheme is described in Fig. 36. LT18 showed that BLMR+ is IND-ENC-CPA and weakIND-UPD-CPA secure, however it is not detIND-UE-CPA secure, since the token contains the encryption key for the nonce value. More precisely, the adversary runs as follows:

- Choose some M_0, call $\mathcal{O}.\mathsf{Enc}(\mathrm{M}_0)$ and receive some C.
- Call $\mathcal{O}.\mathsf{Next}$, choose M_1 (that is distinct from M_0), do $\mathcal{O}.\mathsf{Chall}(\mathrm{C}, \mathrm{M}_1)$ and receive $\tilde{\mathrm{C}}$.
- Call $\mathcal{O}.\mathsf{Next}$, call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, call $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, 2)$ and $\mathcal{O}.\mathsf{Corr}(\mathsf{key}, 0)$.
- Do $\mathsf{BLMR}+.\mathsf{Dec}_{\mathrm{k}_0}(\mathrm{C})$ to see its nonce, do $\mathsf{D}_{\mathrm{k}^2_2}(\tilde{\mathrm{C}})$ to see nonce of challenge ciphertext and compare.

This is a very similar attack to the one LT18 used to demonstrate that BLMR+ is not detIND-UPD-CPA secure.

Although BLMR+ is not detIND-UE-CPA secure, we can prove that it is weakIND-UE-CPA secure.

B.1 BLMR+ is weakIND-UE-CPA Secure.

Trivial wins for a weak model. An additional notion weakIND-UPD-CPA was used by LT18 for proving their BLMR+ scheme secure: if the adversary has access to any token or key which could leak the nonce of a challenge input ciphertext, the trivial win flag is triggered if the adversary gains access to any token which could reveal the nonce of a known (version of the) challenge ciphertext (i.e. if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$, then $\mathsf{twf} \leftarrow 1$ if $\exists \mathsf{e} \in \mathcal{C}^*$ such that e **or** $\mathsf{e}+1 \in \mathcal{T}^*$). This is necessary because the token in BLMR+ contains the symmetric keys that enable decryption and re-encryption of the nonce.

Proof technique of Proposition 6. The proof technique is very similar to the proof of weakIND-UPD-CPA security of BLMR+ in LT18 [LT18a]. We consider two situations of the additional requirements of weakIND-UE-CPA security and provide two proofs based on these situations. We only describe the first proof technique as both proofs use the same strategy. We construct hybrid games across each epoch, such that distinguishing the endpoints represents success in the weakIND-UE-CPA game. Suppose $\mathcal{A}_i$ is an adversary trying to distinguish games in hybrid i. We consider a modified hybrid game in which the first element of ciphertexts is a uniformly random element. We can prove that the ability to notice this change is upper bounded by PRF advantage. Then, we conclude the proof by switching out the nonce inside the encryption in the second component: noticing this change is upper bounded by IND-CPA advantage of an adversary against SKE.

Proposition 6. Let BLMR+ be the updatable encryption scheme described in Fig. 36. For any weakIND-UE-CPA adversary $\mathcal{A}$ against UE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}.\mathsf{Enc}$ before it makes its challenge, there exists an IND-CPA adversary $\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}$ against SKE and an PRF adversary $\mathcal{B}_{6a}^{\mathsf{PRF}}$ against F such that

$$\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda) \leq (n+1)^3 \cdot \left(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}_{6a}^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|} \right).$$

Proof. The additional requirement of weakIND-UE-CPA security states: If the adversary knows a secret key or a token in epoch $\mathsf{e}^* \in \mathcal{I}^*$, then for any $\mathsf{e} \in \mathcal{C}^*$, if the adversary corrupts Δ_{e} or $\Delta_{\mathsf{e}+1}$ then the adversary trivially loses, i.e. $\mathsf{twf} \leftarrow 1$. We consider two situations (whether or not $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) = \emptyset$) that might happen. The reduction can flip a coin at the beginning of the simulation to guess which situation the adversary will produce and set up the simulation appropriately.

Situation 1. Suppose the adversary knows a secret key or a token in epoch $\mathsf{e}^* \in \mathcal{I}^*$.

(Step 1.) We construct a sequence of hybrid games. Define game $\mathcal{G}_i$ as $\mathbf{Exp}^{\mathsf{weakIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{BLMR+},\,\mathcal{A}}$ except for:

- The challenge input $(\bar{\mathrm{M}}, \bar{\mathrm{C}})$, called in epoch j. If $j \leq i$ then return a ciphertext that is an update of $\bar{\mathrm{C}}$, if $j > i$ then return a ciphertext that is an encryption of $\bar{\mathrm{M}}$.
- After $\mathcal{A}$ outputs b', returns b' if $\mathsf{twf} \neq 1$.

Similarly the advantage $\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda)$ is upper bounded by $|\mathbf{Pr}[\mathcal{G}_{-1} = 1] - \mathbf{Pr}[\mathcal{G}_n = 1]|$. For any i, we prove that

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \leq \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}_{6b}^{\mathsf{IND\text{-}CPA}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}_{6a}^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}_i$ is an adversary trying to distinguish $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$. For all queries concerning epochs other than i the responses will be equal in either game, so we assume $\mathcal{A}_i$ asks for a challenge ciphertext in epoch i. That means if the adversary corrupts tokens in epoch i or epoch $i+1$, the trivial win condition is met and the adversary loses.

