Revisiting Updatable Encryption: True Forward Security, Constructions and a Puncturable Perspective

Daniel Slamanig and Christoph Striecks

AIT Austrian Institute of Technology, Vienna, Austria
{firstname.lastname}@ait.ac.at

Abstract. Updatable encryption (UE) allows to periodically rotate encryption keys without the need to decrypt and re-encrypt already encrypted data. In this work, we present an attack which is not covered by prior ciphertext-independent UE security notions and which seems problematic in practice; namely, an adversary would record available information (i.e., ciphertexts, all update tokens) in the lifetime of the system and simply would wait for a *single* key leakage.
To mitigate such an attack, we require a more *fine-grained* ciphertext-update approach where ciphertexts are allowed to expire after some time. Our threefold contribution is as follows:

a) First, we introduce a UE CPA security notion to allow fine-grained updatability. It focuses on UE schemes where the token can only forwardly update the ciphertext and thus reduces complexity compared to prior models. Additionally, it introduces the concept of *expiry epochs*, i.e., ciphertexts can lose the ability of being updatable after a certain time. This is determined at the time of encryption and captures the above mentioned attack.
b) Second, we present and prove secure the first UE scheme with such properties. We construct it from standard assumptions (e.g., the SXDH assumption in prime-order bilinear groups) using the well-known dual system paradigm. To overcome the hurdles towards UE with such strong properties, we require novel construction and adapted proof techniques. Noteworthy, our optimized UE scheme enjoys sublinear key and ciphertext sizes.
c) Finally, as an extension, we introduce a novel approach of constructing UE which significantly departs from previous ones and in particular views UE from the perspective of puncturable encryption (Green and Miers, S&P'15). We introduce a variant of puncturable encryption called ciphertext-puncturable encryption (CPE) which generalizes UE and may be of independent interest.

1 Introduction

When outsourcing the storage of data, the primary measure to protect its confidentiality is encryption. However, a compromise of the respective encryption key(s) will potentially expose the entire data to unauthorized parties and may cause severe damage. Consequently, it is widely considered a good practice to periodically rotate encryption keys. Major providers of cloud storage services such as Google[1], Microsoft[2] or Amazon[3] recommend this practice and sometimes it is even mandated by regulations [Bar16, PCI22]. This raises the immediate question of how to efficiently update already outsourced encrypted data to new keys. An obvious solution for key-rotation is to download the data, decrypt it locally under the old key, re-encrypt it under a new key, and upload it again. Unfortunately, this imposes a significant overhead and soon becomes impractical.

Updatable Encryption. At CRYPTO 2013, Boneh, Lewi, Montgomery, and Raghunathan [BLMR13] proposed the concept of updatable encryption (UE). UE is a symmetric encryption primitive that addresses this problem by allowing to update ciphertexts to new keys without the requirement for decryption by means of a so-called update token. UE schemes can be *ciphertext-dependent* [BLMR13, EPRS17, BEKS20, CLT20] where the update token depends on the specific ciphertext to be updated and, thus, to compute the update token, a part of every ciphertext needs to be downloaded. Or, and from an efficiency point more desirable, quite a number of recent works deal with UE schemes that are *ciphertext-independent* [LT18, KLR19, BDGJ20, Jia20, Nis22, GP22] such that a single compact update token can update *any* ciphertext (and, consequently, the token must be independent of the number of ciphertexts in the system). In the remainder of this work, we focus on UE schemes with

[1] https://cloud.google.com/kms/docs/key-rotation
[2] https://docs.microsoft.com/en-us/azure/storage/blobs/security-recommendations
[3] https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html

ciphertext-independent updates and will simply call them UE schemes. Such an UE scheme consists of the usual algorithms ($\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$) for key-generation, encryption, and decryption. Time is discretized in so-called epochs and Gen produces an initial secret key K_1 (for epoch 1). Additionally, there is an algorithm Next which takes a key K_e and outputs a fresh next-epoch key K_{e+1} along with a so-called update token Δ_{e+1}. This update token can be used by a semi-trusted party to update ciphertexts under key K_e for epoch e to ciphertexts for epoch $e+1$ under key K_{e+1} via an algorithm Update.

Forward- and Post-Compromise Security. For UE, a formal study of forward security (i.e., leaking the current key does not endanger old ciphertexts) and post-compromise security (i.e., leaking the current key does not endanger new ciphertexts) was initiated by Lehmann and Tackmann [LT18]. Security for UE essentially requires that fresh and updated ciphertexts are indistinguishable even if the adversary can compromise keys and update tokens adaptively. A central question is how much information can be given to an adversary to still achieve reasonable security guarantees. For instance, key and ciphertext updates in UE can be *bi-directional* meaning that keys as well as ciphertexts can be updated via the token in the forward *and* the backward direction. This is suboptimal from a security perspective and was already observed by Lehmann and Tackmann [LT18]. Ideally, we want UE schemes where solely the ciphertext can be updated in the forward direction as this would arguably yield the most natural form of UE. However, most known UE constructions have bi-directional key and ciphertext updates [LT18, BDGJ20, Jia20, Nis22], some have key updates in the backward direction [Nis22, GP22], and only one UE scheme has solely ciphertexts updatable in the forward direction [Nis22]. Unfortunately, even having strong guarantees such as the ones provided by the construction in [Nis22], security guarantees for old ciphertexts, i.e., forward security, cannot be truly captured and met by any of the known UE models and constructions respectively. We believe that this constitutes a weakness and should therefore be considered in UE models.

Motivating Stronger Guarantees. When looking at state-of-the-art UE security models [BDGJ20, Jia20, Nis22, GP22][4] it can be observed that they do not capture an attack which seems problematic in practice. Namely, an adversary would record available information (i.e., ciphertexts C_e, all update tokens Δ_{e+1}) in the lifetime of the system and simply would wait for a *single* key leakage $K_{e'}$ in epoch e' with $e' > e$. Such single key leakage allows to completely break confidentiality of all ciphertexts captured before and we dub it "record now, leak later" attack.

Indeed, if we want to mitigate such type of attack, we must introduce a more fine-grained adjustment of the updatability of ciphertexts. The reason is that correctness of UE requires each ciphertext C_e to be updatable *ad infinitum* in the forward direction given the respective update tokens. As a consequence, the above mentioned attack cannot be mitigated in all known models. We will discuss these definitional issues and implications in more detail below.

Revisiting UE Security Models. Our discussion uses Fig. 1 to illustrate the most important aspects of the security notions[5] (starting from [LT18, BDGJ20] up to this work) from its weakest to its strongest form by means of the maximum information available to an adversary. We stick to a single insulated region[6] around challenge epoch e^*, i.e., the epoch in which the adversary needs to distinguish ciphertexts, which simplifies the description.

Let K_e and Δ_{e+1} be the key and the token for epoch e, respectively. The task of an adversary is to distinguish an encryption under a key K_{e^*} in a challenge epoch e^* from one that is updated from some epoch $\tilde{e} < e^*$. A key concept in prior UE models is that of a "firewall" which prevents trivial wins. For the weakest form of UE with bi-directional key and ciphertext updates, there needs to be a firewall e_{start} before and e_{end} after the challenge epoch e^* (indicated by red boxes in Fig. 1). In between, all update tokens can be revealed. Prior to e_{start} and after e_{end}, the adversary can obtain all keys and update tokens. Now the critical restrictions are that in e_{start}, *only* the key is revealed as otherwise in UE schemes with bi- or forward-directional key updates, the adversary could trivially compute a key for the target epoch e^*. In e_{end}, *neither* the key *nor* the update token are revealed. Otherwise, due to

[4] Our focus is on the established game-based security models. However, we want to note that recent works also study UE in composable frameworks and in particular the framework of constructive cryptography [LR21, FMM21].

[5] Our focus is on the CPA notions, but the same issues transfer to stronger notions.

[6] An insulated region [BDGJ20] contains epochs where the adversary is only allowed to query update tokens and cannot trivially decrypt the challenge ciphertext.

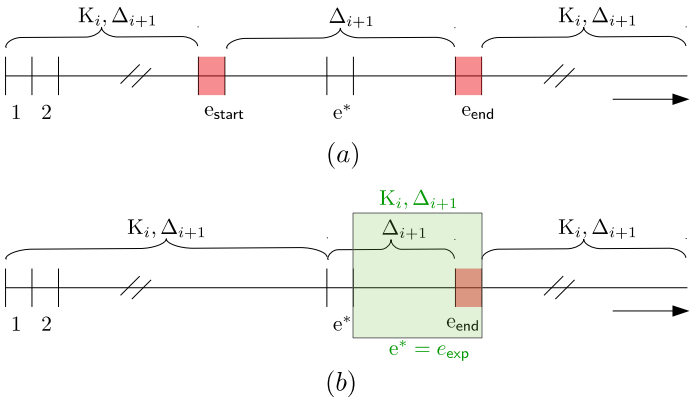

Fig. 1. (a) Example of information an adversary is allowed to obtain in known CPA notions for UE schemes with bi-directional key and ciphertext updates. (b) Example of information an adversary is allowed to obtain in our EE-IND-UE-CPA experiment. First, we can allow to give out a key and a token in e_{start}, and, second, expiry epochs (exemplary $e_{\mathsf{exp}} = e^*$ here) allow to hand out all keys and tokens after e_{exp} (green box). Hence, in this specific case, our model allows even to release *all* tokens and keys except the key for e^* obviously.

correctness, the adversary could update the challenge ciphertext into one epoch where it holds a key and could trivially win.

Let us now dig deeper into the directionality features of keys and in particular look at the case where update tokens *do not* allow key updates in the forward direction. Here, we have to look at the epochs before the challenge epoch e^* as tokens should not even allow to move keys forward. We observe that for such UE schemes, we can remove the firewall at e_{start} entirely and hand out all keys as well as tokens up to e^* to the adversary. This change gives rise to what Nishimaki in [Nis22] calls UE with backward-directional key updates. However, when we look at the epochs after the challenge epoch e^*, the situation gets more subtle. Therefore, let us recall that the correctness of UE requires that ciphertexts can be updated *ad infinitum*, i.e., the update capability never expires (meaning that updates of ciphertexts via a token in any epoch is always possible already by the correctness definition in UE). This in particular means that if old ciphertexts and update tokens are not properly deleted or kept stored intentionally by a server, even if after many updates (or, key-rotations) a newer key leaks, it will still be possible to decrypt an old ciphertext by simply updating it to the respective epoch. Moreover, Jiang Galteland and Pan [GP22] recently formally[7] showed that even UE schemes with no-directional key updates are equivalently secure to UE schemes with backward-directional key updates in prior models. Hence, this means that even "ideal" UE constructions with solely forward-directional ciphertext updates and no-directional key updates cannot give any true *forward-security* guarantees in current models (i.e., once a key leaks, old ciphertexts are in danger).

Extended UE Security with Expiry Epochs. We believe that forward security is an important feature for UE in practice and should be inherently considered in full in the UE regime.[8] This can be enforced by restricting the update capabilities of UE (which were initially not foreseen) and to the best of our knowledge we do not see any other way. In particular, we introduce the concept of expiry epochs such that for every ciphertext, one can decide how long updates should yield decryptable ciphertexts, i.e., encryption in epoch i is performed as $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$ and when epoch e_{exp} is reached, a ciphertext cannot longer be updated into a decryptable ciphertext. Note that an update token should still work for all ciphertexts that have an expiry epoch in the future. Also, by virtually never letting ciphertexts expire, i.e., using $e_{\mathsf{exp}} = 2^\lambda$ for all encryptions with security parameter values $\lambda \in \mathbb{N}$, we are essentially back in the currently strongest models [Nis22, GP22] but with less complex bookkeeping.

This conceptually simple modification has an interesting effect. Namely, as we show in this work, to meet our proposed UE security notion, we at least require the UE scheme to solely allow ciphertext updates via the token in the forward direction. So far such UE schemes are only known to exist by relying on indistinguishability obfuscation (IO) [Nis22]. Despite the simplicity of the conceptual modification, it even requires more and this makes the task of constructing a UE scheme achieving our notion

[7] Informally, this was also mentioned in [MPW22].

[8] By introducing firewalls, prior work offers only a very weak form of forward security by restricting access to tokens artificially.

non-trivial (there exists no such UE scheme so far). Moreover, since UE is inspired mainly by practice, we want constructions from standard assumptions and where key and ciphertext sizes are as compact as possible, but certainly sublinear in the maximum number of possible epochs. While compactness can be achieved in weaker models [BDGJ20, Jia20, Nis22], this important feature turned out to be non-trivial in our model.

UE From a Puncturable Perspective. As an extension, we offer a novel view of UE from the perspective of Puncturable Encryption (PE). We recall that PE, introduced by Green and Miers in [GM15], is a tag-based public-key (or secret-key [SYL+18, AGJ21, BDdK+21]) encryption primitive with an additional puncturing algorithm that takes a secret key and a tag t as input, and produces an updated (punctured) secret key. This key is able to decrypt all ciphertexts *except* those tagged with t and (updated) secret keys can be iteratively punctured on distinct tags. PE is a versatile primitive that has already found numerous applications and in particular where strong forward security is required [GM15, CHN+16, CRRV17, BMO17, DKL+18, GHJL17, DJSS18, DRSS21, DGJ+21].

One can observe that the core goal in UE, i.e., that newer (as well as updated) ciphertexts can no longer be decrypted by older keys, is abstractly reminiscent of puncturing when viewing tags as epochs. As in UE one however has to update ciphertexts *and* keys, puncturing needs to happen on both in a synchronized way. In particular, one needs to guarantee that old (non-updated) ciphertexts are no longer decryptable, while one should be able to include old ciphertexts that are still decryptable (by updating them). Consequently, when puncturing keys, one needs some information which can be used to parametrize ciphertext puncturing, i.e., to exclude certain ciphertexts from being punctured.

In a nutshell, we introduce the concept of Ciphertext Puncturable Encryption (CPE) that can be viewed as a symmetric PE scheme $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{Dec})$ with an additional algorithm ExPunc to control which ciphertexts are excluded from puncturing. Such a scheme is associated to a polynomial sized set of sequence tags, e.g., $(1, \ldots, n)$, as well as an unbounded ciphertext-tag space $\mathcal{T}$. KPunc sequentially punctures keys on sequence tags, i.e., removes the ability to decrypt ciphertexts tagged under them step by step. In addition KPunc can take a set of tags $\mathcal{S} \subseteq \mathcal{T}$ (or a special tag $\forall$) and outputs a puncture token, which can then be used to exclude ciphertexts carrying tags in $\mathcal{S}$ from puncturing (in case of $\forall$ all ciphertext can be excluded). As in PE, in CPE ciphertexts are computed w.r.t. a tag $t \in \mathcal{T}$, but additionally take an "expiry-tag" e_{exp} from the set of sequence tags. The semantics of CPE are now as follows. Old ciphertexts with sequence tags on which a key has been punctured can no longer be decrypted. A puncture token can be used to *exclude* ciphertexts from being punctured and thus updated in a way that they can still be decrypted. However, if e_{exp} is reached, then they are automatically and implicitly punctured and cannot be excluded from puncturing anymore. The tags $t \in \mathcal{T}$ can be used to make the exclusion from puncturing more fine-grained, i.e., puncture tokens that have been computed with respect to some tag t can only exclude ciphertexts tagged with t from puncturing, as long as their e_{exp} has not been reached.

We show that CPE implies UE with expiry epochs and provide a CPE construction from standard assumption. While the so obtained UE is similar to our direct sublinear UE construction, it is slightly less efficient and the security proof is more involved. Besides implying UE, it is however a stronger primitive. We believe that it provides an interesting abstraction for protected outsourced file storage with forward-security and fine-grained secure shredding of files (in the vein of puncturable key wrapping [BGP22], but augmented with efficient key rotation).

Our Contribution. Briefly summarized, our contribution is as follows:

a) First, we simplify and extend the state-of-the-art UE CPA security models [Jia20, Nis22, GP22] to capture the guarantees provided by UE schemes that restrict the function of update tokens to ciphertext updates in the forward direction only. In particular, our model — which we dub $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ — reduces complexity concerning the leakage-profile paradigm which made existing UE models rather cumbersome. Importantly, we introduce *expiry epochs* as a fine-grained updatability feature of UE. By letting ciphertexts expire, we can mitigate the "record now, leak later" attack already discussed above. In such a case, the adversary in our model is able to query *all* update tokens and *almost all* secret keys (excluding secret keys that allow trivial wins only). Moreover, we show that our notion implies the most recent standard CPA UE notion due to [Nis22, GP22] if we set the expiry epoch for all ciphertexts to be exponential in the security parameter (e.g., to 2^λ).

b) Second, we construct a UE scheme that is secure in our model and thus yields the first UE scheme with such strong properties. Moreover, its security is based on standard assumptions. Concretely,

we instantiate our construction from the standard d-Lin assumption (where for $d = 1$ we get SXDH) in prime-order bilinear groups using the well-known dual system paradigm [Wat09, Lew12]. Indeed, to overcome the hurdles towards UE with such strong properties, we require novel construction and adapted proof techniques based on the dual-system groups approach due to Chen-Wee and Gong et al. [CW13, GCTC16]. Noteworthy, our optimized UE scheme enjoys sublinear key and ciphertext sizes. In Fig. 2, we provide a brief comparison of UE schemes.

c) As an extension, we introduce a novel primitive dubbed Ciphertext Puncturable Encryption (CPE) which we believe provides an easier intuition towards UE. The insight here is that updating a ciphertext in UE via a token is related to puncturing operations on ciphertexts. Furthermore, we show a concrete asymptotically efficient construction of such a CPE scheme from the same assumptions used in b). We believe that the ciphertext puncturing in CPE will further increase the applicability of the already very useful concept of puncturable encryption [GM15] and might be of independent interest.

Schemes	key-sizes	ct-sizes	tok-sizes	security	model	dir. (key)	dir. (ct)	assumption
BMLR+ [LT18]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	weak IND-UE-CPA	SM	bi	bi	KH-PRF
RISE [LT18]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	bi	bi	DDH
*SHINE [BDGJ20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	d-IND-UE-CCA	IC	bi	bi	DDH
Jiang [Jia20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	bi	bi	LWE
Nishimaki [Nis22]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	backw	forw	LWE
Nishimaki [Nis22]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	-	forw	IO, OWF
GP [GP22]	$\mathbf{O}(e)$	$\mathbf{O}(e)$	$\mathbf{O}(e)$	r-IND-UE-CPA	SM	backw	forw	PKE
Ours (Sec. 3.3)	$\mathbf{O}(\lambda^2)$	$\mathbf{O}(\log^2 e_{\mathsf{exp}})$	$\mathbf{O}(1)$	EE-IND-UE-CPA	SM	-	forw	SXDH

Fig. 2. Overview of IND-UE-CPA secure (ciphertext-independent) UE schemes. The number of possibly allowed updates is 2^λ and with $e_{\mathsf{exp}} \leq 2^\lambda$ we denote the expiry epoch of a ciphertext, for security parameter λ; with e the current UE epoch. With d-IND-UE-CPA or r-IND-UE-CPA, we mean security under deterministic or randomized updates, where EE-IND-UE-CPA only considers randomized updates and represents our model. IC and SM stand for Ideal Cipher and Standard Model. KH-PRF stands for key-homomorphic PRF.

1.1 Overview of Our Techniques

We give an overview of how to instantiate a UE scheme in such a strong model with a proof sketch. Moreover, we discuss the generalization of UE via the notion of ciphertext puncturable encryption (CPE).

Our Simple UE Construction with Expiry Epochs. We start with our simple construction and use ideas developed within to later give an optimized version with sublinear keys and ciphertexts. We rely on the well-known dual-system paradigm initiated by Waters [Wat09] and, in particular, use an abstraction due to Chen-Wee and Gong et al. [CW13, GCTC16] that can be instantiated from standard assumptions in the standard model. Quite surprisingly, such a dual-system group (DSG) abstraction gives us freedom in constructing the desired UE functionalities. However, we deviate in the proof methodology and have to develop new ideas along the way. Loosely speaking, DSGs have three groups $(\mathbb{G}, \mathbb{H}, G_T)$ with an associated bi-linear map $e : \mathbb{G} \times \mathbb{H} \to G_T$ and a linear mapping function m. Moreover, such groups have the properties:

Subgroup indistinguishability: Two-way sampling of (a) a normal distribution and (b) a semi-functional distribution with higher entropy in $\mathbb{G}$ and $\mathbb{H}$.

Associativity: For all $(g_0, \ldots, g_\lambda) \in \mathbb{G}^{\lambda+1}$ and all $(h_0, \ldots, h_\lambda) \in \mathbb{H}^{\lambda+1}$, it holds that $e(g_0, h_i) = e(g_i, h_0)$, for all $i \in [\lambda]$ and normal distribution samplings.

Orthogonality and Protectiveness: For all $s \leftarrow \mathsf{SampS}(pp; r_s)$, for random coins r_s and public parameters pp, and all $(h_0, \ldots, h_\lambda) \in \mathbb{H}^{\lambda+1}$, it holds that $e(s, h_i) = 1$, for all $i \in [\lambda]$. For all $k \leftarrow \mathsf{SampK}(pp)$ and all $(g_0, \ldots, g_\lambda) \in \mathbb{H}^{\lambda+1}$, it holds that $e(g_0, k) = 1$. For all $h \in \mathbb{H}$, it holds $e(s, h) = \mathsf{SampG_T}(m(h); r_s)$.

Parameter hiding: Normal distributions in $\mathbb{G}$ and $\mathbb{H}$ can be sampled using public parameters pp only. The semi-functional distributions sampling has λ units of information-theoretically embedded entropy (even if pp is given).

