RISE and SHINE: Fast and Secure Updatable Encryption

Colin Boyd[a], Gareth T. Davies*[b], Kristian Gjøsteen[a] and Yao Jiang[a]

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

December 17, 2019

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 keys or plaintexts to an adversary.

We provide a new and highly efficient updatable encryption scheme called SHINE. Ciphertext generation consists of applying one permutation and one exponentiation (per message block), while updating ciphertexts requires just one exponentiation. We also define a new security notion for updatable encryption schemes that implies prior notions (for schemes with randomized and deterministic updates). We prove that SHINE and the previous best scheme, RISE, are secure under our new definition.

*Work partially conducted while employed at NTNU Trondheim and the University of Paderborn.

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 are 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 be provided 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 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.

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 basic goal of UE is to prevent an adversary, who may obtain keys from certain previous points in time (epochs), from obtaining anything useful regarding ciphertexts stored in the current epoch. In addition to obtaining keys from some earlier epochs, a practical adversary may obtain snapshots of the database during different epochs and may have access to some of the update tokens.

UE must therefore, at the least, ensure that an adversary that learns current ciphertexts learns nothing about their decryption. This can be modelled using a standard indistinguishability game, where the adversary must distinguish between encryptions of two messages of the adversary's choice. (Since we are trying to mitigate key compromises, the modelling is complicated by the need to reveal adversarially chosen keys and tokens in addition to ciphertexts, without making the adversary's job trivial.)

In addition to basic confidentiality, a second natural security goal is a form of *unlinkability* between different epochs arising from the ciphertext update procedure. This is modelled by asking an adversary to

[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].

distinguish updated versions of ciphertexts that existed in a prior epoch. Thus we can regard UE security as requiring security across two different "dimensions": for different ciphertexts stored within one epoch, and between ciphertexts stored across different epochs.

These two properties are unfortunately *not* enough. 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. It is possible for a scheme to meet both of the notions that we have outlined previously, but *not* provide security in this context. In this paper we define a new security notion for UE that, unlike any previous notion, captures the above example and maximally protects ciphertexts across *both* dimensions. We also show that this security notion *implies* previous notions, and therefore we claim it is a more natural choice as the right security notion for updatable encryption.

Efficiency. 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 have designed a novel UE scheme that not only satisfies our strong security definition but also is twice as fast as any previous scheme with a comparable security level.

1.1 Related Work

1.1.1 Security Models for UE.

We regard the sequential, epoch-based corruption model of Lehmann and Tackmann [LT18a] (henceforth 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 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 Klooß et al. [KLR19a] (henceforth KLR19) (UP-REENC-CCA) also considered this problem, in the ciphertext-dependent update and CCA-secure setting respectively. KLR19 [[KLR19a], § Appendix A] stated that "an even stronger notion [than

[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, 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].

IND-UPD] might be desirable: namely that fresh and re-encrypted ciphertexts are indistinguishable (which is not guaranteed by (UP-REENC)" – we will close this gap later on in our paper.

In the full version [BLMR15] of their work, 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 with regards to adaptivity, though 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 do not focus on integrity in our work as it is essentially a tangential property (for schemes using 'public-key' techniques such as ours), however we believe that this is an important property and discuss in Section 8 the links between our construction and their framework.

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 unreasonable 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. In (public-key) proxy re-encryption (PRE), a proxy (server) converts an encryption under some public key to something that is decryptable by some other key. 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 symmetric-Elgamal-based scheme of LT18, named RISE, uses a randomized update algorithm and is proven secure in terms of IND-ENC and randIND-UPD under DDH. These proofs invoke 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 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 believe that focusing on direct constructions is the most promising way to achieve efficient and secure updatable encryption.

1.1.3 Related Primitives

Proxy re-encryption. Proxy re-encryption (PRE) is a mechanism for key rotation, but in the public key setting. In a PRE scheme, a ciphertext that is decryptable by some secret key is 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. Consequently, updating the entire ciphertext may not be essential for a PRE scheme to be deemed secure, and 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.

Tokenization. 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 IND-UE security notion and comprehensively analyzing its relation to other, existing security notions (IND-ENC, IND-UPD). Our new notion is strictly stronger even than combinations of prior notions, both in the randomized- and deterministic-update settings. With these results, we show that our single definition is sufficient to ensure that ciphertexts output by the encryption algorithm are indistinguishable from ciphertexts output by the update algorithm. This not only gives us the unlinkability desired by prior works, but also answers the open question posed by KLR19 mentioned earlier. Fig. 14 describes the relationship between our new notion IND-UE and prior notions.

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 we prove that SHINE is detIND-UE secure under the DDH assumption. Our scheme is more efficient than RISE [LT18a], much more efficient than the CCA-secure schemes of KLR19 [KLR19a], and it is approximately as fast as BLMR [BLMR13]. Note that the latter is not even detIND-UPD-secure. A comparison of efficiency and security of these prior schemes and SHINE is given in Fig. 1.

We also further the understanding of 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

	BLMR [BLMR13]	BLMR+ [BLMR13, LT18a]	RISE [LT18a]	SHINE This work
IND-ENC	✓	✓	✓	✓
IND-UPD	✗	$✓^{\mathsf{weak}}$	$✓^{\mathsf{rand}}$	$✓^{\mathsf{det}}$
IND-UE	✗	$✓^{\mathsf{weak}}$	$✓^{\mathsf{rand}}$	$✓^{\mathsf{det}}$
Enc	$1\mathbf{E} + 1\mathbf{P}$	$1\mathbf{E} + 1\mathbf{P} + 1\mathbf{S}$	$2\mathbf{E}$	$1\mathbf{E} + 1\mathbf{S}$
TG	$1\mathbf{X}$	$1\mathbf{X}$	$1\mathbf{E} + 1\mathbf{I}$	$1\mathbf{I}$
Upd	$1\mathbf{E} + 1\mathbf{P}$	$1\mathbf{E} + 1\mathbf{P} + 2\mathbf{S}$	$2\mathbf{E}$	$1\mathbf{E}$

Figure 1: Comparison of security and efficiency – for encryption Enc, token generation TG and ciphertext update Upd– of updatable encryption schemes. $\mathbf{E}$ denotes exponentiation, $\mathbf{I}$ is inversion, $\mathbf{P}$ is evaluating a key-homomorphic PRF, $\mathbf{S}$ is symmetric encryption and $\mathbf{X}$ is an XOR.

notion. It seems very difficult to design a scheme with a randomized update algorithm that is as fast as SHINE, so our contributions to the study of the deterministic-update setting are significant.

Another important contribution is that we further improve on the existing epoch insulation 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 this kind of schemes, and we expect our new techniques to be useful in the future.

We do not claim that SHINE is CCA-secure (it does certainly not have ciphertext integrity), but we note that unlike for RISE, there are no trivial CCA-attacks against SHINE. We discuss this question further in Section 8.

1.3 Further Discussion

We have had to make a number of practical design decisions for our new UE scheme SHINE. 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. Randomized updates seem to be 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 case the adversary is only forbidden from obtaining one token compared to randIND-UE. We believe that this minor security loss is a small price to pay for the efficiency gain, both in terms of reduced computations in the UE encryption and update algorithms and also improved ciphertext expansion.

Bidirectional 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, the *old* key can be derived from the *new* key. In other words, the update algorithm is *bidirectional*. While *unidirectional* update algorithms are desirable, constructing efficient protocols has so far been elusive, and we study bidirectional schemes in this work also: this has technical implications for how security notions are defined.

Other forms of leakage. Modern cloud storage systems usually maintain a lot of meta-data about ciphertexts. Practitioners may therefore doubt that an updatable encryption scheme as a simple drop-in solution will provide the high levels of security defined in this work and previous formal analysis. However, we are motivated to obtain cryptographic schemes that make it *possible* to design cloud storage solutions for applications demanding strong security properties. If our cryptographic schemes leak valuable information, designing secure cloud storage based on these schemes becomes much harder – our goal is to further the understanding of this leakage.

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.

2 Preliminaries

Pseudocode **return** $b' \overset{?}{=} b$ is used as shorthand for **if** $b' = b$ **then return** 1 // **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 and in Def. 13 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.

```
$\mathbf{Exp}^{\mathsf{DDH}\text{-b}}_{\mathbb{G},\mathcal{A}}(\lambda)$ :
  $x, y, r \xleftarrow{\$} \mathbb{Z}_q$
  $X \leftarrow g^x; Y \leftarrow g^y$
  if $\mathrm{b} = 0$
    $Z \leftarrow g^{xy}$
  else
    $Z \leftarrow g^r$
  $\mathrm{b}' \leftarrow \mathcal{A}(g, X, Y, Z)$
  return $\mathrm{b}'$
```

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

```
$\mathbf{Exp}^{\mathsf{PRF}\text{-b}}_{F,\mathcal{A}}(\lambda)$ :
  if $\mathrm{b} = 0$
    $\mathrm{k} \xleftarrow{\$} \mathcal{K}$
    $f(\cdot) \leftarrow F(\mathrm{k}, \cdot)$
  else
    $f(\cdot) \xleftarrow{\$} \{f : \mathcal{X} \to \mathcal{Y}\}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.f}()$
  return $\mathrm{b}'$

$\mathcal{O}.f(x)$ :
  if $x \notin \mathcal{X}$
    return $\perp$
  else
    return $f(x)$
```

Figure 3: PRF experiment $\mathbf{Exp}^{\mathsf{PRF}\text{-b}}_{F,\mathcal{A}}$

Definition 2 (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. 3.

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. 4.

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

Figure 4: Syntax of algorithms defining an Updatable Encryption scheme UE.

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. The definition of correctness is the same as in [KLR19a]: 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 prior work and our schemes have this property.

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*[3]. 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: we stress that <u>all schemes and definitions</u> of security considered in this paper have <u>bi-directional ciphertext updates</u> and <u>bi-directional key updates</u>.

3 Security Models for Updatable Encryption

We consider a number of indistinguishability-based games for assessing security of updatable encryption schemes. The environment provided the challenger attempts to give as much power as possible to adversary $\mathcal{A}$, then after $\mathcal{A}$ has finished running the challenger computes whether or not any of the actions enabled a trivial win. A generic representation of all security games described in this paper is detailed in Fig. 5. The current epoch is advanced by an adversarial call to $\mathcal{O}.\mathsf{Next}$ – simulating $\mathsf{UE.KG}$ and $\mathsf{UE.TG}$ – and keys and tokens (for the current or any prior epoch) can be corrupted via $\mathcal{O}.\mathsf{Corr}$. The adversary can encrypt arbitrary messages via $\mathcal{O}.\mathsf{Enc}$, and update these 'non-challenge' ciphertexts via $\mathcal{O}.\mathsf{Upd}$. 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 $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ (computing this value is actually done by $\mathcal{O}.\mathsf{Next}$, a call to $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ 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 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 $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$) with entries of form $(\mathsf{c}, \mathrm{C}, \mathsf{e})$, where query identifier c is a counter incremented with each new $\mathcal{O}.\mathsf{Enc}$ query.
- $\tilde{\mathcal{L}}$: List of updated versions of challenge ciphertext (created via $\mathcal{O}.\mathsf{Next}$, received by adversary via $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$), 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 $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$).
- $\mathcal{K}$: 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. 6. For security games such as LT18's $\mathsf{IND\text{-}UPD}$ notion, where the adversary must submit as its challenge two ciphertexts (that it received from $\mathcal{O}.\mathsf{Enc}$) and one is updated, the game must also track in which epochs the adversary has updates of these ciphertexts. Later on we will specify a version of our new $\mathsf{IND\text{-}UE}$ notion that allows the

[3]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.

```
$\underline{\mathbf{Setup}(\lambda)}$
  $k_0 \leftarrow \mathsf{UE.KG}(\lambda)$
  $\Delta_0 \leftarrow \perp$;  $\mathsf{e}, \mathsf{c} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\underline{\mathbf{Exp}^{\mathsf{xxIND\text{-}atk\text{-}b}}_{\mathsf{UE}, \mathcal{A}}}$
  do Setup
  $\mathsf{CHALL} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  Create $\tilde{\mathrm{C}}$;  $\tilde{\mathsf{e}} \leftarrow \mathsf{e}$;  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e})\}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}, \mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if}}$ :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$  or
    $\mathsf{xx} \in \{\mathsf{det}, \mathsf{weak}\}$  and  $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
    $\mathsf{xx} = \mathsf{weak}$  and  $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$  and
                    $(\exists \mathsf{e} \in \mathcal{C}^* \ s.t.\ \mathsf{e} \textbf{ or } \mathsf{e}+1 \in \mathcal{T}^*)$
  if $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
  return $\mathrm{b}'$

$\underline{\mathcal{O}.\mathsf{Enc}(\mathrm{m}):}$
  $\mathsf{c} \leftarrow \mathsf{c} + 1$
  $\mathrm{C} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{m})$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}, \mathsf{e})\}$
  return $\mathrm{C}$

$\underline{\mathcal{O}.\mathsf{Next}():}$
  $\mathrm{k}_{\mathsf{e}+1} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
  $\Delta_{\mathsf{e}+1} \xleftarrow{\$} \mathsf{UE.TG}(\mathrm{k}_{\mathsf{e}}, \mathrm{k}_{\mathsf{e}+1})$
  if $\mathsf{phase} = 1$ then
    $\tilde{\mathrm{C}}_{\mathsf{e}+1} \leftarrow \mathsf{UE.Upd}(\Delta_{\mathsf{e}+1}, \tilde{\mathrm{C}}_{\mathsf{e}})$
    $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\mathsf{e}+1}, \mathsf{e}+1)\}$

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

$\underline{\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}}):}$
  if $\hat{\mathsf{e}} > \mathsf{e}$ then
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

$\underline{\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}:}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e})\}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 5: Generic description of challenger behavior in definitions of security for Updatable Encryption scheme UE and adversary $\mathcal{A}$, where $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}, \mathsf{weak}\}$, $\mathsf{IND\text{-}atk} \in \{\mathsf{IND\text{-}ENC}, \mathsf{IND\text{-}UPD}, \mathsf{IND\text{-}UE}\}$. Trivial win conditions, i.e. how to decide the value of twf, are discussed in Section 3.1.

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 and is discussed further in Section 3.1.

	CHALL Input	CHALL Output (in $\tilde{\mathsf{e}}$)
IND-ENC	$\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1$	$\mathsf{UE.Enc}_{\mathrm{k}_{\tilde{\mathsf{e}}}}(\bar{\mathrm{m}}_0)$ **or** $\mathsf{UE.Enc}_{\mathrm{k}_{\tilde{\mathsf{e}}}}(\bar{\mathrm{m}}_1)$
IND-UPD	$\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$	$\mathsf{UE.Upd}_{\Delta_{\tilde{\mathsf{e}}}}(\bar{\mathrm{C}}_0)$ **or** $\mathsf{UE.Upd}_{\Delta_{\tilde{\mathsf{e}}}}(\bar{\mathrm{C}}_1)$
IND-UE	$\bar{\mathrm{m}}, \bar{\mathrm{C}}$	$\mathsf{UE.Enc}_{\mathrm{k}_{\tilde{\mathsf{e}}}}(\bar{\mathrm{m}})$ **or** $\mathsf{UE.Upd}_{\Delta_{\tilde{\mathsf{e}}}}(\bar{\mathrm{C}})$

Figure 6: Intuitive description of challenge inputs and outputs for updatable encryption security notions. Full definitions are given in Section 3.3 and 4.1.

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 $\mathcal{O}.\mathsf{Corr}$ 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 IND-UPD and IND-UE. Note however that the definitions are not restricted to schemes with deterministic updates: such schemes are simply insecure in terms of randIND-UPD and randIND-UE.

3.1 Trivial Win Conditions

Trivial wins via keys and ciphertexts. 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):

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

Trivial wins via direct updates. The following is for analyzing detIND-UE or detIND-UPD security notions, 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 detIND-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 detIND-UE 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 all found indices into list $\mathcal{I}$. Then compute $\mathcal{I}^*$ as follows:

$$\begin{aligned}
&\mathcal{I}^* \leftarrow \{\mathsf{e} \in \{0, ..., n\} | \mathsf{ChallinputEq(e)} = \mathsf{true}\}\\
&\quad \mathsf{true} \leftarrow \mathsf{ChallinputEq(e)} \iff\\
&\qquad (\mathsf{e} \in \mathcal{I}) \vee (\mathsf{ChallinputEq(e\text{-}1)} \wedge \mathsf{e} \in \mathcal{T}^*) \vee (\mathsf{ChallinputEq(e{+}1)} \wedge \mathsf{e{+}1} \in \mathcal{T}^*)
\end{aligned}$$

Additionally, if the adversary submits two ciphertexts $\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1$ as challenge (as in IND-UPD), 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 $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ is shown in Fig. 8.

Trivial wins for a weak model. An additional notion weakIND-UPD 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. We prove that BLMR+ is weakIND-UPD secure in Section 6.

3.2 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 $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$ 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 5.1.

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 $\tilde{\mathsf{e}}$ [4]. 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*.

Definition 3. An *insulated region* with *firewalls* fwl and fwr is a consecutive sequence of epochs $(\mathsf{fwl}, \dots, \mathsf{fwr})$ for which:

- no key in the sequence of epochs $(\mathsf{fwl}, \dots, \mathsf{fwr})$ is corrupted;
- the tokens Δ_{fwl} and $\Delta_{\mathsf{fwr}+1}$ are not corrupted (if they exist);
- all tokens $(\Delta_{\mathsf{fwl}+1}, \dots, \Delta_{\mathsf{fwr}})$ 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. 7 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 $\mathsf{e} \in \neg(\mathcal{T}^* \cup \mathcal{K}^*)$ then
      $j \leftarrow j + 1$
      $\mathsf{fwl}_j \leftarrow \mathsf{e}$
    if $(\mathsf{e} + 1 \notin \mathcal{T}^*) \wedge (\mathsf{e} \notin \mathcal{K}^*)$ then
      $\mathsf{fwr}_j \leftarrow \mathsf{e}$
      $\mathcal{FW} \leftarrow \{(j, \mathsf{fwl}_j, \mathsf{fwr}_j)\}$
```

Figure 7: Algorithm FW-Find for computing all firewalls.