(Step 2.) We consider a modified game $\mathcal{G}_{\mathsf{PRF}}$ which is the same as $\mathcal{G}_i$ except for: the first element of ciphertexts given to the adversary in epoch i is a uniformly random element in $\mathcal{Y}$. More precisely, in epoch i, when $\mathcal{A}_i$ asks for $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Upd}$ or a challenge-equal ciphertext to game $\mathcal{G}^{\mathrm{b}}_{\mathsf{PRF}}$:

- An $\mathcal{O}.\mathsf{Enc}(\mathrm{M})$ query: randomly choose a nonce $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X$, set $X \leftarrow X \cup \{\mathrm{N}\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_i, i; \mathrm{N}, \mathrm{M})\}$. Output C_i.
- An $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{i-1})$ query: proceed if $(\cdot, \mathrm{C}_{i-1}, i-1; \mathrm{N}, \mathrm{M}) \in \mathcal{L}$. If $\mathrm{N} \in X$, then abort the game; otherwise, set $X \leftarrow X \cup \{\mathrm{N}\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_i, i; \mathrm{N}, \mathrm{M})\}$. Output C_i.
- A challenge-equal ciphertext (with the underlying challenge input $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$): proceed if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}}-1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \in \mathcal{L}$. If $\mathrm{N}_1 \in X$, then abort the game; otherwise, set $X \leftarrow X \cup \{\mathrm{N}_1\}$. Randomly choose a nonce $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus X$, set $X \leftarrow X \cup \{\mathrm{N}_0\}$, randomly choose $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$, compute $\tilde{\mathrm{C}}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N}_{\mathrm{b}})$, $(\tilde{\mathrm{C}}_i, i; \mathrm{N}_{\mathrm{b}}, \bar{\mathrm{M}}_{\mathrm{b}}) \in \tilde{\mathcal{L}}$. Output $\tilde{\mathrm{C}}_i$.

We wish to prove that

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \leq \mathbf{Adv}^{\mathcal{G}_{\mathsf{PRF}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}_{6a}^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

If the following results are true, then we have the above result.

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}^1_{\mathsf{PRF}} = 1]| \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}_{6a}^{\mathsf{PRF}}}(\lambda) + \frac{{\mathrm{Q_E}}^2}{|\mathcal{X}|}$$

and

$$|\mathbf{Pr}[\mathcal{G}_{i-1}=1]-\mathbf{Pr}[\mathcal{G}^0_{\mathsf{PRF}}=1]|\leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda)+\frac{\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

We construct an PRF adversary $\mathcal{B}^{\mathsf{PRF}}_{6a}$, detailed in Fig. 37, against F to simulate the responses of queries made by $\mathcal{A}_i$.

The reduction does appropriate bookkeeping for the nonce, message, and ciphertexts in list $\mathcal{L}$. Specifically, in epoch i, $\mathcal{B}^{\mathsf{PRF}}_{6a}$ collects used nonces in list X (initiated as empty set). Initially, the reduction flips a coin $\mathrm{b}\xleftarrow{\$}\{0,1\}$, simulates the challenge response with $\bar{\mathrm{M}}_0$ if $\mathrm{b}=0$; otherwise, simulates the challenge response with $\bar{\mathrm{C}}$. The reduction $\mathcal{B}^{\mathsf{PRF}}_{6a}$ generates all keys and tokens except for k^1_i. In epoch i, $\mathcal{B}^{\mathsf{PRF}}_{6a}$ calls its PRF challenger for help computing $\mathsf{F}(\mathrm{k}^1_i,\mathrm{N})$.

Eventually $\mathcal{B}^{\mathsf{PRF}}_{6a}$ receives b' from $\mathcal{A}_i$, and if $\mathrm{b}'=\mathrm{b}$, then $\mathcal{B}^{\mathsf{PRF}}_{6a}$ guesses that it is interacting with the 'real' PRF, i.e. outputs 0 to the PRF challenger, otherwise $\mathcal{B}^{\mathsf{PRF}}_{6a}$ outputs 1.

When $\mathcal{B}^{\mathsf{PRF}}_{6a}$ interacts with $\mathbf{Exp}^{\mathsf{PRF\text{-}0}}_{F,\,\mathcal{B}^{\mathsf{PRF}}_{6a}}$, the simulation of $\mathcal{G}_{i-1}$ (if $\mathrm{b}=0$) or $\mathcal{G}_i$ (if $\mathrm{b}=1$) is perfect except if a nonce collision during the game has caused an abort, this term is bounded by $\frac{\mathrm{Q_E}^2}{|\mathcal{X}|}$. When $\mathcal{B}^{\mathsf{PRF}}_{6a}$ interacts with $\mathbf{Exp}^{\mathsf{PRF\text{-}1}}_{F,\,\mathcal{B}^{\mathsf{PRF}}_{6a}}$, the simulation of $\mathcal{G}^{\mathrm{b}}_{\mathsf{PRF}}$ to $\mathcal{A}_i$ is perfect. We have the desired result.

(Step 3.) Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}^0_{\mathsf{PRF}}$ from game $\mathcal{G}^1_{\mathsf{PRF}}$. Then we construct a reduction $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$, detailed in Fig. 38, playing the IND-CPA game that runs $\mathcal{A}_i$. We claim that

$$\mathbf{Adv}^{\mathcal{G}_{\mathsf{PRF}}}_{\mathcal{A}_i}(\lambda)\leq \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda).$$

Reduction $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$ generates all keys and tokens except for k_i. In epoch i, $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$ uses the IND-CPA challenger for assistance in computing $\mathsf{SKE.E}(\mathrm{k}^2_i,\mathrm{N})$. In epoch i, the reduction forwards all nonces of $\mathcal{O}.\mathsf{Enc}$ and $\mathcal{O}.\mathsf{Upd}$ to the IND-CPA challenger, and sets the reply in the second part of ciphertext, i.e. C^2_i. For challenge input $(\bar{\mathrm{M}},\bar{\mathrm{C}})$, suppose $\bar{\mathrm{C}}$ has the underlying nonce N_1, $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$ chooses nonce N_0 while encrypting M, sends $(\mathrm{N}_0,\mathrm{N}_1)$ to the IND-CPA challenger as challenge input, and sets the reply in the second part of the challenge ciphertext. The following shows how $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$ simulates the responses of queries made by $\mathcal{A}_i$:

Eventually, $\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}$ sends the guess bit of $\mathcal{A}$ to the IND-CPA challenger. We have the required result.