An exemplary description of our simple UE construction is as follows. Informally (but sufficient for transporting our ideas), the initial UE key is

$$K_1 = (h_0, k_1 \cdot h_1, h_2, \ldots, h_\lambda, m(k_1), pp),$$

generated via $\mathsf{Gen}(\lambda)$. The encryption algorithm $\mathsf{Enc}(K_1, M, e_{\mathsf{exp}} = 2)$ can use this key to compute a (first-epoch) ciphertext, here for simplicity with expiry epoch $e_{\mathsf{exp}} = 2$, for $s = \mathsf{SampS}(pp; r_s)$ and $e(s, k_1) = \mathsf{SampG_T}(m(k_1); r_s)$ with uniform exponent r_s:

$$C_1 = (C_0, C_1, C_2, C_T) = (s \cdot g_0, g_1, g_2, e(s, k_1) \cdot M),$$

The next algorithm $\mathsf{Next}(K_1)$ computes a token

$$\Delta_2 = (D_0, D_1, D_2) = (h'_0, \delta \cdot h'_2, m(k_1 \cdot \delta)),$$

where δ is essentially a "tweak" between epoch 1 and 2 sampled from the normal distribution in $\mathbb{H}$. It is important to "hide" the tweak as otherwise such a token could be used to undo the forward update. The updated key is

$$K_2 = (K'_0, K'_1, K'_3, \ldots, K'_\lambda, K'_T, pp) = (h_0, \underbrace{k_1 \cdot \delta}_{k_2} \cdot \prod_{i=1}^{2} h_i, h_3, \ldots, h_\lambda, m(\underbrace{k_1 \cdot \delta}_{k_2}), pp).$$

See that h_2 is now part of the product in K'_1. Moreover, in the real scheme, we use perfect re-randomization of the key components $K'_1, \ldots, K'_\lambda$, but we skip over it here for depicting purposes. The ciphertext update is performed as follows: first, using the token Δ_2, we compute the tweak δ in the target group G_T by

$$e(s, \delta) = \frac{e(C_0, D_1)}{e(D_0, C_2)} = \frac{e(s, \delta) \cdot \overbrace{e(s, h'_2)}^{1} \cdot \overbrace{e(g_0, \delta)}^{1} \cdot e(g_0, h'_2)}{e(h'_0, g_2)}.$$

Second, by the pairing properties, the updated ciphertext is constructed as

$$C_2 = (C'_0, C'_1, C'_T) = (s \cdot g_0, \prod_{i=1}^{2} g_i, e(s, \underbrace{k_1 \cdot \delta}_{k_2}) \cdot M).$$

See that the element g_2 is now included in the product of C'_1. In the real scheme, we use perfect re-randomization of r_s, but we skip over it here for depicting purposes. Decryption in epoch 2 is done via $M =$

$$\frac{e(C'_1, K'_2) \cdot C'_T}{e(C'_0, K'_1)} = \frac{e(\prod_{i=1}^{2} g_i, h_0) \cdot e(s, k_2) \cdot M}{e(s \cdot g_0, k_2 \cdot \prod_{i=1}^{2} h_i)} = \frac{e(\prod_{i=1}^{2} g_i, h_0) \cdot e(s, k_2) \cdot M}{e(s, k_2) \cdot e(g_0, \prod_{i=1}^{2} h_i)}.$$

See that update and decryption succeed due to the associativity, orthogonality, and protectiveness properties of DSG samplings in $\mathbb{G}$ and $\mathbb{H}$. Moreover, it is straightforward to generalize the above scheme for any epoch e.

The idea of the proof is to first make the challenge ciphertext semi-functional via introducing semi-functional components into such a ciphertext (via subgroup indistinguishability of DSG). It is important to note that such a semi-functional ciphertext can be decrypted by key elements coming from the normal distribution, but will fail with high probability for semi-functional key components. This is exactly what we will use in the remainder of the proof where we carefully introduce uniform randomness into each key component which has key elements associated to the challenge ciphertext. This is done as in the usual dual-system paradigm where we can embed uniform randomness into key components that are not "prefixes" of the challenge ciphertext (via subgroup indistinguishability and parameter hiding of the underlying DSG). This is the main hurdle and we need to develop new proof techniques along the

way as keys and tokens in UE are information-theoretically related (and the adversary can query parts of them). To give a glimpse on the concrete technique, we can observe that one can use the token $\Delta_{e^*} = (D_0, D_1, D_2)$, for challenge epoch e^*, to embed such a random element (since the token is not a prefix of the challenge ciphertext as otherwise it could be used to undo the update operation). See that such a token can be used to compute the key K_{e^*} (without re-randomization but sufficient for depicting the idea) as

$$K_{e^*} = (h_0, \underbrace{k_{e^*-1} \cdot D_1}_{k_{e^*-1} \cdot \delta \cdot h_{e^*}} \cdot \prod_{i=1}^{e^*-1} h_i, h_{e^*+1}, \ldots, h_\lambda, m(k_{e^*}), pp).$$

Via parameter hiding, we embed a uniform element $(\widehat{h})^\alpha$ into the second component of $\Delta_{e^*} = (D_0, D_1, D_2) = (h_0, \delta \cdot (\widehat{h})^\alpha \cdot h_{e^*}, m(k_{e^*}))$ which results in uniform keys $K_{e'}$ from epoch e^* onwards, i.e., with $e' > e^*$, and security readily follows.

We believe that our simple UE scheme above can be useful for scenarios where key updates do not happen very often (as the ciphertexts and keys are linear in the number of epochs). However, for the general scenario of many updates, we are able to lift our above approach to construct a UE scheme that only has sublinear ciphertext and key sizes. This requires more care and a more involved construction as well as proof techniques. (We give more details in the respective sections.) Although the overall construction approach is very similar to the one sketched scheme above, we additionally need a clever encoding (due to Drijvers, Gorbunov, Neven, and Wee [DGNW20, Sec. 4.2]) with an adapted novel dual-system proof. The hurdle to overcome is that we now have more key elements where we need to embed uniform randomness since the token Δ_{e^*} cannot be longer used to compute the key in epoch e^*. This results in a slightly more involved proof with a larger security loss, but yields a UE construction with sublinear keys and ciphertexts in the number of allowed updates from standard assumptions in the standard model.

Ciphertext-Puncturable Encryption (CPE) Construction. The CPE construction follows the approach of the encoding-based approach for UE above. However, one additional feature of CPE is that ciphertexts are tagged, which results in an even more involved proof strategy compared to the UE construction with sublinear keys and ciphertexts above. The hurdle is that the adversary can now query tokens in specific epochs which cannot puncture the challenge ciphertext, but have key elements that could be used to derive information on the challenge key. Fortunately, this can be solved via carefully embedding also randomness in such tokens, again utilizing our dual-system proof methodology and security can be shown. The final CPE scheme has ciphertext and key sizes sublinearly in the number of allowed punctures.

1.2 Notation

For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$, and let $\lambda \in \mathbb{N}$ be the security parameter. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For an algorithm A, let $y \leftarrow A(\lambda, x)$ be the process of running A on input (λ, x) with access to uniformly random coins and assigning the result to y. (We may omit to mention the λ-input explicitly and assume that all algorithms take λ as input.) To make the random coins r explicit, we write $A(\lambda, x; r)$. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in λ. A function f is negligible if its absolute value is smaller than the inverse of any polynomial (i.e., if $\forall c \exists k_0 \forall \lambda \geq k_0 : |f(\lambda)| < 1/\lambda^c$).

1.3 Outline of the paper

In Sec. 2, we present our security model with expiry epochs and discuss relations to previous models. In Sec. 3, we present two UE construction, one with linear and one with sublinear keys and ciphertexts and prove them secure in our UE model. Finally, in Sec. 4 we introduce Ciphertext Puncturable Encryption (CPE), present a concrete construction thereof and discuss how we can instantiate UE from CPE. Moreover, we briefly discuss other applications of CPE.

2 Updatable Encryption with Expiry Epochs

We define UE with expiry epochs in ciphertexts which allow ciphertexts to being excluded from updates. We build on the recent UE models [Jia20, Nis22, GP22], but dealing solely with the natural form of updatability namely security for UE schemes with forward-directional ciphertext updates. This allows to reduce the security model complexity compared to prior UE models significantly (i.e., extra bookkeeping of gained adversarial information via leakage profiles is not necessary).

The main idea of UE with expiry epochs is the following. On the very high level, all operations are bound to discrete epochs $1, 2, \ldots$ where keys and ciphertexts as well as so-called update tokens are associated to. System setup Gen creates a first-epoch symmetric key K_1. With this key, one can create a first-epoch ciphertext $C_{1,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_1, M, e_{\mathsf{exp}})$, for some message M and expiry epoch e_{exp}, and, e.g., outsource $C_{1,e_{\mathsf{exp}}}$ to some semi-trusted third-party. With probabilistic algorithm Next, K_1 can be updated to K_2 while also an update token Δ_2 is generated. With Δ_2, a semi-trusted third-party, e.g., an outsourced service provider, can update C_1 to $C_{2,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_2, C_{1,e_{\mathsf{exp}}})$ such that $C_{2,e_{\mathsf{exp}}}$ is "consistent" with K_2. Correctness guarantees that decryption of $C_{2,e_{\mathsf{exp}}}$ yields $M = \mathsf{Dec}(K_2, C_{2,e_{\mathsf{exp}}})$ as intended if the ciphertext is not expired already (and so on). More formally:

Definition 1. *A UE scheme* UE *with message space* $\mathcal{M}$ *consist of the PPT algorithms* (Gen, Next, Enc, Update, Dec)*:*

$\mathsf{Gen}(\lambda)$*: on input security parameter* λ*, the key generation algorithm outputs an initial (symmetric) key* K_1*.*

$\mathsf{Next}(K_e)$*: on input key* K_e*, the key update algorithm outputs an updated key* K_{e+1} *for the next epoch together with an update token* Δ_{e+1}*.*

$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$*: on input key* K_e*, a message* $M \in \mathcal{M}$*, and expiry epoch* e_{exp}*, encryption outputs a ciphertext* $C_{e,e_{\mathsf{exp}}}$ *or* $\perp$*.*

$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$*: on input an update token* Δ_{e+1} *and a ciphertext* $C_{e,e_{\mathsf{exp}}}$*, decryption outputs an updated ciphertext* $C_{e+1,e_{\mathsf{exp}}}$ *or* $\perp$*.*

$\mathsf{Dec}(K_e, C_{e,e_{\mathsf{exp}}})$*: on input key* K_e *and a ciphertext* $C_{e,e_{\mathsf{exp}}}$*, decryption outputs* $M \in \mathcal{M} \cup \{\perp\}$*.*

Correctness. For all $\lambda, e \in \mathbb{N}$, for $K_1 \leftarrow \mathsf{Gen}(\lambda)$, for all $i \in \{1, \ldots, e\}$, for all $(K_{i+1}, \Delta_{i+1}) \leftarrow \mathsf{Next}(K_i)$, for all $M \in \mathcal{M}$, for all $e_{\mathsf{exp}} \in \mathbb{N}$, for all $j \in \{1, \ldots, e+1\}$, for all $C_{j,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_j, M, e_{\mathsf{exp}})$, we require that $M = \mathsf{Dec}(K_e, C_{e,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq j$, also for all $C_{j'+1,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_{j'+1}, C_{j',e_{\mathsf{exp}}})$ with $j' \in \{j+1, \ldots, e\}$.

Security notion. We particularly consider the attack that an adversary can use a token Δ_e to update a key K_e to K_{e-1} (i.e., yielding a key that is consistent with epoch-$(e-1)$ ciphertexts). This is because one could build an UE scheme that has "backward-directional"[9] key updates and this would still be secure in common UE models. Indeed, formally shown only recently by Galteland Jiang and Pan [GP22], UE with backward-directional key updates are equivalently secure in common UE models to UE schemes that do not allow such key updates. Since we want ciphertexts that can expire, the token in the expiry epoch should not be of help to update a key from the next epoch to the current expiry one.

We will dub our CPA security notion with expiry epochs $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$. Essentially, it ensures that fresh and updated ciphertexts are indistinguishable even if the adversary has access to keys and update tokens adaptively. We define:

Definition 2 ($\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ security). *A UE scheme* UE *is* $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$*-secure iff for any PPT adversary* A*, the advantage function*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda) = 1\right] - 1/2 \right|$$

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}$ *is defined as in Fig. 3.*

On post-compromise and forward security. Since tokens cannot be used to update keys, we achieve post-compromise security (PCS) – where by PCS we loosely speaking mean: once an old key leaks, future ciphertexts are not in danger even in the presence of tokens. Moreover, by requiring in our model that a current-epoch key cannot be updated via a token and tokens cannot be used to update ciphertexts from C_e to C_{e-1} together with expiry epochs allows us to achieve forward security (FS) when keys are

[9] We refer to [Nis22] for nomenclature direction discussions and definitions on UE.

Experiment $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{Next}',\mathsf{Update}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUpdC}^*}(\lambda)$
if $b = b'$ and A is valid, then return 1 else return 0

Oracles
$\mathsf{Enc}'(M, e_{\mathsf{exp}})$: run $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_{e,e_{\mathsf{exp}}}), c = c + 1$. Return $C_{e,e_{\mathsf{exp}}}$.
Next' : run $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$. If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e + 1$.
$\mathsf{Update}'(C_{e-1,e_{\mathsf{exp}}})$: if $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\bot$. Run $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_e, C_{e-1,e_{\mathsf{exp}}})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_{e,e_{\mathsf{exp}}}), c = c + 1$. Return $C_{e,e_{\mathsf{exp}}}$.
$\mathsf{Corrupt}(\mathsf{inp}, e')$: if $e' > e$, return $\bot$. If $\mathsf{inp} = \mathsf{key}$, set $\mathcal{K}^* = \mathcal{K}^* \cup \{e'\}$ and return $K_{e'}$. If $\mathsf{inp} = \mathsf{token}$, set $\mathcal{D}^* = \mathcal{D}^* \cup \{e'\}$ and return $\Delta_{e'}$.
$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\bot$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\bot$. If $b = 0$, set $C^*_{e,0} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$, else $C^*_{e,1} \leftarrow \mathsf{Update}(\Delta_e, C_{e-1,e_{\mathsf{exp}}})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,b}$.
$\mathsf{GetUpdC}^*$: If $\mathsf{phase} = 0$ or $e > e^*_{\mathsf{exp}}$, return $\bot$. Set $\mathcal{C}^* := \mathcal{C}^* \cup (e, C^*_{e,b})$ and return $C^*_{e,b}$.

A **is valid iff**:
1) For all $e' \in \mathcal{K}^*$, $(e', C^*_{e',b}) \notin \mathcal{C}^*$ holds. (No trivial win via retrieved keys.)
2) For all $e' \in \mathcal{K}^*$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ and $(e'-1, C^*_{e'-1,b}) \in \mathcal{C}^*$, $e' - 1 \notin \mathcal{D}^*$ holds. (No trivial win via updating a retrieved challenge ciphertext via a token.)

Fig. 3. Our EE-IND-UE-CPA security notion for UE schemes with expiry epochs.

leaked beyond that expiry epoch of a ciphertext — where by FS we mean, again loosly speaking: once a key leaks, expired ciphertexts are not in danger even in presence of tokens. Recall that by correctness of UE schemes *without* expiry epochs, such strong from of FS *cannot* be met (i.e., once a key leaks, old ciphertexts are immediately in danger when access to all tokens is granted). Indeed, known UE schemes do not support expiry epochs and have to mitigate this issue by artificially disallowing access to a certain token after the challenge ciphertext was produced, which however results in only a very weak form of FS.

2.1 Relation to Other UE Security Notions

Our security notions imply IND-UE-CPA security [GP22, Nis22, Jia20]. Moreover, we can even show that our notion implies a simple and plausible ciphertext indistinguishability notion where challenge ciphertexts with different expiry epochs are indistinguishable in the same challenge epoch.

EE-IND-UE-CPA implies IND-UE-CPA for UE with $e_{\mathsf{exp}} = 2^\lambda$. The security notions of IND-UE-CPA as given in [GP22, Nis22, Jia20] are closely related to ours with the exception that ours allows expiry epochs. In the following, we show that our notions imply the prior CPA notion for an UE scheme with $e_{\mathsf{exp}} = 2^\lambda$ for large enough $\lambda \in \mathbb{N}$ via a simple reduction. Before that, we recap the IND-UE-CPA notion.

We adapt the most recent IND-UE-CPA notion from [GP22] (which is also used in [Nis22, Jia20]) and left out unnecessary details such as deterministic updates, CCA, and some sets for the leakage profiles. Indeed, to show our implication, we require that any successful IND-UE-CPA adversary yields a successful EE-IND-UE-CPA adversary. The contrary does not hold. The reason is that in such a hypothetical reduction, we cannot simulate encryption queries and a valid challenge with different expiry epochs, and certainly cannot provide the token $\Delta_{e^*_{\mathsf{exp}}+1}$ to the EE-IND-UE-CPA adversary as this would yield a trivial win via the non-validity of the adversary in the IND-UE-CPA security experiment. Notably, here the currently strongest notion of IND-UE-CPA is presented, i.e., security for backward-directional key updates.

Definition 3 (IND-UE-CPA **security** [GP22])**.** *A UE scheme* UE *is* IND-UE-CPA*-secure iff for any PPT adversary A, the advantage function*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda) = 1\right] - 1/2 \right|$$

Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{Next}',\mathsf{Update}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUpdC}^*}(\lambda)$
if $b = b'$ and A is valid, then return 1 else return 0

Oracles
$\mathsf{Enc}'(M)$: run $C_e \leftarrow \mathsf{Enc}(K_e, M)$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_e), c = c + 1$. Return C_e.
Next' : run $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$. If $\mathsf{phase} = 1$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e + 1$.
$\mathsf{Update}'(C_{e-1})$: if $(\cdot, e-1, C_{e-1}) \notin \mathcal{L}^*$, return $\bot$. Run $C_e \leftarrow \mathsf{Update}(\Delta_e, C_{e-1})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_e), c = c + 1$. Return C_e.
$\mathsf{Corrupt}(\mathsf{inp}, e')$: if $e' > e$, return $\bot$. If $\mathsf{inp} = \mathsf{key}$, set $\mathcal{K}^* = \mathcal{K}^* \cup \{e'\}$ and return $K_{e'}$. If $\mathsf{inp} = \mathsf{token}$, set $\mathcal{D}^* = \mathcal{D}^* \cup \{e'\}$ and return $\Delta_{e'}$.
$\mathsf{Chall}(M, C_{e-1})$: if $\mathsf{phase} = 1$, return $\bot$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1}) \notin \mathcal{L}^*$, return $\bot$. If $b = 0$, set $C^*_{e,0} \leftarrow \mathsf{Enc}(K_e, M)$, else $C^*_{e,1} \leftarrow \mathsf{Update}(\Delta_e, C_{e-1})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, and return $C^*_{e,b}$.
$\mathsf{GetUpdC}^*$: If $\mathsf{phase} = 0$, return $\bot$. Set $\mathcal{C}^* := \mathcal{C}^* \cup (e, C^*_{e,b})$ and return $C^*_{e,b}$.

A **is valid iff**:
1) For all $e' \in \mathcal{K}^*$, $(e', C^*_{e',b}) \notin \mathcal{C}^*$ holds. (No trivial win via retrieved keys.)
2) For all $e' \in \mathcal{K}^*$ with $e^* < e'$ and $(e' - 1, C^*_{e'-1,b}) \in \mathcal{C}^*$, $e' - 1 \notin \mathcal{D}^*$ holds. (No trivial win via updating a retrieved challenge ciphertext via a token.)

Fig. 4. The IND-UE-CPA security notion for UE schemes.

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}$ *is defined as in Figure 4.*

In [Nis22, GP22], efficient UE schemes were presented that fulfill the IND-UE-CPA notion. Those schemes allow key updates in the backwards direction. We briefly want to mention that any potential UE scheme that allows key updates in the backward direction *cannot* be EE-IND-UE-CPA-secure. The reason is that our notion allows the adversary to query the update token $\Delta_{e_{\mathsf{exp}}+1}$ in the expiry epoch and the key $K_{e_{\mathsf{exp}}+1}$ right after the expiry epoch of the challenge ciphertext. In a backward-directional key update setting, the adversary can take $K_{e_{\mathsf{exp}}+1}$ and $\Delta_{e_{\mathsf{exp}}+1}$ to derive $K_{e_{\mathsf{exp}}}$ which results in a adversarial win in our notion. Hence, this yields a trivial UE scheme separation between IND-UE-CPA and EE-IND-UE-CPA security as we strictly have to rely on UE schemes with no-directional key updates.

However, we can show that any EE-IND-UE-CPA-secure UE scheme with fixed $e_{\mathsf{exp}} = 2^\lambda$ is also IND-UE-CPA secure relating the notions for exponential expiry-epoch values. Let UE_∞ be a UE scheme with expiry epochs set always to $e_{\mathsf{exp}} = 2^\lambda$ for large-enough $\lambda \in \mathbb{N}$.

Lemma 1. *If* UE_∞ *is EE-IND-UE-CPA-secure, then* UE_∞ *is IND-UE-CPA-secure. Concretely, for any PPT adversary* A *in the IND-UE-CPA security notion there is a PPT distinguisher* D *in the EE-IND-UE-CPA security notion such that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE}_\infty,A}(\lambda) \leq \mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE}_\infty,D}(\lambda).$$

Proof. D starts A with λ and has to present a consistent view for A. Therefore, D forwards oracles queries for $\mathsf{Enc}', \mathsf{Next}', \mathsf{Update}', \mathsf{Corrupt}, \mathsf{Chall}$ and $\mathsf{GetUpdC}^*$ directly to it's EE-IND-UE-CPA challenger (using $e_{\mathsf{exp}} = 2^\lambda$).

Since we have $e < e_{\mathsf{exp}}$ always in the reduction (as the current epoch e evolves only sequentially), those queries are consistent. The only crucial point here is the validity of D. For D to be valid, we need that A does not have queried 1) keys in challenge-equal epochs and 2) update tokens in epochs for which A has queried a key in the next epoch (but depending on e_{exp} in A's view). However, since $e < e_{\mathsf{exp}}$ always in the reduction, those validity constraints for A are essentially D's validity constraints. Hence, if A is valid and successful in the sense of IND-UE-CPA on UE_∞, D yields a successful PPT adversary on UE_∞ in the EE-IND-UE-CPA notion. □

Ciphertext Indistinguishability for Different Expiry Epochs. Next we introduce a simple and natural relaxation of our EE-IND-UE-CPA security notion, which we dub EE-IND-UE-ENC, to show security of ciphertexts under different expiry epochs but in the same challenge epoch. See that such a property is not immediately achieved in our stronger EE-IND-UE-CPA notion since there we guarantee

indistinguishability of fresh and updated ciphertexts that have the same expiry epoch in the same challenge epoch. However, as we show, our EE-IND-UE-CPA implies the EE-IND-UE-ENC notion.