3.2.1 Example of Epoch Corruption and Trivial Wins

In Fig. 8 we indicate the trivial win conditions and insulated regions for a particular adversarial corruption strategy, in the experiment for IND-UE* (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

[4] 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

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 $\mathsf{FW\text{-}Find}$ 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	$\{7$	$\tilde{\mathrm{e}}\}$	9
Key	$\times$	k_1	$\times$	$\times$	$\times$	$\times$	k_6	$\times$	$\times$	k_9
Token		$\times$	$\times$	Δ_3	Δ_4	$\times$	$\times$	$\times$	Δ_8	$\times$
Challenge ciphertexts	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\tilde{\mathrm{C}}_7$	$\tilde{\mathrm{C}}_8$	$\times$
Challenge input	$\times$	$\bar{\mathrm{C}}$	$\bar{\mathrm{C}}_2$	$\bar{\mathrm{C}}_3$	$\bar{\mathrm{C}}_4$	$\times$	$\times$	$\times$	$\times$	$\times$

Figure 8: An example of trivial win conditions and insulated regions incurred by an adversary playing $\mathsf{IND\text{-}UE}^*$, where $\times$ indicates the keys/tokens/ciphertexts not revealed to the adversary, and $\{\}$ indicates insulated regions.

3.3 Existing Definitions of Security

Here we describe existing security notions for UE security given by LT18, including formal definitions for their $\mathsf{IND\text{-}ENC}$ and $\mathsf{xxIND\text{-}UPD}$ notions. We will define our new security notion in Section 4.1 and compare the relation among all of these notions in Section 4.4.

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 $\mathsf{IND\text{-}ENC}$ advantage of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 5 and Fig. 9.

```
$\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$:
  $\underline{\mathsf{CHALL}}$:
    $(\mathrm{m}_0, \mathrm{m}_1) \leftarrow \mathcal{A}$
    if $|\mathrm{m}_0| \neq |\mathrm{m}_1|$ then
      return $\perp$
    $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathrm{e}}}, \mathrm{m}_{\mathrm{b}})$
    return $\tilde{\mathrm{C}}$
```

Figure 9: Challenge call definition for $\mathsf{IND\text{-}ENC}$ security experiment; the full experiment is defined in Fig. 5.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$:
  $\underline{\mathsf{CHALL}}$:
    $(\bar{\mathrm{C}}_0, \bar{\mathrm{C}}_1) \leftarrow \mathcal{A}$
    if $|\bar{\mathrm{C}}_0| \neq |\bar{\mathrm{C}}_1|$ or $(\bar{\mathrm{C}}_0, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$
     or $(\bar{\mathrm{C}}_1, \tilde{\mathrm{e}}\text{-}1) \notin \mathcal{L}$ then
      return $\perp$
    $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{UE.Upd}(\Delta_{\tilde{\mathrm{e}}}, \bar{\mathrm{C}}_{\mathrm{b}})$
    return $\tilde{\mathrm{C}}$
```

Figure 10: Challenge call definition for $\mathsf{xxIND\text{-}UPD}$ security experiment for $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}, \mathsf{weak}\}$; the full experiment is defined in Fig. 5.

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 $\mathsf{xxIND\text{-}UPD}$ advantage, for $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}, \mathsf{weak}\}$, of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiments $\mathbf{Exp}^{\mathsf{xxIND\text{-}UPD\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ are given in Fig. 5 and Fig. 10.

4 On the Security of Updates

In this section we present a new notion of security for updatable encryption schemes, which we denote IND-UE. This notion captures both security of fresh encryptions (i.e. implies IND-ENC) and unlinkability (i.e. implies IND-UPD). We first explain the new notion and then describe its relation to previous notions.

4.1 A New Definition of Security

In the security game for IND-UE, the adversary submits one message and a ciphertext from an earlier epoch that the adversary received via a call to $\mathcal{O}$.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 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-UE advantage, for $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}, \mathsf{weak}\}$, of an adversary $\mathcal{A}$ against UE is defined as

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

where the experiment $\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}$ is given in Fig. 5 and Fig. 11.

Note that randIND-UE is strictly stronger than detIND-UE, since the adversary has strictly more capabilities.

A generalized version of xxIND-UE, denoted xxIND-UE*, is also given in Fig. 11. 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 is a special case of xxIND-UE*. Under some fairly weak requirements (that all schemes discussed in this paper satisfy) we can prove that xxIND-UE implies xxIND-UE* as well – we prove this result in Section. 4.3.

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$ :
  CHALL :
    $(\bar{\mathrm{m}}, \bar{\mathrm{C}}) \leftarrow \mathcal{A}$
    if $(\bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1) \notin \mathcal{L}$ then
      return $\perp$
    if $\mathrm{b} = 0$ then
      $\tilde{\mathrm{C}} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{m}})$
    else
      $\tilde{\mathrm{C}} \leftarrow \mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathrm{C}})$
    return $\tilde{\mathrm{C}}$
```

```
$\mathbf{Exp}^{\mathsf{xxIND\text{-}UE^*\text{-}b}}_{\mathsf{UE},\,\mathcal{A}}(\lambda)$ :
  CHALL :
    $(\bar{\mathrm{m}}, (\bar{\mathrm{C}}, \mathsf{e}')) \leftarrow \mathcal{A}$
    if $(\bar{\mathrm{C}}, \mathsf{e}') \notin \mathcal{L}$ then
      return $\perp$
    if $\mathrm{b} = 0$ then
      $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{m}})$
    else
      $\tilde{\mathrm{C}}_{\mathsf{e}'} \leftarrow \bar{\mathrm{C}}$
      for $j \in \{\mathsf{e}'+1, \ldots, \tilde{\mathsf{e}}\}$ do
        $\tilde{\mathrm{C}}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{\mathrm{C}}_{j-1})$
    return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
```

Figure 11: Challenge call definition for xxIND-UE and xxIND-UE* security experiments for $\mathsf{xx} \in \{\mathsf{rand}, \mathsf{det}, \mathsf{weak}\}$; the full experiment is defined in Fig. 5.

Remark 1. The definition of xxIND-UE is more concise and intuitively easier to understand than xxIND-UE*, however in Theorem 1.1 and Theorem 1.2 in Section 4.3 we show that xxIND-UE* $\Longleftrightarrow$ xxIND-UE, and in particular our proof techniques mean that all results in this paper that hold for xxIND-UE, also hold for xxIND-UE*, vice versa.

4.2 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. $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{UE.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{m}; r)$.

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 7. [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) $\mathrm{k}_{\mathsf{old}} \overset{\$}{\leftarrow} \mathsf{UE.KG}(\lambda)$ following distributions of Δ are identical:

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

Notice that BLMR, $\mathsf{BLMR+}$, 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. $\mathsf{SimTG} : \Delta \overset{\$}{\leftarrow} \mathcal{TS}$.

Definition 8. [Randomness-preserving [KLR19a]] Let UE be an updatable encryption scheme. We say that $\mathsf{UE.Upd}$ (for UE) is randomness-preserving if the following holds: First, as usually assumed, UE encrypts with uniformly chosen randomness. Second, all keys $(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}}) \overset{\$}{\leftarrow} \mathsf{UE.KG}(\lambda)$, tokens $\Delta^{\mathsf{new}} \overset{\$}{\leftarrow} \mathsf{UE.TG}(\mathrm{k}^{\mathsf{old}}, \mathrm{k}^{\mathsf{new}})$, 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 9. [Update-preserving] Let UE be an updatable encryption scheme. with deterministic update algorithm We say that $\mathsf{UE.Upd}$ (for UE) is update-preserving if the following holds: for any two sequence of key pairs with the same start key and end key $(\mathrm{k}_i, \mathrm{k}_{i+1}, ..., \mathrm{k}_j)$ and $(\mathrm{k}'_i, \mathrm{k}'_{i+1}, ..., \mathrm{k}'_j)$, where $\mathrm{k}_i(= \mathrm{k}'_i), \mathrm{k}_{i+1}, \mathrm{k}'_{i+1}, ..., \mathrm{k}_{j-1}, \mathrm{k}'_{j-1}, \mathrm{k}_j(= \mathrm{k}'_j) \overset{\$}{\leftarrow} \mathsf{UE.KG}(\lambda)$, and tokens $\Delta_l \overset{\$}{\leftarrow} \mathsf{UE.TG}(\mathrm{k}_{l-1}, \mathrm{k}_l)$, $\Delta'_l \overset{\$}{\leftarrow} \mathsf{UE.TG}(\mathrm{k}'_{l-1}, \mathrm{k}'_l)$, f or 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 $l = i+1, ..., j$.

The diagrams in Fig. 12 show how the property works, with the left-hand side indicating keys and tokens and the right-hand side showing ciphertexts.

$$\begin{tikzcd}
& \mathrm{k}_{i+1} \arrow[r, "\Delta_{i+2}, \dots, \Delta_{j-1}"'] & \mathrm{k}_{j-1} \arrow[dr, "\Delta_j"'] & \\
\mathrm{k}_i \arrow[ur, bend left, "\Delta_{i+1}"'] \arrow[dr, bend right, "\Delta'_{i+1}"'] & & & \mathrm{k}_j \\
& \mathrm{k}'_{i+1} \arrow[r, "\Delta'_{i+2}, \dots, \Delta'_{j-1}"'] & \mathrm{k}'_{j-1} \arrow[ur, "\Delta'_j"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathrm{C}_{i+1} \arrow[r, "\Delta_{i+2}, \dots, \Delta_{j-1}"'] & \mathrm{C}_{j-1} \arrow[dr, "\Delta_j"'] & \\
\mathrm{C}_i \arrow[ur, bend left, "\Delta_{i+1}"'] \arrow[dr, bend right, "\Delta'_{i+1}"'] & & & \mathrm{C}_j \\
& \mathrm{C}'_{i+1} \arrow[r, "\Delta'_{i+2}, \dots, \Delta'_{j-1}"'] & \mathrm{C}'_{j-1} \arrow[ur, "\Delta'_j"'] &
\end{tikzcd}$$

Figure 12: Keys and updated ciphertexts in Definition 9

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 10 (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 7) 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.

4.3 IND-UE **implies** IND-UE*

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

randIND-UE implies randIND-UE*.

Theorem 1.1. 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* adversary $\mathcal{A}$ against UE, there exists an randIND-UE adversary $\mathcal{B}_{1.1}$ against UE such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE^*}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{UE},\ \mathcal{B}_{1.1}}(\lambda).$$

Proof. We construct an reduction $\mathcal{B}_{1.1}$: 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 challenger and respond with a valid challenge output to the adversary.

More precisely, when the adversary makes the randIND-UE* challenge query, the reduction make its own randIND-UE 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* game. We have the required result. □

detIND-UE implies detIND-UE*.

Proof technique of Theorem 1.2. 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* game is an epoch in which the detIND-UE 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 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 challenger will stay in epoch $\hat{\mathsf{fwl}}$. When $\mathcal{A}$ makes the detIND-UE* challenge queries (if the trivial win conditions of the detIND-UE* game are not satisfied then the trivial win conditions of the detIND-UE game will be not satisfied as well), the reduction makes its detIND-UE 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 1.2. 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 $\mathsf{detIND\text{-}UE}^*$ adversary $\mathcal{A}$ against UE, there exists an $\mathsf{detIND\text{-}UE}$ adversary $\mathcal{B}_{1.2}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE}^*}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq (n+1)^2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.2}}(\lambda),$$

Proof. We use three steps to prove this result.

(Step 1.) Consider a modified version of $\mathsf{detIND\text{-}UE}^*$. For $\mathrm{b} \in \{0,1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT}_1\text{-}\mathrm{b}}$ to be the same as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE}^*\text{-}\mathrm{b}}$ 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 $\tilde{\mathsf{e}}$, 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

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

(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 $\mathcal{O}.\mathsf{Enc}$ 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.

- challenge, 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 $\mathcal{B}_{1.2}$, detailed in Fig. 13, that is playing the $\mathsf{detIND\text{-}UE}$ game and runs $\mathcal{A}$. We claim that

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.2}}(\lambda).$$

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 $\mathbf{Exp}^{\mathsf{detIND\text{-}UE}^*\text{-}\mathrm{b}}$), 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 $\mathsf{detIND\text{-}UE}$ 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 $\mathsf{detIND\text{-}UE}$ 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 $\mathsf{detIND\text{-}UE}$ 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 $\mathsf{detIND\text{-}UE}$ challenger, $\mathcal{B}_{1.2}$ updates the received ciphertext to the challenge epoch to reply $\mathcal{A}$.

Eventually $\mathcal{B}_{1.2}$ receives b' from $\mathcal{A}$, and simply outputs b' to its $\mathsf{detIND\text{-}UE}$ challenger. When $\mathcal{B}_{1.2}$ interacts with $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}b}}_{\mathsf{UE},\,\mathcal{B}_{1.2}}$, $\mathcal{B}_{1.2}$ can perfectly simulate $\mathbf{Exp}^{\mathsf{INT}_2\text{-}\mathrm{b}}_{\mathsf{UE},\,\mathcal{A}}$ to $\mathcal{A}$. Then we have the required result.