Situation 2. Suppose the adversary knows none of the secret keys and tokens in epoch $\mathsf{e}^*\in\mathcal{I}^*$.

Since the adversary never knows the nonce in the challenge $\bar{\mathrm{C}}$, we do not need to worry if the adversary knows a token in the challenge epoch or the next epoch will make the adversary trivially win the game.

We use the firewall technique to construct hybrid games: in hybrid i, we embed within the i-th insulated region. This means that to the left of the i-th insulated region the game responds with an update of the challenge input ciphertext and to the right of the i-th insulated region it gives an encryption of the challenge input message. Similarly the advantage $\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda)$ is upper bounded by $(n+1)^2\cdot|\mathbf{Pr}[\mathcal{G}^1_l=1]-\mathbf{Pr}[\mathcal{G}^0_1=1]|$. For any $1\leq i\leq l$, we prove that

$$|\mathbf{Pr}[\mathcal{G}^1_i=1]-\mathbf{Pr}[\mathcal{G}^0_i=1]|\leq \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda)+2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{PRF}}_{6a}}(\lambda)+\frac{2\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}^0_i$ from game $\mathcal{G}^1_i$ in hybrid i.

As the above step 2 proof, we consider a modified hybrid game in which the first element of ciphertexts in epoch fwl_i is a uniformly random element in $\mathcal{Y}$. The difference here is that if $\mathcal{A}_i$ asks for encryption queries or challenge queries in an epoch within the i-th insulated region, the reduction will simulate these queries in epoch fwl_i and then output the updated version (updated from epoch fwl_i to the queried epoch). Since the update algorithm of BLMR+ is deterministic, this simulation is valid.

Similarly we can prove the modified hybrid game is indistinguishable from the original hybrid game, and that the distinguishing advantage is upper bounded by the PRF advantage. Finally, similarly to the above step 3 proof (the difference is the same as the difference mentioned in the former paragraph), the advantage is upper bounded by IND-CPA advantage of SKE. We have the following result:

$$\mathbf{Adv}^{\mathsf{weakIND\text{-}UE\text{-}CPA}}_{\mathsf{BLMR+},\,\mathcal{A}}(\lambda)\leq(n+1)^3\cdot\left(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda)+2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\,\mathcal{B}^{\mathsf{IND\text{-}CPA}}_{6b}}(\lambda)+\frac{2\mathrm{Q_E}^2}{|\mathcal{X}|}\right).$$

□

Reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ playing $\mathbf{Exp}^{\mathsf{PRF\text{-}b}}_{\mathsf{F},\mathcal{B}_{6a}}$ in hybrid i

1 : **do Setup**
2 : $\bar{\mathrm{M}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 : $\mathsf{phase} \leftarrow 1$
4 : Create $\tilde{\mathrm{C}}$ **with** $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$, **get** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
5 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
6 : **if** $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) = \emptyset$
7 : **return** ABORT
8 : $\mathsf{twf} \leftarrow 1$ **if**
9 : $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ **or**
10 : $(\exists \mathsf{e} \in \mathcal{C}^* : \mathsf{e} \in \mathcal{T}^*$ **or** $\mathsf{e}+1 \in \mathcal{T}^*)$
11 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
12 : $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
13 : **return** b'
14 : **if** $\mathrm{b}' = \mathrm{b}$
15 : **return** 0
16 : **else**
17 : **return** 1

Setup(λ)

18 : $\mathrm{b} \xleftarrow{\$} \{0,1\}$
19 : $\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
20 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
21 : **for** $j \in \{0, ..., n\}$ **do**
22 : $\mathrm{k}_j^{\P} \xleftarrow{\$} \mathsf{BLMR+.KG}(\lambda)$
23 : $\Delta_{j+1}^{\diamond} \xleftarrow{\$} \mathsf{BLMR+.TG}(\mathrm{k}_j, \mathrm{k}_{j+1})$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

24 : **if** $\mathsf{e} \neq i$
25 : $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{M})$
26 : **if** $\mathsf{e} = i$
27 : $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
28 : **call** $y \leftarrow \mathcal{O}.f(\mathrm{N})$
29 : $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{M}$ / embed
30 : $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
31 : $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
32 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
33 : **return** C_{e}

$\mathcal{O}.\mathsf{Next}$

34 : $\mathsf{e} \leftarrow \mathsf{e}+1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$

35 : **if** $(\cdot, \mathrm{C}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathrm{N}, \mathrm{M}) \notin \mathcal{L}$ **or** $(\mathsf{e} = i$ **and** $\mathrm{N} \in X)$
36 : **return** $\perp$
37 : **if** $\mathsf{e} \neq i, i+1$
38 : $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}\text{-}1})$
39 : **if** $\mathsf{e} = i+1$
40 : $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{F}(\mathrm{k}_{\mathsf{e}}^1, \mathrm{N}) \otimes \mathrm{M}, \mathsf{SKE.E}(\mathrm{k}_{\mathsf{e}}^2, \mathrm{N}))$
41 : **if** $\mathsf{e} = i$
42 : $X \leftarrow X \cup \{\mathrm{N}\}$
43 : **call** $y \leftarrow \mathcal{O}.f(\mathrm{N})$; $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{M}$ / embed
44 : $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
45 : $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
46 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
47 : **return** C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

48 : **if** $\hat{\mathsf{e}} > \mathsf{e}$ **or** $\mathsf{e} = i$ **or** $(\mathsf{e} = i{+}1$ **and** $\mathsf{inp} = \mathsf{token})$
49 : **return** $\perp$
50 : **if** $\mathsf{inp} = \mathsf{key}$
51 : $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
52 : **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
53 : **if** $\mathsf{inp} = \mathsf{token}$
54 : $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
55 : **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ **with** $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$