Essentially, such a notion is strongly related to the common IND-UE-ENC notions [LT18], but adapted to expiry epochs. Particularly, we allow the adversary to output different expiry epochs as a challenge together with two chosen messages for a challenge epoch. The goal of the adversary is to distinguish which of the challenge messages was encrypted for which expiry epoch depending on a uniform bit b. Essentially, b determines if M_0 is encrypted for $e_{\mathsf{exp},0}$ or for $e_{\mathsf{exp},1}$ while M_1 is encrypted for the other expiry epoch mimicking that no PPT adversary can distinguish two different messages for two different expiry epochs within a target epoch.

The intuition is that in our EE-IND-UE-CPA notion the adversary is capable of querying encryptions (through Enc') for different expiry epochs in the challenge epoch. While this intuitively yields ciphertext indistinguishability for different expiry epochs, we want to make it more formal next. The key feature is that the adversary queries the challenge oracle Chall with two messages and two expiry epochs of its choice. In the challenge phase, it has to distinguish for which expiry epoch which message was encrypted, where the choice in the challenge ciphertexts is determined by a uniformly random bit b which the adversary has to guess to succeed in the notion.

Definition 4 (EE-IND-UE-ENC security). *A UE scheme* UE *is* EE-IND-UE-ENC*-secure iff for any PPT adversary* A*, the advantage function*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}(\lambda) = 1\right] - 1/2 \right|$$

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}$ *is defined as in Figure 5.*

Experiment $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{Next}',\mathsf{Update}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUpdC}^*}(\lambda)$
if $b = b'$ and A is valid, then return 1 else return 0

Oracles
$\mathsf{Enc}'(M)$: run $C_e \leftarrow \mathsf{Enc}(K_e, M)$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_e), c = c + 1$. Return C_e.
Next' : run $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$. If $\mathsf{phase} = 1$, run $C^*_{e+1,0} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,0})$ and $C^*_{e+1,1} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,1})$. Set $e = e + 1$.
$\mathsf{Update}'(C_{e-1})$: if $(\cdot, e-1, C_{e-1}) \notin \mathcal{L}^*$, return $\bot$. Run $C_e \leftarrow \mathsf{Update}(\Delta_e, C_{e-1})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_e), c = c + 1$. Return C_e.
$\mathsf{Corrupt}(\mathsf{inp}, e')$: if $e' > e$, return $\bot$. If $\mathsf{inp} = \mathsf{key}$, set $\mathcal{K}^* = \mathcal{K}^* \cup \{e'\}$ and return $K_{e'}$. If $\mathsf{inp} = \mathsf{token}$, set $\mathcal{D}^* = \mathcal{D}^* \cup \{e'\}$ and return $\Delta_{e'}$.
$\mathsf{Chall}(M^*_0, M^*_1, e^*_{\mathsf{exp},0}, e^*_{\mathsf{exp},1})$: if $\mathsf{phase} = 1$, return $\bot$. Set $\mathsf{phase} = 1$. If $b = 0$, set $C^*_{e,0} \leftarrow \mathsf{Enc}(K_e, M_b, e^*_{\mathsf{exp},0})$, else $C^*_{e,1} \leftarrow \mathsf{Enc}(K_e, M_{1-b}, e^*_{\mathsf{exp},1})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,0}, C^*_{e,1})$, $e^* = e$, and return $C^*_{e,0}, C^*_{e,1}$.
$\mathsf{GetUpdC}^*$: If $\mathsf{phase} = 0$, return $\bot$. Set $\mathcal{C}^* := \mathcal{C}^* \cup \{(e, C^*_{e,0}, C^*_{e,1})\}$ and return $(C^*_{e,0}, C^*_{e,1})$.

A **is valid iff**:
1) For all $e' \in \mathcal{K}^*$, $(e', C^*_{e',0}, C^*_{e',1}) \notin \mathcal{C}^*$ holds. (No trivial win via retrieved keys.)
2) For all $e' \in \mathcal{K}^*$ with $e^* < e' \leq \max(e^*_{\mathsf{exp},0}, e^*_{\mathsf{exp},1})$ and $(e'-1, C^*_{e'-1,0}, C^*_{e'-1,1}) \in \mathcal{C}^*$, $e' - 1 \notin \mathcal{D}^*$ holds. (No trivial win via updating the retrieved challenge ciphertexts via a token.)

Fig. 5. The EE-IND-UE-ENC security notion for UE schemes with expiry epochs.

Lemma 2. *If* UE *is an EE-IND-UE-CPA-secure UE scheme with expiry epochs, then* UE *is EE-IND-UE-ENC-secure. Concretely, for any PPT adversary* A *there is a distinguisher* D *such that*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},D}(\lambda).$$

Proof. D starts A with λ and has to present a consistent view for A. Therefore, D forwards oracles queries for $\mathsf{Enc}', \mathsf{Next}', \mathsf{Update}'$, and $\mathsf{Corrupt}$ directly to it's EE-IND-UE-CPA challenger. The answers of $\mathsf{Enc}', \mathsf{Update}'$, and $\mathsf{Corrupt}$ provide a consistent view for A as they are the same in both experiments. Next' switches the epochs in the pre-challenge epochs.

If A queries Chall with $(M^*_0, M^*_1, e^*_{\mathsf{exp},0}, e^*_{\mathsf{exp},1})$, D queries its Chall-oracle with $M^*_0, M^*_1, e^*_{\mathsf{exp},0}$ and its Enc'-oracle with $(M^*_b, e^*_{\mathsf{exp},1})$, for uniform $b \leftarrow \{0,1\}$. D retrieves $C^*_{e,0}$ from Chall and $C^*_{e,1}$ from Enc'

which D both forwards to A as the answer of A's Chall-oracle. In half of the cases, this yields a consistent contribution for A's challenge ciphertexts.

In the challenge phase, Next' switches the epochs and lets the EE-IND-UE-CPA challenger updates the challenge ciphertext $C^*_{e,0}$ while $C^*_{e,1}$ is updated via the Update'-oracle (which is possible since $C^*_{e,1}$ is a valid ciphertext queried from D's Enc'-oracle). If A queries the updated challenge ciphertexts, D returns those.

For D to be valid, we need that A does not have queried 1) keys in challenge-equal epochs and 2) update tokens in epochs for which A has queried a key in the next epoch (depending on $\max(e^*_{\mathsf{exp},0}, e^*_{\mathsf{exp},1})$). Those validity constraints for A directly transfer to D's validity constraints. Hence, if A is valid and successful in the sense of EE-IND-UE-ENC on UE, D yields a successful PPT adversary on UE in the EE-IND-UE-CPA notion. □

3 Constructions of UE with Expiry Epochs

We continue with constructing EE-IND-UE-CPA-secure UE schemes. Before that, we recap necessary building blocks such as pairings, group generator, and dual-system groups which provide us with a modular approach towards our goal.

3.1 Preliminaries and Dual System Groups

Notations. We write $\mathbf{v} = (v_i)_{i\in[n]}$, for $n \in \mathbb{N}$. We may also write vectors in bold fonts which depends on the context, i.e., we use a component-wise multiplication of vectors, i.e., $\mathbf{v} \cdot \mathbf{v}' = (v_1, \ldots, v_n) \cdot (v'_1, \ldots, v'_n) = (v_1 \cdot v'_1, \ldots, v_n \cdot v'_n)$.

Pairings. Let $\mathbb{G}, \mathbb{H}, G_T$ be cyclic groups. A *pairing* $e : \mathbb{G} \times \mathbb{H} \to G_T$ is a map that is *bilinear* (i.e., for all $g, g' \in \mathbb{G}$ and $h, h' \in \mathbb{H}$, we have $e(g \cdot g', h) = e(g, h) \cdot e(g', h)$ and $e(g, h \cdot h') = e(g, h) \cdot e(g, h')$), *non-degenerate* (i.e., for generators $g \in \mathbb{G}, h \in \mathbb{H}$, we have that $e(g, h) \in G_T$ is a generator), and *efficiently computable*.

Group generator. Let $\mathsf{G}(\lambda, n')$ be a group generator that generates the tuple $(\mathbb{G}, \mathbb{H}, G_T, N, g, h, (g_{p_i})_{i\in[n']}, e)$, for a pairing $e : \mathbb{G} \times \mathbb{H} \to G_T$, for composite-order groups $\mathbb{G}, \mathbb{H}, G_T$, all of known group order $N = p_1 \cdots p_{n'}$, generators $g, h, (g_{p_i})_{i\in[n']}$, and for $\Theta(\lambda)$-bit primes $(p_i)_i$.

Dual System Groups. We recap a special variant of dual-system groups (DSGs) due to Gong et al. [GCTC16] (which is based on [CW13]). We only need a relaxed version of their DSG. We want emphasize that we left out unnecessary features and did not add anything to the syntax, correctness, or security. Hence, we can safely assume that our relaxed version is implied by the full DSG version from [GCTC16] and we give a concrete prime-order instantiation from the standard d-Lin assumption in the standard model inSubsection A.Starting from the initial work by Waters [Wat09], the richness of the dual-system paradigm was demonstrated in several prior works already (e.g., [LW10, LW11, OT12, CW13, HKS15, AHY15, GCD+16, GCTC16, GWW19, GW20]). The concept of [GCTC16] is particularly useful as it provides us with functionalities that are also essential in the UE paradigm with solely ciphertext updates in the forward direction. This connection is new and enriches the applications of the dual-system paradigm. Our (relaxed) DSG DSG consists of the PPT algorithms (SampP, $\mathsf{SampG}, \mathsf{SampH}, \mathsf{SampS}, \mathsf{SampK}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}}$):

$\mathsf{SampP}(\lambda, n)$: sample $(\mathbb{G}, \mathbb{H}, G_T, N, (g_{p_i})_{i\in[n']}, e) \leftarrow \mathsf{G}(\lambda, n')$, for fixed integer n'. Define $m : \mathbb{H} \to \mathbb{G}_T$ to be linear map, let $\widehat{g}$ and $\widehat{h}$ be group elements generated by g_s and h_s, respectively (see below). Further, $pars, \widehat{pars}$ may contain arbitrary information. Output public parameters $pp = (\mathbb{G}, \mathbb{H}, G_T, N, e, m, pars)$ and secret parameters $sp = (\widehat{g}, \widehat{h}, \widehat{pars})$

$\mathsf{SampG}(pp)$: output $\mathbf{g} = (g_0, \ldots, g_n) \in \mathbb{G}^{n+1}$.

$\mathsf{SampS}(pp)$: output $S \in \mathbb{G}$.

$\mathsf{SampH}(pp)$: output $\mathbf{h} = (h_0, \ldots, h_n) \in \mathbb{H}^{n+1}$.

$\mathsf{SampK}(pp)$: output $K \in \mathbb{H}$.

$\widehat{\mathsf{SampG}}(pp, sp)$: output $\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_n) \in \mathbb{G}^{n+1}$ and $g_s \in \mathbb{G}$.

$\widehat{\mathsf{SampH}}(pp, sp)$: output $\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n) \in \mathbb{H}^{n+1}$ and $h_s, h_a \in \mathbb{H}$.

Remark. $\mathsf{SampG}, \mathsf{SampS}, \mathsf{SampH}$ and SampK sample from a "normal" distribution (used for correctness) while $\widehat{\mathsf{SampG}}$ and $\widehat{\mathsf{SampH}}$ sample from a "semi-functional" distribution (used in the security proof). When

proving UE security, we can switch UE ciphertexts and keys to semi-functional ones. The essence of dual system is then carried out, namely, semi-functional ciphertexts and keys are incompatible meaning that we can derive at a stage where the UE ciphertexts carry a uniformly random group element and indistinguishability can be shown.

Correctness. For all $\lambda, n \in \mathbb{N}$, for all pp (generated via $\mathsf{SampP}(\lambda, n)$):

Projectiveness. $m(h)^s = e(\mathsf{SampS}(pp; s), h)$, for all $s \in \mathbb{Z}_N^*$ and $h \in \mathbb{H}$.

Orthogonality. $e(S, h_i) = 1$ and $e(g_0, K) = 1$, for all $i \in [n]$, $(h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$, $S \leftarrow \mathsf{SampS}(pp)$, $(g_0, \ldots) \leftarrow \mathsf{SampH}(pp)$, and $K \leftarrow \mathsf{SampK}(pp)$.

Associativity. $e(g_0, h_i) = e(g_i, h_0)$, for all $i \in [n]$, $(g_0, \ldots, g_n) \leftarrow \mathsf{SampG}(pp)$ and $(h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$.

$\mathbb{G}$-$\mathbb{H}$-*subgroups.* The outputs of $\mathsf{SampG}(pp)$ and $\mathsf{SampS}(pp)$ are uniformly distributed over the generators of non-trivial subgroups of $\mathbb{G}^{n+1}$ and $\mathbb{G}$, respectively. The outputs of $\mathsf{SampH}(pp)$ and $\mathsf{SampK}(pp)$ are uniformly distributed over the generators of non-trivial subgroups of $\mathbb{H}^{n+1}$ and $\mathbb{H}$, respectively.

Security. For all $\lambda, n \in \mathbb{N}$, for all $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, n)$:

Orthogonality. $m(\widehat{h}) = 1$.

Non-degeneracy. $\widehat{h}$ lies in a subgroup of h_s, g_s lies in a subgroup of $\widehat{g}$.

Left-subgroup indistinguishability (LS). For any PPT D, $\mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},D}(\lambda, n) :=$

$$\left|\Pr\left[D(pp, \mathbf{g}) = 1\right] - \Pr\left[D(pp, \mathbf{g}\widehat{\mathbf{g}}) = 1\right]\right|$$

is negligible in λ, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$ and $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.

Right-subgroup indistinguishability (RS). For any PPT D, $\mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, n) :=$

$$\left|\Pr\left[D(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{h}) = 1\right] - \Pr\left[D(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{h}\widehat{\mathbf{h}}) = 1\right]\right|$$

is negligible in λ, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and $(\widehat{\mathbf{h}}, \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

Parameter-hiding. The distributions $\{pp, \widehat{g}, \widehat{h}, \widehat{\mathbf{g}}, \widehat{\mathbf{h}}\}$ and $\{pp, \widehat{g}, \widehat{h}, \widehat{\mathbf{g}}\widehat{\mathbf{g}}', \widehat{\mathbf{h}}\widehat{\mathbf{h}}'\}$ are identically distributed, for $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_n), g_s) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n), h_s, h_a) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $\widehat{\mathbf{g}}' = (1, g_s^{\gamma_1}, \ldots, g_s^{\gamma_n})$, and $\widehat{\mathbf{h}}' = (1, h_s^{\gamma_1}, \ldots, h_s^{\gamma_n})$, for $\gamma_1, \ldots, \gamma_n \leftarrow \mathbb{Z}_N$.

Computational non-degeneracy (ND). For any PPT D, $\mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D}(\lambda, n) :=$

$$\left|\Pr\left[D(pp, \mathbf{S} \cdot \mathbf{g}\widehat{\mathbf{g}}, K \cdot \widehat{h}^{\alpha}, e(S, K)) = 1\right] - \Pr\left[D(pp, \mathbf{S} \cdot \mathbf{g}\widehat{\mathbf{g}}, K \cdot \widehat{h}^{\alpha}, R = 1\right]\right|$$

is negligible in λ, for $\mathbf{S} = (S, 1, \ldots)$, $S \leftarrow \mathsf{SampS}(pp)$, $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $K \leftarrow \mathsf{SampK}(pp)$, $\alpha \leftarrow \mathbb{Z}_N$, and $R \leftarrow G_T$.

Remark. The properties have the following implications which we will need later on. From orthogonality and projectiveness, we retrieve $e(S, \widehat{h}) = 1$. By projectiveness, it holds $m(K)^s \cdot m(K')^s = e(\mathsf{SampS}(pp; s), K) \cdot e(\mathsf{SampS}(pp; s), K') = m(K \cdot K')^s$, for $s \in \mathbb{Z}_N^*$, and $K, K' \in \mathbb{H}$. Moreover, by projectiveness and $\mathbb{G}$-subgroups, we have $m(K)^s \cdot m(K)^{s'} = e(\mathsf{SampS}(pp; s), K) \cdot e(\mathsf{SampS}(pp; s'), K) = e(g^{s+s'}, K) = m(K)^{s+s'}$, for $K \in \mathbb{H}$ and suitable generator $g \in \mathbb{G}$.

3.2 Constructing UE Schemes with Expiry Epochs

The main idea to construct UE schemes with expiry epochs from DSG is the following. In each epoch e, keys and ciphertexts are consistent in the sense that the associativity, orthogonality, and protectiveness properties of DSG ensure correctness. A token Δ_{e+1} is generated with a tweak δ that "lifts" keys and ciphertexts from epoch e to $e + 1$. It is important to hide such a tweak in the token (since otherwise a token can undo a ciphertext update).

Construction of UE from DSG

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(pp, \mathsf{poly}(\lambda))$, set $n = \mathsf{poly}(\lambda)$, sample $(h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$, $k_1 \leftarrow \mathsf{SampK}(pp)$ and return $K_1 = (h_0, k_1 \cdot h_1, k_2, \ldots, h_n, m(k_1), pp)$.

$\mathsf{Next}(K_e)$: for $K_e = (T_0, T_1, T_{e+1}, \ldots, T_n, m(k_e), pp)$, sample $\delta \leftarrow \mathsf{SampK}(pp)$ and $(h_0, \ldots, h_n), (h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)$, and return

$$\Delta_{e+1} = (h_0, \delta \cdot h_{e+1}, m(k_e \cdot \delta))$$

$$K_{e+1} = (T_0 h'_0, T_1 \cdot \delta \cdot T_{e+1} \prod_{i=1}^{e+1} h'_i, T_{e+2} h'_{e+2}, \ldots, T_n h'_n, m(k_e \cdot \delta), pp).$$

Remark. δ is the "tweak" towards the new key; h_0, h_{e+1} hide the tweak while the h'_i-values re-randomize the new key.

$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: for $K_e = (\ldots, m(k_e), pp)$, sample $(g_0, \ldots, g_n) \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$. If $e_{\mathsf{exp}} > n(\lambda)$, then set $e_{\mathsf{exp}} = \lfloor \mathsf{poly}(\lambda) \rfloor$. Return

$$C_e = (S g_0, \prod_{i=1}^{e} g_i, g_{e+1}, \ldots, g_{e_{\mathsf{exp}}}, m(k_e)^s \cdot M).$$

$\mathsf{Update}(\Delta_{e+1}, C_e)$: for $\Delta_{e+1} = (D_0, D_1, m(k_{e+1}))$ and $C_e = (S_0, S_1, S_{e+1}, \ldots, S_{e_{\mathsf{exp}}}, S_T)$, sample $S \leftarrow \mathsf{SampS}(pp; s')$, for $s' \leftarrow \mathbb{Z}_N^*$, $(g'_0, \ldots, g'_n) \leftarrow \mathsf{SampG}(pp)$, compute $m(\delta)^s = \frac{e(S_0, D_1)}{e(D_0, S_{e+1})}$, and return

$$C_{e+1} = (S_0 S g'_0, S_1 S_{e+1} \prod_{i=1}^{e+1} g'_i, S_{e+2} g'_{e+2}, \ldots, S_{e_{\mathsf{exp}}} g'_{e_{\mathsf{exp}}}, S_T m(\delta)^s m(k_{e+1})^{s'}).$$

$\mathsf{Dec}(K_e, C_e)$: for $K_e = (T_0, T_1, \ldots)$ and $C_e = (S_0, S_1, \ldots, S_T)$ return

$$M = S_{G_T} \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)}.$$

Correctness of UE

1) Correct decryption of epoch-e ciphertexts:

$$M = S_T \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)} = m(k_e)^s \cdot M \cdot \frac{e(h_0, \prod_{i=1}^{e} g_i)}{e(S g_0, k_e \prod_{i=1}^{e} h_i)} = \frac{m(k_e)^s}{e(S, k_e)} \cdot M = M,$$

where $m(k_e)^s = e(S, k_e)$, for some $s \in \mathbb{Z}_N^*$, $e(S, h_i) = 1$, for all $i \in [e]$, and $e(g_0, k_e) = 1$ due to projectiveness, orthogonality, and associativity.

2) Correct updates of epoch-e to epoch-$(e+1)$ ciphertexts:

$$C_{e+1} = (S'' g''_0, \prod_{i=1}^{e+1} g''_i, g''_{e+2}, \ldots, g''_{e_{\mathsf{exp}}}, m(k_{e+1})^{s+s'} \cdot M),$$

where $S'' = \mathsf{SampS}(pp; s+s')$, $(g''_0, \ldots, g''_n) = \mathsf{SampG}(pp; s+s')$, $m(k_{e+1})^{s+s'} = m(k_e)^s \cdot m(\delta)^s \cdot m(k_{e+1})^{s'}$, for $m(\delta)^s = \frac{e(S_0, D_1)}{e(D_0, S_{e+1})} = \frac{e(S g_0, \delta h_{e+1})}{e(h_0, g_{e+1})}$, due to to projectiveness, orthogonality, associativity, and $\mathbb{G}$-$\mathbb{H}$-subgroups.

Fig. 6. Construction of UE from DSG with correctness.

Blinding the tweak with a uniform epoch-specific term h_{e+1} helps us where the corresponding element g_{e+1} is *only* present in ciphertexts for epoch e, but will not be made available in ciphertexts for epoch $e+1$. This construction strategy also ensures that the token cannot be used to update keys in any direction as such a token is not "compatible" with neither K_e nor K_{e+1}. Moreover, the update via a token only works as long as e_{exp} is not reached and "update" ciphertext elements $\ldots, g_{e_{\mathsf{exp}}}$ are available. In that sense, the ciphertext as well as the key will shrink the further epochs progress.