□

```
Reduction $\mathcal{B}_{1.2}$ playing $\mathbf{Exp}^{\mathsf{detIND\text{-}UE}\text{-}b}_{\mathsf{UE},\ \mathcal{B}_{1.2}}$ :
  do Setup
  $\mathcal{FL}, \tilde{\mathcal{FL}}, \mathcal{L}, \tilde{\mathcal{L}} \leftarrow \emptyset$
  $\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \dots, n\}$
  $\bar{\mathrm{m}}, (\bar{\mathrm{C}}, \mathsf{e}') \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do CHALL with $(\bar{\mathrm{m}}, (\bar{\mathrm{C}}, \mathsf{e}'))$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}, \mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
  $\mathsf{twf} \leftarrow 1$ if :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
  if ABORT occurred or $(\cdot, \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}}) \notin \mathcal{FW}$
    or $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
  return $\mathrm{b}'$

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ :
  $\mathsf{c} \leftarrow \mathsf{c} + 1$
  if $\mathsf{e} \notin \{\hat{\mathsf{fwl}}+1, \dots, \hat{\mathsf{fwr}}\}$ then
    call $\mathcal{O}.\mathsf{Enc}(\mathrm{m})$, get $\mathrm{C}_{\mathsf{e}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwl}}+1, \dots, \hat{\mathsf{fwr}}\}$ then
    call $\mathcal{O}.\mathsf{Enc}(\mathrm{m})$, get $\mathrm{C}_{\hat{\mathsf{fwl}}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\hat{\mathsf{fwl}}}, \hat{\mathsf{fwl}})\}$
    for $j \in \{\hat{\mathsf{fwl}}+1, \dots, \mathsf{e}\}$ do
      $\mathrm{C}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathrm{C}_{j-1})$
    $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$ :
  if $\mathsf{e} \in \{1, \dots, \hat{\mathsf{fwl}}\text{-}1\}$ then
    for $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \in \mathcal{L}$ do
      call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$, get $\mathrm{C}_{\mathsf{e}}$
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
    call $\mathcal{O}.\mathsf{Next}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwr}}+1, \dots, n\}$ then
    call $\mathcal{O}.\mathsf{Next}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwl}}, \dots, \hat{\mathsf{fwr}}\text{-}1\}$ then
    $\mathsf{e} \leftarrow \mathsf{e}\text{+}1$
    $\Delta_{\mathsf{e}} \xleftarrow{\$} \mathsf{SimTG}(\lambda)$
  if $\mathsf{e} = \hat{\mathsf{fwr}}$ then
    for $j \in \{\hat{\mathsf{fwl}}+1, \dots, \hat{\mathsf{fwr}}+1\}$ do
      call $\mathcal{O}.\mathsf{Next}$
      for $(\mathsf{c}, \mathrm{C}_{j\text{-}1}, j\text{-}1) \in \mathcal{L}$ do
        call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{j\text{-}1})$, get $\mathrm{C}_j$
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_j, j)\}$
      for $(\tilde{\mathrm{C}}_{j\text{-}1}, j\text{-}1) \in \tilde{\mathcal{L}}$ do
        call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}(\tilde{\mathrm{C}}_{j\text{-}1}))$, get $\tilde{\mathrm{C}}_j$
        $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_j, j)\}$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$ :
  if $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1) \notin \mathcal{L} \cup \mathcal{FL}$ then
    return $\perp$
  if $\mathsf{e} \in \{1, \dots, \hat{\mathsf{fwl}}\}$ or $\mathsf{e} \in \{\hat{\mathsf{fwr}}\text{+}2, \dots, n\}$ then
    call $\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$, get $\mathrm{C}_{\mathsf{e}}$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwl}}\text{+}1, \dots, \hat{\mathsf{fwr}}\}$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{UE.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}-1})$
    $\mathcal{FL} \leftarrow \mathcal{FL} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e})\}$
  if $\mathsf{e} = \hat{\mathsf{fwr}}\text{+}1$ then
    find $(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}) \in \mathcal{L}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}})$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    if $\hat{\mathsf{e}} \in \{1, \dots, \hat{\mathsf{fwl}}\text{-}1\}$ or $\hat{\mathsf{e}} \in \{\hat{\mathsf{fwr}}\text{+}2, \dots, n\}$ then
      call $\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$, get $\Delta_{\hat{\mathsf{e}}}$
    if $\hat{\mathsf{e}} \in \{\hat{\mathsf{fwl}}\text{+}1, \dots, \hat{\mathsf{fwr}}\}$ then
      find $\Delta_{\hat{\mathsf{e}}}$
    return $\Delta_{\hat{\mathsf{e}}}$

do CHALL with $(\bar{\mathrm{m}}, (\bar{\mathrm{C}}, \mathsf{e}'))$ :
  if $\tilde{\mathsf{e}} \notin \{\hat{\mathsf{fwl}}, \dots, \hat{\mathsf{fwr}}\}$ or $(\mathsf{c}, \bar{\mathrm{C}}, \mathsf{e}') \notin \mathcal{L}$ then
    ABORT
  find $(\mathsf{c}, \mathrm{C}_{\hat{\mathsf{fwl}}-1}, \hat{\mathsf{fwl}}-1) \in \mathcal{L}$
  call CHALL with $(\bar{\mathrm{m}}, \mathrm{C}_{\hat{\mathsf{fwl}}-1})$, get $\tilde{\mathrm{C}}_{\hat{\mathsf{fwl}}}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{\mathrm{C}}_{\hat{\mathsf{fwl}}}, \hat{\mathsf{fwl}})\}$
  for $j \in \{\hat{\mathsf{fwl}}\text{+}1, \dots, \hat{\mathsf{fwr}}\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \tilde{\mathrm{C}}_{j-1})$
    $\tilde{\mathcal{FL}} \leftarrow \tilde{\mathcal{FL}} \cup \{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ :
  if $\mathsf{e} \in \{1, \dots, \hat{\mathsf{fwl}}\text{-}1\}$ then
    return $\perp$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwl}}\}$ then
    find $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwl}}\text{+}1, \dots, \hat{\mathsf{fwr}}\}$ then
    find $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{FL}}$
  if $\mathsf{e} \in \{\hat{\mathsf{fwr}}\text{+}1, \dots, n\}$ then
    call $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$, get $\tilde{\mathrm{C}}_{\mathsf{e}}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 13: Reduction $\mathcal{B}_{1.2}$ for proof of Theorem 1.2. The underlined oracles shows how $\mathcal{B}_{1.2}$ simulates the responses of this oracles made by $\mathcal{A}$. $\mathcal{B}_{1.2}$ calls oracles to its detIND-UE challenger, and updates the response to $\mathcal{A}$.

4.4 Relation to Prior Notions

In Fig. 14 we show the relationship between the new and existing UE security notions. Note that our new notion is strictly stronger than the IND-ENC and IND-UPD notions presented by LT18, and is in fact stronger than the combination of the prior notions. The relationships are proven via Theorems 1.3 to 1.8 which follow next. Since similar proof techniques are used in Theorems 1.3, 1.4, 1.5 and 1.6, of these we give full proof details only for Theorem 1.3.

Theorem 1 (Informal Theorem)**.** The relationship among the security notions IND-UE, IND-ENC and IND-UPD are as in Fig. 14. This is proven via Theorems 1.3 to 1.8.

$$\begin{tikzcd}
& \text{IND-ENC+randIND-UPD} \arrow[d, Rightarrow, "/" marking, "\text{Thm. 1.7}"] & & & \text{IND-ENC+detIND-UPD} \arrow[d, Rightarrow, "/" marking, "\text{Thm. 1.7}"] & \\
& \text{randIND-UE} \arrow[rrr, Rightarrow, bend left, "\text{Def. 6}"] \arrow[dl, "\text{Thm. 1.3}"'] \arrow[dr, "\text{Thm. 1.5}"] & & & \text{detIND-UE} \arrow[lll, Rightarrow, bend left, "/" marking, "\text{Thm.1.8}"] \arrow[dl, "\text{Thms. 1.4/1.2}"'] \arrow[dr, "\text{Thm. 1.6}"] & \\
\text{IND-ENC} & & \text{randIND-UPD} & \text{IND-ENC} & & \text{detIND-UPD}
\end{tikzcd}$$

Figure 14: Relations between IND-ENC, IND-UPD and IND-UE.

Theorem 1.3. 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 IND-ENC adversary $\mathcal{A}$ against UE, there exists an randIND-UE adversary $\mathcal{B}_{1.3}$ against UE such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.3}}(\lambda).$$

Proof. We construct a reduction $\mathcal{B}_{1.3}$ running the randIND-UE experiment which will simulate the responses of queries made by the IND-ENC adversary $\mathcal{A}$. To provide a valid non-challenge ciphertext to its own challenger, $\mathcal{B}_{1.3}$ 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 $\mathcal{B}_{1.3}$, and so on.

1. $\mathcal{B}_{1.3}$ chooses $\mathrm{b} \xleftarrow{\$} \{0,1\}$.
2. $\mathcal{B}_{1.3}$ receives the setup parameters from its randIND-UE challenger, chooses $\mathrm{m} \xleftarrow{\$} \mathcal{MS}$ and calls $\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ which returns some C^0. Then $\mathcal{B}_{1.3}$ calls $\mathcal{O}.\mathsf{Next}$ once and sends the setup parameters to $\mathcal{A}$.
3. (a) Whenever $\mathcal{B}_{1.3}$ receives the queries $\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}$ from $\mathcal{A}$, $\mathcal{B}_{1.3}$ sends these queries to its IND-UE challenger, and forwards the responses to $\mathcal{A}$.

   (b) Whenever $\mathcal{O}.\mathsf{Next}$ is called by $\mathcal{A}$, $\mathcal{B}_{1.3}$ 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 $\mathcal{O}.\mathsf{Next}$.
4. At some point, in epoch $\tilde{\mathsf{e}}$ (for its game), $\mathcal{B}_{1.3}$ receives the challenge query $(\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1)$ from $\mathcal{A}$. Then $\mathcal{B}_{1.3}$ sends $(\bar{\mathrm{m}}_{\mathrm{b}}, \mathrm{C}^{\tilde{\mathsf{e}}-1})$ as challenge to its own randIND-UE challenger. After receiving the challenge ciphertext, $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$, from its challenger, $\mathcal{B}_{1.3}$ sends $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ to $\mathcal{A}$.
5. $\mathcal{B}_{1.3}$ continues to answer $\mathcal{A}$'s queries using its own oracles, now including $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$.
6. Finally $\mathcal{B}_{1.3}$ receives the output bit b' from $\mathcal{A}$. If $\mathrm{b} = \mathrm{b}'$ then $\mathcal{B}_{1.3}$ returns 0. Otherwise $\mathcal{B}_{1.3}$ returns 1.

We now bound the advantage of $\mathcal{B}_{1.3}$. The point is that whenever $\mathcal{B}_{1.3}$ returns a random encryption to $\mathcal{A}$, $\mathcal{B}_{1.3}$'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 $\mathcal{B}_{1.3}$ returns a "correct" value to $\mathcal{A}$ (an encryption of $\bar{\mathrm{m}}_0$ or $\bar{\mathrm{m}}_1$), then $\mathcal{B}_{1.3}$'s probability of winning is the same as the probability that $\mathcal{A}$ wins.

First note that, as usual,

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{UE},\mathcal{B}_{1.3}} = |\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}1}}_{\mathsf{UE},\ \mathcal{B}_{1.3}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}0}}_{\mathsf{UE},\ \mathcal{B}_{1.3}} = 1]|.$$

We claim that $\mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}1}}_{\mathsf{UE},\ \mathcal{B}_{1.3}} = 1] = 1/2$ because in this case $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$ is independent of b and so b' must also be independent of b. Then we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{UE},\mathcal{B}_{1.3}} &= \left|\frac{1}{2} - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}0}}_{\mathsf{UE},\ \mathcal{B}_{1.3}} = 1]\right| \\
&= \left|\frac{1}{2} - \left(\frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] + \frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 0]\right)\right| \\
&= \left|\frac{1}{2} - \frac{1}{2}\cdot\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] - \frac{1}{2}\left(1 - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 1]\right)\right| \\
&= \left|\frac{1}{2}\cdot\left(\mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}0}}_{\mathsf{UE},\ \mathcal{A}} = 1] - \mathbf{Pr}[\mathbf{Exp}^{\mathsf{IND\text{-}ENC\text{-}1}}_{\mathsf{UE},\ \mathcal{A}} = 1]\right)\right| \\
&= \frac{1}{2}\cdot\mathbf{Adv}^{\mathsf{IND\text{-}ENC}}_{\mathsf{UE},\ \mathcal{A}}.
\end{aligned}$$

□

A remark on Theorem 1.4. Directly proving that detIND-UE implies IND-ENC is very challenging, and in fact the difficulty is the same as proving that detIND-UE implies detIND-UE*. Since we have proved detIND-UE implies detIND-UE* in Theorem 1.2 in Section 4.3, we do not repeat the similar proof approach here. We can just prove detIND-UE* implies IND-ENC, which is easy. We follow a very similar approach to the proof of Theorem 1.3. The detIND-UE* (reduction) adversary can ask for tokens almost as freely as the IND-ENC adversary without incurring the trivial win conditions. But since there should at least one token, in an epoch before (include) challenge epoch $\tilde{\mathsf{e}}$, is unknown to the adversary, we again have to run our reduction out of step with the IND-ENC adversary (essentially creating an artificial epoch).

Theorem 1.4. 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 IND-ENC adversary $\mathcal{A}$ against UE, there exists a detIND-UE* adversary $\mathcal{B}_{1.4}$ against UE such that

$$\mathbf{Adv}^{\mathsf{IND\text{-}ENC}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) \leq 2\cdot\mathbf{Adv}^{\mathsf{detIND\text{-}UE}^*}_{\mathsf{UE},\ \mathcal{B}_{1.4}}(\lambda).$$

Proof. Similar to the proof strategy in Theorem 1.3, we construct a detIND-UE* adversary $\mathcal{B}_{1.4}$ against UE to simulate the responses to queries made by IND-ENC adversary $\mathcal{A}$. $\mathcal{B}_{1.4}$ chooses $\mathrm{b} \xleftarrow{\$} \{0,1\}$. Since $\mathcal{B}_{1.4}$ is allowed to takes its challenge ciphertext from any epoch in its detIND-UE* 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 1.3, there is now no need for $\mathcal{B}_{1.4}$ to generate a random ciphertext for each epoch. Also $\mathcal{B}_{1.4}$ will never ask for $\mathcal{O}.\mathsf{Upd}(\mathrm{C}^0)$ to its own detIND-UE* challenger.

When receiving the challenge query $(\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1)$ from $\mathcal{A}$, the reduction $\mathcal{B}_{1.4}$ sends $(\bar{\mathrm{m}}_\mathrm{b}, \mathrm{C}^0)$ as challenge to its own detIND-UE* 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* game. The result follows using the same calculation as in the proof of Theorem 1.3.

□

Theorem 1.5. 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 adversary $\mathcal{A}$ against UE, there exists an randIND-UE adversary $\mathcal{B}_{1.5}$ against UE such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UPD}}_{\mathsf{UE},\ \mathcal{A}}(\lambda) \leq 2\cdot\mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{UE},\ \mathcal{B}_{1.5}}(\lambda).$$

Proof. Similarly to the proof of Theorem 1.3, we construct a randIND-UE reduction $\mathcal{B}_{1.5}$ against UE to simulate the responses of queries made by randIND-UPD adversary $\mathcal{A}$. However in this case it is not necessary for the reduction to be out-of-step with the adversary. $\mathcal{B}_{1.5}$ first chooses $\mathrm{b} \xleftarrow{\$} \{0,1\}$. $\mathcal{B}_{1.5}$ 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 $\mathcal{A}$, $\mathcal{B}_{1.5}$ samples a random message m, and sends $(\mathrm{m}, \bar{\mathrm{C}}_\mathrm{b})$ to the randIND-UE challenger. The result follows using a similar calculation to that in the proof of Theorem 1.3.

□

Theorem 1.6. Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme For any detIND-UPD adversary $\mathcal{A}$ against UE, there exists an detIND-UE adversary $\mathcal{B}_{1.6}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UPD}}_{\mathsf{UE},\,\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.6}}(\lambda).$$

Proof. The proof follows exactly the same steps as that of Theorem 1.5, in addition to the observation that in the det 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 1.7. Each of the following hold for xx $\in \{$rand, det$\}$.
(i). Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme and let α_{ENC} be the IND-ENC advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE$'$ such that $\mathcal{A}$'s IND-ENC advantage against UE$'$ is (still) α_{ENC}, and there exists an xxIND-UE adversary $\mathcal{B}$ against UE$'$ with advantage 1.
(ii). Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme and let α_{UPD} be the xxIND-UPD advantage of an adversary $\mathcal{A}$ against UE. Then there exists a modified scheme UE$'$ such that $\mathcal{A}$'s xxIND-UPD advantage against UE$'$ is (still) α_{UPD}, and there exists an xxIND-UE adversary $\mathcal{B}$ against UE$'$ with advantage 1.
(iii). Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme and let α_{ENC} be the IND-ENC advantage of an adversary $\mathcal{A}_{\mathsf{ENC}}$ against UE and α_{UPD} be the xxIND-UPD advantage of an adversary . Then there exists a modified scheme UE$'$ such that $\mathcal{A}_{\mathsf{ENC}}$'s IND-ENC advantage against UE$'$ is (still) α_{ENC}, $\mathcal{A}_{\mathsf{UPD}}$'s xxIND-UPD advantage against UE$'$ is (still) α_{UPD}, and there exists an xxIND-UE 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 the epoch number in which the ciphertext was initially created. This does not affect an adversary's ability to win the IND-ENC or IND-UPD games but trivially breaks IND-UE security. □

A remark on Theorem 1.8. We construct an updatable encryption scheme which is detIND-UE secure but not randIND-UE secure to prove that detIND-UE does not imply randIND-UE. Note that in Section 5 we will prove that SHINE is detIND-UE secure (yet it is trivially not randIND-UE secure), which provides an example to support this result. However the proof that SHINE is detIND-UE 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 1.8. We use a IND-UE secure UE scheme to construct a new UE scheme UE$^{\mathsf{new}}$, where we use a PRF to make a part of the new update algorithm deterministic. Because of this UE$^{\mathsf{new}}$ will not be randIND-UE secure. In order to bound the detIND-UE security of UE$^{\mathsf{new}}$, we need to make sure the newly added deterministic part of the updates will not make Enc(m) and Upd(C) distinguishable – this is where we need the PRF.

Theorem 1.8. Let UE $= \{$UE.KG, UE.TG, UE.Enc, UE.Dec, UE.Upd$\}$ be an updatable encryption scheme, and define a new updatable encryption scheme UE$^{\mathsf{new}}$ in Fig. 15, built using pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. Then, for any detIND-UE adversary $\mathcal{A}$ against 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 adversary $\mathcal{B}_{1.8}$ against UE such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE^{new}},\,\mathcal{A}}(\lambda) \leq (n+1) \cdot (\mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.8}}(\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{F,\,\mathcal{B}^{\mathsf{PRF}}} + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}),$$

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