56 : **if** $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \notin \mathcal{L}$ **or** $\mathrm{N}_1 \in X$
57 : **return** $\perp$
58 : $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$; $X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
59 : **call** $y_{\mathrm{b}} \leftarrow \mathcal{O}.f(\mathrm{N}_{\mathrm{b}})$; $\tilde{\mathrm{C}}_i^1 \leftarrow y_{\mathrm{b}} \otimes \bar{\mathrm{M}}_{\mathrm{b}}$ / embed
60 : $\tilde{\mathrm{C}}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N}_{\mathrm{b}})$
61 : $\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
62 : **for** $j \in \{0, ..., i-1\}$ **do**
63 : $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{M}}_1, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1))$ / left
64 : **for** $j \in \{i+1, ..., n\}$ **do**
65 : $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{M}}_0, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0))$ / right
66 : $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
67 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

69 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
70 : **find** $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
71 : **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$

Figure 37: Reduction $\mathcal{B}_{6a}^{\mathsf{PRF}}$ for proof of Proposition 6. Recall that in the PRF game in Definition 3, the oracle $\mathcal{O}.f$ responds to query input N with either $\mathsf{F}(\mathrm{k}, \mathrm{N})$ or a random value. On line 22, ¶ indicates k_i^1 are skipped in the generation; on line 23, $\diamond$ indicates Δ_i and Δ_{i+1} are skipped in the generation.

Reduction $\mathcal{B}_{6b}^{\text{IND-CPA}}$ playing $\mathbf{Exp}_{\text{SKE},\mathcal{B}_{6b}}^{\text{IND-CPA}}$ in hybrid i

1 : **do Setup**
2 : $\bar{\mathrm{M}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
3 : $\mathsf{phase} \leftarrow 1$
4 : Create $\tilde{\mathrm{C}}$ **with** $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$, **get** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
5 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
6 : **if if** $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$
7 : **return** ABORT
8 : $\mathsf{twf} \leftarrow 1$**if**
9 : $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ **or**
10 : $(\exists \mathsf{e} \in \mathcal{C}^* : \mathsf{e} \in \mathcal{T}^*$ **or** $\mathsf{e}+1 \in \mathcal{T}^*)$
11 : **if** ABORT occurred **or** $\mathsf{twf} = 1$
12 : $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
13 : **return** b'
14 : **if** $\mathrm{b}' = \mathrm{b}$
15 : **return** 0
16 : **else**
17 : **return** 1

Setup(λ)

18 : $\mathrm{b} \xleftarrow{\$} \{0,1\}$
19 : $\Delta_0 \leftarrow \perp$; $\mathsf{e} \leftarrow 0$; $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
20 : $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$
21 : **for** $j \in \{0, ..., n\}$ **do**
22 : $\mathrm{k}_j^{\circledast} \xleftarrow{\$} \mathsf{BLMR+.KG}(\lambda)$
23 : $\Delta_{j+1}^{\diamond} \xleftarrow{\$} \mathsf{BLMR+.TG}(\mathrm{k}_j, \mathrm{k}_{j+1})$

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

24 : **if** $\mathsf{e} \neq i$
25 : $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{M})$
26 : **if** $\mathsf{e} = i$
27 : $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
28 : $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$
29 : **call** $y \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})$
30 : $\mathrm{C}_i^2 \leftarrow y$ // embed
31 : $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
32 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
33 : **return** C_{e}

$\mathcal{O}.\mathsf{Next}$

34 : $\mathsf{e} \leftarrow \mathsf{e}+1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$

35 : **if** $(\cdot, \mathrm{C}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathrm{N}, \mathrm{M}) \notin \mathcal{L}$ **or** $(\mathsf{e} = i$ **and** $\mathrm{N} \in X)$
36 : **return** $\perp$
37 : **if** $\mathsf{e} \neq i, i+1$
38 : $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}\text{-}1})$
39 : **if** $\mathsf{e} = i+1$
40 : $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{F}(\mathrm{k}_{\mathsf{e}}^1, \mathrm{N}) \otimes \mathrm{M}, \mathsf{SKE.E}(\mathrm{k}_{\mathsf{e}}^2, \mathrm{N}))$
41 : **if** $\mathsf{e} = i$
42 : $X \leftarrow X \cup \{\mathrm{N}\}$; $\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}$
43 : **call** $y \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})$; $\mathrm{C}_i^2 \leftarrow y$ // embed
44 : $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
45 : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{M})\}$
46 : **return** C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

47 : **if** $\hat{\mathsf{e}} > \mathsf{e}$ **or** $\mathsf{e} = i$ **or** $(\mathsf{e} = i$+1 **and** $\mathsf{inp} = \mathsf{token})$
48 : **return** $\perp$
49 : **if** $\mathsf{inp} = \mathsf{key}$
50 : $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
51 : **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
52 : **if** $\mathsf{inp} = \mathsf{token}$
53 : $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
54 : **return** $\Delta_{\hat{\mathsf{e}}}$

Create $\tilde{\mathrm{C}}$ **with** $(\bar{\mathrm{M}}_0, \bar{\mathrm{C}})$

55 : **if** $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{M}}_1) \notin \mathcal{L}$ **or** $\mathrm{N}_1 \in X$
56 : **return** $\perp$
57 : $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$; $X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
58 : $\tilde{\mathrm{C}}_i^1 \xleftarrow{\$} \mathcal{Y}$
59 : **call** CHALL **with** $(\mathrm{N}_0, \mathrm{N}_1)$, **get** $\tilde{\mathrm{C}}_i^2$ // embed
60 : $\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
61 : **for** $j \in \{0, ..., i-1\}$ **do**
62 : $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{M}}_1, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1))$ // left
63 : **for** $j \in \{i+1, ..., n\}$ **do**
64 : $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{M}}_0, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0))$ // right
65 : $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
66 : **return** $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

67 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
68 : **find**$(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
69 : **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$

Figure 38: Reduction $\mathcal{B}_{6b}^{\text{IND-CPA}}$ for proof of Proposition 6; the simulation is almost the same as $\mathcal{B}_{6a}^{\text{PRF}}$ except for the underlined simulations. Recall that in the IND-CPA game in Definition 14, the encryption oracle $\mathcal{O}.\mathsf{E}$ replies to input N with $\mathsf{SKE.Enc}(\mathrm{k}, \mathrm{N})$. On line 22, $\circledast$ indicates k_i is skipped in the generation; on line 23, $\diamond$ indicates Δ_i and Δ_{i+1} are skipped in the generation.