Definition 5 (UE Construction). *The construction of a UE scheme* UE *from DSG* DSG *with correctness is shown in Fig. 6. Security is proved in Theorem 1.*

We now prove security of UE and provide detailed information about the game hops in the beginning of the proof of the following theorem.

Theorem 1. *If* DSG *is a DSG scheme, then* UE *is* IND-UE-CPA*-secure. Concretely, for any PPT adversary A and $n = \mathsf{poly}(\lambda)$, it holds:*

$$\mathsf{Adv}^{\text{ind-ue-cpa}}_{\mathsf{UE}, A}(\lambda) \leq \mathsf{Adv}^{\text{ls}}_{\mathsf{DSG}, D_1}(\lambda, n) + 2 \cdot \mathsf{Adv}^{\text{rs}}_{\mathsf{DSG}, D_2}(\lambda, n)/n + \mathsf{Adv}^{\text{nd}}_{\mathsf{DSG}, D_3}(\lambda, n).$$

Proof. We proceed in a sequence of games:

Game 0. The IND-UE-CPA experiment.

Game 1. Fresh encryption of $C^*_{e,1}$ in Chall-oracle by using Dec and Enc instead of Update. This is a conceptional change. Here, we use the fact that fresh ciphertexts and updated ciphertexts are indistinguishable.

Game 2. The challenge ciphertext(s) are semi-functional (via left-subgroup indistinguishability (LS)). This is a property that comes from the underlying DSG assumption and is a common hybrid step in proving constructions secure in the dual-system paradigm.

Game 3. The keys and tokens are generated directly in the Next' oracle (instead of calling Next). Moreover, the token is used to generate the next key. We observe that a token Δ_{e^*} can be used to compute the corresponding key K_{e^*}. This is a crucial change as we prepare the embedding of uniform randomness into Δ_{e^*}. See that such a token by construction must not carry a prefix of the challenge ciphertext in challenge epoch e^*.

Game 4. The token Δ_{e^*} and the keys K_e, for challenge epoch e^* and $e \geq e^*$, are pseudo-normal (via right-subgroup indistinguishability (RS)). This is a common next step in many dual-system proofs to prepare the embedding of uniform elements.

Game 5. The token Δ_{e^*} and the keys K_e, for challenge epoch e^* and $e \geq e^*$, are pseudo-normal semi-functional (via parameter-hiding). This is further common step in many dual-system proofs to prepare the embedding of a uniform element. The result is that Δ_{e^*} now carries a uniform element $(\widehat{h})^\alpha$. By the argument we prepared in Game 3, this also ensures that keys K_e, for $e \geq e^*$, also have such a uniform blinding term.

Game 6. The token Δ_{e^*} and the keys K_e, for challenge epoch e^* and $e \geq e^*$, are semi-functional (via right-subgroup indistinguishability (RS)). This is a common next step in many dual-system proofs to finalize the embedding of a uniform element.

Game 7. The message in the challenge ciphertext(s) is a uniform G_T-element (via computational non-degeneracy (ND)). In this game, the message is independent of the bit b and security readily follows.

Let $S_{A,j}$ be the event that A succeeds in Game j. We highlight changes boxed.

Lemma 3 (Game 0 to Game 1). *For any PPT adversary A, it holds:* $|\Pr[S_{A,0}] - \Pr[S_{A,1}]| = 0$.

Proof. This is a conceptional change. The Chall oracle in Game 1 is as follows:

$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. If $b = 0$, set $C^*_{e,0} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$, else run decryption as

$$\boxed{M' \leftarrow \mathsf{Dec}(K_{e-1}, C_{e-1,e_{\mathsf{exp}}})} \text{ and } \boxed{C^*_{e,1} \leftarrow \mathsf{Enc}(K_e, M', e_{\mathsf{exp}})}.$$

Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,b}$.

Due to correctness (i.e., via perfect re-randomization), the ciphertexts derived from Enc and Update for an epoch e yield the same distribution. Hence, such a change cannot be detected by A.

Lemma 4 (Game 1 to Game 2). *For any PPT adversary A there is a distinguisher D on LS such that* $|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},D}(\lambda, n)$.

Proof. The input is provided as $(pp, \mathbf{T})$, where $\mathbf{T} = (T_0, \ldots, T_n)$ is either $\mathbf{g}$ or $\mathbf{g}\widehat{\mathbf{g}}$, for $\mathbf{g} = (g_0, \ldots, g_n) \leftarrow \mathsf{SampG}(pp)$ and $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_n), \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$. The Chall oracle in Game 2 is as follows:

$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{\begin{aligned} C^*_{e,0} &= (S \cdot T_0, \prod_{i=1}^{e} T_i, T_{e+1}, \ldots, T_{e_{\mathsf{exp}}}, m(k_e)^s \cdot M) \\ C^*_{e,1} &= (S \cdot T_0, \prod_{i=1}^{e} T_i, T_{e+1} \ldots, T_{e_{\mathsf{exp}}}, m(k_e)^s \cdot M') \end{aligned}}$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, $\boxed{S \leftarrow \mathsf{SampS}(pp; s)}$, and $\boxed{s \leftarrow \mathbb{Z}^*_N}$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,b}$.

If $\mathbf{T} = \mathbf{g}$, then the challenge ciphertext(s) are distributed as in Game 1. If $\mathbf{T} = \mathbf{g}\widehat{\mathbf{g}}$, then the challenge ciphertext(s) are distributed as in Game 2.

Lemma 5 (Game 2 to Game 3). *For any PPT adversary A, it holds:* $|\Pr[S_{A,2}] - \Pr[S_{A,3}]| = 0$.

Proof. This is a conceptional change. The Next'-oracle in Game 3 is as follows (where we directly compute Δ_{e+1} and K_{e+1} instead of calling Next, use Δ_{e+1} to compute K_{e+1}, and store k_i-elements):

Next' : for $K_e = (T_0, T_1, T_{e+1}, \ldots, T_n, m(k_e), pp)$, sample

$$\boxed{\delta \leftarrow \mathsf{SampK}(pp)} \text{ and } \boxed{(h_0, \ldots, h_n), (h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)}, \text{ store } \boxed{k_{e+1} := k_e \cdot \delta}$$

and compute

$$\begin{aligned}
\Delta_{e+1} &= (h_0, \delta \cdot h_{e+1}, m(k_{e+1})) =: (D_0, D_1, D_2) \\
K_{e+1} &= (T_0 \boxed{D_0} h'_0, T_1 \cdot \boxed{D_1} T_{e+1} h'_{e+1} \prod_{i=1}^{e} \boxed{h_i} h'_i, T_{e+2} \boxed{h_{e+2}} h'_{e+2}, \ldots, \\
&\quad T_n \boxed{h_n} h'_n, \boxed{D_2}, pp).
\end{aligned}$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e + 1$.

See that we can safely use the above boxed h_i-elements used to compute Δ_{e+1} for K_{e+1} as those elements are perfectly re-randomized by the h'_i-elements (due to the $\mathbb{G}$-$\mathbb{H}$-subgroups property) which yields a consistent distribution for K_{e+1}. (Looking ahead, in the remaining game hops, we will use such a change to embed a uniformly random element in Δ_{e^*} and K_e, for $e \geq e^*$.)

Lemma 6 (Game 3 to Game 4). *For any PPT adversary A there is a distinguisher D on RS such that* $|\Pr[S_{A,3}] - \Pr[S_{A,4}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, n)/n$.

Proof. The input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T} = (T_0, \ldots, T_n)$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for $\mathbf{h} = (h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n), \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, and $\mathbf{g}\widehat{\mathbf{g}} = (g_0\widehat{g}_0, \ldots, g_n\widehat{g}_n)$. The reduction guesses the challenge epoch $e^* \leftarrow [n]$. The Next' (epoch $e^* - 1$ only) and Chall oracles in Game 4 are:

Next' : for $K_{e^*-1} = (T^*_0, T^*_1, T^*_{e^*}, \ldots, T^*_n, m(k_{e^*-1}), pp)$, sample $\delta \leftarrow \mathsf{SampK}(pp)$ and $(h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)$, store $k_{e^*} := k_{e^*-1} \cdot \delta$, and compute

$$\begin{aligned}
\Delta_{e^*} &= (\boxed{T_0}, \delta \cdot \boxed{T_{e^*}}, m(k_{e^*})) =: (D_0, D_1, D_2) \\
K_{e^*} &= (T^*_0 D_0 h'_0, T^*_1 \cdot D_1 \cdot T^*_{e^*} h'_{e^*} \prod_{i=1}^{e^*-1} \boxed{T_i} h'_i, T_{e^*+1} \boxed{T_{e^*+1}} h'_{e^*+1}, \ldots, \\
&\quad T_n \boxed{T_n} h'_n, D_2), pp).
\end{aligned}$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e^*,b} \leftarrow \mathsf{Update}(\Delta_{e^*}, C^*_{e^*-1,b})$. Set $e = e + 1$.
$\mathsf{Chall}(M, C_{e^*-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e^* - 1, C_{e^*-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{\begin{aligned}
C^*_{e^*,0} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, m(k_{e^*})^s \cdot M) \\
C^*_{e^*,1} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, m(k_{e^*})^s \cdot M'),
\end{aligned}}$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, for $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}^*_N$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e^*,b})$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e^*,b}$.

If $\mathbf{T} = \mathbf{h}$, then the token Δ_{e^*} and the keys K_e, for $e \geq e^*$, are distributed as in Game 3. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the token Δ_{e^*} and the keys K_e, for $e \geq e^*$, are distributed as in Game 4. This game hop introduces a loss of a factor $n = \mathsf{poly}(\lambda)$.

Lemma 7 (Game 4 to Game 5). *For any PPT adversary A, it holds:* $|\Pr[S_{A,4}] - \Pr[S_{A,5}]| = 0$.

Proof. We use the parameter-hiding property which states that the distributions

$$\{pp, \widehat{g}, \widehat{h}, \widehat{\mathbf{g}}, \widehat{\mathbf{h}}\} \text{ and } \{pp, \widehat{g}, \widehat{h}, \widehat{\mathbf{g}}\widehat{\mathbf{g}}', \widehat{\mathbf{h}}\widehat{\mathbf{h}}'\}$$

are identically distributed, for $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_n), g_s) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n), h_s, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $\widehat{\mathbf{g}}' = (1, g_s^{\gamma_1}, \ldots, g_s^{\gamma_n})$, and $\widehat{\mathbf{h}}' = (1, h_s^{\gamma_1}, \ldots, h_s^{\gamma_n})$, for $\gamma_1, \ldots, \gamma_n \leftarrow \mathbb{Z}_N$.

This is reminiscent of Lemma 11 in [GCTC16] and we information-theoretically embed $(\widehat{h})^\alpha$. This results in a pseudo-normal semi-functional token Δ_{e^*} and keys K_e, for $e \geq e^*$. As shown in [GCTC16], due to non-degeneracy, we have that $(h_s)^{\alpha'}$ can be replaced by some suitable $(\widehat{h})^\alpha$, for suitable $\alpha, \alpha' \in \mathbb{Z}_N$. This can be embeded into Δ_{e^*} (which the adversary can query) since no (prefix) element $g_{e^*}\widehat{g}_{e^*}$ is present in the challenge ciphertext(s). Since Δ_{e^*} is used to compute keys K_e, for $e \geq e^*$, the blinding term $(\widehat{h})^\alpha$ is embedded in all remaining epoch keys as well. This is crucial as we need to blind all occurrences of K_{e^*}. Particular, see that $k_{e^*}k_{e^*-1}^{-1} = \delta$ which is why we start from the token Δ_{e^*} which has information on δ embedded. The Next' (epoch $e^* - 1$ only) and Chall oracles in Game 5 are:

Next' : for $K_{e^*-1} = (T_0^*, T_1^*, T_{e^*}^*, \ldots, T_n^*, m(k_{e^*-1}), pp)$, sample $\boxed{\alpha \leftarrow \mathbb{Z}_N^*}$, $\delta \leftarrow \mathsf{SampK}(pp)$, $(h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)$, store $k_{e^*} := k_{e^*-1} \cdot \delta$, and compute

$$\Delta_{e^*} = (h_0\widehat{h}_0, \delta \cdot \boxed{(\widehat{h})^\alpha} \cdot h_{e^*}\widehat{h}_{e^*}, m(k_{e^*})) =: (D_0, D_1, D_2)$$

$$K_{e^*} = (T_0^* D_0 h'_0, T_1^* \cdot D_1 \cdot h'^{\,}_{e^*} \prod_{i=1}^{e^*-1} h_i \boxed{\widehat{h}_i h_s^{\gamma_i}} h'_i, h_{e^*+1} \boxed{\widehat{h}_{e^*+1} h_s^{\gamma_{e^*+1}}} h'_{e^*+1},$$

$$\ldots, h_n \boxed{\widehat{h}_n h_s^{\gamma_n}} h'_n, D_2, pp).$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e^*,b} \leftarrow \mathsf{Update}(\Delta_{e^*}, C^*_{e^*-1,b})$. Set $e = e + 1$.
$\mathsf{Chall}(M, C_{e^*-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e^* - 1, C_{e^*-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$C^*_{e^*,0} = (Sg_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i \boxed{\widehat{g}_i g_s^{\gamma_i}}, g_{e^*+1} \boxed{\widehat{g}_{e^*+1} g_s^{\gamma_{e^*+1}}}, \ldots, g_{e_{\mathsf{exp}}} \boxed{\widehat{g}_{e_{\mathsf{exp}}} g_s^{\gamma_{e_{\mathsf{exp}}}}}, m(k_{e^*})^s \cdot M),$$

$$C^*_{e^*,1} = (Sg_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i \boxed{\widehat{g}_i g_s^{\gamma_i}}, g_{e^*+1} \boxed{\widehat{g}_{e^*+1} g_s^{\gamma_{e^*+1}}}, \ldots, g_{e_{\mathsf{exp}}} \boxed{\widehat{g}_{e_{\mathsf{exp}}} g_s^{\gamma_{e_{\mathsf{exp}}}}}, m(k_{e^*})^s \cdot M'),$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, for $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$, and $(g_0, \ldots, g_n) \leftarrow \mathsf{SampG}(pp)$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e^*,b})$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e^*,b}$.

It is important to note that essentially $(\widehat{h})^\alpha$ "blinds" the key element k_{e^*} which is used in the epoch-e^* challenge ciphertext via $m(k_{e^*})$. See that all remaining keys in epochs $e > e^*$ also have this blinding term.

Lemma 8 (Game 5 to Game 6). *For any PPT adversary A there is a distinguisher D on RS such that* $|\Pr[S_{A,5}] - \Pr[S_{A,6}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, n)/n$.

Proof. The input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T} = (T_0, \ldots, T_n)$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for $\mathbf{h} = (h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n), \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, and $\mathbf{g}\widehat{\mathbf{g}} = (g_0\widehat{g}_0, \ldots, g_n\widehat{g}_n)$. The reduction guesses $e^* \leftarrow [n]$. The Next' (epoch $e^* - 1$ only) and Chall oracles in Game 6 are as follows:

Next' : for $K_{e^*-1} = (T_0^*, T_1^*, T_{e^*}^*, \ldots, T_n^*, m(k_{e^*-1}), pp)$, sample $\alpha \leftarrow \mathbb{Z}_N^*$, $\delta \leftarrow \mathsf{SampK}(pp)$, $(h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)$, store $k_{e^*} := k_{e^*-1} \cdot \delta$, and compute

$$\Delta_{e^*} = (\boxed{T_0}, \delta \cdot (\widehat{h})^\alpha \cdot \boxed{T_{e^*}}, m(k_{e^*})) =: (D_0, D_1, D_2)$$

$$K_{e^*} = (T_0^* D_0 h'_0, T_1^* D_1 T_{e^*}^* h'^{\,}_{e^*} \prod_{i=1}^{e^*-1} \boxed{T_i} h'_i, T_{e^*+1}^* \boxed{T_{e^*+1}} h'_{e^*+1}, \ldots, T_n^* \boxed{T_n} h'_n, D_2, pp).$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e^*,b} \leftarrow \mathsf{Update}(\Delta_{e^*}, C^*_{e^*-1,b})$. Set $e = e + 1$.

$\mathsf{Chall}(M, C_{e^*-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e^* - 1, C_{e^*-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{\begin{aligned} C^*_{e^*,0} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, m(k_{e^*})^s \cdot M), \\ C^*_{e^*,1} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, m(k_{e^*})^s \cdot M'), \end{aligned}}$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, for $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e^*,b})$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e^*,b}$.

If $\mathbf{T} = \mathbf{h}$, then the token Δ_{e^*} and the keys K_e, for $e \geq e^*$, are distributed as in Game 6. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the token Δ_{e^*} and the keys K_e,for $e \geq e^*$, are distributed as in Game 5. This game hop introduces a loss of a factor $n = \mathsf{poly}(\lambda)$.

Lemma 9 (Game 6 to Game 7). *For any PPT adversary A there is a distinguisher D on ND such that $|\Pr[S_{A,6}] - \Pr[S_{A,7}]| \leq \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D}(\lambda, n)$.*

Proof. The input is provided as $(pp, \mathbf{S}\mathbf{g}\widehat{\mathbf{g}}, K \cdot (\widehat{h})^\alpha, \mathbf{T})$, where $\mathbf{T}$ is either $e(S, K)$ or $R \leftarrow G_T$, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $(\mathbf{h}) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}}, \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $S \leftarrow \mathsf{SampS}(pp)$, $\mathbf{S} = (1, 0, \ldots)$, $K \leftarrow \mathsf{SampK}(pp)$. We implicitly set $\boxed{\delta = K \cdot k^{-1}_{e^*-1}}$ where k_{e^*-1} is known to the reduction. The Next' (only for $e^* - 1$) and Chall oracles are:

Next' : for $K_{e^*-1} = (T^*_0, T^*_1, T^*_{e^*}, \ldots, T^*_n, m(k_{e^*-1}), pp)$, sample $\alpha \leftarrow \mathbb{Z}_N^*$, $(h_0, \ldots, h_n)(h'_0, \ldots, h'_n) \leftarrow \mathsf{SampH}(pp)$, and compute

$$\begin{aligned} \Delta_{e^*} &= (h_0, \boxed{\delta \cdot (\widehat{h})^\alpha} \cdot h_{e^*}, m(\boxed{K \cdot (\widehat{h})^\alpha})) =: (D_0, D_1, D_2) \\ K_{e^*} &= (T^*_0 D_0 h'_0, T^*_1 D_1 T^*_{e^*} h'_{e^*} \prod_{i=1}^{e^*-1} h_i h'_i, T^*_{e^*+1} h_{e^*+1} h'_{e^*+1}, \ldots, T^*_n h_n h'_n, D_2, pp). \end{aligned}$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e^*,b} \leftarrow \mathsf{Update}(\Delta_{e^*}, C^*_{e^*-1,b})$. Set $e = e + 1$.
$\mathsf{Chall}(M, C_{e^*-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e^* - 1, C_{e^*-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{\begin{aligned} C^*_{e^*,0} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, \mathbf{T} \cdot M) \\ C^*_{e^*,1} &= (S \cdot g_0\widehat{g}_0, \prod_{i=1}^{e^*} g_i\widehat{g}_i, g_{e^*+1}\widehat{g}_{e^*+1}, \ldots, g_{e_{\mathsf{exp}}}\widehat{g}_{e_{\mathsf{exp}}}, \mathbf{T} \cdot M') \end{aligned}}$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e^*,b})$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e^*,b}$.

See that $m(K \cdot (\widehat{h})^\alpha) = m(K)$ holds. If $\mathbf{T} = e(S, K)$, then the challenge ciphertext(s) are distributed as in Game 6. If $\mathbf{T} = R$, then the challenge ciphertext(s) are distributed as in Game 7.

Lemma 10 (Game 7). *For any PPT adversary A, $\Pr[S_{A,7}] = 1/2$ holds.*

Proof. In Game 7, for (uniform) $b \in \{0, 1\}$, we provide A with challenge ciphertext(s) that include a uniform G_T-element instead of a A-chosen b-dependent message. Hence, b is completely hidden from A's view.

Taking Lemmata 3, 4, 5, 6, 7, 8, 9, and 10 together, shows Theorem 1. □

3.3 UE with Expiry Epochs with Sublinear Keys and Ciphertexts

We now give an adaption of our UE scheme with sublinear key and ciphertext sizes dubbed UE^*. Such a construction is more involved than the construction and the proof of UE. We start with encoding of epochs as a first step.

Encoding of epochs. We use the encoding function of the recent work due to Drijvers, Gorbunov, Neven, and Wee [DGNW20, Sec. 4.2]. They give a function e that maps tags $\mathbf{t} = (t_1, \ldots) \in \{1,2\}^{\leq \lambda-1}$, for $\lambda = \lfloor \log_2 n \rfloor$, to epochs $[n]$:

$$e(\mathbf{t}) = 1 + \sum_{i=1}^{|\mathbf{t}|} (1 + (2^{\lambda-i} - 1)(t_i - 1)).$$

Moreover, $\mathbf{t}$ is the inverse function that maps epochs $e \in [n]$ to tags $\{1,2\}^{\leq \lambda-1}$:

$$\mathbf{t}(e) = \begin{cases} \varepsilon & \text{if } e = 1 \\ \mathbf{t}(e-1)||1 & \text{if } |\mathbf{t}(e-1)| < \lambda - 1 \\ \bar{\mathbf{t}}||2 & \text{if } |\mathbf{t}(e-1)| = \lambda - 1, \end{cases}$$

for longest string $\bar{\mathbf{t}}$ such that $\bar{\mathbf{t}}||1$ is a prefix of $t(e-1)$. Furthermore, they define sets $\Gamma_{\mathbf{t}} \subset \{1,2\}^{\leq \lambda-1}$ for each $\mathbf{t}$ such that:

$$\Gamma_{\mathbf{t}} = \{\mathbf{t}\} \cup \{\bar{\mathbf{t}}||2 : \bar{\mathbf{t}}||1 \text{ prefix of } \mathbf{t}\}.$$

The properties of $\Gamma_{\mathbf{t}}$ are: (1) $\mathbf{t} \preceq \mathbf{t}' \Leftrightarrow \exists \mathbf{u} \in \Gamma_{\mathbf{t}}$ such that $\mathbf{u}$ is a prefix of $\mathbf{t}'$, (2) $\forall \mathbf{t}$, it holds $\Gamma_{\mathbf{t}(e(\mathbf{t})+1)} = \Gamma_{\mathbf{t}} \backslash \{\mathbf{t}\}$ if $|\mathbf{t}| = \lambda - 1$ or $\Gamma_{\mathbf{t}(e(\mathbf{t})+1)} = (\Gamma_{\mathbf{t}} \backslash \{\mathbf{t}\}) \cup \{\mathbf{t}||1, \mathbf{t}||2\}$ otherwise, (3) $\forall \mathbf{t}' \succ \mathbf{t}$, it holds $\forall \mathbf{u}' \in \Gamma_{\mathbf{t}'}, \exists \mathbf{u} \in \Gamma_{\mathbf{t}}$ such that $\mathbf{u}$ is a prefix of $\mathbf{u}'$.