```
UE^new.KG(λ):
    $\mathrm{k} \xleftarrow{\$} \mathsf{UE.KG}(\lambda)$
    return $\mathrm{k}$

UE^new.TG(k_e, k_{e+1}):
    $\Delta'_{\mathsf{e}+1} \leftarrow \mathsf{UE.TG}(\mathrm{k}_{\mathsf{e}}, \mathrm{k}_{\mathsf{e}+1})$
    $\mathrm{fk}_{\mathsf{e}+1} \xleftarrow{\$} \mathcal{K}$
    return $(\Delta'_{\mathsf{e}+1}, \mathrm{fk}_{\mathsf{e}+1})$

UE^new.Enc(k_e, m):
    $r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
    $\mathrm{C}'_{\mathsf{e}} \xleftarrow{\$} \mathsf{UE.Enc}(\mathrm{k}_{\mathsf{e}}, m)$
    return $(r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$

UE^new.Dec(k_e, C_e):
    parse $\mathrm{C}_{\mathsf{e}} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
    $\mathrm{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathrm{k}_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
    return $\mathrm{m}'$

UE^new.Upd(Δ_{e+1}, C_e):
    parse $\Delta_{\mathsf{e}+1} = (\Delta'_{\mathsf{e}+1}, \mathrm{fk}_{\mathsf{e}+1})$
    parse $\mathrm{C}_{\mathsf{e}} = (r_{\mathsf{e}}, \mathrm{C}'_{\mathsf{e}})$
    $r_{\mathsf{e}+1} \leftarrow F(\mathrm{fk}_{\mathsf{e}+1}, r_{\mathsf{e}})$
    $\mathrm{C}'_{\mathsf{e}+1} \leftarrow \mathsf{UE.Upd}(\Delta'_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}})$
    return $(r_{\mathsf{e}+1}, \mathrm{C}'_{\mathsf{e}+1})$
```

Figure 15: Updatable encryption scheme $\mathsf{UE}^{\mathsf{new}}$ for PRF F and updatable encryption scheme UE.

Proof. $\mathsf{UE}^{\mathsf{new}}$ is not randIND-UE 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 secure.
$\mathsf{UE}^{\mathsf{new}}$ is detIND-UE secure. We proceed in three steps.

(Step 1.)

Consider a modified version of detIND-UE. For $\mathrm{b} \in \{0, 1\}$, define experiments $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ to be the same as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}b}}$ except that the experiments randomly pick $\mathsf{e}^* \leftarrow \{0, ..., n\}$, 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:

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

(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

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

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

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

Suppose $\mathcal{A}$ is an adversary trying to distinguish $\mathbf{Exp}_{\mathsf{UE^{new}},\mathcal{A}}^{\mathsf{INT_2\text{-}b}}$ from $\mathbf{Exp}_{\mathsf{UE^{new}},\mathcal{A}}^{\mathsf{INT_1\text{-}b}}$. We construct a PRF reduction $\mathcal{B}^{\mathsf{PRF}}$, detailed in Fig. 16, 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 (initiated 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 $\mathsf{PRF}$:
  do Setup
  $\bar{\mathrm{m}}, (r, \bar{\mathrm{C}}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do $\mathsf{CHALL}$ with $(\bar{\mathrm{m}}, (r, \bar{\mathrm{C}}))$, 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 $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

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

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

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

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

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

do $\mathsf{CHALL}$ 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 $r \in X$ then
    return $\perp$
  $X \leftarrow X \cup \{r\}$
  if $\mathrm{b} = 0$ then
    $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}} \leftarrow \mathsf{UE^{new}.Enc}(\mathrm{k}_{\tilde{\mathsf{e}}}, \bar{\mathrm{m}})$
  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 $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

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

Figure 16: Reduction $\mathcal{B}^{\mathsf{PRF}}$ for proof of Theorem 1.8, $\mathcal{B}^{\mathsf{PRF}}$ 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. 5. Recall that the PRF advantage in Definition 2, $\mathcal{O}.f$ 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 $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$, and 1 if $\mathcal{A}$ guesses that it is in $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$.

When $\mathcal{B}^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}^{\mathsf{PRF\text{-}0}}$, it can simulate $\mathbf{Exp}^{\mathsf{INT_1\text{-}b}}$ 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 $\mathbf{Exp}^{\mathsf{INT_2\text{-}b}}$, thus we have the required result.

(Step 3.) Now we conclude that the advantage of winning $\mathsf{INT_2}$ is upper bounded by $\mathsf{detIND\text{-}UE}$ advantage (against UE). Suppose an $\mathsf{INT_2}$ adversary $\mathcal{A}$ is trying to attack $\mathsf{UE^{new}}$. We construct a $\mathsf{detIND\text{-}UE}$ reduction $\mathcal{B}_{1.8}$, detailed in Fig. 17, attacking UE and runs $\mathcal{A}$. $\mathcal{B}_{1.8}$ first guesses when $\mathcal{A}$ is going to ask a challenge query (assume e^*) and in that epoch $\mathcal{B}_{1.8}$ does a bookkeeping for the randomness in X (initiated as empty set).

```
Reduction $\mathcal{B}_{1.8}$ playing detIND-UE :
  do Setup
  $\bar{m}, (r, \bar{C}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do CHALL with $(\bar{m}, (r, \bar{C}))$, get $\tilde{C}_{\mathsf{e}^*}$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}, \mathcal{O}.\mathsf{Upd}\tilde{C}}(\tilde{C}_{\mathsf{e}^*})$
  $\mathsf{twf} \leftarrow 1$ if :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
  if ABORT occurred or $\mathsf{twf} = 1$ then
    $b' \xleftarrow{\$} \{0, 1\}$
  return $b'$

Setup($\lambda$)
  $k_0 \leftarrow \mathsf{UE.KG}(\lambda)$;
  $\Delta_0 \leftarrow \perp$;  $\mathsf{e} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$;
  $\mathsf{e}^* \xleftarrow{\$} \{0, ..., n\}$;
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}, X \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(m)$ :
  call $\mathcal{O}.\mathsf{Enc}(m)$, get $C'$
  $r \xleftarrow{\$} \mathcal{X}$
  return $(r, C')$

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

$\mathcal{O}.\mathsf{Upd}((r_{\mathsf{e}-1}, C'_{\mathsf{e}-1}))$ :
  if $(\cdot, (r_{\mathsf{e}-1}, C'_{\mathsf{e}-1}), \mathsf{e} - 1) \notin \mathcal{L}$ then
    return $\perp$
  call $\mathcal{O}.\mathsf{Upd}(C'_{\mathsf{e}-1})$, get $C'_{\mathsf{e}}$
  if $\mathsf{e} \neq \mathsf{e}^*$ then
    $r_{\mathsf{e}} = F(\mathrm{fk}_{\mathsf{e}}, r)$
  if $\mathsf{e} = \mathsf{e}^*$ then
    if $r_{\mathsf{e}-1} \in X$ then
      return ABORT
    $X \leftarrow X \cup \{r_{\mathsf{e}-1}\}$
    $r_{\mathsf{e}} \xleftarrow{\$} \mathcal{X}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, (r_{\mathsf{e}}, C'_{\mathsf{e}}), \mathsf{e})\}$
  return $(r_{\mathsf{e}}, C'_{\mathsf{e}})$

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

do CHALL with $(\bar{m}, (r, \bar{C}))$ :
  if $(\cdot, (r, \bar{C}), \tilde{\mathsf{e}} - 1) \notin \mathcal{L}$ or $\mathsf{e}^* \neq \tilde{\mathsf{e}}$ or $r \in X$ then
    return $\perp$
  $X \leftarrow X \cup \{r\}$
  call CHALL with $(\bar{m}, \bar{C})$, get $\tilde{C}'$          embed
  $\tilde{r}_{\tilde{\mathsf{e}}} \xleftarrow{\$} \mathcal{X}$
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{C}'), \tilde{\mathsf{e}})\}$
  return $(\tilde{r}_{\tilde{\mathsf{e}}}, \tilde{C}')$

$\mathcal{O}.\mathsf{Upd}\tilde{C}$
  call $\mathcal{O}.\mathsf{Upd}\tilde{C}$, get $\tilde{C}'_{\mathsf{e}}$          embed
  $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{((\tilde{r}_{\mathsf{e}}, \tilde{C}'_{\mathsf{e}}), \mathsf{e})\}$
  return $(\tilde{r}_{\mathsf{e}}, \tilde{C}'_{\mathsf{e}})$
```

Figure 17: Reduction $\mathcal{B}_{1.8}$ for proof of Theorem 1.8.

Eventually $\mathcal{B}_{1.8}$ receives b' from $\mathcal{A}$, and simply outputs b'. Then $\mathcal{B}_{1.8}$ perfectly simulates $\mathbf{Exp}^{\mathsf{INT}_2\text{-}b}$ to $\mathcal{A}$. We have the required result

$$\mathbf{Adv}^{\mathsf{INT}_2}_{\mathsf{UE}^{\mathsf{new}},\,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{UE},\,\mathcal{B}_{1.8}}(\lambda).$$

□

5 The SHINE Scheme

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 two flavors: SHINE1 is presented in Fig. 18 and takes in short messages and only uses a single permutation. The second flavor, SHINE2, is given in Fig. 19 is for applications with long messages, and uses a family of permutations.

Our proof of security, given as Theorem 2, bounds an adversary's detIND-UE advantage by DDH, and is provided in the ideal cipher model. The proof refers to SHINE1 but is extendable to SHINE2, and thus we refer to SHINE to mean the family containing both schemes.

The message block in SHINE1 and the final message block in SHINE2 must of course be appropriately padded to allow application of the permutation, and this permutation's block size must allow – via some encoding to group elements – exponentiation by the epoch key. (As far as our proofs are concerned, we just need this object to be any invertible bijection from the message space to the exponentiation group.) In practice, the permutation – which takes as input some message and some initialization vector IV– could be deployed using a variable-output-length sponge construction, or a blockcipher or authenticated encryption scheme with a fixed key and suitably large IV space (for example, AES-GCM-256 with 80-bit IV). We discuss this further in Section 8.

```
SHINE1.KG(λ):
    $k \xleftarrow{\$} \mathbb{Z}_q^*$
    return $k$

SHINE1.TG(k_e, k_{e+1}):
    $\Delta_{e+1} \leftarrow \frac{k_{e+1}}{k_e}$
    return $\Delta_{e+1}$

SHINE1.Enc(k_e, m):
    $\mathsf{IV} \xleftarrow{\$} \mathcal{V}$
    $C_e \leftarrow (\pi(\mathsf{IV}||m))^{k_e}$
    return $C_e$

SHINE1.Dec(k_e, C_e):
    $a \leftarrow \pi^{-1}(C_e^{1/k_e})$
    parse $a$ as $\mathsf{IV}'||m'$
    return $m'$

SHINE1.Upd(Δ_{e+1}, C_e):
    $C_{e+1} \leftarrow (C_e)^{\Delta_{e+1}}$
    return $C_{e+1}$
```

Figure 18: Updatable encryption scheme SHINE1 for permutation π and λ-bit prime q.

5.1 Proof Challenges in Schemes with Deterministic Updates

We now highlight the difficulties in creating security proofs for single-component updatable encryption schemes. In both variants of SHINE all components are raised to the epoch key, so the update mechanism transforms a ciphertext for epoch e to one for e + 1 by raising this value to $\frac{k_{e+1}}{k_e}$. Randomness is used in creation of the initial ciphertext (via IV) but updates are completely deterministic, and thus in any reduction it is necessary to provide consistent ciphertexts to the adversary (i.e. the IV value must be consistent). The (cryptographic) separation gained by using the firewall technique (see Section 3.2 for discussion and definition) assists with producing (updates of) non-challenge ciphertexts, but embedding any challenge value while also providing answers to the $\mathcal{O}$.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.

```
$\mathsf{SHINE2.KG}(\lambda)$ :
  $\mathrm{k} \xleftarrow{\$} \mathbb{Z}_q^*$
  return $\mathrm{k}$

$\mathsf{SHINE2.TG}(\mathrm{k_e}, \mathrm{k_{e+1}})$ :
  $\Delta_{\mathsf{e}+1} \leftarrow \frac{\mathrm{k_{e+1}}}{\mathrm{k_e}}$
  return $\Delta_{\mathsf{e}+1}$

$\mathsf{SHINE2.Enc}(\mathrm{k_e}, \mathrm{m})$ :
  parse $\mathrm{m} = (\mathrm{m}_1, \ldots, \mathrm{m}_n)$
  $\mathsf{IV} \xleftarrow{\$} \mathcal{V}$
  $\mathrm{C}^0 \leftarrow \big(\pi_0(\mathsf{IV})\big)^{\mathrm{k_e}}$
  for $i = 1, \ldots, n$ do
    $\mathrm{C}^i \leftarrow \big(\pi_i(\mathsf{IV} \oplus \mathrm{m}_i)\big)^{\mathrm{k_e}}$
  $\mathrm{C_e} \leftarrow (\mathrm{C}^0, \ldots \mathrm{C}^n)$
  return $\mathrm{C_e}$

$\mathsf{SHINE2.Dec}(\mathrm{k_e}, \mathrm{C_e})$ :
  parse $\mathrm{C_e} = (\mathrm{C}^0, \ldots, \mathrm{C}^n)$
  $a_0 \leftarrow \big(\mathrm{C}^0\big)^{1/\mathrm{k_e}}$
  $\mathsf{IV}' \leftarrow \pi_0^{-1}(a_0)$
  for $i = 1, \ldots, n$ do
    $a_i \leftarrow \big(\mathrm{C}^i\big)^{1/\mathrm{k_e}}$
    $\mathrm{m}'_i \leftarrow \pi_i^{-1}(a_i) \oplus \mathsf{IV}'$
  $\mathrm{m}' \leftarrow (\mathrm{m}'_1, \ldots, \mathrm{m}'_n)$
  return $\mathrm{m}'$

$\mathsf{SHINE2.Upd}(\Delta_{\mathsf{e}+1}, \mathrm{C_e})$ :
  parse $\mathrm{C_e} = (\mathrm{C}^0, \ldots, \mathrm{C}^n)$
  for $i = 0, \ldots, n$ do
    $\mathrm{C}^i_{\mathsf{e}+1} \leftarrow (\mathrm{C}^i_{\mathsf{e}})^{\Delta_{\mathsf{e}+1}}$
  return $\mathrm{C}_{\mathsf{e}+1}$
```

Figure 19: Updatable encryption scheme SHINE2 for family of permutations $\{\pi_0, \ldots, \pi_n\}$ and λ-bit prime q.

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

5.2 Proof Method of Theorem 2: Constructing a Hybrid argument across Insulated Regions

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

Notice that the non-corrupted key space is the union set of all insulated regions, i.e. $\{0, 1, \ldots, n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j, \ldots, \mathsf{fwr}_j\}$. 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

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

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 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. 20, to check if this event happens.

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.

We note that the problem of challenge insulation in schemes with deterministic updates was also observed independently and concurrently 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