C The RISE Scheme of Lehmann and Tackmann

In this section we discuss the Elgamal-based updatable encryption scheme RISE, developed by Lehmann and Tackmann [LT18a] and given in Fig. 39. LT18 showed[10] that RISE is randIND-ENC and randIND-UPD-CPA, under DDH. KLR19 observed that for RISE, knowledge of an update token allows the storage host to create arbitrary ciphertexts for messages of its choice: this is a very undesirable feature for an UE scheme, and is not possible for SHINE. This emphasizes the importance of ciphertext integrity for UE schemes.

$\mathsf{RISE.KG}(\lambda)$

$x \xleftarrow{\$} \mathbb{Z}_q^*$
$\mathrm{k_e} \leftarrow (x, g^x)$
return $\mathrm{k_e}$

$\mathsf{RISE.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$

parse $\mathrm{k_e} = (x, y)$
parse $\mathrm{k_{e+1}} = (x', y')$
$\Delta_{\mathsf{e}+1} \leftarrow (\frac{x'}{x}, y')$
return $\Delta_{\mathsf{e}+1}$

$\mathsf{RISE.Enc}(\mathrm{k_e}, \mathrm{M})$

parse $\mathrm{k_e} = (x, y)$
$r \xleftarrow{\$} \mathbb{Z}_q$
$\mathrm{C_e} \leftarrow (y^r, g^r \cdot \mathrm{M})$
return $\mathrm{C_e}$

$\mathsf{RISE.Dec}(\mathrm{k_e}, \mathrm{C_e})$

parse $\mathrm{k_e} = (x, y)$
parse $\mathrm{C_e} = (\mathrm{C}_1, \mathrm{C}_2)$
$\mathrm{M}' \leftarrow \mathrm{C}_2 \cdot \mathrm{C}_1^{-1/x}$
return M'

$\mathsf{RISE.Upd}(\Delta_{\mathsf{e}+1}, \mathrm{C_e})$

parse $\Delta_{\mathsf{e}+1} = (\Delta, y')$
parse $\mathrm{C_e} = (\mathrm{C}_1, \mathrm{C}_2)$
$r' \xleftarrow{\$} \mathbb{Z}_q$
$\mathrm{C}'_1 \leftarrow \mathrm{C}_1^{\Delta} \cdot y'^{r'}$
$\mathrm{C}'_2 \leftarrow \mathrm{C}_2 \cdot g^{r'}$
$\mathrm{C_{e+1}} \leftarrow (\mathrm{C}'_1, \mathrm{C}'_2)$
return $\mathrm{C_{e+1}}$

Figure 39: Updatable encryption scheme RISE [LT18a] for λ-bit prime q.

C.1 RISE is randIND-UE-CPA Secure

We show that RISE is randIND-UE-CPA under DDH. First, we adapt (an extended version of) the Oracle-DDH experiment to the epoch-based corruption model found in updatable encryption, in a way that ensures that it still reduces to DDH. In this way, we lift a lot of the bookkeeping and complexity. Then, the reduction from randIND-UE-CPA to this oracle-based game is straightforward. We believe that this two-step proof strategy may be useful for proving security of other UE schemes, under any of the security notions discussed so far.

Oracle-Decision-Diffie Hellman for RISE. We give an experiment $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$, where each exponent represents an epoch key, and corruption of keys and tokens is represented: the game allows the adversary to acquire the difference of two exponents in the form $\mathsf{t}_i = \frac{\mathsf{e}_i}{\mathsf{e}_{i-1}}$. In each 'epoch' the adversary can possibly ask for a challenge via $\mathcal{O}.\mathsf{Chall}$, which returns either a 'real' DDH tuple, with the epoch key defined as one of the exponents, or a random tuple. The game is given in Fig. 40. Just as in the games for UE, the challenger keeps track of the epochs in which the adversary has access to 'updates' of the 'challenge' (via CL^*), and access to the keys (exponents, via EL^*). If these overlap then the adversary can trivially extract from the challenge value whether it is 'real' or 'random' and win, so this is of course ruled out. The syntax also follows the UE games in the sense that once the adversary asks for a challenge, it can only ask for 'later' challenges (i.e. with a higher index) from this oracle – it can of course move this challenge 'backwards' into earlier epochs/indices by applying 'token' t.

Definition 15. Fix a cyclic group $\mathbb{G}$ of prime order q with generator g. The advantage of an algorithm $\mathcal{A}$ solving the *Oracle-Decision Diffie-Hellman for* RISE ($\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$) problem for $\mathbb{G}$ and g is

$$\mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{A}} = \left|\mathbf{Pr}[\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}_{\mathbb{G},\,\mathcal{A}}(\lambda) = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}_{\mathbb{G},\,\mathcal{A}}(\lambda) = 1]\right|$$

where the experiment $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}\mathrm{b}}_{\mathbb{G},\,\mathcal{A}}$ is given in Fig. 40.

[10] On 19th December 2019, the authors updated the full version of their paper [LT18b] to include a new proof of randIND-ENC security, fixing a flaw in the prior proof methodology.