The main construction idea is the following. The first ingredient is the above encoding mapping epochs to encoded tags. See that such tags have only length of $\lambda - 1$ while allowing 2^λ epochs. One can think of it as a binary-tree encoding as discussed in detail in [DGNW20, Sec. 4.2] where nodes are labeled as time epochs. However, this is not sufficient and we need a second ingredient, namely group elements from the DSG, to support such an encoding in the final UE scheme. Inspired by [DGNW20] but used in the DSG context, we can carefully stick together all ingredients and we adapt our security proof for UE to prove security also for UE^*. With this, we are able to lift our linear-size UE construction UE to a sublinear-size UE scheme UE^*. Again, keys and ciphertexts in each epoch are consistent. The keys and the ciphertexts are now generated differently where the encoding function comes in. This results in several key and ciphertext parts where all key parts now carry the main key elements as well. For ciphertext, the main hurdle is the common randomness that blinds the message part where such a randomness has to be included in all ciphertext parts.

Definition 6 (UE^* construction). *The construction of a UE scheme UE^* from DSG and correctness is shown in Fig. 7 and in Fig. 8, respectively. Security is proved in Theorem 2.*

In general, constructing the ciphertexts yields no problem (given our DSG abstraction), the keys now have several parts where we need to embed uniform randomness. This is the main hurdle in the proof and we therefor have to carefully embed uniform randomness in a step-by-step fashion. First, we start with making ciphertexts semi-functional. Then, we can embed randomness via parameter hiding in the token Δ_{e^*} to it semi-functional (see that it does not contain any prefix of the challenge ciphertext elements). Afterwards, we embed uniform randomness in the key $\mathcal{K}_{e'}$ (if queried by the adversary) or the key $\mathcal{K}_{e^*_{\mathsf{exp}}}$ (if $\mathcal{K}_{e'}$ was not queried). See that such keys can be made semi-functional trivially since if $\mathcal{K}_{e'}$ is queried, by the validity requirement, no token $\Delta_{e'}$ is allowed to be queried. Otherwise, if $\mathcal{K}_{e'}$ is not queried, $\mathcal{K}_{e^*_{\mathsf{exp}}}$ can be made semi-functional since such a key does not have any prefix element of the challenge ciphertext (as the challenge ciphertext has already expired in e^*_{exp}). In that sense, security follows readily and we formally prove the theorem below.

Theorem 2. *If DSG is a DSG scheme, then UE^* is $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$-secure. Concretely, for any PPT adversary A, it holds:*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE}^*,A}(\lambda) \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},D_1}(\lambda,\lambda) + 2 \cdot (|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + 1) \cdot \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D_2}(\lambda,\lambda)/\mathsf{poly}(\lambda) + \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D_3}(\lambda,\lambda)/\mathsf{poly}(\lambda).$$

Construction of UE^* from DSG

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(pp, \lambda)$, sample $(h_0, \ldots, h_\lambda) \leftarrow \mathsf{SampH}(pp)$, $k_1 \leftarrow \mathsf{SampK}(pp)$ and return $K_1 = (\{K'_1\}, m(k_1), pp)$, with $K'_1 = (h_0, k_1 \cdot h_1, \ldots, h_\lambda)$.

$\mathsf{Next}(K_e)$: for $K_e = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(K_e), pp)$, if $|\mathbf{t}(e)| < \lambda - 1$, then find $K'_{\mathbf{t}(e)} = (T_0, T_1, T_{|\mathbf{t}(e)|+1}, \ldots, T_\lambda)$, compute

$$K'_{\mathbf{t}(e+1)} = (T_0, T_1 \cdot T_{|\mathbf{t}(e)|+1}, T_{|\mathbf{t}(e)|+2}, \ldots, T_\lambda),$$
$$K'_{\mathbf{t}(e+2)} = (T_0, T_1 \cdot T^2_{|\mathbf{t}(e)|+1}, T_{|\mathbf{t}(e)|+2}, \ldots, T_\lambda),$$

Sample $\delta \leftarrow \mathsf{SampK}(pp)$ and $(h_0, \ldots, h_\lambda), (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda})_{\mathbf{u} \in [|\Gamma_{\mathbf{t}(e+1)}|]} \leftarrow \mathsf{SampH}(pp)$, for $\mathbf{t}(e+1) = (\mathbf{t}(e+1)_1, \ldots, \mathbf{t}(e+1)_{|\mathbf{t}(e+1)|})$, compute

$$\Delta_{e+1} = (h_0, \delta \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, m(k_e \cdot \delta)),$$

$$K''_{\mathbf{u}} = (T'_{\mathbf{u},0} h_{\mathbf{u},0}, T'_{\mathbf{u},1} \cdot \delta \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T'_{\mathbf{u},|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T'_{\mathbf{u},\lambda} h_{\mathbf{u},\lambda}),$$

for $\{K'_{\mathbf{u}} = (T'_{\mathbf{u},0}, \ldots, T'_{\mathbf{u},\lambda}) : \mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e+1)}\}$. Set $K_{e+1} = (\{K''_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_e \cdot \delta), pp)$ and return (Δ_{e+1}, K_{e+1}).

$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: for $K_e = (\ldots, m(K_e), pp)$, sample $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$, and return

$$C_{e,e_{\mathsf{exp}}} = (\{(Sg_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{u_i}, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, g_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s \cdot M).$$

$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$: if $e < e_{\mathsf{exp}}$, return $\perp$. For $\Delta_{e+1} = (D_0, D_1, m(k_{e+1}))$ and $C_{e,e_{\mathsf{exp}}} = (\{C'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(k_e)^s \cdot M)$, if $|\mathbf{t}(e)| < \lambda - 1$, then find $C'_{\mathbf{t}(e)} = (S_0, S_1, S_{|\mathbf{t}(e)|+1}, \ldots, S_\lambda)$, compute

$$C'_{\mathbf{t}(e+1)} = (S_0, S_1 \cdot S_{|\mathbf{t}(e)|+1}, S_{|\mathbf{t}(e)|+2}, \ldots, S_\lambda),$$
$$C'_{\mathbf{t}(e+2)} = (S_0, S_1 \cdot S^2_{|\mathbf{t}(e)|+1}, S_{|\mathbf{t}(e)|+2}, \ldots, S_\lambda).$$

Sample $S \leftarrow \mathsf{SampS}(pp; s')$, for $s' \leftarrow \mathbb{Z}_N^*$, $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampG}(pp)$, compute $m(\delta)^s = \frac{e(S_{\mathbf{t}(e+1),0}, D_1)}{e(D_0, S_{\mathbf{t}(e+1),1})}$ and

$$C''_{\mathbf{u}} = (S_{\mathbf{u},0} Sg_{\mathbf{u},0}, S_{\mathbf{u},1} \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{u_i}, S_{\mathbf{u},|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, S_{\mathbf{u},\lambda} g_{\mathbf{u},\lambda}),$$

for $\{C'_{\mathbf{u}} = (S_{\mathbf{u},0}, \ldots, S_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}$. Return $C_{e+1,e_{\mathsf{exp}}} = (\{C''_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_e \cdot \delta)^{s+s'} \cdot M)$.

$\mathsf{Dec}(K_e, C_{e,e_{\mathsf{exp}}})$: for $K_e = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(K_e), pp)$ and $C_{e,e_{\mathsf{exp}}} = (\{C'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, S_T)$, find $K'_{\mathbf{t}(e)} = (T_0, T_1, \ldots)$ and $C'_{\mathbf{t}(e)} = (S_0, S_1, \ldots)$, and return

$$M = S_T \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)}.$$

Fig. 7. Construction of UE^* from DSG.

1) Correct decryption of epoch-e ciphertexts:

$$M = S_T \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)} = m(k_e)^s \cdot M \cdot \frac{e(h_0, \prod_{i=1}^{e} g_i)}{e(Sg_0, k_e \prod_{i=1}^{e} h_i)} = \frac{m(k_e)^s}{e(S, k_e)} \cdot M = M,$$

where $m(k_e)^s = e(S, k_e)$, for some $s \in \mathbb{Z}_N^*$, $e(S, h_i) = 1$, for all $i \in [e]$, and $e(g_0, k_e) = 1$ due to projectiveness, orthogonality, and associativity.

2) Correct updates of epoch-e to epoch-$(e+1)$ ciphertexts $C_{e+1,e_{\mathsf{exp}}} = (\{C''_{\mathbf{u}} : \mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_{e+1})^{s+s'} \cdot M)$:

$$C''_{\mathbf{u}} = (S_{\mathbf{u},0} Sg_{\mathbf{u},0}, S_{\mathbf{u},1} \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{\mathbf{u}_i}, S_{\mathbf{u},|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, S_{\mathbf{u},\lambda} g_{\mathbf{u},\lambda}),$$

where $S \leftarrow \mathsf{SampS}(pp; s')$, for $s' \leftarrow \mathbb{Z}_N^*$, $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampG}(pp)$, $m(k_{e+1})^{s+s'} = m(k_e)^s \cdot m(\delta)^s \cdot m(k_{e+1})^{s'}$, for $m(\delta)^s = \frac{e(S_{\mathbf{t}(e+1),0}, D_1)}{e(D_0, S_{\mathbf{t}(e+1),1})}$, due to to projectiveness, orthogonality, associativity, and $\mathbb{G}$-$\mathbb{H}$-subgroups.

Fig. 8. Correctness of UE^*.

Proof. We proceed similarly as in the proof of Theorem 1. However, we have to carefully introduce randomness in token Δ_{e^*} and keys K_e, for $e \geq e'$, for a specific "window" epoch $e' \leq e^*_{\mathsf{exp}}$, in a slightly different way to make them semi-functional. Let $S_{A,j}$ be the event that A succeeds in Game j. We proceeds with main changes here solely and box them.

Lemma 11 (Game 0 to Game 1). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,0}] - \Pr[S_{A,1}]| = 0.$$

Proof. This is a conceptional change (exactly as in the proof of Lemma 3). The Chall oracle in Game 1 is as follows:

$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. If $b = 0$, set $C^*_{e,0} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$, else run $\boxed{M' \leftarrow \mathsf{Dec}(K_{e-1}, C_{e-1,e_{\mathsf{exp}}})}$ and $\boxed{C^*_{e,1} \leftarrow \mathsf{Enc}(K_e, M', e_{\mathsf{exp}})}$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,b}$.

Due to correctness (i.e., via perfect re-randomization; cf. Fig. 8), the ciphertexts derived from Enc and Update for an epoch e yield the same distribution. Hence, such a change cannot be detected by A.

Lemma 12 (Game 1 to Game 2). *For any PPT adversary A there is a distinguisher D on LS such that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},D}(\lambda, \lambda).$$

Proof. The input is provided as $(pp, \mathbf{T})$, where $\mathbf{T} = (T_0, \ldots, T_\lambda)$ is either $\mathbf{g}$ or $\mathbf{g}\widehat{\mathbf{g}}$, for $\mathbf{g} = (g_0, \ldots, g_\lambda) \leftarrow \mathsf{SampG}(pp)$ and $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_\lambda), \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$. The Chall oracle in Game 2 is as follows:

$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{\begin{aligned} C^*_{e,0} = (\{(ST_0 g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} T_i^{\mathbf{u}_i} g_{\mathbf{u},i}^{\mathbf{u}_i}, T_{|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_\lambda g_{\mathbf{u},\lambda}) \\ : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s \cdot M) \end{aligned}}$$

$$\boxed{\begin{aligned} C^*_{e,1} = (\{(ST_0 g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} T_i^{\mathbf{u}_i} g_{\mathbf{u},i}^{\mathbf{u}_i}, T_{|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_\lambda g_{\mathbf{u},\lambda}) \\ : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s \cdot M') \end{aligned}}$$

for $\boxed{(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda})_{\mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(e)}\setminus\Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)}$, for $\boxed{S \leftarrow \mathsf{SampS}(pp; s)}$ with $\boxed{s \leftarrow \mathbb{Z}_N^*}$, and for $M' \leftarrow \mathsf{Dec}(K_{e-1}, C_{e-1,e_{\mathsf{exp}}})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,b})$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,b}$.

If $\mathbf{T} = \mathbf{g}$, then the challenge ciphertext(s) are distributed as in Game 1. If $\mathbf{T} = \mathbf{g}\widehat{\mathbf{g}}$, then the challenge ciphertext(s) are distributed as in Game 2.

Lemma 13 (Game 2 to Game 3.1.0). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,2}] - \Pr[S_{A,3.1.0}]| = 0.$$

Proof. This is a conceptional change. The Next'-oracle in Game 3.0.0 is as follows (where we alter K_{e+1} after calling Next):

Next' : run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{Next}(K_e)$. For $K'_{e+1} = (\{(T_{\mathbf{u},0}, T_{\mathbf{u},1}, T_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$ and $\Delta'_{e+1} = (D_0, D_1, D_2)$, sample $(h_0, \ldots, h_\lambda), (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda})_{\mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and compute

$$\boxed{\Delta_{e+1} = (D_0 h_0, D_1 \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_2),}$$

$$\boxed{K'_{\mathbf{u}} = (T_{\mathbf{u},0} h_{\mathbf{u},0}, T_{\mathbf{u},1} \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},\lambda} h_{\mathbf{u},\lambda}),}$$ for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$. Set

$$\boxed{K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)}.$$

If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e + 1$.

See that we can use the above boxed $\mathbb{H}$-elements to perfectly re-randomize Δ'_{e+1} and K'_{e+1} which is due to the $\mathbb{G}$-$\mathbb{H}$-subgroups property which ensures that the above token Δ_{e+1} and key K_{e+1} have the same distribution as the token and key output by Next, respectively.

Lemma 14 (Game 3.i.0 to Game 3.i.1). *For any PPT adversary A there is a distinguisher D on RS such that*

$$|\Pr[S_{A,3.i.0}] - \Pr[S_{A,3.i.1}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda,\lambda)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1]$.

Proof. The input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T} = (T_0,\ldots,T_\lambda)$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for $\mathbf{h} = (h_0,\ldots,h_\lambda) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0,\ldots,\widehat{h}_\lambda),\cdot,\cdot) \leftarrow \widehat{\mathsf{SampH}}(pp,sp)$, and $\mathbf{g}\widehat{\mathbf{g}} = (g_0\widehat{g}_0,\ldots,g_\lambda\widehat{g}_\lambda)$. We guess the challenge, expiry, and "window" epochs $\hat{e}^*, \hat{e}^*_{\mathsf{exp}}, \hat{e}' \leftarrow [\lfloor \mathsf{poly}(\lambda)\rfloor]$ and abort if $e^* \neq \hat{e}^*$ or $e^*_{\mathsf{exp}} \neq \hat{e}^*_{\mathsf{exp}}$ or $\hat{e}^* < \hat{e}' \leq \hat{e}^*_{\mathsf{exp}}$. The Next' and Chall oracles are as follows:

Next' : run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{Next}(K_e)$. For $K'_{e+1} = (\{(T_{\mathbf{u},j,0}, T_{\mathbf{u},j,1}, T_{\mathbf{u},j,|\mathbf{u}|+1},\ldots,T_{\mathbf{u},j,\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}, j \in [|\Gamma_{\mathbf{t}(e+1)|}]], m(K_e), pp)$ (we can assume a natural order in K'_{e+1} indexed by j) and $\Delta'_{e+1} = (D_0, D_1, D_2)$, sample $(h_0,\ldots,h_\lambda), (h_{\mathbf{u},0},\ldots,h_{\mathbf{u},\lambda})_{\mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and compute

$$\Delta_{e+1} = \begin{cases} (D_0 \boxed{T_0}, D_1 \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} \boxed{T_i^{\mathbf{t}(e+1)_i}}, D_2) & (i=0 \wedge e = \hat{e}^*-1) \\ (D_0 h_0, D_1 \cdot (\widehat{h})^\alpha \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_2) & (i \geq 1 \wedge e = \hat{e}^*-1) \\ (D_0 \boxed{T_0}, D_1 \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} \boxed{T_i^{\mathbf{t}(e+1)_i}}, D_2) & \text{otherwise,} \end{cases}$$

$$K'_{\mathbf{u}} = \begin{cases} (T_{\mathbf{u},j,0}h_{\mathbf{u},0}, T_{\mathbf{u},j,1}\cdot(\widehat{h})^\alpha \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},j,\lambda}h_{\mathbf{u},\lambda}) & (e = \hat{e}'-1) \\ (T_{\mathbf{u},j,0}h_{\mathbf{u},0}, T_{\mathbf{u},j,1}\cdot(\widehat{h})^\alpha \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},j,\lambda}h_{\mathbf{u},\lambda}) & (j < i \wedge e = \hat{e}^*_{\mathsf{exp}}) \\ (T_{\mathbf{u},j,0}\boxed{T_0}, T_{\mathbf{u},j,1}\cdot\prod_{i=1}^{|\mathbf{u}|}\boxed{T_i^{\mathbf{u}_i}}, T_{\mathbf{u},j,|\mathbf{u}|+1}\boxed{T_{|\mathbf{u}|+1}},\ldots,T_{\mathbf{u},j,\lambda}\boxed{T_\lambda}) & (j = i \wedge e = \hat{e}^*_{\mathsf{exp}}) \\ (T_{\mathbf{u},j,0}h_{\mathbf{u},0}, T_{\mathbf{u},j,1}\cdot\prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},j,\lambda}h_{\mathbf{u},\lambda}) & (j > i \wedge e = \hat{e}^*_{\mathsf{exp}}) \\ (T_{\mathbf{u},j,0}h_{\mathbf{u},0}, T_{\mathbf{u},j,1}\cdot\prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},j,\lambda}h_{\mathbf{u},\lambda}) & \text{otherwise,} \end{cases}$$

for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$ and $\boxed{\alpha \leftarrow \mathbb{Z}_N}$. Set $K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$. If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e+1$.

$\mathsf{Chall}(M, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, \hat{e}^*-1, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$C^*_{\hat{e}^*,0} = (\{(S \boxed{g_0\widehat{g}_0} g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}} g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}} g_{\mathbf{u},|\mathbf{u}|+1},\ldots,\boxed{g_\lambda\widehat{g}_\lambda} g_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1,\ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, m(k_{\hat{e}^*})^s \cdot M)$$

$$C^*_{\hat{e}^*,1} = (\{(S \boxed{g_0\widehat{g}_0} g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}} g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}} g_{\mathbf{u},|\mathbf{u}|+1},\ldots,\boxed{g_\lambda\widehat{g}_\lambda} g_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1,\ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, m(k_{\hat{e}^*})^s \cdot M'),$$

for $(g_{\mathbf{u},0},\ldots,g_{\mathbf{u},\lambda})_{\mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(\hat{e}^*)}\setminus\Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, for $S \leftarrow \mathsf{SampS}(pp;s)$ with for $s \leftarrow \mathbb{Z}^*_N$, and for $M' \leftarrow \mathsf{Dec}(K_{e-1}, C_{e-1,e_{\mathsf{exp}}})$, . Set $\mathcal{C}^* = \mathcal{C}^* \cup (\hat{e}^*, C^*_{\hat{e}^*,b})$, $e^* = \hat{e}^*$, $e^*_{\mathsf{exp}} = \hat{e}^*_{\mathsf{exp}}$, and return $C^*_{\hat{e}^*,b}$.

See that by validity condition 2), A is not allowed to query a token $\Delta_{e'}$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ when it has queried a key $K_{e'}$. Hence, we can safely embed an $\widehat{h}$-element in $K_{e'}$ since no information on $\widehat{h}$ is given out in $\Delta_{e'}$ due to orthogonality (i.e., $m(\widehat{h}) = 1$). Moreover, if $\mathbf{T} = \mathbf{h}$, then the token Δ_{e^*} and the key $K_{e^*_{\mathsf{exp}}+1}$ are distributed as in Game 3.i.0. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the token Δ_{e^*} and the key $K_{e^*_{\mathsf{exp}}+1}$ are distributed as in Game 3.i.1. This reduction looses a polynomial factor due to guessing.

Lemma 15 (Game 3.i.1 to Game 3.i.2). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,3.i.1}] - \Pr[S_{A,3.i.2}]| = 0,$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1]$.

Proof. This step is exactly the same as in the proof of Theorem 1; however, we have to do this for all i to ensure that all key elements and the token are semi-functional, i.e., for all key elements in the set $\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}$ and for the token Δ_{e^*}, we can information-theoretically embed $(\widehat{h})^\alpha$ step-by-step for each i.

Lemma 16 (Game 3.i.2 to Game 3.i.3). *For any PPT adversary A there is a distinguisher D on RS such that*

$$|\Pr[S_{A,3.i.4}] - \Pr[S_{A,3.i.5}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda,\lambda)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1]$.

Proof. In this step, we undo the game hop from Lemma 14. As a result, we now have a uniform element $(\widehat{h})^\alpha$ embedded in the i-th key and in the token. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the token Δ_{e^*} and the key $K_{e^*_{\mathsf{exp}}+1}$ are distributed as in Game 3.i.2. If $\mathbf{T} = \mathbf{h}$, then the token Δ_{e^*} and the key $K_{e^*_{\mathsf{exp}}+1}$ are distributed as in Game 3.i.3 where up to the i-th key element of $K_{e^*_{\mathsf{exp}}+1}$, all elements are semi-functional. This reduction looses a polynomial factor.