```
Check(inp, ê; e; fwl, fwr) :
  if $\hat{\mathsf{e}} > \mathsf{e}$ then
    return $\perp$
  if inp = key and $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$ then
    return ABORT
  if inp = token and $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \mathsf{fwr}+1\}$ then
    return ABORT
```

Figure 20: Algorithm Check, used in proofs in this section. In Check, $\hat{\mathsf{e}}$ is the epoch in the adversary's request, and e is the current epoch.

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 (or IND-ENC) 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.

5.3 SHINE is detIND-UE Secure

In Theorem 2, we show that SHINE is detIND-UE in the ideal cipher model, if DDH holds. The loss incurred by this proof is the normal n^3 and also the number of encryption queries the adversary makes before it makes its challenge: to avoid the issues described above we not only need to guess the location of the challenge firewall 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 that is chosen randomly from all possible key-permutation possibilities of the appropriate length. Since the SHINE scheme exponentiates the output of the permutation by the epoch key to encrypt, our reduction can thus 'program' the transformation from permutation outputs to group elements.

Theorem 2. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE1 be the updatable encryption scheme described in Fig. 18. For any detIND-UE adversary $\mathcal{A}$ against SHINE that asks at most $\mathrm{Q_E}$ queries to $\mathcal{O}$.Enc before it makes its challenge, there exists an adversary $\mathcal{B}_2$ against DDH such that

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE}}_{\mathsf{SHINE},\ \mathcal{A}}(\lambda) \leq 2(n+1)^3 \cdot \mathrm{Q_E} \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\ \mathcal{B}_2}(\lambda).$$

Proof. Play hybrid games. We begin by partitioning non-corrupted key space as follows: $\{0, 1, \ldots, n\} \backslash \mathcal{K}^* = \cup_{(j, \mathsf{fwl}_j, \mathsf{fwr}_j) \in \mathcal{FW}} \{\mathsf{fwl}_j, \ldots, \mathsf{fwr}_j\}$, where fwr_i and fwr_i are firewalls of the i-th insulated region. Recall the definition from Section 3.2 and firewall computing algorithm FW-Find in Fig. 7: $\mathsf{fwl}_i, \mathsf{fwr}_i$ 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 $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}b}}_{\mathsf{SHINE},\ \mathcal{A}}$ 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 $\tilde{\mathsf{e}}$, input $(\bar{\mathrm{m}}, \bar{\mathrm{C}})$): If $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ then return a ciphertext with respect to $\bar{\mathrm{C}}$, if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ return a ciphertext with $\bar{\mathrm{m}}$, and if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ 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 $\mathcal{G}_1^0$ is $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}0}}_{\mathsf{SHINE},\ \mathcal{A}}$, 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 $\mathcal{G}_l^1$ is

$\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}1}}_{\mathsf{SHINE},\ \mathcal{A}}$, 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 $\mathbf{Pr}[\mathcal{G}_i^{\mathrm{b}} = 1|\neg \mathrm{E}] = \frac{1}{2}$ for any $1 \leq i \leq n+1$ and $\mathrm{b} \in \{0,1\}$. Then

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

Thus we have that

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

Notice that all queries in $\mathcal{G}_{i-1}^1$ and $\mathcal{G}_i^0$ 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 $l(\leq n+1)$, $|\mathbf{Pr}[\mathcal{G}_l^1 = 1] - \mathbf{Pr}[\mathcal{G}_1^0 = 1]| \leq \sum_{i=1}^{l} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. We prove that for any $1 \leq i \leq l$, $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq 2\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},}(\lambda)$. 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 $\mathcal{G}_i^0$ from $\mathcal{G}_i^1$. 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 $\mathcal{B}_2$, detailed in Fig. 21, that is playing the standard DDH game and will simulate the responses of queries made by adversary $\mathcal{A}_i$. The reduction $\mathcal{B}_2$ 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. $\mathcal{B}_2$ guesses the values of $\mathrm{h}, \mathsf{fwl}_i, \mathsf{fwr}_i$.
2. $\mathcal{B}_2$ 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. $\mathcal{B}_2$ computes the public values of keys in an epoch:
    - $\mathsf{e} \notin \{\mathsf{fwl}_i, ..., \mathsf{fwr}_i\}$: $\mathcal{B}_2$ computes $\mathsf{PK_e} = g^{\mathrm{k_e}}$;
    - $\mathsf{e} \in \{\mathsf{fwl}_i, ..., \mathsf{fwr}_i\}$: $\mathcal{B}_2$ 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 $\mathcal{B}_2$ uses tokens $\Delta_{\mathsf{fwl}_i+1}, ..., \Delta_{\mathsf{fwr}_i}$ to compute the remaining public key values $\mathsf{PK_e}$ in the insulated region.

To simulate a non-challenge ciphertext that is:

- not the h-th query to $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_2$ 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 $\mathrm{C_e} = \mathsf{PK}_{\mathsf{e}}^r$. 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 $\mathcal{O}.\mathsf{Enc}$: $\mathcal{B}_2$ 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 $\Delta_0, \ldots, \Delta_{\mathsf{fwl}_i - 1}$ 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: $\mathcal{B}_2$ simulates $\mathsf{SHINE.Upd}(\bar{\mathrm{C}})$ using the tokens that it created itself.

- within the i-th insulated region: $\mathcal{B}_2$ simulates $\mathsf{SHINE.Upd}(\bar{\mathrm{C}})$ if $\mathrm{b} = 1$, and simulates $\mathsf{SHINE.Enc}(\bar{\mathrm{m}})$ if $\mathrm{b} = 0$. More precisely, $\mathcal{B}_2$ 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

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

  If $\mathrm{b} = 0$, the reduction will give X to $\pi(\mathsf{IV} || \bar{\mathrm{m}})$, and Z to $\tilde{\mathrm{C}}_{\mathsf{fwl}_i}$ (recall $\mathsf{PK}_{\mathsf{fwl}_i} = Y$) since $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} = \mathsf{SHINE.Enc}(\bar{\mathrm{m}}) = \pi(\mathsf{IV} || \bar{\mathrm{m}})^{\mathrm{k}_{\mathsf{fwl}_i}}$. Furthermore, the reduction uses tokens $\Delta_{\mathsf{fwl}_i + 1}, \ldots, \Delta_{\mathsf{fwr}_i}$ 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: $\mathcal{B}_2$ simulates $\mathsf{SHINE.Enc}(\bar{\mathrm{m}})$ using the keys that it created itself.

Eventually, $\mathcal{B}_2$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_2$ guesses that it has seen real DDH tuples (returns 0 to its DDH challenger), otherwise, $\mathcal{B}_2$ guesses that it has seen random DDH tuples (returns 1).

If $\mathcal{B}_2$ receives a real DDH tuple, then $\mathcal{B}_2$ perfectly simulates the environment of $\mathcal{A}_i$ in $\mathcal{G}_i^{\mathrm{b}}$. If $\mathcal{B}_2$ receives a random DDH tuple, then $\mathcal{B}_2$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 1.3 we have $\mathbf{Adv}_{\mathbb{G}, \mathcal{B}_2}^{\mathsf{DDH}}(\lambda) = \frac{1}{2} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$.

□

5.4 SHINE is IND-ENC Secure.

As a corollary of Theorem 1.4 and Theorem 2, SHINE is IND-ENC – however we can give a tighter proof – eliminating the Q_E term – by directly proving the IND-ENC security of SHINE. The proof follows a similar strategy to that of Theorem 2, with one hybrid for each insulated region.

Proposition 3. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g, and let SHINE1 be the updatable encryption scheme described in Fig. 18. For any IND-ENC adversary $\mathcal{A}$ against SHINE, there exists an adversary $\mathcal{B}_3$ against DDH such that

$$\mathbf{Adv}_{\mathsf{SHINE}, \mathcal{A}}^{\mathsf{IND\text{-}ENC}}(\lambda) \leq 2(n+1)^3 \cdot \mathbf{Adv}_{\mathbb{G}, \mathcal{B}_3}^{\mathsf{DDH}}(\lambda).$$

Proof. Similarly to the proof of Theorem 2, we use the firewall technique and construct hybrid games. For $\mathrm{b} \in \{0, 1\}$, define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}_{\mathsf{SHINE}, \mathcal{A}}^{\mathsf{IND\text{-}ENC\text{-}b}}$ except for:

- The game randomly pick two numbers $\mathsf{fwl}_i, \mathsf{fwr}_i$ 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 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 with $\bar{\mathrm{m}}_1$, if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ return a ciphertext with $\bar{\mathrm{m}}_0$, and if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ return a ciphertext with $\bar{\mathrm{m}}_\mathrm{b}$.

- After $\mathcal{A}$ outputs b': returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition triggers.

```
Reduction $\mathcal{B}_2$ playing DDH in hybrid i:
  receive $(g, X, Y, Z)$
  do Setup
  $\bar{\mathrm{m}}, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do CHALL with $(\bar{\mathrm{m}}, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
  $\mathsf{twf} \leftarrow 1$ if :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
  if ABORT occurred or $\mathsf{twf} = 1$
    or $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$ then
      $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
      return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

Setup($\lambda$)
  $\mathrm{b} \xleftarrow{\$} \{0, 1\}; \mathrm{k}_0 \leftarrow \mathsf{SHINE.KG}(\lambda)$
  $\Delta_0 \leftarrow \perp$;  $\mathsf{e}, \mathsf{c} \leftarrow 0$;  $\mathsf{phase}, \mathsf{twf} \leftarrow 0$;
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
  $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}; \mathrm{h} \xleftarrow{\$} \{1, \ldots, \mathrm{Q_E}\}$
  for $j \in \{0, \ldots, \mathsf{fwl}_i\text{-}1\} \cup \{\mathsf{fwr}_i\text{+}1, \ldots, n\}$ do
    $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}{}^{**}; \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$
  if $\mathrm{b} = 0$ then
    $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y; \mathrm{C} \xleftarrow{\$} \mathbb{G}$
  else
    $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y^{\mathrm{k}_{\mathsf{fwl}_i - 1}}; \mathrm{C} \leftarrow X$
  for $j \in \{\mathsf{fwl}_i\text{+}1, \ldots, \mathsf{fwr}_i\}$ do
    $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ :
  $\mathsf{c} \leftarrow \mathsf{c} + 1$
  if $\mathsf{c} = \mathrm{h}$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}; \mathsf{inf} \leftarrow \mathsf{e}$
  else
    $\mathsf{inf} \xleftarrow{\$} \mathbb{Z}_q^*; \pi(\mathsf{IV}||\mathrm{m}) \leftarrow g^{\mathsf{inf}}; \mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
  return $\mathrm{C}_{\mathsf{e}}$
```

```
$\mathcal{O}.\mathsf{Next}$ :
  $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$ :
  if $(\mathsf{c}, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}) \notin \mathcal{L}$ then
    return $\perp$
  if $\mathsf{c} = \mathrm{h}$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathrm{C}_{\mathsf{e}-1}^{\Delta_{\mathsf{e}}}$
  else
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^{\mathsf{inf}}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

do CHALL with $(\bar{\mathrm{m}}, \bar{\mathrm{C}})$ :
  if $(\mathrm{h}, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$ then
    ABORT
  if $\mathrm{b} = 0$ then
    $\pi(\mathsf{IV}||\bar{\mathrm{m}}) \leftarrow X; \tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
  else
    $\pi(\mathsf{IV}||\bar{\mathrm{m}}) \xleftarrow{\$} \mathbb{G}; \tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z^{\prod_{j=\mathsf{inf}+1}^{\mathsf{fwl}_i - 1} \Delta_j}$
  for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow \bar{\mathrm{C}}^{(\prod_{k=0}^{j} \Delta_k)/(\prod_{k=0}^{\tilde{\mathsf{e}}-1} \Delta_k)}$          *left*
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$          *embed*
  for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathsf{IV}||\bar{\mathrm{m}}))^{k_j}$          *right*
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  find $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 21: Reduction $\mathcal{B}_2$ for proof of Theorem 2. 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 is used to track the fixed additional information over updated ciphertext, it is equal to an epoch if $\mathsf{c} = \mathrm{h}$, otherwise it is a random value used in the encryption and updating. ** indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

Similarly to the computation in Theorem 2, we have

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

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

We construct a reduction $\mathcal{B}_3$, detailed in Fig. 22, that is playing the standard DDH game and runs $\mathcal{A}_i$. The reduction $\mathcal{B}_3$ flips a coin b, and simulates $\mathcal{G}_i^{\mathrm{b}}$ by using DDH tuples (X, Y, Z) to output $\mathsf{SHINE.Enc}(\bar{\mathrm{m}}_{\mathrm{b}})$ in the i-th insulated region. If $\mathcal{A}_i$ guess b correctly, then $\mathcal{B}_3$ guesses its real DDH tuples, otherwise, $\mathcal{B}_3$ guess its random DDH tuples. If $\mathcal{B}_3$ receives a real DDH tuple, then $\mathcal{B}_3$ perfectly simulates the input of $\mathcal{A}_i$ in $\mathcal{G}_i^{\mathrm{b}}$. If $\mathcal{B}_3$ receives a random DDH tuple, then $\mathcal{B}_3$ wins with probability 1/2. After some computation similar to that in the proof of Theorem 1.3 we have that $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_3}(\lambda) = \frac{1}{2}|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. □

```
Reduction $\mathcal{B}_3$ playing DDH in hybrid i:
  receive $(g, X, Y, Z)$
  do Setup
  $\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1 \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1)$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
  $\mathsf{twf} \leftarrow 1$ if :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$
  if ABORT occurred or $\mathsf{twf} = 1$
    or $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

Setup$(\lambda)$
  $\mathrm{b} \xleftarrow{\$} \{0,1\}; \mathrm{k}_0 \leftarrow \mathsf{SHINE.KG}(\lambda)$
  $\Delta_0 \leftarrow \perp;\ \mathsf{e} \leftarrow 0;\ \mathsf{phase}, \mathsf{twf} \leftarrow 0;$
  $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
  $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
  $\mathsf{PK}_{\mathsf{fwl}_i} \leftarrow Y$
  for $j \in \{\mathsf{fwl}_i{+}1, \ldots, \mathsf{fwr}_i\}$ do
    $\Delta_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{PK}_j \leftarrow \mathsf{PK}_{j-1}^{\Delta_j}$
  for $j \in \{0, \ldots, \mathsf{fwl}_i{-}1\} \cup \{\mathsf{fwr}_i{+}1, \ldots, n\}$ do
    $\mathrm{k}_j \xleftarrow{\$} \mathbb{Z}_q^*; \Delta_j \leftarrow \frac{\mathrm{k}_j}{\mathrm{k}_{j-1}}{}^{**}; \mathsf{PK}_j \leftarrow g^{\mathrm{k}_j}$

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ :
  $r \xleftarrow{\$} \mathbb{Z}_q^*; \pi(\mathsf{IV}||\mathrm{m}) \leftarrow g^r; \mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; r)\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$ :
  $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$ :
  if $(\cdot, \mathrm{C}_{\mathsf{e}-1}, \mathsf{e}-1; r) \notin \mathcal{L}$ then
    return $\perp$
  $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{PK}_{\mathsf{e}}^r$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; r)\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{m}}_1)$ :
  $\pi(\mathsf{IV}||\bar{\mathrm{m}}_{\mathrm{b}}) \leftarrow X; \pi(\mathsf{IV}||\bar{\mathrm{m}}_{\mathrm{b}\oplus 1}) \xleftarrow{\$} \mathbb{G}$
  $\tilde{\mathrm{C}}_{\mathsf{fwl}_i} \leftarrow Z$
  for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathsf{IV}||\bar{\mathrm{m}}_1))^{\mathrm{k}_j}$                left
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow \tilde{\mathrm{C}}_{j-1}^{\Delta_j}$                          embed
  for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\pi(\mathsf{IV}||\bar{\mathrm{m}}_0))^{\mathrm{k}_j}$                right
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n}\{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  find $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 22: Reduction $\mathcal{B}_3$ for proof of Proposition 3. Embedding DDH tuples to challenge ciphertexts: moving the challenge ciphertexts $\mathsf{Enc}(\bar{\mathrm{m}}_{\mathrm{b}})$ within the i-th insulated region to be random. ** indicates Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

6 The BLMR Scheme of Boneh et al.

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. 24, represented the first UE construction.

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 11 (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 12. 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. 23.