$\mathbf{Exp}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}(\lambda)$

1 : $\mathsf{phase}, i^* \leftarrow 0$
2 : $\mathsf{EL}^*, \mathsf{CL}^* \leftarrow \emptyset$
3 : **if** $\mathrm{b} = 1$
4 : $\quad w_1, ..., w_n \xleftarrow{\$} \mathbb{Z}_q^*$
5 : **else**
6 : $\quad w_1, ..., w_n \leftarrow 0$
7 : $\mathsf{e}_1, ..., \mathsf{e}_n \xleftarrow{\$} \mathbb{Z}_q^*$
8 : $x_1, ..., x_n \xleftarrow{\$} \mathbb{Z}_q^*$
9 : **for** $i \in \{0, ...n\}$ **do**
10 : $\quad \mathsf{s}_i \leftarrow g^{\mathsf{e}_i}$
11 : $\quad X_i \leftarrow g^{x_i}$
12 : $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Open},\mathcal{O}.\mathsf{Difr},\mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, ..., \mathsf{s}_n\})$
13 : **if** $\mathsf{EL}^* \cap \mathsf{CL}^* \neq \emptyset$
14 : $\quad \mathrm{b}' \xleftarrow{\$} \{0, 1\}$
15 : **return** b'

$\mathcal{O}.\mathsf{Open}(i)$

16 : **update** EL^*
17 : **return** e_i

$\mathcal{O}.\mathsf{Difr}(i)$

18 : $\mathsf{t}_i \leftarrow \dfrac{\mathsf{e}_i}{\mathsf{e}_{i-1}}$
19 : **update** $\mathsf{EL}^*, \mathsf{CL}^*$

$\mathcal{O}.\mathsf{Chall}(i)$

20 : **if** $\mathsf{phase} = 1$ **and** $i < i^*$
21 : $\quad$ **return** $\perp$
22 : $\mathsf{CL}^* \leftarrow \mathsf{CL}^* \cup \{i\}$
23 : $Z_i \leftarrow \mathsf{s}_i^{x_i} \cdot g^{w_i}$
24 : **if** $\mathsf{phase} = 0$
25 : $\quad i^* \leftarrow i$
26 : $\quad \mathsf{phase} \leftarrow 1$
27 : **return** (X_i, Z_i)

Figure 40: $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ experiment for $\mathbb{G}$ of order q (λ-bit prime) and generator g.

Proposition 7. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let RISE be the updatable encryption scheme described in Fig. 39. For any randIND-UE-CPA adversary $\mathcal{A}$ against RISE, there exists an adversary $\mathcal{B}_7$ against DDH such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{RISE},\,\mathcal{A}}(\lambda) = 2(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_7}(\lambda).$$

This theorem is proven by Lemmas 7.1 and 7.2.

Lemma 7.1. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g. For any adversary $\mathcal{A}$ against $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$, there exists an adversary $\mathcal{B}_{7.1}$ against DDH such that

$$\mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = (n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{7.1}}(\lambda),$$

where $n+1$ is the number of exponents in the $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ game.

Proof. We use a sequence of game hops and a hybrid argument. Define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}$ except for $\mathcal{O}.\mathsf{Chall}$: if called in index j, if $j < i$ then return a 'real' sample (with $w_j = 0$), and if $j > i$ return a 'random sample' ($w_j \xleftarrow{\$} \mathbb{Z}_q^*$). Thus $\mathcal{G}_0^1$ is $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-1}}_{\mathbb{G}}$, i.e. all challenges result in 'random' DDH tuples, and $\mathcal{G}_n^0$ is $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-0}}_{\mathbb{G}}$, i.e. all challenges result in 'real' DDH tuples. Thus distinguishing $\mathcal{G}_0^1$ from $\mathcal{G}_n^0$ is the task of distinguishing $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-1}}_{\mathbb{G},\,\mathcal{A}}$ from $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-0}}_{\mathbb{G},\,\mathcal{A}}$ for adversary $\mathcal{A}$.

Notice that all queries in $\mathcal{G}_{i-1}^0$ and $\mathcal{G}_i^1$ have the equal responses (For $j \leq i-1$, returns a real sample. For $j > i-1$, returns a random sample). We have $\mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = \sum_{i=0}^{n} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. Then we prove that $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq (n+1)^2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},}(\lambda)$ for any i.

Let $\mathcal{A}_i$ be an adversary trying to distinguish $\mathcal{G}_i^1$ from $\mathcal{G}_i^0$. For all queries concerning epochs other than i the responses will be equal in either game, so we assume that $\mathcal{A}_i$ asks for a challenge ciphertext in epoch i and this is where we will embed in our reduction. We construct a reduction $\mathcal{B}_{7.1}$, detailed in Fig. 41, that is playing the standard DDH game (Fig. 2) and runs $\mathcal{A}_i$. This reduction guesses the locations of the firewalls around the challenge query: if $\mathcal{A}_i$ adds any of the epochs within this insulated region to its EL^* list then the reduction fails. fwl and fwr could take any value in $\{0, ..., n\}$, so this loss is upper bounded by $(n+1)^2$.