Lemma 17 (Game 3.$|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1$.3 to Game 4). *For any PPT adversary A there is a distinguisher D on ND such that*

$$|\Pr\left[S_{A,3.|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1.3}\right] - \Pr[S_{A,4}]| \leq \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D}(\lambda,n)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+1]$.

Proof. The input is provided as $(pp, \mathbf{S}\mathbf{g}\widehat{\mathbf{g}}, K\cdot(\widehat{h})^\alpha, \mathbf{T})$, where $\mathbf{T}$ is either $e(S,K)$ or $R \leftarrow G_T$, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}},\cdot) \leftarrow \widehat{\mathsf{SampG}}(pp,sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $S \leftarrow \mathsf{SampS}(pp)$, $\mathbf{S} = (S,1,0,\ldots)$, $K \leftarrow \mathsf{SampK}(pp)$. We guess the challenge, expiry, and "window" epochs $\hat{e}^*, \hat{e}^*_{\mathsf{exp}}, \hat{e}' \leftarrow [\lfloor \mathsf{poly}(\lambda)\rfloor]$ and abort if $e^* \neq \hat{e}^*$ or $e^*_{\mathsf{exp}} \neq \hat{e}^*_{\mathsf{exp}}$ or $\hat{e}^* < \hat{e}' \leq \hat{e}^*_{\mathsf{exp}}$. The Next' and Chall oracles are as follows:

Next' : run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{Next}(K_e)$. (We additionally assume that δ which is sampled in Next is available in Next' as well as k_1 sampled in Gen in the beginning of the experiment.) For $K'_{e+1} = (\{(T_{\mathbf{u},0}, T_{\mathbf{u},1}, T_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$, and $\Delta'_{e+1} = (D_0, D_1, D_2)$, store $\delta_{e+1} = \delta$, $k_{e+1} = \delta \cdot k_e$ (for $e = \hat{e}^* - 1$, we set $K \cdot k^{-1}_{\hat{e}^*-1}$ as "delta" and K as key in epoch e^*, implicitly), sample $(h_0,\ldots,h_\lambda), (h_{\mathbf{u},0},\ldots,h_{\mathbf{u},\lambda})_{\mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and compute

$$\Delta_{e+1} = \begin{cases} (h_0, \boxed{K\cdot(\widehat{h})^\alpha \cdot k_e^{-1}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, \boxed{m(k_e \cdot K \cdot (\widehat{h})^\alpha \cdot k_e^{-1})}) & (e = \hat{e}^* - 1) \\ (D_0 h_0, D_1 \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_2) & \text{otherwise} \end{cases}$$

and $K'_{\mathbf{u}} =$

$$\begin{cases} (T_{\mathbf{u},0}h_{\mathbf{u},0}, T_{\mathbf{u},1}\cdot \boxed{K\cdot(\widehat{h})^\alpha \cdot \delta_{\hat{e}^*}^{-1}\cdot k_{\hat{e}^*-1}^{-1}} \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},\lambda}h_{\mathbf{u},\lambda}) & (e = \hat{e}' - 1) \\ (T_{\mathbf{u},j,0}h_{\mathbf{u},0}, T_{\mathbf{u},j,1}\cdot\prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j,\lambda}h_{\mathbf{u},\lambda}) & \text{otherwise} \end{cases}$$

for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$ and $\alpha \leftarrow \mathbb{Z}_N$. Set $K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$. If $\mathsf{phase} = 1$ and $e < e^*_{\mathsf{exp}}$, run $C^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, C^*_{e,b})$. Set $e = e+1$.

$\mathsf{Chall}(M, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, \hat{e}^*-1, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$C^*_{\hat{e}^*,0} = (\{(\boxed{Sg_0\widehat{g}_0}\, g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}}\, g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}}\, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, \boxed{g_\lambda\widehat{g}_\lambda}\, g_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1,\ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, \mathbf{T}\cdot M)$$

$$C^*_{\hat{e}^*,1} = (\{(\boxed{Sg_0\widehat{g}_0}\, g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}}\, g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}}\, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, \boxed{g_\lambda\widehat{g}_\lambda}\, g_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1,\ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, \mathbf{T}\cdot M'),$$

for $(g_{\mathbf{u},0},\ldots,g_{\mathbf{u},\lambda})_{\mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(\hat{e}^*)}\setminus\Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, for $S \leftarrow \mathsf{SampS}(pp;s)$ with for $s \leftarrow \mathbb{Z}_N^*$, and for $M' \leftarrow \mathsf{Dec}(K_{\hat{e}^*-1}, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}})$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (\hat{e}^*, C^*_{\hat{e}^*,b})$, $e^* = \hat{e}^*$, $e^*_{\mathsf{exp}} = \hat{e}^*_{\mathsf{exp}}$, and return $C^*_{\hat{e}^*,b}$.

See that $m(K \cdot (\widehat{h})^{\alpha}) = m(K)$ holds. Hence, no information on $(\widehat{h})^{\alpha}$ is given out via m in Δ_{e^*}. Moreover, if the adversary queries $K_{e'}$ (by validity, it is not allowed to have queried $\Delta_{e'-1}$), then $(\widehat{h})^{\alpha}$ hides $K_{e'}$. As a consequence, all following keys including $K_{e^*_{\mathsf{exp}}+1}$ also have the blinding term $(\widehat{h})^{\alpha}$. Now, if $\mathbf{T} = e(S, K)$, then the challenge ciphertext(s) are distributed as in Game $3.|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + 1.3$. If $\mathbf{T} = R$, then the challenge ciphertext(s) are distributed as in Game 4.

Lemma 18 (Game 4). *For any PPT adversary A,* $\Pr[S_{A,4}] = 1/2$ *holds.*

Proof. In Game 4, for (uniform) $b \in \{0, 1\}$, we provide A with challenge ciphertext(s) that include a uniform G_T-element instead of a A-chosen b-dependent message. Hence, b is completely hidden from A's view.

Taking Lemmata 11, 12, 13, 14, 15, 16, 17, and 18 together, shows Theorem 2. □

4 Extension: UE from a Puncturable Encryption Perspective

We introduce a novel primitive dubbed Ciphertext Puncturable Encryption (CPE), provide a CPE security model, and show how to construct UE with expiry epochs from CPE. Finally, we give an instantiation of CPE under standard assumptions. CPE essentially views UE from the perspective of puncturable encryption [GM15].

4.1 Ciphertext Puncturable Encryption

Recall that the distinguishing feature of CPE is that, besides sequence tags in keys and ciphertexts, it uses tags for ciphertexts and tokens can be associated to tags or not. Only ciphertexts with tags in tokens can excluded from being punctured. However, there is one exception, a token can be constructed to be working for all ciphertexts denoted by symbol $\forall$ in the token. Moreover, because of tags, CPE is stronger than our UE definition as it allows the adversary to even query more tokens (since those can be crafted in a more fine-grained way via tags now). Noteworthy, keys are agnostic of tags and, hence, the restriction of querying keys are the same in UE and CPE. As a consequence, CPE yields an even more fine-grained primitive compared to what our UE definition offers and might be of independent interest.

Definition 7. *A Ciphertext Puncturable Encryption (CPE) scheme* CPE *for sequences* $(1, \ldots, \mathsf{poly}(\lambda))$[10]*, ciphertext-tag space* $\mathcal{T}$*, and message space* $\mathcal{M}$ *consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{ExPunc}, \mathsf{Dec})$*:*

$\mathsf{Gen}(\lambda)$: on input security parameter λ, outputs initial key K_1.
$\mathsf{KPunc}(K_e, \mathcal{S})$: on input key K_e for sequence element e and ciphertext tags $\mathcal{S} \subseteq \mathcal{T}$ or $\mathcal{S} = \forall$, if $\mathcal{S} \subseteq \mathcal{T}$, outputs a punctured key K_{e+1} and tag-specific tokens $(\Delta_{e+1,t})_{t \in \mathcal{S}}$; otherwise, if $\mathcal{S} = \forall$, outputs a punctured key K_{e+1} and (universal) token $\Delta_{e+1,\forall}$.
$\mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})$: on input key K_e, ciphertext tag $t \in \mathcal{T}$, message $M \in \mathcal{M}$, and expiry sequence element e_{exp}, outputs a ciphertext $C_{e,t,e_{\mathsf{exp}}}$ or $\perp$.
$\mathsf{ExPunc}(\Delta_{e+1,t'}, C_{e,t,e_{\mathsf{exp}}})$: on input token $\Delta_{e+1,t'}$ and ciphertext $C_{e,t,e_{\mathsf{exp}}}$, outputs a ciphertext $C_{e+1,t,e_{\mathsf{exp}}}$ if $t' \in \{t, \forall\}$ and $e < e_{\mathsf{exp}}$; otherwise outputs $\perp$.
$\mathsf{Dec}(K_e, C_{e',t,e_{\mathsf{exp}}})$: on input key K_e and ciphertext $C_{e,t,e_{\mathsf{exp}}}$, outputs message $M \in \mathcal{M}$ if $e = e'$; otherwise outputs $\perp$.

Correctness. For all $\lambda \in \mathbb{N}$, for $e \in [\lfloor \mathsf{poly}(\lambda) \rfloor]$, for $K_1 \leftarrow \mathsf{Gen}(\lambda)$, for all $i \in \{1, \ldots, e\}$, for any $\mathcal{S} \in \mathcal{T} \cup \{\forall\}$, for all $(K_{i+1}, \Delta_{i+1,\mathcal{S}}) \leftarrow \mathsf{KPunc}(K_i, \mathcal{S})$, for all $M \in \mathcal{M}$, for all expiry sequence elements $e_{\mathsf{exp}} \in \mathbb{N}$, for all $t \in \mathcal{T}$, for all $j \in \{1, \ldots, e+1\}$, for all $C_{j,t,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_j, t, M, e_{\mathsf{exp}})$, we require that $M = \mathsf{Dec}(K_e, C_{e,t,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq e$, also for $C_{j'+1,t,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(\Delta_{j'+1,t'}, C_{j',t,e_{\mathsf{exp}}})$ with $t' \in \{t, \forall\}$ and $j' \in \{j+1, \ldots, e\}$.

Security notion. We define $\mathsf{IND\text{-}CPE\text{-}CPA}$ which guarantees that freshly generated ciphertexts cannot be distinguished from ones that are exclude from puncturing similarly to UE, but we give more power to the adversary as it is allowed to even query more tokens. This is particularly the case in the epoch e' where the adversary queried a key $K_{e'}$ and the challenge ciphertext is not expired, i.e., $e' \leq e_{\mathsf{exp}}$. In

[10] We can use any arbitrarily ordered sequence, but stick to integers for simplicity.

our UE definition, the adversary is not allowed to query a token $\Delta_{e'}$ while in CPE, we allow querying tokens $\Delta_{e',\mathcal{S}}$ that do not incorporate update capabilities for the challenge tag t^*, i.e., $t^* \notin \mathcal{S}$ or $\mathcal{S} \neq \forall$. This introduces more subtleties not only in the construction but also in the security proof. Due to the versatile features of our DSG abstraction (which allows also querying of tags adaptively), we are able to formally prove security.

Definition 8 (IND-CPE-CPA security). *A CPE scheme* CPE *is* IND-CPE-CPA*-secure iff for any PPT adversary A, the advantage function*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda) = 1\right] - 1/2 \right|$$

is negligible in λ, where $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}$ is defined as in Fig. 9.

Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 0, \mathcal{S} = \emptyset, c = 0, \Delta_{1,\forall} = \perp$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{KPunc}',\mathsf{ExPunc}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{CPuncC}^*}(\lambda)$
if $b = b'$ and A is valid, then return 1 else return 0

Oracles
$\mathsf{Enc}'(t, M, e_{\mathsf{exp}})$: run $C_{e,t,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_{e,t,e_{\mathsf{exp}}}), c = c + 1$. Return $C_{e,t,e_{\mathsf{exp}}}$.
$\mathsf{KPunc}'(\mathcal{S}')$: run $(K_{e+1}, (\Delta_{e+1,t})_{t\in\mathcal{S}'}) \leftarrow \mathsf{KPunc}(K_e, \mathcal{S}')$. If $\mathsf{phase} = 1, e < e^*_{\mathsf{exp}}$, and $t^* \in \mathcal{S}'$ or $\mathcal{S}' = \forall$, run $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,t^*}, C^*_{e,t^*,b})$ (if $t^* \in \mathcal{S}'$) or $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,\forall}, C^*_{e,t^*,b})$ (if $\mathcal{S}' = \forall$). Set $e = e + 1$ and $\mathcal{S} = \mathcal{S}'$.
$\mathsf{ExPunc}'(C_{e-1,t,e_{\mathsf{exp}}})$: if $(\cdot, e-1, C_{e-1,t,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, or if $t \notin \mathcal{S}$ and $\mathcal{S} \neq \forall$, return $\perp$. Run $C_{e,t,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(\Delta_{e,t}, C_{e-1,t,e_{\mathsf{exp}}})$ (if $t \in \mathcal{S}$) or $C_{e,t,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(\Delta_{e,\forall}, C_{e-1,t,e_{\mathsf{exp}}})$ (if $\mathcal{S} = \forall$) and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, C_{e,t,e_{\mathsf{exp}}}), c = c + 1$. Return $C_{e,t,e_{\mathsf{exp}}}$.
$\mathsf{Corrupt}(\mathsf{inp}, e')$: if $e' > e$, return $\perp$. If $\mathsf{inp} = \mathsf{key}$, set $\mathcal{K}^* = \mathcal{K}^* \cup \{e'\}$ and return $K_{e'}$. If $\mathsf{inp} = \mathsf{token}$, set $\mathcal{D}^* = \mathcal{D}^* \cup \{e', \mathcal{S}\}$ and return $(\Delta_{e',t})_{t\in\mathcal{S}}$.
$\mathsf{Chall}(M, C_{e-1,t,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, or if $t \notin \mathcal{S}$ and $\mathcal{S} \neq \forall$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,t,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. If $b = 0$, set $C^*_{e,t,0} \leftarrow \mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})$, else $C^*_{e,t,1} \leftarrow \mathsf{ExPunc}(\Delta_{e,t}, C_{e-1,t,e_{\mathsf{exp}}})$ (if $t \in \mathcal{S}$) or $C^*_{e,t,1} \leftarrow \mathsf{ExPunc}(\Delta_{e,\forall}, C_{e-1,t,e_{\mathsf{exp}}})$ (if $\mathcal{S} = \forall$). Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,t,b})$, $e^* = e$, $t^* = t$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,t,b}$.
CPuncC^* : If $\mathsf{phase} = 0$ or $e > e^*_{\mathsf{exp}}$, return $\perp$. Set $\mathcal{C}^* := \mathcal{C}^* \cup (e, C^*_{e,t,b})$ and return $C^*_{e,t,b}$.

A **is valid iff**:
1) For all $e' \in \mathcal{K}^*$, $(e', C^*_{e',b}) \notin \mathcal{C}^*$ holds. (No trivial win via retrieved keys.)
2) For all $e' \in \mathcal{K}^*$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ and $(e'-1, C^*_{e'-1,t^*,b}) \in \mathcal{C}^*$, $(e'-1, \mathcal{S}') \notin \mathcal{D}^*$ for $\mathcal{S}' = \forall$ or $t^* \in \mathcal{S}'$ holds. (No trivial win via updating a retrieved challenge ciphertext via a token.)

Fig. 9. Our IND-CPE-CPA security notion for CPE.

4.2 Generic UE Construction from CPE

We can see that UE is essentially an epoch-based version of CPE (i.e., where sequence elements correspond to epochs) and we set $\mathcal{S} = \forall$ as input to key puncturing KPunc. Moreover, we fix the ciphertext-tag set of CPE to a singleton $\mathcal{T} = \{t\}$ for any arbitrary tag t. Particularly, we use key puncturing for generating the next UE key (i.e., "puncturing" on the CPE sequence element e) and use exclude puncturing for ciphertexts to update ciphertext to the next epoch (where ciphertext with expired epochs are punctured). Encryption and decryption directly map to UE's encryption and decryption functionality, respectively. Let $\mathsf{CPE} = (\mathsf{CPE.Gen}, \mathsf{CPE.KPunc}, \mathsf{CPE.Enc}, \mathsf{CPE.ExPunc}, \mathsf{CPE.Dec})$ be a CPE with tag space $\mathcal{T} = \{t\}$ and message space $\mathcal{M}_{\mathsf{CPE}}$. We construct a UE scheme $\mathsf{UE} = (\mathsf{Gen}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$ with message space $\mathcal{M} := \mathcal{M}_{\mathsf{CPE}}$. The construction with correctness is given in Fig. 10. We prove security in Theorem 3.

Theorem 3. *If* CPE *is* IND-CPE-CPA *secure, then* UE *is* EE-IND-UE-CPA *secure. Concretely, for any PPT adversary A there is a distinguisher D in the* IND-CPE-CPA *security experiment, such that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},D}(\lambda) \geq \mathsf{Adv}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda).$$

$\mathsf{Gen}(\lambda)$: return $K_1 \leftarrow \mathsf{CPE.Gen}(\lambda)$. (Implicitly, we set $\mathcal{M} = \mathcal{M}_{\mathsf{CPE}}$.)
$\mathsf{Next}(K_e)$: return $(K_{e+1}, \Delta_{e+1}) gets \mathsf{CPE.KPunc}(K_e, \forall)$.
$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(K_e, t, e_{\mathsf{exp}}, M)$.
$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$: return $C_{e+1,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.ExPunc}(C_{e,e_{\mathsf{exp}}}, \Delta_{e+1})$.
$\mathsf{Dec}(K_e, C_e)$: return $M := \mathsf{CPE.Dec}(K_e, C_{e,e_{\mathsf{exp}}})$.

Correctness of CPE: See that this directly translates from the CPE scheme, i.e., the ciphertexts that were computed by Enc and/or updated via Update can be decrypted by Dec if the keys are consistent (i.e., for the same epoch) and the ciphertext is not expired.

Fig. 10. Construction of UE from CPE with correctness.

Proof. We show the theorem by constructing a PPT distinguisher D in the IND-CPE-CPA security experiment with CPE as defined in Figure 9 from any successful PPT adversary A in the EE-IND-UE-CPA security with UE as defined in Figure 3. D runs $A(\lambda)$. Let $\mathsf{CPE.Enc}', \mathsf{KPunc}, \mathsf{ExPunc}, \mathsf{CPE.Corrupt}, \mathsf{CPE.Chall}, \mathsf{CPE.CPuncC}^*$ be the CPE oracles. A's oracle queries are answered as follows:

$\mathsf{Enc}'(M, e_{\mathsf{exp}})$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}'(M, t, e_{\mathsf{exp}})$, for t the element in the singleton $\mathcal{T}$.

Next' : run $\mathsf{KPunc}(\forall)$.

$\mathsf{Update}'(C_{e-1,e_{\mathsf{exp}}})$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(C_{e-1,e_{\mathsf{exp}}})$.

$\mathsf{Corrupt}(\mathsf{inp}, e')$: return the result of $\mathsf{CPE.Corrupt}(\mathsf{inp}, e')$. (This is either a key or a token depending on inp.)

$\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})$: return $C^*_{e,b} \leftarrow \mathsf{CPE.Chall}(M, C_{e-1,e_{\mathsf{exp}}})$.

CPuncC^* : return $C^*_{e,b} \leftarrow \mathsf{CPE.CPuncC}^*$.

We conclude that D provides a consistent view for A. If A is a successful PPT adversary in the EE-IND-UE-CPA security experiment with UE (see that also the validity conditions of CPE subsumes the validity conditions of UE), then D is a successful PPT adversary in the IND-CPE-CPA security experiment with CPE. □

4.3 CPE from Standard Assumptions

We construct a CPE scheme from DSG (and, hence, from standard assumptions) along the way of the construction of UE with sublinear parameter sizes in Construction 6. The distinguishing feature to UE^* is that tokens can be associated to ciphertext tags (i.e., which ciphertexts should be excluded from puncturing) instead of universal token as in UE^* that work always for all ciphertext in the respective epoch.

The proof of security is slightly more involved than proving UE schemes with sublinear parameter sizes in Theorem 2 and augments such a proof in the sense that the adversary is now able to query even more tokens. That means that we cannot simply argue that a queried key $K_{e'}$ can be made independent of the challenge key K_{e^*}, for $e^* < e' \leq e^*_{\mathsf{exp}}$, since in CPE the adversary is allowed to query a token $\Delta_{e'}$ (where such a token now could be used to retrieve information on the challenge key). Fortunately, due to properties of the DSG, we can also embed uniform randomness in such a token (since such token must however not have any prefix elements of the challenge ciphertext; otherwise, validity would not be achieved as the challenge ciphertext could be trivially decrypted).

Definition 9 (CPE Construction). *The construction of a CPE scheme* CPE *from DSG* DSG *is shown in Fig. 11 and correctness in Fig. 12. (We box main changes to the* DSG*-to-*UE^* *construction with sublinear parameter sizes as both constructions rely on the same construction paradigm.) We prove security in 4.*

Construction of CPE from DSG

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(\boxed{\lambda+1})$, set $\boxed{\mathcal{T} = \mathbb{Z}_N^*}$, sample $(h_0, \ldots, h_{\lambda+1}) \leftarrow \mathsf{SampH}(pp)$, $k_1 \leftarrow \mathsf{SampK}(pp)$ and return $K_1 = (\{K_1'\}, m(k_1), pp)$, with $K_1' = (h_0, k_1 \cdot h_1, \ldots, h_\lambda)$.
Remark. We assume that tags in $\mathcal{T}$ are integers in $\mathbb{Z}_N^*$ for simplicity. Also see that we use $\lambda+1$ as input to SampP as this will give us one additional element for embedding the identity.