```
$\mathbf{Exp}^{\mathsf{IND\text{-}CPA\text{-}b}}_{\mathsf{SKE},\ \mathcal{A}}(\lambda)$ :
  $\mathrm{k} \xleftarrow{\$} \mathsf{KG}$
  $(\mathrm{m}_0, \mathrm{m}_1, \mathrm{st}) \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\lambda)$
  if $|\mathrm{m}_0| \neq |\mathrm{m}_1|$ then
    return $\perp$
  $\tilde{\mathrm{C}} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathrm{k}, \mathrm{m}_\mathrm{b})$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{E}}(\tilde{\mathrm{C}})$
  return $\mathrm{b}'$

$\mathcal{O}.\mathsf{E}(\mathrm{m})$ :
  $\mathrm{C} \leftarrow \mathsf{SKE.Enc}(\mathrm{k}, \mathrm{m})$
  return $\mathrm{C}$
```

Figure 23: The experiments defining IND-CPA security for Symmetric Encryption schemes

```
$\mathsf{BLMR.KG}(\lambda)$ :
  $\mathrm{k}_\mathsf{e} \xleftarrow{\$} \mathsf{F.KG}(\lambda)$
  return $\mathrm{k}_\mathsf{e}$

$\mathsf{BLMR.TG}(\mathrm{k}_\mathsf{e}, \mathrm{k}_{\mathsf{e}+1})$ :
  $\Delta_{\mathsf{e}+1} \leftarrow \mathrm{k}_\mathsf{e} \oplus \mathrm{k}_{\mathsf{e}+1}$
  return $\Delta_{\mathsf{e}+1}$

$\mathsf{BLMR.Enc}(\mathrm{k}_\mathsf{e}, \mathrm{m})$ :
  $\mathrm{N} \xleftarrow{\$} \chi$
  $\mathrm{C}^1_\mathsf{e} \leftarrow \mathsf{F}(\mathrm{k}_\mathsf{e}, \mathrm{N}) \otimes \mathrm{m}$
  $\mathrm{C}_\mathsf{e} \leftarrow (\mathrm{C}^1_\mathsf{e}, \mathrm{N})$
  return $\mathrm{C}_\mathsf{e}$

$\mathsf{BLMR.Dec}(\mathrm{k}_\mathsf{e}, \mathrm{C}_\mathsf{e})$ :
  parse $\mathrm{C}_\mathsf{e} = (\mathrm{C}^1_\mathsf{e}, \mathrm{N})$
  $\mathrm{m}' \leftarrow \mathrm{C}^1_\mathsf{e} \otimes \mathsf{F}(\mathrm{k}_\mathsf{e}, \mathrm{N})$
  return $\mathrm{m}'$

$\mathsf{BLMR.Upd}(\Delta_{\mathsf{e}+1}, \mathrm{C}_\mathsf{e})$ :
  parse $\mathrm{C}_\mathsf{e} = (\mathrm{C}^1_\mathsf{e}, \mathrm{N})$
  $\mathrm{C}_{\mathsf{e}+1} \leftarrow (\mathrm{C}^1_\mathsf{e} \otimes \mathsf{F}(\Delta_{\mathsf{e}+1}, \mathrm{N}), \mathrm{N})$
  return $\mathrm{C}_{\mathsf{e}+1}$
```

Figure 24: Updatable encryption scheme BLMR [BLMR13] for key-homomorphic PRF F.

Note that BLMR is trivially insecure in terms of IND-UPD (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.

LT18 detailed an extension of BLMR, denoted BLMR+, where the nonce is encrypted: this scheme is described in Fig. 25. LT18 showed that BLMR+ is IND-ENC and weakIND-UPD secure, however it is not detIND-UE 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 secure.

```
BLMR+.KG(λ) :
    $\mathrm{k}^1 \xleftarrow{\$} \mathsf{F.KG}(\lambda)$
    $\mathrm{k}^2 \xleftarrow{\$} \mathsf{SKE.KG}(\lambda)$
    $\mathrm{k} \leftarrow (\mathrm{k}^1, \mathrm{k}^2)$
    return $\mathrm{k}$

BLMR+.TG(k_e, k_{e+1}) :
    parse $\mathrm{k}_\mathsf{e} = (\mathrm{k}_\mathsf{e}^1, \mathrm{k}_\mathsf{e}^2), \mathrm{k}_{\mathsf{e}+1} = (\mathrm{k}_{\mathsf{e}+1}^1, \mathrm{k}_{\mathsf{e}+1}^2)$
    $\Delta_{\mathsf{e}+1} \leftarrow (\mathrm{k}_\mathsf{e}^1 \oplus \mathrm{k}_{\mathsf{e}+1}^1, (\mathrm{k}_\mathsf{e}^2, \mathrm{k}_{\mathsf{e}+1}^2))$
    return $\Delta_{\mathsf{e}+1}$

BLMR+.Enc(k_e, m) :
    parse $\mathrm{k}_\mathsf{e} = (\mathrm{k}_\mathsf{e}^1, \mathrm{k}_\mathsf{e}^2)$
    $\mathrm{N} \xleftarrow{\$} \chi$
    $\mathrm{C}_\mathsf{e}^1 \leftarrow \mathsf{F}(\mathrm{k}_\mathsf{e}^1, \mathrm{N}) \otimes \mathrm{m}$
    $\mathrm{C}_\mathsf{e}^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_\mathsf{e}^2, \mathrm{N})$
    $\mathrm{C}_\mathsf{e} \leftarrow (\mathrm{C}_\mathsf{e}^1, \mathrm{C}_\mathsf{e}^2)$
    return $\mathrm{C}_\mathsf{e}$

BLMR+.Dec(k_e, C_e) :
    parse $\mathrm{k}_\mathsf{e} = (\mathrm{k}_\mathsf{e}^1, \mathrm{k}_\mathsf{e}^2)$
    parse $\mathrm{C}_\mathsf{e} = (\mathrm{C}_\mathsf{e}^1, \mathrm{C}_\mathsf{e}^2)$
    $\mathrm{N} \leftarrow \mathsf{SKE.D}(\mathrm{k}_\mathsf{e}^2, \mathrm{C}_\mathsf{e}^2)$
    $\mathrm{m}' \leftarrow \mathrm{C}_\mathsf{e}^1 \otimes \mathsf{F}(\mathrm{k}_\mathsf{e}^1, \mathrm{N})$
    return $\mathrm{m}'$

BLMR+.Upd(Δ_{e+1}, C_e) :
    parse $\Delta_{\mathsf{e}+1} = (\Delta'_{\mathsf{e}+1}, (\mathrm{k}_\mathsf{e}^2, \mathrm{k}_{\mathsf{e}+1}^2))$
    parse $\mathrm{C}_\mathsf{e} = (\mathrm{C}_\mathsf{e}^1, \mathrm{C}_\mathsf{e}^2)$
    $\mathrm{N} \leftarrow \mathsf{SKE.D}(\mathrm{k}_\mathsf{e}^2, \mathrm{C}_\mathsf{e}^2)$
    $\mathrm{C}_{\mathsf{e}+1}^1 \leftarrow \mathrm{C}_\mathsf{e}^1 \otimes \mathsf{F}(\Delta'_{\mathsf{e}+1}, \mathrm{N})$
    $\mathrm{C}_{\mathsf{e}+1}^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_{\mathsf{e}+1}^2, \mathrm{N})$
    $\mathrm{C}_{\mathsf{e}+1} \leftarrow (\mathrm{C}_{\mathsf{e}+1}^1, \mathrm{C}_{\mathsf{e}+1}^2)$
    return $\mathrm{C}_{\mathsf{e}+1}$
```

Figure 25: Updatable encryption scheme BLMR+ [BLMR13, LT18a] for key-homomorphic PRF F and symmetric key encryption scheme SKE.

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

6.1 BLMR+ is weakIND-UE Secure.

Proof technique of Proposition 4. The proof technique is very similar the proof of weakIND-UPD security of BLMR+ in LT18 [LT18a]. We consider two situations of the additional requirements of weakIND-UE 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 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 4. Let BLMR+ be the updatable encryption scheme described in Fig. 25. For any weakIND-UE adversary $\mathcal{A}$ against UE that asks at most Q_E queries to $\mathcal{O}.\mathsf{Enc}$ before it makes its challenge, there exists an IND-CPA adversary $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ against SKE and an PRF adversary $\mathcal{B}_4^{\mathsf{PRF}}$ against F such that

$$\mathbf{Adv}^{\mathsf{weakIND\text{-}UE}}_{\mathsf{BLMR+},\, \mathcal{A}}(\lambda) \leq (n+1)^3 \cdot \left(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\, \mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\, \mathcal{B}_4^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q}_\mathrm{E}}^2}{|\mathcal{X}|} \right).$$

Proof. The additional requirement of weakIND-UE 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 $\mathbf{Exp}^{\mathsf{weakIND\text{-}UE\text{-}b}}_{\mathsf{BLMR+},\ \mathcal{A}}$ except for:

- The challenge input $(\bar{\mathrm{m}}, \bar{\mathrm{C}})$, called in epoch j. If $j \le i$ 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 $\mathbf{Adv}^{\mathsf{weakIND\text{-}UE}}_{\mathsf{BLMR+},\ \mathcal{A}}(\lambda)$ is upper bounded by $|\mathbf{Pr}[\mathcal{G}_{-1} = 1] - \mathbf{Pr}[\mathcal{G}_n = 1]|$. For any i, we prove that

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \le \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SKE},\ \mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\ \mathcal{B}_4^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

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

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}_{i-1} = 1]| \le \mathbf{Adv}^{\mathcal{G}_{\mathsf{PRF}}}(\lambda) + 2\mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\ \mathcal{B}_4^{\mathsf{PRF}}}(\lambda) + \frac{2{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

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

$$|\mathbf{Pr}[\mathcal{G}_i = 1] - \mathbf{Pr}[\mathcal{G}^1_{\mathsf{PRF}} = 1]| \le \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\ \mathcal{B}_4^{\mathsf{PRF}}}(\lambda) + \frac{{\mathrm{Q_E}}^2}{|\mathcal{X}|}$$

and

$$|\mathbf{Pr}[\mathcal{G}_{i-1} = 1] - \mathbf{Pr}[\mathcal{G}^0_{\mathsf{PRF}} = 1]| \le \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{F},\ \mathcal{B}_4^{\mathsf{PRF}}}(\lambda) + \frac{{\mathrm{Q_E}}^2}{|\mathcal{X}|}.$$

We construct an PRF adversary $\mathcal{B}_4^{\mathsf{PRF}}$, detailed in Fig. 26, 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 $\mathcal{L}$.

```
Reduction $\mathcal{B}_4^{\mathsf{PRF}}$ playing $\mathsf{PRF}$ in hybrid i:
  do Setup
  $\bar{\mathrm{m}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
  if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) = \emptyset$ then
    return ABORT
  twf ← 1 if :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ or
    $(\exists \mathsf{e} \in \mathcal{C}^*\ s.t.\ \mathsf{e}$ or $\mathsf{e}+1 \in \mathcal{T}^*)$
  if ABORT occurred or $\mathsf{twf} = 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

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

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ :
  if $\mathsf{e} \neq i$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Enc}(\mathrm{k}_{\mathsf{e}}, \mathrm{m})$
  if $\mathsf{e} = i$ then
    $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
    $y \leftarrow \mathcal{O}.f(\mathrm{N})$;  $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{m}$            embed
    $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{m})\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$ :
  $\mathsf{e} \leftarrow \mathsf{e} + 1$
```

```
$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}\text{-}1})$ :
  if $(\cdot, \mathrm{C}_{\mathsf{e}\text{-}1}, \mathsf{e}\text{-}1; \mathrm{N}, \mathrm{m}) \notin \mathcal{L}$ or
    $(\mathsf{e} = i$ and $\mathrm{N} \in X)$ then
    return $\perp$
  if $\mathsf{e} \neq i, i+1$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow \mathsf{BLMR+.Upd}(\Delta_{\mathsf{e}}, \mathrm{C}_{\mathsf{e}\text{-}1})$
  if $\mathsf{e} = i+1$ then
    $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{F}(\mathrm{k}_{\mathsf{e}}^1, \mathrm{N}) \otimes \mathrm{m}, \mathsf{SKE.E}(\mathrm{k}_{\mathsf{e}}^2, \mathrm{N}))$
  if $\mathsf{e} = i$ then
    $X \leftarrow X \cup \{\mathrm{N}\}$
    $y \leftarrow \mathcal{O}.f(\mathrm{N})$;  $\mathrm{C}_i^1 \leftarrow y \otimes \mathrm{m}$            embed
    $\mathrm{C}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$
    $\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \mathsf{e}; \mathrm{N}, \mathrm{m})\}$
  return $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  if $\hat{\mathsf{e}} > \mathsf{e}$ or $\mathsf{e} = i$ or
    $(\mathsf{e} = i{+}1$ and $\mathsf{inp} = \mathsf{token})$ then
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{C}})$ :
  if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{m}}_1) \notin \mathcal{L}$ or $\mathrm{N}_1 \in X$ then
    return $\perp$
  $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$ ;  $X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
  $y_{\mathrm{b}} \leftarrow \mathcal{O}.f(\mathrm{N}_{\mathrm{b}})$; $\tilde{\mathrm{C}}_i^1 \leftarrow y_{\mathrm{b}} \otimes \bar{\mathrm{m}}_{\mathrm{b}}$            embed
  $\tilde{\mathrm{C}}_i^2 \leftarrow \mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N}_{\mathrm{b}})$
  $\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
  for $j \in \{0, ..., i-1\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{m}}_1, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1))$      left
  for $j \in \{i+1, ..., n\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{m}}_0, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0))$      right
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  find $(\tilde{\mathrm{C}}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  return $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 26: Reduction $\mathcal{B}_4^{\mathsf{PRF}}$ for proof of Proposition 4. Recall that in the PRF game in Definition, 2, the oracle $\mathcal{O}.f$ responds to query input N with either $\mathsf{F}(\mathrm{k}, \mathrm{N})$ or a random value. ** indicates k_i^1 are skipped in the generation. *** indicates Δ_i and Δ_{i+1} are skipped in the generation.

Specifically, in epoch i, $\mathcal{B}_4^{\mathsf{PRF}}$ 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 $\mathcal{B}_4^{\mathsf{PRF}}$ generates all keys and tokens except for k_i^1. In epoch i, $\mathcal{B}_4^{\mathsf{PRF}}$ calls its PRF challenger for help computing $\mathsf{F}(\mathrm{k}_i^1, \mathrm{N})$. Eventually $\mathcal{B}_4^{\mathsf{PRF}}$ receives b' from $\mathcal{A}_i$, and if $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_4^{\mathsf{PRF}}$ guesses that it is interacting with the 'real' PRF, i.e. outputs 0 to the PRF challenger, otherwise $\mathcal{B}_4^{\mathsf{PRF}}$ outputs 1.

When $\mathcal{B}_4^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}_{F,\,\mathcal{B}_4^{\mathsf{PRF}}}^{\mathsf{PRF\text{-}0}}$, 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 $\mathcal{B}_4^{\mathsf{PRF}}$ interacts with $\mathbf{Exp}_{F,\,\mathcal{B}_4^{\mathsf{PRF}}}^{\mathsf{PRF\text{-}1}}$, the simulation of $\mathcal{G}_{\mathsf{PRF}}^{\mathrm{b}}$ to $\mathcal{A}_i$ is perfect. We have the desired result.

(Step 3.) Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}_{\mathsf{PRF}}^0$ from game $\mathcal{G}_{\mathsf{PRF}}^1$. Then we construct a reduction $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$, detailed in Fig. 27, playing the $\mathsf{IND\text{-}CPA}$ game that runs $\mathcal{A}_i$. We claim that

$$\mathbf{Adv}_{\mathcal{A}_i}^{\mathcal{G}_{\mathsf{PRF}}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}^{\mathsf{IND\text{-}CPA}}(\lambda).$$

Reduction $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ generates all keys and tokens except for k_i. In epoch i, $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ uses the $\mathsf{IND\text{-}CPA}$ challenger for assistance in computing $\mathsf{SKE.E}(\mathrm{k}_i^2, \mathrm{N})$. In epoch i, the reduction forwards all nonces of $\mathcal{O}.\mathsf{Enc}$ and $\mathcal{O}.\mathsf{Upd}$ to the $\mathsf{IND\text{-}CPA}$ challenger, and sets the reply in the second part of ciphertext, i.e. C_i^2. For challenge input $(\bar{\mathrm{m}}, \bar{\mathrm{C}})$, suppose $\bar{\mathrm{C}}$ has the underlying nonce N_1, $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ chooses nonce N_0 while encrypting m, sends $(\mathrm{N}_0, \mathrm{N}_1)$ to the $\mathsf{IND\text{-}CPA}$ challenger as challenge input, and sets the reply in the second part of the challenge ciphertext. The following shows how $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ simulates the responses of queries made by $\mathcal{A}_i$:

Eventually, $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ sends the guess bit of $\mathcal{A}$ to the $\mathsf{IND\text{-}CPA}$ challenger. We have the required result.