Reduction $\mathcal{B}_{7.1}$ playing $\mathbf{Exp}^{\mathsf{DDH\text{-}b}}_{\mathbb{G},\,\mathcal{B}_{7.1}}(\lambda)$ in hybrid i

1 : **receive** (g, X, Y, Z)
2 : $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \ldots, n\}$
3 : $w_{i+1}, \ldots, w_n \xleftarrow{\$} \mathbb{Z}_q^*$
4 : $w_0, \ldots, w_{i-1} \leftarrow 0$
5 : $\mathsf{s}_i \leftarrow Y$
6 : **for** $j \in \{0, \ldots, i\text{-}1\} \cup \{i{+}1, \ldots, n\}$ **do**
7 : $\quad x_j \xleftarrow{\$} \mathbb{Z}_q^*; X_j \leftarrow g^{x_j}$
8 : **for** $j \in \{0, \ldots, \mathsf{fwl}\text{-}1\} \cup \{\mathsf{fwr}{+}1, \ldots, n\}$ **do**
9 : $\quad \mathsf{e}_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{t}_j \leftarrow \frac{\mathsf{e}_j}{\mathsf{e}_{j\text{-}1}}^{\dagger}; \mathsf{s}_j \leftarrow g^{\mathsf{e}_j}$
10 : **for** $j \in \{\mathsf{fwl}{+}1, \ldots, \mathsf{fwr}\}$ **do**
11 : $\quad \mathsf{t}_j \xleftarrow{\$} \mathbb{Z}_q^*$
12 : **for** $j \in \{\mathsf{fwl}, \ldots, i\text{-}1\}$ **do**
13 : $\quad \mathsf{s}_j \leftarrow Y^{-\prod_{k=j+1}^{i} \mathsf{t}_k}$
14 : **for** $j \in \{i{+}1, \ldots, \mathsf{fwr}\}$ **do**
15 : $\quad \mathsf{s}_j \leftarrow Y^{\prod_{k=i+1}^{j} \mathsf{t}_k}$
16 : $\mathrm{b}' \leftarrow \mathcal{A}_i^{\mathcal{O}.\mathsf{Open}, \mathcal{O}.\mathsf{Difr}, \mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\})$
17 : **if** ABORT occurred **then**
18 : $\quad$ **return** $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
19 : **return** b'

$\mathcal{O}.\mathsf{Open}(j)$

20 : **if** $j \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$
21 : $\quad$ **return** ABORT
22 : **return** e_i

$\mathcal{O}.\mathsf{Difr}(j)$

23 : **if** $j \in \{\mathsf{fwl}, \mathsf{fwr}{+}1\}$
24 : $\quad$ **return** ABORT
25 : **return** t_j

$\mathcal{O}.\mathsf{Chall}(j)$

26 : **if** $j = i$
27 : $\quad X_j \leftarrow X$
28 : $\quad Z_j \leftarrow Z$
29 : **else**
30 : $\quad Z_j \leftarrow \mathsf{s}_j{}^{x_j} g^{w_j}$
31 : **return** (X_j, Z_j)

Figure 41: Reduction $\mathcal{B}_{7.1}$ for proof of Lemma 7.1. On line 9, † indicates t_0 and $\mathsf{t}_{\mathsf{fwr}+1}$ are skipped in the computation.

For all challenge queries smaller than i the reduction needs to faithfully respond with a 'real' tuple, that is the exponent of Z_j is the product of the exponent used in s_j and the exponent in X_j. These queries must still be consistent with each other, which is why even though the reduction is free to choose x_j it must compute the correct value of s_j. For challenge queries larger than i the reduction produces a random value.

Note that $\mathcal{A}_i$ will have its own CL^* and EL^* lists and $\mathcal{B}_{7.1}$ will simulate these, however we omit this calculation for readability.

If $\mathcal{B}_{7.1}$ is playing $\mathbf{Exp}^{\mathsf{DDH\text{-}1}}_{\mathbb{G},\,\mathcal{B}_{7.1}}$ then it receives a random tuple from its challenger and thus provides a random response to $\mathcal{O}.\mathsf{Chall}(i)$, creating a perfect simulation of $\mathcal{G}_i^1$ to $\mathcal{A}_i$. If $\mathcal{B}_{7.1}$ is playing $\mathbf{Exp}^{\mathsf{DDH\text{-}0}}_{\mathbb{G},\,\mathcal{B}_{7.1}}$ then its tuple is real, providing a perfect simulation of $\mathcal{G}_i^0$. We have the required result. □

Lemma 7.2. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let RISE be the updatable encryption scheme described in Fig. 39. For any randIND-UE-CPA adversary $\mathcal{A}$ against RISE, there exists an adversary $\mathcal{B}_{7.2}$ against $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE\text{-}CPA}}_{\mathsf{RISE},\,\mathcal{A}}(\lambda) = 2 \cdot \mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{B}_{7.2}}(\lambda).$$

Proof. The reduction $\mathcal{B}_{7.2}$ is given in Fig. 42. The reduction $\mathcal{B}_{7.2}$ is playing $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ game and runs $\mathcal{A}$. $\mathcal{B}_{7.2}$ flips a coin b, and simulates $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{RISE},\,\mathcal{A}}$ by interacting with its own $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ challenger.

To simulate updated non-challenge ciphertexts (i.e. respond to $\mathcal{O}.\mathsf{Upd}$ queries), $\mathcal{B}_{7.2}$ must track the underlying messages for these encryptions so that it can generate valid 'fresh' encryptions using the 'public key' values $\{\mathsf{s}_1, \ldots, \mathsf{s}_n\}$ received from its $\mathcal{O}$-$\mathsf{DDH}^{\mathsf{RISE}}$ challenger. Since updated non-challenge ciphertexts are of the form $\mathrm{C}_{\mathsf{e}} = (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$, where r is a fresh random value, the s_i values allow $\mathcal{B}_{7.2}$ to successfully simulate updated non-challenge ciphertexts.