$\mathsf{KPunc}(K_e, \mathcal{S})$: for $K_e = (\{K_{\mathbf{u}}' : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(K_e), pp)$, if $|\mathbf{t}(e)| < \lambda - 1$, then find $K'_{\mathbf{t}(e)} = (T_0, T_1, T_{|\mathbf{t}(e)|+1}, \ldots, T_\lambda)$, compute

$$K'_{\mathbf{t}(e+1)} = (T_0, T_1 \cdot T_{|\mathbf{t}(e)|+1}, T_{|\mathbf{t}(e)|+2}, \ldots, T_\lambda),$$
$$K'_{\mathbf{t}(e+2)} = (T_0, T_1 \cdot T_{|\mathbf{t}(e)|+1}^2, T_{|\mathbf{t}(e)|+2}, \ldots, T_\lambda),$$

Sample $\delta \leftarrow \mathsf{SampK}(pp)$ and $(h_0, \ldots, h_{\lambda+1}), (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda+1})_{\mathbf{u} \in [|\Gamma_{\mathbf{t}(e+1)}|]} \leftarrow \mathsf{SampH}(pp)$, for $\mathbf{t}(e+1) = (\mathbf{t}(e+1)_1, \ldots, \mathbf{t}(e+1)_{|\mathbf{t}(e+1)|})$, compute

$$\Delta_{e+1} = \begin{cases} (\Delta_{e+1,t})_{t \in \mathcal{S}} = (h_0, \delta \cdot \boxed{h_{\lambda+1}^t} \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, m(k_e \cdot \delta))_{t \in \mathcal{S}} & (\text{if } \mathcal{S} \neq \forall) \\ (h_0, \delta \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, m(k_e \cdot \delta)) & (\text{if } \mathcal{S} = \forall) \end{cases}$$

$$K_{\mathbf{u}}'' = (T'_{\mathbf{u},0} h_{\mathbf{u},0}, T'_{\mathbf{u},1} \cdot \delta \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T'_{\mathbf{u},|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T'_{\mathbf{u},\lambda} h_{\mathbf{u},\lambda}),$$

for $\{K_{\mathbf{u}}' = (T'_{\mathbf{u},0}, \ldots, T'_{\mathbf{u},\lambda}) : \mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e+1)}\}$. Set $K_{e+1} = (\{K_{\mathbf{u}}'' : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_e \cdot \delta), pp)$ and return (Δ_{e+1}, K_{e+1}).

$\mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})$: for $K_e = (\ldots, m(K_e), pp)$, sample $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$, and return $C_{e,t,e_{\mathsf{exp}}} =$

$$(\{(Sg_{\mathbf{u},0} \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{u_i}, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, g_{\mathbf{u},\lambda}, \boxed{g_{\mathbf{u},\lambda+1}^t}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s \cdot M).$$

$\mathsf{ExPunc}(\Delta_{e+1,t'}, C_{e,t,e_{\mathsf{exp}}})$: if $e < e_{\mathsf{exp}}$ or $\boxed{t' \notin \{t, \forall\}}$, return $\perp$. For $\Delta_{e+1,t} = (D_0, D_1, m(k_{e+1}))$ and $C_{e,t,e_{\mathsf{exp}}} = (\{C_{\mathbf{u}}' : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(k_e)^s \cdot M)$, if $|\mathbf{t}(e)| < \lambda - 1$, then find $C'_{\mathbf{t}(e)} = (S_0, S_1, S_{|\mathbf{t}(e)|+1}, \ldots, S_\lambda)$, compute

$$C'_{\mathbf{t}(e+1)} = (S_0, S_1 \cdot S_{|\mathbf{t}(e)|+1}, S_{|\mathbf{t}(e)|+2}, \ldots, \boxed{S_{\lambda+1}}),$$
$$C'_{\mathbf{t}(e+2)} = (S_0, S_1 \cdot S_{|\mathbf{t}(e)|+1}^2, S_{|\mathbf{t}(e)|+2}, \ldots, \boxed{S_{\lambda+1}}).$$

Sample $S \leftarrow \mathsf{SampS}(pp; s')$, for $s' \leftarrow \mathbb{Z}_N^*$, $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampG}(pp)$, compute $m(\delta)^s = \frac{e(S_0, D_1)}{e(D_0, S_1 \boxed{S_{\lambda+1}})}$ (where $S_{\lambda+1}$ is only present if $t' \neq \forall$) and

$$C_{\mathbf{u}}'' = (S_{\mathbf{u},0} S g_{\mathbf{u},0}, S_{\mathbf{u},1} \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{u_i}, S_{\mathbf{u},|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, S_{\mathbf{u},\lambda} g_{\mathbf{u},\lambda}, \boxed{S_{\mathbf{u},\lambda+1} g_{\mathbf{u},\lambda+1}^t}),$$

for $\{C_{\mathbf{u}}' = (S_{\mathbf{u},0}, \ldots, S_{\mathbf{u},\lambda}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}$. Return $C_{e+1,t,e_{\mathsf{exp}}} = (\{C_{\mathbf{u}}'' : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_e \cdot \delta)^{s+s'} \cdot M)$.

$\mathsf{Dec}(K_e, C_{e,t,e_{\mathsf{exp}}})$: for $K_e = (\{K_{\mathbf{u}}' : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(K_e), pp)$ and $C_{e,t,e_{\mathsf{exp}}} = (\{C_{\mathbf{u}}' : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, S_T)$, find $K'_{\mathbf{t}(e)} = (T_0, T_1, \ldots)$ and $C'_{\mathbf{t}(e)} = (S_0, S_1, \ldots)$, and return

$$M = S_T \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)}.$$

Fig. 11. Construction of CPE from DSG.

1) Correct decryption ciphertexts:

$$M = S_T \cdot \frac{e(T_0, S_1)}{e(S_0, T_1)} = m(k_e)^s \cdot M \cdot \frac{e(h_0, \prod_{i=1}^e g_i)}{e(Sg_0, k_e \prod_{i=1}^e h_i)} = \frac{m(k_e)^s}{e(S, k_e)} \cdot M = M,$$

where $m(k_e)^s = e(S, k_e)$, for some $s \in \mathbb{Z}_N^*$, $e(S, h_i) = 1$, for all $i \in [e]$, and $e(g_0, k_e) = 1$ due to projectiveness, orthogonality, and associativity.

2) Correctness for excluding ciphertexts $C_{e+1,t,e_{\mathsf{exp}}} = (\{C_{\mathbf{u}}'' : \mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e+1)}\}, m(k_{e+1})^{s+s'} \cdot M)$ from puncturing:

$$C_{\mathbf{u}}'' = (S_{\mathbf{u},0} S g_{\mathbf{u},0}, S_{\mathbf{u},1} \prod_{i=1}^{|\mathbf{u}|} g_{\mathbf{u},i}^{\mathbf{u}_i}, S_{\mathbf{u},|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, S_{\mathbf{u},\lambda} \boxed{g_{\mathbf{u},\lambda+1}^t}),$$

where $S \leftarrow \mathsf{SampS}(pp; s')$, for $s' \leftarrow \mathbb{Z}_N^*$, $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampG}(pp)$, $m(k_{e+1})^{s+s'} = m(k_e)^s \cdot m(\delta)^s \cdot m(k_{e+1})^{s'}$, for $m(\delta)^s = \frac{e(S_{\mathbf{t}(e+1),0}, D_1)}{e(D_0, S_{\mathbf{t}(e+1),1})}$, due to to projectiveness, orthogonality, associativity, and $\mathbb{G}$-$\mathbb{H}$-subgroups.

Fig. 12. Correctness of CPE.

Theorem 4. *If* DSG *is a DSG scheme, then* CPE *is* IND-CPE-CPA*-secure. Concretely, for any PPT adversary A, it holds:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda) \leq \mathsf{Adv}^{\mathsf{ls}}_{\mathsf{DSG},D_1}(\lambda,\lambda+1) + 2\cdot(|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q)\cdot \\ \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D_2}(\lambda,\lambda+1)/\mathsf{poly}(\lambda) + \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D_3}(\lambda,\lambda+1)/\mathsf{poly}(\lambda),$$

with q number of tag-based token queries.

Proof. We proceed similarly as in the proof of Theorem 2 for UE^* and only focus on the main differences here as both proofs follow the same proof paradigm. However, since more tokens can be given out in CPE compared to UE, the proof is more complex as those tokens might reveal information on the challenge keys which has to dealt with. Moreover, we map the symbol $\forall$ to the integer 0 for ease of proof exposition. Let $S_{A,j}$ be the event that A succeeds in Game j. We highlight changes boxed.

Lemma 19 (Game 0 to Game 1). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,0}] - \Pr[S_{A,1}]| = 0.$$

Proof. This is a conceptional change as in the proof of Lemma 11 where we change the behavior of the challenge oracle such that we use a fresh encryption instead of updating the ciphertext provided by A. Since the change is agnostic of the tag (and independent of the tokens), this can be carried out exactly as in Lemma 11.

Lemma 20 (Game 1 to Game 2). *For any PPT adversary A there is a distinguisher D on LS such that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \mathsf{Adv}^{\mathsf{ls}}_{\mathsf{DSG},D}(\lambda,\lambda+1).$$

Proof. The input is provided as $(pp,\mathbf{T})$, where $\mathbf{T} = (T_0,\ldots,T_{\lambda+1})$ is either $\mathbf{g}$ or $\mathbf{g}\widehat{\mathbf{g}}$, for $\mathbf{g} = (g_0,\ldots,g_{\lambda+1}) \leftarrow \mathsf{SampG}(pp)$ and $(\widehat{\mathbf{g}} = (\widehat{g}_0,\ldots,\widehat{g}_{\lambda+1}),\cdot) \leftarrow \widehat{\mathsf{SampG}}(pp,sp)$. The Chall oracle in Game 2 is as follows:

$\mathsf{Chall}(M, C_{e-1,t,e_{\mathsf{exp}}})$: if $\mathsf{phase}=1$, or if $t\notin\mathcal{S}$ and $\mathcal{S}\neq\forall$, return $\perp$. Set $\mathsf{phase}=1$. If $(\cdot,e-1,C_{e-1,t,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$\boxed{C^*_{e,t,0} = (\{(ST_0 g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} T_i^{\mathbf{u}_i} g_{\mathbf{u},i}^{\mathbf{u}_i}, T_{|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T^t_{\lambda+1} g^t_{\mathbf{u},\lambda+1}) : \mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(e)}\setminus\Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s\cdot M)}$$

$$\boxed{C^*_{e,t,1} = (\{(ST_0 g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} T_i^{\mathbf{u}_i} g_{\mathbf{u},i}^{\mathbf{u}_i}, T_{|\mathbf{u}|+1} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T^t_{\lambda+1} g^t_{\mathbf{u},\lambda+1}) : \mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(e)}\setminus\Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}\}, m(k_e)^s\cdot M'),}$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, for $(g_{\mathbf{u},0},\ldots,g_{\mathbf{u},\lambda+1})_{\mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(e)}\setminus\Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp;s)$, for $s\leftarrow\mathbb{Z}_N^*$. Set $\mathcal{C}^* = \mathcal{C}^*\cup(e,C^*_{e,t,b})$, $e^*=e$, $t^*=t$, $e^*_{\mathsf{exp}}=e_{\mathsf{exp}}$, and return $C^*_{e,t,b}$.

If $\mathbf{T}=\mathbf{g}$, then the challenge ciphertext(s) are distributed as in Game 1. If $\mathbf{T}=\mathbf{g}\widehat{\mathbf{g}}$, then the challenge ciphertext(s) are distributed as in Game 2.

Lemma 21 (Game 2 to Game 3.1.0). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,2}] - \Pr[S_{A,3.1.0}]| = 0.$$

Proof. This is a conceptional change. The Next'-oracle in Game 3.0.0 is as follows (where we alter K_{e+1} after calling Next):

$\mathsf{KPunc}'(\mathcal{S}')$: run $(K'_{e+1},\Delta'_{e+1}) \leftarrow \mathsf{KPunc}(K_e,\mathcal{S}')$. For $K'_{e+1} = (\{(T_{\mathbf{u},0},T_{\mathbf{u},1},T_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},\lambda}) : \mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$ and $\Delta'_{e+1} = (D_{t,0},D_{t,1},D_{t,2})_{t\in\mathcal{S}'\cup\{\forall\}}$, sample $(h_{t,0},\ldots,h_{t,\lambda+1})_{t\in\mathcal{S}'\cup\{\forall\}}, (h_{\mathbf{u},0},\ldots,h_{\mathbf{u},\lambda+1})_{\mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and compute

$$\Delta_{e+1} = (D_{t,0}h_{t,0}, D_{t,1}\cdot\boxed{h^t_{t,\lambda+1}}\cdot\prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t,i}^{\mathbf{t}(e+1)_i}, D_{t,2})_{t\in\mathcal{S}'},$$

$$K'_{\mathbf{u}} = (T_{\mathbf{u},0}h_{\mathbf{u},0}, T_{\mathbf{u},1}\prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},|\mathbf{u}|+1}h_{\mathbf{u},|\mathbf{u}|+1},\ldots,T_{\mathbf{u},\lambda}h_{\mathbf{u},\lambda}),$$

for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$. Set $K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$. If $\mathsf{phase} = 1, e < e^*_{\mathsf{exp}}$, and $t^* \in \mathcal{S}'$ or $\mathcal{S}' = \forall$, run $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,t^*}, C^*_{e,t^*,b})$ or $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,\forall}, C^*_{e,t^*,b})$, respectively. Set $e = e + 1$.

See that the essential difference to the proof of Lemma 13 is the construction of the tokens where there might be tag-associated elements embedded now which require an according re-randomization with such a tag. (See that we set $t = 0$ if $\mathcal{S}' = \forall$.)

Lemma 22 (Game 3.i.0 to Game 3.i.1). *For any PPT adversary A there is a distinguisher D on RS such that*

$$|\Pr[S_{A,3.i.0}] - \Pr[S_{A,3.i.1}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, \lambda+1)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q]$ and q the number of tag-based tokens queried in $e^* - 1$ and $\hat{e}'$.

Proof. The input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T} = (T_0, \ldots, T_{\lambda+1})$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for $\mathbf{h} = (h_0, \ldots, h_{\lambda+1}) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_{\lambda+1}), \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, and $\mathbf{g}\widehat{\mathbf{g}} = (g_0\widehat{g}_0, \ldots, g_{\lambda+1}\widehat{g}_{\lambda+1})$. We guess the challenge, expiry, and "window" sequence elements $\hat{e}^*, \hat{e}^*_{\mathsf{exp}}, \hat{e}' \leftarrow [\lfloor \mathsf{poly}(\lambda) \rfloor]$ and abort if $e^* \neq \hat{e}^*$ or $e^*_{\mathsf{exp}} \neq \hat{e}^*_{\mathsf{exp}}$, $\hat{e}' \leq \hat{e}^*$ or $\hat{e}' > \hat{e}^*_{\mathsf{exp}}$. The KPunc' and Chall oracles are as follows:

$\mathsf{KPunc}'(\mathcal{S}')$: run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{KPunc}(K_e, \mathcal{S}')$. For $K'_{e+1} = (\{(T_{\mathbf{u},j'',0}, T_{\mathbf{u},j'',1}, T_{\mathbf{u},j'',|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j'',\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}, j'' \in [|\Gamma_{\mathbf{t}(e+1)}|]\}, m(K_e), pp)$ (we can assume a natural order in K'_{e+1}) and $\Delta'_{e+1} = (D_{t,0}, D_{t,1}, D_{t,2})_{t \in \mathcal{S}' \cup \{\forall\}}$, sample $(h_{t,0}, \ldots, h_{t,\lambda+1})_{t \in \mathcal{S} \cup \{\forall\}}, (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and for the (j, j')-th query (j-th token with its j'-th tag-based part, $(j, j') \in [q'] \times [q_t]$ with $q = q' + q_t$) compute

$$\Delta'_{e+1} = \begin{cases} (D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot (\widehat{h})^{\alpha} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t_{j'},i}^{\mathbf{t}(e+1)_i}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & (j + j' < i + 1 \wedge (e = \hat{e}^* - 1 \vee e = \hat{e}' - 1)) \\ (D_{t_{j'},0} \boxed{T_0}, D_{t_{j'},1} \cdot \boxed{T_{\lambda+1}^{t_{j'}}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} \boxed{T_i^{\mathbf{t}(e+1)_i}}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & (j + j' = i + 1 \wedge (e = \hat{e}^* - 1 \vee e = \hat{e}' - 1)) \\ (D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & \text{otherwise,} \end{cases}$$

$$K'_{\mathbf{u}} = \begin{cases} (T_{\mathbf{u},j'',0} \boxed{T_0}, T_{\mathbf{u},j'',1} \cdot \prod_{i=1}^{|\mathbf{u}|} \boxed{T_i^{\mathbf{u}_i}}, T_{\mathbf{u},j'',|\mathbf{u}|+1} \boxed{T_{|\mathbf{u}|+1}}, \ldots, T_{\mathbf{u},j'',\lambda} \boxed{T_\lambda}) & (q + j'' = i \wedge e = e^*_{\mathsf{exp}}) \\ (T_{\mathbf{u},j,0} h_{\mathbf{u},0}, T_{\mathbf{u},j,1} \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j,\lambda} h_{\mathbf{u},\lambda}) & \text{otherwise,} \end{cases}$$

for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$, and $\alpha \leftarrow \mathbb{Z}_N$. Set $K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$. If $\mathsf{phase} = 1, e < e^*_{\mathsf{exp}}$, and $t^* \in \mathcal{S}'$ or $\mathcal{S}' = \forall$, run $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,t^*}, C^*_{e,t^*,b})$ or $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,\forall}, C^*_{e,t^*,b})$, respectively. Set $e = e + 1$.

$\mathsf{Chall}(M, C_{\hat{e}^*-1,t^*,\hat{e}^*_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, \hat{e}^* - 1, C_{\hat{e}^*-1,t^*,\hat{e}^*_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$C^*_{\hat{e}^*,t^*,0} = (\{(S \boxed{g_0\widehat{g}_0} g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}} g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots,$$

$$\boxed{(g_{\lambda+1}\widehat{g}_{\lambda+1})^{t^*}} g_{\mathbf{u},\lambda+1}^{t^*}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, m(k_{\hat{e}^*})^s \cdot M),$$

$$C^*_{\hat{e}^*,t^*,1} = (\{(S \boxed{g_0\widehat{g}_0} g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}} g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}} g_{\mathbf{u},|\mathbf{u}|+1}, \ldots,$$

$$\boxed{(g_{\lambda+1}\widehat{g}_{\lambda+1})^{t^*}} g_{\mathbf{u},\lambda+1}^{t^*}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, m(k_{\hat{e}^*})^s \cdot M'),$$

for $M' \leftarrow \mathsf{Dec}(K_{e^*-1}, C_{e^*-1,e_{\mathsf{exp}}})$, for $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda+1})_{\mathbf{u}=(\mathbf{u}_1,\ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (\hat{e}^*, C^*_{\hat{e}^*,t^*,b})$, $e^* = \hat{e}^*$, $e^*_{\mathsf{exp}} = \hat{e}^*_{\mathsf{exp}}$, and return $C^*_{\hat{e}^*,t^*,b}$.

If $\mathbf{T} = \mathbf{h}$, then the token $\Delta_{e^*,t}$ with $t \in \{t^*, \forall\}$, the token $\Delta_{e',t}$ with $t \notin \{t^*, \forall\}$, and the key $K_{e'}$ (if $K_{e'}$ is queried) or the key $K_{e^*_{\mathsf{exp}}+1}$ (if $K_{e'}$ is not queried) are distributed as in Game 3.i.0. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then those are distributed as in Game 3.i.1. This reduction looses a polynomial factor.

Lemma 23 (Game 3.i.1 to Game 3.i.2). *For any PPT adversary A, it holds:*

$$|\Pr[S_{A,3.i.1}] - \Pr[S_{A,3.i.2}]| = 0,$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q]$ and a large-enough $e(\lambda)$ polynomial in λ and $q$the number of tag-based token queries.

Proof. This step is the same as in the proof of Theorem 2; we have to do this for all i, i.e., for all key elements in the set$\Gamma_{\mathbf{t}(e')}$ (if $K_{e'}$ is queried) or in $\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}$ (if $K_{e'}$ is not queried) and for all tag-based token queries q. As a result, we can information-theoretically embed $(\widehat{h})^\alpha$ step-by-step for each i. See that by validity condition 2), A is not allowed to query a token $\Delta_{e',t}$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ and $t = t^*$ or $t = \mathcal{S}$ when it has queried a key $K_{e'}$. Hence, we can use the same argument here to embed an $\widehat{h}$-element in $K_{e'}$ since no information on $\widehat{h}$ is given out in $\Delta_{e'}$ due to orthogonality (i.e., $m(\widehat{h}) = 1$). Moreover, since $K_{e^*_{\mathsf{exp}}+1}$ does not have any prefixes of the challenge ciphertext, we can safely embed $(\widehat{h})^\alpha$ (if $K_{e'}$ was not queried; otherwise, $(\widehat{h})^\alpha$ has already been embedded in such a key).

Lemma 24 (Game 3.i.2 to Game 3.i.3). *For any PPT adversary A there is a distinguisher D on RS such that*

$$|\Pr[S_{A,3.i.4}] - \Pr[S_{A,3.i.5}]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, \lambda+1)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q]$ and q number of tag-based token queries.

Proof. In this step, we undo the game hop from Lemma 22. As a result, we now have a uniform element $(\widehat{h})^\alpha$ embedded in key and in the token queries for $K_{e'}, K_{e^*_{\mathsf{exp}}+1}$ and $\Delta_{e^*}, \Delta_{e'}$. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the tokens $\Delta_{e^*}, \Delta_{e'}$ and the keys $K_{e'}, K_{e^*_{\mathsf{exp}}+1}$, are distributed as in Game 3.i.2. If $\mathbf{T} = \mathbf{h}$, then those are distributed as in Game 3.i.3. This reduction looses a polynomial factor.

Lemma 25 (Game 3.$|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q$.3 to Game 4). *For any PPT adversary A there is a distinguisher D on ND such that*

$$\left|\Pr\left[S_{A,3.|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}|+q.5}\right] - \Pr[S_{A,4}]\right| \leq \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D}(\lambda, \lambda+1)/\mathsf{poly}(\lambda),$$

for $i \in [|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + 1]$ and q number of tag-based token queries.