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

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 $\mathbf{Adv}_{\mathsf{BLMR+},\,\mathcal{A}}^{\mathsf{weakIND\text{-}UE}}(\lambda)$ is upper bounded by $(n+1)^2 \cdot |\mathbf{Pr}[\mathcal{G}_l^1 = 1] - \mathbf{Pr}[\mathcal{G}_1^0 = 1]|$. For any $1 \leq i \leq l$, we prove that

$$|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \leq \mathbf{Adv}_{\mathsf{SKE},\,\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}^{\mathsf{IND\text{-}CPA}}(\lambda) + 2\mathbf{Adv}_{\mathsf{F},\,\mathcal{B}_4^{\mathsf{PRF}}}^{\mathsf{PRF}}(\lambda) + \frac{2\mathrm{Q_E}^2}{|\mathcal{X}|}.$$

Suppose $\mathcal{A}_i$ is an adversary trying to distinguish game $\mathcal{G}_i^0$ from game $\mathcal{G}_i^1$ 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 $\mathsf{BLMR+}$ 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 $\mathsf{IND\text{-}CPA}$ advantage of SKE. We have the following result:

$$\mathbf{Adv}_{\mathsf{BLMR+},\,\mathcal{A}}^{\mathsf{weakIND\text{-}UE}}(\lambda) \leq (n+1)^3 \cdot \left(\mathbf{Adv}_{\mathsf{SKE},\,\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}^{\mathsf{IND\text{-}CPA}}(\lambda) + 2\mathbf{Adv}_{\mathsf{F},\,\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}}^{\mathsf{PRF}}(\lambda) + \frac{2\mathrm{Q_E}^2}{|\mathcal{X}|} \right).$$

□

```
Reduction $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ playing IND-CPA in hybrid i:
  do Setup
  $\bar{\mathrm{m}}_0, \bar{\mathrm{C}} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  phase $\leftarrow 1$
  do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{C}})$, get $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}})$
  if $\mathcal{I}^* \cap (\mathcal{K}^* \cup \mathcal{T}^*) \neq \emptyset$ then
    return ABORT
  $\underline{\mathsf{twf} \leftarrow 1 \textbf{ if}}$ :
    $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ or
    $(\exists \mathsf{e} \in \mathcal{C}^*\ s.t.\ \mathsf{e}$ or $\mathsf{e}+1 \in \mathcal{T}^*)$
  if ABORT occurred or twf $= 1$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
    return $\mathrm{b}'$
  if $\mathrm{b}' = \mathrm{b}$ then
    return 0
  else
    return 1

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

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$ :
  if $\mathsf{e} \neq i$ then
    $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{BLMR{+}.Enc}(\mathrm{k}_\mathsf{e}, \mathrm{m})$
  if $\mathsf{e} = i$ then
    $\mathrm{N} \xleftarrow{\$} \mathcal{X} \setminus X; X \leftarrow X \cup \{\mathrm{N}\}$
    $\underline{\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}}$
    $\underline{\mathrm{C}_i^2 \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})}$                    embed
    $\underline{\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_\mathsf{e}, \mathsf{e}; \mathrm{N}, \mathrm{m})\}$
  return $\mathrm{C}_\mathsf{e}$

$\mathcal{O}.\mathsf{Next}$ :
  $\mathsf{e} \leftarrow \mathsf{e} + 1$
```

```
$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e\text{-}1}})$ :
  if $(\cdot, \mathrm{C}_{\mathsf{e\text{-}1}}, \mathsf{e\text{-}1}; \mathrm{N}, \mathrm{m}) \notin \mathcal{L}$ or
    $(\mathsf{e} = i$ and $\mathrm{N} \in X)$ then
    return $\perp$
  if $\mathsf{e} \neq i, i+1$ then
    $\mathrm{C}_\mathsf{e} \leftarrow \mathsf{BLMR{+}.Upd}(\Delta_\mathsf{e}, \mathrm{C}_{\mathsf{e\text{-}1}})$
  if $\mathsf{e} = i+1$ then
    $\mathrm{C}_\mathsf{e} \leftarrow (\mathsf{F}(\mathrm{k}_\mathsf{e}^1, \mathrm{N}) \otimes \mathrm{m}, \mathsf{SKE.E}(\mathrm{k}_\mathsf{e}^2, \mathrm{N}))$
  if $\mathsf{e} = i$ then
    $X \leftarrow X \cup \{\mathrm{N}\}$
    $\underline{\mathrm{C}_i^1 \xleftarrow{\$} \mathcal{Y}}$
    $\underline{\mathrm{C}_i^2 \leftarrow \mathcal{O}.\mathsf{E}(\mathrm{N})}$                    embed
    $\underline{\mathrm{C}_i \leftarrow (\mathrm{C}_i^1, \mathrm{C}_i^2)}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_\mathsf{e}, \mathsf{e}; \mathrm{N}, \mathrm{m})\}$
  return $\mathrm{C}_\mathsf{e}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  if $\hat{\mathsf{e}} > \mathsf{e}$ or $\mathsf{e} = i$ or $(\mathsf{e} = i{+}1$ and $\mathsf{inp} = \mathsf{token})$ then
    return $\perp$
  if $\mathsf{inp} = \mathsf{key}$ then
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
    return $\mathrm{k}_{\hat{\mathsf{e}}}$
  if $\mathsf{inp} = \mathsf{token}$ then
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
    return $\Delta_{\hat{\mathsf{e}}}$

do CHALL with $(\bar{\mathrm{m}}_0, \bar{\mathrm{C}})$ :
  if $(\cdot, \bar{\mathrm{C}}, \tilde{\mathsf{e}} - 1; \mathrm{N}_1, \bar{\mathrm{m}}_1) \notin \mathcal{L}$ or $\mathrm{N}_1 \in X$ then
    return $\perp$
  $\mathrm{N}_0 \xleftarrow{\$} \mathcal{X} \setminus (X \cup \{\mathrm{N}_1\})$ ; $X \leftarrow X \cup \{\mathrm{N}_0, \mathrm{N}_1\}$
  $\underline{\tilde{\mathrm{C}}_i^1 \xleftarrow{\$} \mathcal{Y}}$
  $\underline{\textbf{call } \mathsf{CHALL} \textbf{ with } (\mathrm{N}_0, \mathrm{N}_1), \textbf{ get } \tilde{\mathrm{C}}_i^2}$          embed
  $\tilde{\mathrm{C}}_i \leftarrow (\tilde{\mathrm{C}}_i^1, \tilde{\mathrm{C}}_i^2)$
  for $j \in \{0, ..., i-1\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_1) \otimes \bar{\mathrm{m}}_1, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_1))$     left
  for $j \in \{i+1, ..., n\}$ do
    $\tilde{\mathrm{C}}_j \leftarrow (\mathsf{F}(\mathrm{k}_j^1, \mathrm{N}_0) \otimes \bar{\mathrm{m}}_0, \mathsf{SKE.E}(\mathrm{k}_j^2, \mathrm{N}_0))$     right
  $\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^n \{(\tilde{\mathrm{C}}_j, j)\}$
  return $\tilde{\mathrm{C}}_{\tilde{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  find $(\tilde{\mathrm{C}}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}$
  return $\tilde{\mathrm{C}}_\mathsf{e}$
```

Figure 27: Reduction $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ for proof of Proposition 4, the simulation is almost the same as reduction $\mathcal{B}_4^{\mathsf{IND\text{-}CPA}}$ does except for the underlined simulations. Recall the IND-CPA game in Definition 12: encryption oracle $\mathcal{O}.\mathsf{E}$ replies to input N with $\mathsf{SKE.Enc}(\mathrm{k}, \mathrm{N})$. ** indicates k_i are skipped in the generation. *** indicates Δ_i and Δ_{i+1} are skipped in the generation.

7 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. 28. 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.

```
RISE.KG(λ) :
  $x \xleftarrow{\$} \mathbb{Z}_q^*$
  $k_e \leftarrow (x, g^x)$
  return $k_e$

RISE.TG($k_e, k_{e+1}$) :
  parse $k_e = (x, y), k_{e+1} = (x', y')$
  $\Delta_{e+1} \leftarrow (\frac{x'}{x}, y')$
  return $\Delta_{e+1}$

RISE.Enc($k_e$, m) :
  parse $k_e = (x, y)$
  $r \xleftarrow{\$} \mathbb{Z}_q$
  $C_e \leftarrow (y^r, g^r \cdot m)$
  return $C_e$

RISE.Dec($k_e, C_e$) :
  parse $k_e = (x, y)$
  parse $C_e = (C_1, C_2)$
  $m' \leftarrow C_2 \cdot C_1^{-1/x}$
  return $m'$

RISE.Upd($\Delta_{e+1}, C_e$) :
  parse $\Delta_{e+1} = (\Delta, y')$
  parse $C_e = (C_1, C_2)$
  $r' \xleftarrow{\$} \mathbb{Z}_q$
  $C_1' \leftarrow C_1^{\Delta} \cdot y'^{r'}$
  $C_2' \leftarrow C_2 \cdot g^{r'}$
  $C_{e+1} \leftarrow (C_1', C_2')$
  return $C_{e+1}$
```

Figure 28: Updatable encryption scheme RISE [LT18a] for λ-bit prime q.

7.1 Proofs in LT18

In their IND-ENC and IND-UPD proofs for RISE, LT18 employ game hopping to reduce to DDH. An issue arises in both proofs, which we identify here using the IND-ENC proof [[LT18b], § D.1]:

Game 0 is a faithful simulation of the IND-ENC game. In Game 1, the reduction does not update ciphertexts when $\mathcal{O}$.Upd and $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ are called, and just encrypts as if the values were fresh. This requires extra bookkeeping to keep track of the $(\mathrm{m}, \mathrm{C}, \mathrm{e})$ tuples called previously but does not change the adversary's view. For Game 2, a hybrid argument separates challenge ciphertexts from actual epoch keys. Updates are randomized so in the case that the adversary doesn't know $\Delta_{\tilde{e}}$ or $\Delta_{\tilde{e}+1}$, the challenge ciphertext given to the adversary is different to the one used for $\mathcal{O}$.Enc queries. This is justified using the *key-anonymity* [BBDP01] property of Elgamal (under DDH). Unfortunately, the use of this property doesn't work here since keys are linked across epochs. The reduction between Game 1 and Game 2 must embed its DDH challenge (g, g^x, g^y, g^c) in some epoch, where x will play the role of the epoch key (thus let's call g^x the public key for this epoch) and y will be the randomness used in to construct the challenge ciphertext. It does not know when the challenge query will come from the IND-ENC algorithm so if the algorithm asks for an $\mathcal{O}$.Enc query in some epoch, it must use the 'public key' g^x to create the encryption: $((g^x)^r, g^r \cdot \mathrm{m})$. This provides a correct ciphertext, but now the reduction can no longer answer a $\mathcal{O}.\mathsf{Corr}(\mathrm{k})$ query for this epoch – regardless of when the algorithm asks for the challenge ciphertext. Such issues emphasize the difficulty in constructing proofs that successfully embed the challenge.

The problem can be solved by partitioning the epoch continuum using firewalls, i.e. incurring an $(n+1)^3$ loss (guessing the firewalls incurs an $(n+1)^2$ loss, and the hybrid argument across epochs incurs an $n+1$ loss). In fact, a proof is also possible for our new stronger notion, randIND-UE. In Section 7.2 we use novel techniques to provide a proof that RISE in fact meets the stronger notion randIND-UE.

7.2 RISE is randIND-UE Secure

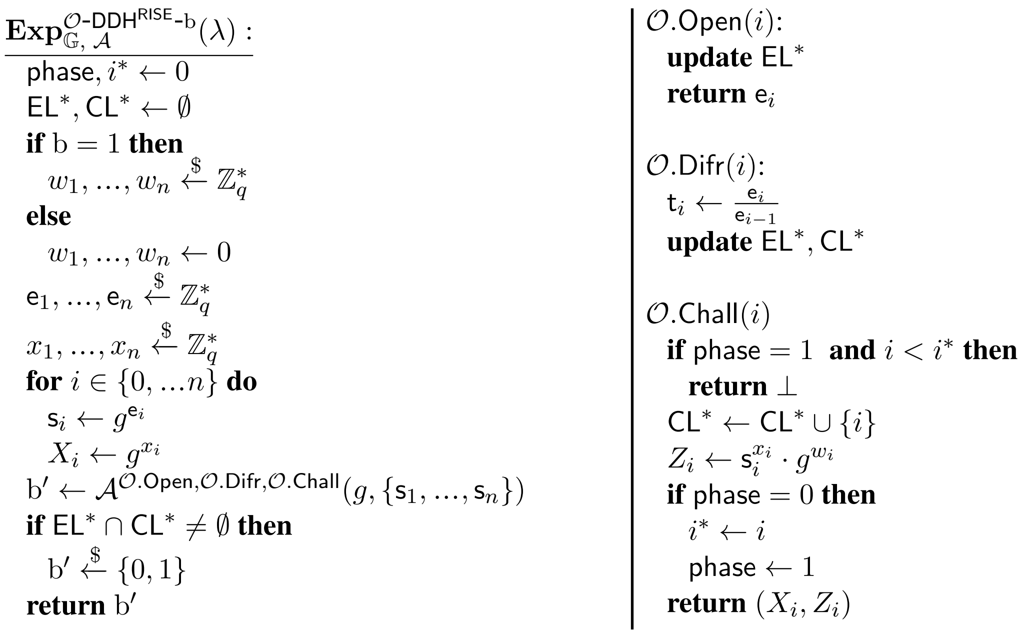
We now show that RISE is randIND-UE under DDH. First, we adapt (an extended version of) the Oracle-DDH experiment to the epoch-based corruption model found in updatable encryption, in a way that ensures that it still reduces to DDH. In this way, we lift a lot of the bookkeeping and complexity. Then, the reduction from randIND-UE 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 $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$, 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. 29. 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 13. 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 ($\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$) problem for $\mathbb{G}$ and g is

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

where the experiment $\mathbf{Exp}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}$ is given in Fig. 29.



```
$\mathbf{Exp}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}\text{-b}}_{\mathbb{G},\,\mathcal{A}}(\lambda)$ :
  phase, $i^* \leftarrow 0$
  $\mathsf{EL}^*, \mathsf{CL}^* \leftarrow \emptyset$
  if $\mathrm{b} = 1$ then
    $w_1, \ldots, w_n \xleftarrow{\$} \mathbb{Z}_q^*$
  else
    $w_1, \ldots, w_n \leftarrow 0$
  $\mathsf{e}_1, \ldots, \mathsf{e}_n \xleftarrow{\$} \mathbb{Z}_q^*$
  $x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_q^*$
  for $i \in \{0, \ldots n\}$ do
    $\mathsf{s}_i \leftarrow g^{\mathsf{e}_i}$
    $X_i \leftarrow g^{x_i}$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Open},\mathcal{O}.\mathsf{Difr},\mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\})$
  if $\mathsf{EL}^* \cap \mathsf{CL}^* \neq \emptyset$ then
    $\mathrm{b}' \xleftarrow{\$} \{0,1\}$
  return $\mathrm{b}'$

$\mathcal{O}.\mathsf{Open}(i)$:
  update $\mathsf{EL}^*$
  return $\mathsf{e}_i$

$\mathcal{O}.\mathsf{Difr}(i)$:
  $\mathsf{t}_i \leftarrow \frac{\mathsf{e}_i}{\mathsf{e}_{i-1}}$
  update $\mathsf{EL}^*, \mathsf{CL}^*$

$\mathcal{O}.\mathsf{Chall}(i)$
  if phase $= 1$ and $i < i^*$ then
    return $\perp$
  $\mathsf{CL}^* \leftarrow \mathsf{CL}^* \cup \{i\}$
  $Z_i \leftarrow \mathsf{s}_i^{x_i} \cdot g^{w_i}$
  if phase $= 0$ then
    $i^* \leftarrow i$
    phase $\leftarrow 1$
  return $(X_i, Z_i)$
```

Figure 29: $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$ experiment for $\mathbb{G}$ of order q (λ-bit prime) and generator g.

Theorem 5. 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. 28. For any randIND-UE adversary $\mathcal{A}$ against RISE, there exists an adversary $\mathcal{B}_5$ against DDH such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{RISE},\,\mathcal{A}}(\lambda) = 2(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_5}(\lambda).$$

This theorem is proven by Lemmas 5.1 and 5.2.