Reduction $\mathcal{B}_{7.2}$ playing $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{B}_{7.2}}$

$1:$ **receive** $g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\}$
$2:$ $\mathsf{e} \leftarrow 0; \mathsf{phase} \leftarrow 0; \mathcal{L} \leftarrow \emptyset$
$3:$ $\bar{\mathrm{M}}_0, \mathrm{C} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
$4:$ $\mathsf{phase} \leftarrow 1$
$5:$ **if** $(\cdot, \mathrm{C} = (\mathrm{C}_1, \mathrm{C}_2), \tilde{\mathsf{e}} - 1; \bar{\mathrm{M}}_1) \notin \mathcal{L}$
$6:$ **return** $\perp$
$7:$ **call** $(X_{\tilde{\mathsf{e}}}, Z_{\tilde{\mathsf{e}}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\tilde{\mathsf{e}})$
$8:$ $\mathrm{b} \xleftarrow{\$} \{0,1\}$
$9:$ $\tilde{\mathrm{C}} \leftarrow (Z_{\tilde{\mathsf{e}}}, X_{\tilde{\mathsf{e}}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$ // embed
$10:$ $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
$11:$ **if** $\mathrm{b}' = \mathrm{b}$
$12:$ **return** 0
$13:$ **else**
$14:$ **return** 1

$\mathcal{O}.\mathsf{Enc}(\mathrm{M})$

$15:$ $r \xleftarrow{\$} \mathbb{Z}_q^*$
$16:$ $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$
$17:$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{M})\}$
$18:$ **return** C_{e}

$\mathcal{O}.\mathsf{Next}$

$19:$ $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$

$20:$ **if** $(\cdot, \mathrm{C}_{\mathsf{e}-1}, \cdot; \mathrm{M}) \notin \mathcal{L}$
$21:$ **return** $\perp$
$22:$ $r \xleftarrow{\$} \mathbb{Z}_q^*$
$23:$ $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}_{\mathsf{e}}^r, g^r \cdot \mathrm{M})$
$24:$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{M})\}$
$25:$ **return** C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

$26:$ **if** $\hat{\mathsf{e}} > \mathsf{e}$
$27:$ **return** $\perp$
$28:$ **if** $\mathsf{inp} = \mathsf{key}$
$29:$ **call** $\mathsf{e}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Open}(\hat{\mathsf{e}})$
$30:$ $\mathrm{k}_{\hat{\mathsf{e}}} \leftarrow (\mathsf{e}_{\hat{\mathsf{e}}}, g^{\mathsf{e}_{\hat{\mathsf{e}}}})$
$31:$ **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
$32:$ **if** $\mathsf{inp} = \mathsf{token}$
$33:$ **call** $\mathsf{t}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Difr}(\hat{\mathsf{e}})$
$34:$ $\Delta_{\hat{\mathsf{e}}} \leftarrow (\mathsf{t}_{\hat{\mathsf{e}}}, \mathsf{s}_{\hat{\mathsf{e}}})$
$35:$ **return** $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$

$36:$ **call** $(X_{\mathsf{e}}, Z_{\mathsf{e}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\mathsf{e})$
$37:$ $\tilde{\mathrm{C}} \leftarrow (Z_{\mathsf{e}}, X_{\mathsf{e}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$
$38:$ **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$

Figure 42: Reduction $\mathcal{B}_{7.2}$ for proof of Lemma 7.2.

To simulate challenge ciphertext (i.e. respond to challenge query or $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$), $\mathcal{B}_{7.2}$ must embed using its own challenge. Recall that in the $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ experiment in Fig. 40, a call to $\mathcal{O}.\mathsf{Chall}(i)$ will result in a response $Z_i = g^{\mathrm{k}_{\tilde{\mathsf{e}}} x_i}$ or $g^{\mathrm{k}_{\tilde{\mathsf{e}}} x_i + w_i}$, and $X_i = g^{x_i}$. When $\mathcal{B}_{7.2}$ receives a challenge query $(\bar{\mathrm{M}}_0, \mathrm{C})$ (where $\mathrm{C} = (g^{r_2 \mathrm{k}_{\tilde{\mathsf{e}}-1}}, g^{r_2} \bar{\mathrm{M}}_1)$ for some $\bar{\mathrm{M}}_1$) from $\mathcal{A}$, $\mathcal{B}_{7.2}$ tries to simulate $\mathsf{RISE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{M}}_0) = (g^{r_1 \mathrm{k}_{\tilde{\mathsf{e}}}}, g^{r_1} \bar{\mathrm{M}}_0)$ or $\mathsf{RISE.Upd}(\Delta_{\tilde{\mathsf{e}}-1}, \mathrm{C}) = (\mathrm{C}_1^{\Delta_{\tilde{\mathsf{e}}-1}} g^{\mathrm{k}_{\tilde{\mathsf{e}}} r_3}, \mathrm{C}_2 g^{r_3}) = (g^{\mathrm{k}_{\tilde{\mathsf{e}}}(r_2 + r_3)}, g^{r_2 + r_3} \bar{\mathrm{M}}_1)$, where r_1, r_3 are fresh random values. $\mathcal{B}_{7.2}$ will embed $Z_{\tilde{\mathsf{e}}}$ to the first part of the challenge ciphertext and embed $X_{\tilde{\mathsf{e}}}$ to the second part of the challenge ciphertext, i.e. $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} = (Z_{\tilde{\mathsf{e}}}, X_{\tilde{\mathsf{e}}} \cdot \bar{\mathrm{M}}_{\mathrm{b}})$. Similarly, $\mathcal{B}_{7.2}$ can simulate the response of $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ using the same approach.

Eventually, $\mathcal{B}_{7.2}$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{7.2}$ returns 0 to its $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ challenger, otherwise, $\mathcal{B}_{7.2}$ returns 1.

If $\mathcal{B}_{7.2}$ interacting with $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}_{\mathbb{G},\,\mathcal{B}_{7.2}}$, then it perfectly simulates $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}CPA\text{-}b}}_{\mathsf{RISE},\,\mathcal{A}}$ to $\mathcal{A}$. If $\mathcal{B}_{7.2}$ interacting with $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}_{\mathbb{G},\,\mathcal{B}_{7.2}}$, then it wins with probability 1/2. After some computation similar to that in the proof of Theorem 2.3 we have the desired result. □