Proof. The input is provided as $(pp, \mathbf{S}\mathbf{g}\widehat{\mathbf{g}}, K \cdot (\widehat{h})^\alpha, \mathbf{T})$, where $\mathbf{T}$ is either $e(S,K)$ or $R \leftarrow G_T$, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $S \leftarrow \mathsf{SampS}(pp)$, $\mathbf{S} = (S, 1, 0, \ldots)$, $K \leftarrow \mathsf{SampK}(pp)$. We guess the challenge, expiry, and "window" sequence elements $\hat{e}^*, \hat{e}^*_{\mathsf{exp}}, \hat{e}' \leftarrow [\lfloor e(\lambda) \rfloor]$ and abort if $e^* \neq \hat{e}^*$ or $e^*_{\mathsf{exp}} \neq \hat{e}^*_{\mathsf{exp}}$ or $\hat{e}^* < \hat{e}' \leq \hat{e}^*_{\mathsf{exp}}$. The Next' and Chall oracles are as follows:

$\mathsf{KPunc}'(\mathcal{S}')$: run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{KPunc}(K_e, \mathcal{S}')$. (We additionally assume that δ which is sampled in KPunc is available in KPunc' as well as k_1 sampled in Gen in the beginning of the experiment.) For $K'_{e+1} = (\{(T_{\mathbf{u},j'',0}, T_{\mathbf{u},j'',1}, T_{\mathbf{u},j'',|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j'',\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}, j'' \in [|\Gamma_{\mathbf{t}(e+1)}|]\}, m(K_e), pp)$ (we can assume a natural order in K'_{e+1}) and $\Delta'_{e+1} = (D_{t,0}, D_{t,1}, D_{t,2})_{t \in \mathcal{S}' \cup \{\forall\}}$, store $\delta_{e+1} = \delta$, $k_{e+1} = \delta \cdot k_e$ (for $e = \hat{e}^* - 1$, we set $K \cdot k^{-1}_{\hat{e}^*-1}$ as "delta" and K as key in epoch e^*, implicitly), sample $(h_{t,0}, \ldots, h_{t,\lambda+1})_{t \in \mathcal{S} \cup \{\forall\}}, (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and for the (j,j')-th (j-th token with its j'-th tag-based part) query compute

$$\Delta'_{e+1} = \begin{cases} (D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \boxed{K \cdot (\widehat{h})^\alpha \cdot k_e^{-1} \cdot \delta_{e+1}^{-1}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t_{j'},i}^{\mathbf{t}(e+1)_i}, m(\boxed{K \cdot (\widehat{h})^\alpha}))_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & (e = \hat{e}^* - 1) \\ (D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \boxed{(K \cdot (\widehat{h})^\alpha \cdot \prod_{e^*}^{e'} \delta_i^{-1} \cdot k_{e'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t_{j'},i}^{\mathbf{t}(e+1)_i}, m(\boxed{K \cdot (\widehat{h})^\alpha \cdot \prod_{e^*}^{e'} \delta_i^{-1}}))_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & (e = \hat{e}' - 1) \\ (D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}} & \text{otherwise,} \end{cases}$$

$$K'_{\mathbf{u}} = \begin{cases} (T_{\mathbf{u},j'',0} h_{\mathbf{u},0}, T_{\mathbf{u},j'',1} \cdot \boxed{K \cdot (\widehat{h})^\alpha \cdot \prod_{i=e^*+1}^{e^*_{\mathsf{exp}}+1} \delta_i} \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j'',|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j'',\lambda} h_{\mathbf{u},\lambda}) & (e = e^*_{\mathsf{exp}}) \\ (T_{\mathbf{u},j,0} h_{\mathbf{u},0}, T_{\mathbf{u},j,1} \cdot \prod_{i=1}^{|\mathbf{u}|} h_{\mathbf{u},i}^{\mathbf{u}_i}, T_{\mathbf{u},j,|\mathbf{u}|+1} h_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j,\lambda} h_{\mathbf{u},\lambda}) & \text{otherwise,} \end{cases}$$

for all $\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}$, and $\alpha \leftarrow \mathbb{Z}_N$. Set $K_{e+1} = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}\}, m(K_e), pp)$. If $\mathsf{phase} = 1, e < e^*_{\mathsf{exp}}$, and $t^* \in \mathcal{S}'$ or $\mathcal{S}' = \forall$, run $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,t^*}, C^*_{e,t^*,b})$ or $C^*_{e+1,t^*,b} \leftarrow \mathsf{ExPunc}(\Delta_{e+1,\forall}, C^*_{e,t^*,b})$, respectively. Set $e = e + 1$.

$\mathsf{Chall}(M, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, \hat{e}^* - 1, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. Set

$$C^*_{\hat{e}^*,t^*,0} = (\{(\boxed{Sg_0\widehat{g}_0}\, g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}}\, g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}}\, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots,$$
$$(\boxed{g_\lambda\widehat{g}_{\lambda+1}}\, g_{\mathbf{u},\lambda+1})^{t^*}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, \mathbf{T} \cdot M)$$
$$C^*_{\hat{e}^*,t^*,1} = (\{(\boxed{Sg_0\widehat{g}_0}\, g_{\mathbf{u},0}, \prod_{i=1}^{|\mathbf{u}|} \boxed{(g_i\widehat{g}_i)^{\mathbf{u}_i}}\, g_{\mathbf{u},i}^{\mathbf{u}_i}, \boxed{g_{|\mathbf{u}|+1}\widehat{g}_{|\mathbf{u}|+1}}\, g_{\mathbf{u},|\mathbf{u}|+1}, \ldots,$$
$$(\boxed{g_{\lambda+1}\widehat{g}_{\lambda+1}}\, g_{\mathbf{u},\lambda+1})^{t^*}) : \mathbf{u} = (u_1, \ldots) \in \Gamma_{\mathbf{t}(\hat{e}^*)} \setminus \Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}\}, \mathbf{T} \cdot M'),$$

for $M' \leftarrow \mathsf{Dec}(K_{\hat{e}^*-1}, C_{\hat{e}^*-1,\hat{e}^*_{\mathsf{exp}}})$, for $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda+1})_{\mathbf{u}=(\mathbf{u}_1,\ldots)\in\Gamma_{\mathbf{t}(\hat{e}^*)}\setminus\Gamma_{\mathbf{t}(\hat{e}^*_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (\hat{e}^*, C^*_{\hat{e}^*,t^*,b})$, $e^* = \hat{e}^*$, $e^*_{\mathsf{exp}} = \hat{e}^*_{\mathsf{exp}}$, and return $C^*_{\hat{e}^*,t^*,b}$.

See that $m(K \cdot (\widehat{h})^\alpha) = m(K)$ holds. Hence, no information on $(\widehat{h})^\alpha$ is given out via m in Δ_{e^*}. Moreover, if the adversary queried $K_{e'}$ (by validity, it is not allowed to have queried $\Delta_{e'-1,t}$ with $t \in \{t^*, \forall\}$), then $(\widehat{h})^\alpha$ hides all key elements in $K_{e'}$. Otherwise, if the adversary did not query $K_{e'}$, then $(\widehat{h})^\alpha$ blinds the key elements in $K_{e^*_{\mathsf{exp}}}$. Now, if $\mathbf{T} = e(S, K)$, then the challenge ciphertext(s) are distributed as in Game $3.|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q.3$. If $\mathbf{T} = R$, then the challenge ciphertext(s) are distributed as in Game 4.

Lemma 26 (Game 4). *For any PPT adversary A, $\Pr[S_{A,4}] = 1/2$ holds.*

Proof. In Game 4, for (uniform) $b \in \{0, 1\}$, we provide A with challenge ciphertext(s) that include a uniform G_T-element instead of a A-chosen b-dependent message. Hence, b is completely hidden from A's view.

Taking Lemmata 19, 20, 21, 22, 23, 24, 25, and 26 together, shows Theorem 4. □

Applications of CPE beyond UE. CPE provides an interesting abstraction for outsourced file storage with forward-security and fine-grained secure shredding of files. In a recent work, Backendal, Günther and Paterson [BGP22] introduced such a so-called protected file storage (PFS) setting and show how this can be instantiated via puncturable key wrappingIn (introduced in the same work). Loosely speaking, Backendal et al. achieve forward-security via key-rotation (but this requires to download, decrypt and re-encrypt of all file encryption keys) and the shredding of files is achieved via key-puncturing.

We observe that the concept of sequence tags in CPE (used as expiry epochs when instantiating UE from CPE) allows to implement the fine-grained forward-security aspect via efficient key-rotation (though in contrast to [BGP22] via help of the server). Moreover, the ciphertext-tag space in CPE provides an additional dimension for granularity which allows to implement a secure fine-grained shredding of files, i.e., via puncturing of the ciphertext (by excluding them from updates). We hope that CPE will find additional applications in this and beyond this context and leave a more detailed study to future work.

Acknowledgements. We thank the anonymous reviewers for valuable feedback. This project has received funding from the European Union's Horizon 2020 ECSEL Joint Undertaking project under grant agreement n°783119 (SECREDAS) and n°826610 (COMP4DRONES), and by the Austrian Science Fund (FWF) and netidee SCIENCE grant P31621-N38 (PROFET).

References

[AGJ21] Nimrod Aviram, Kai Gellert, and Tibor Jager. Session resumption protocols and efficient forward security for TLS 1.3 0-rtt. *J. Cryptol.*, 34(3):20, 2021.

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. A framework for identity-based encryption with almost tight security. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 521–549. Springer, Heidelberg, November / December 2015.

[Bar16] Elaine Barker. Recommendation for key management. NIST Special Publication 800-57 Part 1, Revision 4, 2016. http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4.

[BDdK+21] Colin Boyd, Gareth T. Davies, Bor de Kock, Kai Gellert, Tibor Jager, and Lise Millerjord. Symmetric key exchange with full forward security and robust synchronization. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021*, 2021.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 464–493. Springer, Heidelberg, August 2020.

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

[BGP22] Matilda Backendal, Felix Günther, and Kenneth G. Paterson. Puncturable key wrapping and its applications. Cryptology ePrint Archive, Paper 2022/1209, ASIACRYPT 2022 (to appear), 2022. https://eprint.iacr.org/2022/1209.

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

[BMO17] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. Forward and backward private searchable encryption from constrained cryptographic primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1465–1482. ACM Press, October / November 2017.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016.

[CLT20] Long Chen, Yanan Li, and Qiang Tang. CCA updatable encryption against malicious re-encryption attacks. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 590–620. Springer, Heidelberg, December 2020.

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-ciphertext secure fully homomorphic encryption. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 213–240. Springer, Heidelberg, March 2017.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.

[CW14] Jie Chen and Hoeteck Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014. https://eprint.iacr.org/2014/265.

[DGJ+21] David Derler, Kai Gellert, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. Journal of Cryptology, 2021.

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020.

[DJSS18] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 425–455. Springer, Heidelberg, April / May 2018.

[DKL+18] David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 219–250. Springer, Heidelberg, March 2018.

[DRSS21] David Derler, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Fine-grained forward secrecy: Allow-list/deny-list encryption and applications. In *FC*, 2021.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 98–129. Springer, Heidelberg, August 2017.

[FMM21] Andrés Fabrega, Ueli Maurer, and Marta Mularczyk. A fresh approach to updatable symmetric encryption. Cryptology ePrint Archive, Report 2021/559, 2021. https://eprint.iacr.org/2021/559.

[GCD+16] Junqing Gong, Jie Chen, Xiaolei Dong, Zhenfu Cao, and Shaohua Tang. Extended nested dual system groups, revisited. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 133–163. Springer, Heidelberg, March 2016.

[GCTC16] Junqing Gong, Zhenfu Cao, Shaohua Tang, and Jie Chen. Extended dual system group and shorter unbounded hierarchical identity based encryption. *Des. Codes Cryptogr.*, 80(3):525–559, 2016.

[GHJL17] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548. Springer, Heidelberg, April / May 2017.

[GM15] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320. IEEE Computer Society Press, May 2015.

[GP22] Yao Jiang Galteland and Jiaxin Pan. Backward-leak uni-directional updatable encryption from public key encryption. Cryptology ePrint Archive, Report 2022/324, 2022. https://eprint.iacr.org/2022/324.

[GW20] Junqing Gong and Hoeteck Wee. Adaptively secure ABE for DFA from k-Lin and more. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 278–308. Springer, Heidelberg, May 2020.

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. ABE for DFA from k-Lin. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 732–764. Springer, Heidelberg, August 2019.

[HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Heidelberg, March / April 2015.

[Jia20] Yao Jiang. The direction of updatable encryption does not matter much. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 529–558. Springer, Heidelberg, December 2020.

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

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, April 2012.

[LR21] Françoise Levy-dit-Vehel and Maxime Roméas. A composable look at updatable encryption. Cryptology ePrint Archive, Report 2021/538, 2021. https://eprint.iacr.org/2021/538.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 685–716. Springer, Heidelberg, April / May 2018.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

[MPW22] Peihan Miao, Sikhar Patranabis, and Gaven Watson. Unidirectional updatable encryption and proxy re-encryption from DDH or LWE. Cryptology ePrint Archive, Report 2022/311, 2022. https://eprint.iacr.org/2022/311.

[Nis22] Ryo Nishimaki. The direction of updatable encryption does matter. In *Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 194–224. Springer, 2022.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, December 2012.

[PCI22] PCI SSC. Ci security standards council. payment card industry data security standard: Requirements and testing procedures, v4.0. https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf, 2022.

[SYL+18] Shifeng Sun, Xingliang Yuan, Joseph K. Liu, Ron Steinfeld, Amin Sakzad, Viet Vo, and Surya Nepal. Practical backward-secure searchable encryption from symmetric puncturable encryption. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 763–780. ACM Press, October 2018.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

A Concrete Instantiation Under the d-Lin Assumption

For completeness, we provide the concrete DSG instantiation of Gong et al. [GCTC16]. The pairing operation is defined as $\widehat{e}((\mathbf{a}_1, \mathbf{a}_2), (\mathbf{b}_1, \mathbf{b}_2)) = e(\mathbf{a}_1, \mathbf{b}_1)/e(\mathbf{a}_2, \mathbf{b}_2)$. Let π_L, π_M, π_R be function that map

the leftmost d columns, the $d+1$-th column, and rightmost column of a matrix. We require the d-LIN assumption:

d-LIN assumption. For any PPT adversary D, we have that the function

$$\mathsf{Adv}^{\mathsf{d-LIN}}_{\mathsf{G},D}(\lambda) := |\Pr\left[D(pars, g^{a_{d+1}(s_1+\cdots+s_d)}) = 1\right] - \Pr\left[D(pars, g^{a_{d+1}(s_1+\cdots+s_d)+s_{d+1}}) = 1\right]|$$

is negligible in λ, where $(\mathbb{G}, \mathbb{H}, G_T, p, e) \leftarrow \mathsf{G}(\lambda, 1)$, $s_1, \ldots, s_{d+1}, a_1, \ldots, a_d \leftarrow \mathbb{Z}_p$, for $pars := (\mathbb{G}, \mathbb{H}, G_T, p, e, g, h, g^{a_1}, \ldots, g^{a_{d+1}}, g^{a_1 s_1}, \ldots, g^{a_d s_d})$, for generators g and h of $\mathbb{G}$ and $\mathbb{H}$, respectively.[11]

The DSG construction (adapted mostly verbatim from [GCTC16]) is as follows (we omit the algorithm SampGT since we directly use the respective values):

$(pp, sp) \leftarrow \mathsf{SampP}(\lambda, n)$: sample $(\mathbb{G}_1, \mathbb{G}_2, G'_T, p, g_1, g_2, g_T, g'_1, g'_2, e') \leftarrow \mathsf{G}(\lambda, 1)$ and set $\mathbb{G} := \mathbb{G}_1^{d+2} \times \mathbb{G}_1^{d+2}, \mathbb{H} := \mathbb{G}_2^{d+2} \times \mathbb{G}_2^{d+2}, G_T := G'_T, e := e', g := g_1, h := g_2$. Furthermore, sample matrices $\mathbf{B}, \mathbf{B}^* \leftarrow \mathsf{GL}_{d+2}(\mathbb{Z}_p)$ with $\mathbf{B}^\top \mathbf{B}^* = \mathbf{I}_{d+2}$ and $\mathbf{A}_0, \ldots, \mathbf{A}_n \leftarrow \mathbb{Z}_p^{(d+2)\times(d+2)}$, and sample diagonal matrix $\mathbf{R} \in \mathsf{GL}_{d+2}(\mathbb{Z}_p)$ with the right-most two diagonal entries being 1. Then, set

$$\begin{aligned}
\mathbf{D} := \pi_L(\mathbf{B}), \mathbf{D}_i := \pi_L(\mathbf{B}\mathbf{A}_i), \mathbf{D}^* := \pi_L(\mathbf{B}^*\mathbf{R}), \mathbf{D}^*_i := \pi_L(\mathbf{B}^*\mathbf{A}_i^\top \mathbf{R}) \\
\mathbf{m} := \pi_M(\mathbf{B}), \mathbf{m}_i := \pi_M(\mathbf{B}\mathbf{A}_i), \mathbf{m}^* := \pi_M(\mathbf{B}^*\mathbf{R}), \mathbf{m}^*_i := \pi_M(\mathbf{B}^*\mathbf{A}_i^\top \mathbf{R}), \\
\mathbf{f} := \pi_R(\mathbf{B}), \mathbf{f}_i := \pi_R(\mathbf{B}\mathbf{A}_i), \mathbf{f}^* := \pi_R(\mathbf{B}^*\mathbf{R}), \mathbf{f}^*_i := \pi_R(\mathbf{B}^*\mathbf{A}_i^\top \mathbf{R}),
\end{aligned}$$

for all $i \in [n] \cup \{0\}$, and function $m((g_2^{\mathbf{b}_1}, g_2^{\mathbf{b}_2})) := e(g_1, g_2)^{\mathbf{b}_1}$, for all $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{Z}_p^{d+2}$. Define $\widehat{g} := (g^{\mathbf{0}}, g^{\mathbf{f}}), \widehat{h} := (h^{\mathbf{0}}, h^{\mathbf{f}^*})$ and output

$$\begin{aligned}
pp &:= (\mathbb{G}, \mathbb{H}, G_T, p, g, h, g_T, \widehat{e}, m, g^{\mathbf{D}}, g^{\mathbf{D}_0}, \ldots, g^{\mathbf{D}_n}, h^{\mathbf{D}^*}, h^{\mathbf{D}^*_0}, \ldots, h^{\mathbf{D}^*_n}) \\
sp &:= (\widehat{g}, \widehat{h}, g^{\mathbf{m}}, g^{\mathbf{m}_0}, \ldots, g^{\mathbf{m}_n}, g^{\mathbf{f}}, g^{\mathbf{f}_0}, \ldots, g^{\mathbf{f}_n}, h^{\mathbf{m}^*}, h^{\mathbf{m}^*_0}, \ldots, h^{\mathbf{m}^*_n}, h^{\mathbf{f}^*}, h^{\mathbf{f}^*_0}, \ldots, h^{\mathbf{f}^*_n}).
\end{aligned}$$

$\mathbf{g} \leftarrow \mathsf{SampG}(pp)$: sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output $\mathbf{g} := ((g^{\mathbf{D}_0\mathbf{s}}, g^{\mathbf{D}\mathbf{s}}), (g^{\mathbf{0}}, g^{\mathbf{D}_1\mathbf{s}}), \ldots, (g^{\mathbf{0}}, g^{\mathbf{D}_n\mathbf{s}}))$.

$\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$: sample $\widehat{s} \leftarrow \mathbb{Z}_p$ and output $\widehat{\mathbf{g}} := ((g^{\widehat{s}\mathbf{f}_0}, g^{\widehat{s}\mathbf{f}}), (g^{\mathbf{0}}, g^{\widehat{s}\mathbf{f}_1}), \ldots, (g^{\mathbf{0}}, g^{\widehat{s}\mathbf{f}_n}))$ and $g_s := (g^{\mathbf{0}}, g^{\widehat{s}\mathbf{f}})$.

$\mathbf{h} \leftarrow \mathsf{SampH}(pp, sp)$: sample $\mathbf{r} \leftarrow \mathbb{Z}_p^d$ and output $\mathbf{h} := ((h^{\mathbf{0}}, h^{\mathbf{D}^*\mathbf{r}}), (h^{\mathbf{0}}, h^{\mathbf{D}^*_1\mathbf{r}}), \ldots, (h^{\mathbf{0}}, h^{\mathbf{D}^*_n\mathbf{r}}))$.

$\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp)$: sample $\widehat{r} \leftarrow \mathbb{Z}_p$ and output $\widehat{\mathbf{h}} := ((h^{\mathbf{0}}, h^{\widehat{r}\mathbf{f}^*}), (h^{\mathbf{0}}, h^{\widehat{r}\mathbf{f}^*_1}), \ldots, (h^{\mathbf{0}}, h^{\widehat{r}\mathbf{f}^*_n}))$ and $h_s := (h^{\mathbf{0}}, h^{\widehat{r}\mathbf{f}^*}), h_a := (h^{\mathbf{0}}, h^{\widehat{r}\mathbf{m}^*})$.

$S \leftarrow \mathsf{SampS}(pp)$: sample $\mathbf{s} \leftarrow \mathbb{Z}_p^{d+2}$ and output $S := (g^{\mathbf{s}}, g^{\mathbf{0}})$.

$K \leftarrow \mathsf{SampK}(pp)$: sample $\mathbf{k} \leftarrow \mathbb{Z}_p^{d+2}$ and output $K := (h^{\mathbf{D}^*\mathbf{k}}, h^{\mathbf{D}_0\mathbf{k}})$.

Correctness and security. All correctness and security claims carry over from [GCTC16] since no changes in the assumptions or distributions were made.

[11] The original definition of d-LIN requires $a_1, \ldots, a_d, s_{d+1} \leftarrow \mathbb{Z}_p$; however, as in [CW14, Remark 12], we allow for a negligible difference of $(d+1)/p$ in the $\mathsf{Adv}^{\mathsf{d-LIN}}_{\mathsf{G},D}$ function.