Lemma 5.1. Let $\mathbb{G}$ be a group of order q (a λ-bit prime) with generator g. For any adversary $\mathcal{A}$ against $\mathcal{O}\text{-DDH}^{\mathsf{RISE}}$, there exists an adversary $\mathcal{B}_{5.1}$ against DDH such that

$$\mathbf{Adv}^{\mathcal{O}\text{-DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{A}}(\lambda) = (n+1)^3 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G},\,\mathcal{B}_{5.1}}(\lambda),$$

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

Proof. We use a sequence of game hops and a hybrid argument. Define game $\mathcal{G}_i^{\mathrm{b}}$ as $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-b}}$ 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 $\mathbf{Exp}_{\mathbb{G}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}$, i.e. all challenges result in 'random' DDH tuples, and $\mathcal{G}_n^0$ is $\mathbf{Exp}_{\mathbb{G}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}$, 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 $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}$ from $\mathbf{Exp}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}$ for adversary $\mathcal{A}$.

Notice that all queries in $\mathcal{G}_{i-1}^0$ and $\mathcal{G}_i^1$ have the equal responses (For $j \le i-1$, returns a real sample. For $j > i-1$, returns a random sample). We have $\mathbf{Adv}_{\mathbb{G},\,\mathcal{A}}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}(\lambda) = \sum_{i=0}^{n} |\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]|$. Then we prove that $|\mathbf{Pr}[\mathcal{G}_i^1 = 1] - \mathbf{Pr}[\mathcal{G}_i^0 = 1]| \le (n+1)^2 \cdot \mathbf{Adv}_{\mathbb{G},}^{\mathsf{DDH}}(\lambda)$ 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 $\mathcal{B}_{5.1}$, detailed in Fig. 30, 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 $\{0, \ldots, n\}$, so this loss is upper bounded by $(n+1)^2$.

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 $\mathcal{B}_{5.1}$ will simulate these, however we omit this calculation for readability.

If $\mathcal{B}_{5.1}$ is playing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{B}_{5.1}}^{\mathsf{DDH}\text{-}1}$ 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 $\mathcal{B}_{5.1}$ is playing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{B}_{5.1}}^{\mathsf{DDH}\text{-}0}$ then its tuple is real, providing a perfect simulation of $\mathcal{G}_i^0$. We have the required result.

```
Reduction $\mathcal{B}_{5.1}$ playing $\mathbf{Exp}_{\mathbb{G},\,\mathcal{B}_{5.1}}^{\mathsf{DDH}\text{-b}}(\lambda)$
 in hybrid $i$ :
  receive $(g, X, Y, Z)$
  $\mathsf{fwl}, \mathsf{fwr} \xleftarrow{\$} \{0, \ldots, n\}$
  $w_{i+1}, \ldots, w_n \xleftarrow{\$} \mathbb{Z}_q^*$
  $w_0, \ldots, w_{i-1} \leftarrow 0$
  $\mathsf{s}_i \leftarrow Y$
  for $j \in \{0, \ldots, i\text{-}1\} \cup \{i\text{+}1, \ldots, n\}$ do
    $x_j \xleftarrow{\$} \mathbb{Z}_q^*; X_j \leftarrow g^{x_j}$
  for $j \in \{0, \ldots, \mathsf{fwl}\text{-}1\} \cup \{\mathsf{fwr}\text{+}1, \ldots, n\}$ do
    $\mathsf{e}_j \xleftarrow{\$} \mathbb{Z}_q^*; \mathsf{t}_j \leftarrow \frac{\mathsf{e}_j}{\mathsf{e}_{j\text{-}1}}{}^{**}; \mathsf{s}_j \leftarrow g^{\mathsf{e}_j}$
  for $j \in \{\mathsf{fwl}\text{+}1, \ldots, \mathsf{fwr}\}$ do
    $\mathsf{t}_j \xleftarrow{\$} \mathbb{Z}_q^*$
  for $j \in \{\mathsf{fwl}, \ldots, i\text{-}1\}$ do
    $\mathsf{s}_j \leftarrow Y^{-\prod_{k=j+1}^{i} \mathsf{t}_k}$
  for $j \in \{i\text{+}1, \ldots, \mathsf{fwr}\}$ do
    $\mathsf{s}_j \leftarrow Y^{\prod_{k=i+1}^{j} \mathsf{t}_k}$
  $\mathrm{b}' \leftarrow \mathcal{A}_i^{\mathcal{O}.\mathsf{Open}, \mathcal{O}.\mathsf{Difr}, \mathcal{O}.\mathsf{Chall}}(g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\})$
  if ABORT occurred then
    $\mathrm{b}' \xleftarrow{\$} \{0, 1\}$
  return $\mathrm{b}'$

$\mathcal{O}.\mathsf{Open}(j)$:
  if $j \in \{\mathsf{fwl}, \ldots, \mathsf{fwr}\}$ then
    ABORT
  return $\mathsf{e}_i$

$\mathcal{O}.\mathsf{Difr}(j)$:
  if $j \in \{\mathsf{fwl}, \mathsf{fwr}\text{+}1\}$ then
    ABORT
  return $\mathsf{t}_j$

$\mathcal{O}.\mathsf{Chall}(j)$
  if $j = i$ then
    $X_j \leftarrow X; Z_j \leftarrow Z$
  if $j \neq i$ then
    $Z_j \leftarrow \mathsf{s}_j{}^{x_j} g^{w_j}$
  return $(X_j, Z_j)$
```

Figure 30: Reduction $\mathcal{B}_{5.1}$ for proof of Lemma 5.1. ** indicates t_0 and $\mathsf{t}_{\mathsf{fwr}+1}$ are skipped in the computation.

□

Lemma 5.2. 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. 28. For any $\mathsf{randIND\text{-}UE}$ adversary $\mathcal{A}$ against RISE, there exists an adversary $\mathcal{B}_{5.2}$ against $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ such that

$$\mathbf{Adv}^{\mathsf{randIND\text{-}UE}}_{\mathsf{RISE},\,\mathcal{A}}(\lambda) = 2 \cdot \mathbf{Adv}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}}_{\mathbb{G},\,\mathcal{B}_{5.2}}(\lambda).$$

Proof. The reduction $\mathcal{B}_{5.2}$ is given in Fig. 31. The reduction $\mathcal{B}_{5.2}$ is playing $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ game and runs $\mathcal{A}$. $\mathcal{B}_{5.2}$ flips a coin b, and simulates $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}b}}_{\mathsf{RISE},\,\mathcal{A}}$ by interacting with its own $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ challenger.

To simulate updated non-challenge ciphertexts (i.e. respond to $\mathcal{O}.\mathsf{Upd}$ queries), $\mathcal{B}_{5.2}$ 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 $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ challenger. Since updated non-challenge ciphertexts are of the form $\mathrm{C}_{\mathsf{e}} = (\mathsf{s}^r_{\mathsf{e}}, g^r \cdot \mathrm{m})$, where r is a fresh random value, the s_i values allow $\mathcal{B}_{5.2}$ to successfully simulate updated non-challenge ciphertexts.

To simulate challenge ciphertext (i.e. respond to challenge query or $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$), $\mathcal{B}_{5.2}$ must embed using its own challenge. Recall that in the $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ experiment in Fig. 29, 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 $\mathcal{B}_{5.2}$ 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 $\mathcal{A}$, $\mathcal{B}_{5.2}$ 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. $\mathcal{B}_{5.2}$ 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, $\mathcal{B}_{5.2}$ can simulate the response of $\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$ using the same approach.

Eventually, $\mathcal{B}_{5.2}$ receives the output bit b' from $\mathcal{A}_i$. If $\mathrm{b}' = \mathrm{b}$, then $\mathcal{B}_{5.2}$ returns 0 to its $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ challenger, otherwise, $\mathcal{B}_{5.2}$ returns 1.

If $\mathcal{B}_{5.2}$ interacting with $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}0}_{\mathbb{G},\,\mathcal{B}_{5.2}}$, then it perfectly simulates $\mathbf{Exp}^{\mathsf{randIND\text{-}UE\text{-}b}}_{\mathsf{RISE},\,\mathcal{A}}$ to $\mathcal{A}$. If $\mathcal{B}_{5.2}$ interacting with $\mathbf{Exp}^{\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}\text{-}1}_{\mathbb{G},\,\mathcal{B}_{5.2}}$, then it wins with probability 1/2. After some computation similar to that in the proof of Theorem 1.3 we have the desired result. $\square$

```
Reduction $\mathcal{B}_{5.2}$ playing $\mathcal{O}\text{-}\mathsf{DDH}^{\mathsf{RISE}}$ game:
  **receive** $g, \{\mathsf{s}_1, \ldots, \mathsf{s}_n\}$
  $\mathsf{e} \leftarrow 0; \mathsf{phase} \leftarrow 0; \mathcal{L} \leftarrow \emptyset$
  $\bar{\mathrm{m}}_0, \mathrm{C} \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr}}(\lambda)$
  $\mathsf{phase} \leftarrow 1$
  **if** $(\cdot, \mathrm{C} = (\mathrm{C}_1, \mathrm{C}_2), \tilde{\mathsf{e}} - 1; \bar{\mathrm{m}}_1) \notin \mathcal{L}$ **then**
    **return** $\perp$
  **call** $(X_{\tilde{\mathsf{e}}}, Z_{\tilde{\mathsf{e}}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\tilde{\mathsf{e}})$
  $\mathrm{b} \xleftarrow{\$} \{0, 1\}$
  $\tilde{\mathrm{C}} \leftarrow (Z_{\tilde{\mathsf{e}}}, X_{\tilde{\mathsf{e}}} \cdot \bar{\mathrm{m}}_{\mathrm{b}})$
  $\mathrm{b}' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}}(\tilde{\mathrm{C}})$
  **if** $\mathrm{b}' = \mathrm{b}$ **then**
    **return** $0$
  **else**
    **return** $1$

$\mathcal{O}.\mathsf{Enc}(\mathrm{m})$
  $r \xleftarrow{\$} \mathbb{Z}_q^*$
  $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}^r_{\mathsf{e}}, g^r \cdot \mathrm{m}); \mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{m})\}$
  **return** $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Next}$
  $\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(\mathrm{C}_{\mathsf{e}-1})$ :
  **if** $(\cdot, \mathrm{C}_{\mathsf{e}-1}, \cdot; \mathrm{m}) \notin \mathcal{L}$ **then**
    **return** $\perp$
  $r \xleftarrow{\$} \mathbb{Z}_q^*$
  $\mathrm{C}_{\mathsf{e}} \leftarrow (\mathsf{s}^r_{\mathsf{e}}, g^r \cdot \mathrm{m}); \mathcal{L} \leftarrow \mathcal{L} \cup \{(\cdot, \mathrm{C}_{\mathsf{e}}, \cdot; \mathrm{m})\}$
  **return** $\mathrm{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$ :
  **if** $\hat{\mathsf{e}} > \mathsf{e}$ **then**
    **return** $\perp$
  **if** $\mathsf{inp} = \mathsf{key}$ **then**
    **call** $\mathsf{e}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Open}(\hat{\mathsf{e}})$
    $\mathrm{k}_{\hat{\mathsf{e}}} \leftarrow (\mathsf{e}_{\hat{\mathsf{e}}}, g^{\mathsf{e}_{\hat{\mathsf{e}}}})$
    **return** $\mathrm{k}_{\hat{\mathsf{e}}}$
  **if** $\mathsf{inp} = \mathsf{token}$ **then**
    **call** $\mathsf{t}_{\hat{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Difr}(\hat{\mathsf{e}})$
    $\Delta_{\hat{\mathsf{e}}} \leftarrow (\mathsf{t}_{\hat{\mathsf{e}}}, \mathsf{s}_{\hat{\mathsf{e}}})$
    **return** $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathrm{C}}$
  **call** $(X_{\mathsf{e}}, Z_{\mathsf{e}}) \leftarrow \mathcal{O}.\mathsf{Chall}(\mathsf{e})$
  $\tilde{\mathrm{C}} \leftarrow (Z_{\mathsf{e}}, X_{\mathsf{e}} \cdot \bar{\mathrm{m}}_{\mathrm{b}})$
  **return** $\tilde{\mathrm{C}}_{\mathsf{e}}$
```

Figure 31: Reduction $\mathcal{B}_{5.2}$ for proof of Theorem 5.2.

8 Conclusions, Discussion, and Future Work

In this work we provided a new updatable encryption scheme, SHINE, and a new definition of security IND-UE (that implies prior notions) in which we prove its security. 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.

One clear avenue for future work is achieving integrity protection and CCA security in an efficient manner (i.e. without using the generic constructions of Klooß et al.). KLR19's Theorem 1 states that an updatable encryption scheme (consisting of separated components, a symmetric encryption scheme and an update mechanism) that fulfils a number of properties provides an updatable encryption scheme that is secure in terms of CCA attacks and integrity of ciphertexts. Judging whether or not SHINE can meet the required properties is particularly challenging: the properties required of the 'SKE' scheme (i.e. the combination of permutation and exponentiation) are strong-IND-CCA and strong-INT-CTXT, and directly proving these properties for any instantiation is not immediate. Instead, we conjecture that a more direct approach should be possible for proving (possibly equivalent) strong properties for UE schemes if SHINE's permutation is instantiated using authenticated encryption. We suggest that this efficient construction may be suitable for many applications of updatable encryption, though a more thorough analysis is needed.

We do not wish to be overly prescriptive regarding implementation details for SHINE, or indeed other SHINE-like systems. In the proof of Theorem 2 (and Proposition 3), we require that π is a random (unkeyed) permutation, however we do not need any specific and strong properties that are provided by modern constructions of blockciphers and sponges. As far as the proof goes, and in practice, the specific property that we want from this permutation is that given a ciphertext and the inverse of the epoch key k_{e} (yielding $\pi(\mathsf{IV}||\mathrm{m})$), the only way to extract useful information about m 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. If we were to specify that π must be, for example, a keyed blockcipher, then the adversary in the security games would acquire this key in an $\mathcal{O}.\mathsf{Corr}$ query (for any epoch) and this would make simulation and embedding very challenging. We stress that this is a proof issue rather than a practical problem, and using (for example) a blockcipher key of all zeros should not yield any security loss compared with using a random key.

Acknowledgements. This research was funded by the Research Council of Norway under Project No. 248166. The second author 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.

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 NDSS 2005*. The Internet Society, 2005. Cited on page 6.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Proceedings of ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, 2001. Cited on page 38.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In *Proceedings of CRYPTO 2013 I*, volume 8042 of *LNCS*, pages 410–428. Springer, 2013. Cited on pages 3, 5, 6, 8, 32, and 33.

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

[CCFL17] Christian Cachin, Jan Camenisch, Eduarda Freire-Stögbuchner, and Anja Lehmann. Updatable tokenization: Formal definitions and provably secure constructions. In *Proceedings of Financial Cryptography and Data Security 2017*, volume 10322 of *LNCS*, pages 59–75. Springer, 2017. Cited on page 6.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of ACM CCS 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 *Proceedings of CRYPTO 2008*, volume 5157 of *LNCS*, pages 1–20. Springer, 2008. Cited on page 27.

[DDLM19] Alex Davidson, Amit Deo, Ela Lee, and Keith Martin. Strong post-compromise secure proxy re-encryption. In *Proceedings of ACISP 2019*, volume 11547 of *LNCS*, 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 *Proceedings of SECRYPT 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 *Proceedings of CRYPTO 2017 III*, volume 10403 of *LNCS*, 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*, 2017:527, 2017. Cited on page 5.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In *Proceedings of ACM CCS 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 *Proceedings of EUROCRYPT 2019 I*, volume 11476 of *LNCS*, pages 68–99. Springer, 2019. Cited on pages 3, 4, 6, 8, 12, 14, 15, and 16.

[KLR19b] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. *IACR Cryptology ePrint Archive*, 2019:222, 2019. Cited on page 26.

[KRS$^+$03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of FAST 2003*. USENIX, 2003. Cited on page 3.

[Lee17] Ela Lee. Improved security notions for proxy re-encryption to enforce access control. In *Proceedings of LATINCRYPT 2017*, volume 11368 of *LNCS*, 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 4.

[LT18a] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In *Proceedings of EUROCRYPT 2018 III*, volume 10822 of *LNCS*, pages 685–716. Springer, 2018. Cited on pages 3, 4, 6, 12, 33, and 38.

[LT18b] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. *IACR Cryptology ePrint Archive*, 2018:118, 2018. Cited on page 38.

[MS18] Steven Myers and Adam Shull. Practical revocation and key rotation. In *Proceedings of CT-RSA 2018*, volume 10808 of *LNCS*, pages 157–178. Springer, 2018. Cited on page 6.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In *Proceedings of EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 327–346. Springer, 1999. Cited on page 32.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949. Cited on page 27.