Puncture 'Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts

Daniel Slamanig and Christoph Striecks

AIT Austrian Insitute 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. This is achieved by means of an update token that allows to perform the ciphertext update via any semi-trusted party. Unfortunately, apart from a recent UE construction from indistinguishability obfuscation (Nishimaki, ePrint'21), in all existing constructions the update token provides additional functionality and at least allows to downgrade keys. Such a leakage is undesirable and leads to rather involved and complex security models. A recent UE model due to Jiang (Asiacrypt'20), extending the model of Boyd et al. (Crypto'20), explicitly considers these directionality and leakage issues, and left open the construction of UE schemes where keys cannot be transformed via a token in any direction (aka UE schemes with "no-directional" key updates).

In this work, we solve the problem via our threefold contribution:

i) We introduce a simpler and cleaner UE CPA security notion extending prior models. It focuses on UE schemes with no-directional key updates and thus avoids the use of rather complex leakage profiles for UE. Moreover, 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 inherently requires the no-directional key update feature of UE schemes.

ii) We introduce a novel approach of constructing UE with no-directional key updates which significantly departs from previous ones and in particular views UE from the perspective of puncturable encryption (Green and Miers, S&P'15). As a stepping stone, we introduce a variant of puncturable encryption called ciphertext puncturable encryption (CPE). This turns out to be a useful abstraction on our way to construct UE and may be of independent interest.

iii) Finally, we present a CPE instantiation from standard assumptions (i.e., the standard d-Lin assumption in prime-order bilinear groups) which via *ii*) yields the first UE scheme with no-directional key updates and expiry epochs from standard assumptions.

Keywords: Updatable Encryption, Puncturable Encryption

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 some industries even require it [PCI16]. 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, especially if the amount of outsourced data is huge.

As a remedy, Boneh et. al [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. It consists of the usual algorithms ($\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$)

[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`

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_i and outputs a fresh next-epoch key K_{i+1} along with a so-called update token Δ_{i+1}. This update token can be be used by a semi-trusted party to update ciphertexts under key K_i for epoch i to ciphertexts for epoch $i+1$ under key K_{i+1} via an algorithm Update. 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, UE schemes can be ciphertext-independent [LT18, KLR19, BDGJ20, Jia20, Nis21] such that a single compact update token Δ_{i+1} can update *any* ciphertext from epoch i to $i+1$. In the remainder of this work, we focus on UE schemes with ciphertext-independent update and will simply call them UE schemes. Security for UE essentially guarantees that the updated ciphertexts are indistinguishable from fresh encryptions, with Boyd et al. [BDGJ20] representing the state-of-the-art model which was recently extended by Jiang [Jia20] to consider directionality.

UE schemes might use deterministic or randomized updates. While encryption clearly needs to be randomized, the situation is less clear for updates. Deterministic updates avoid the use of randomness on the server side and enable an easier design of CCA secure UE schemes (cf. [KLR19] and [BDGJ20]). However, as a consequence of being deterministic, they require a rather severe weakening of the security model and in particular prevent the adversary from seeing the update token before the challenge epoch. While omitting this single token may sound like a minor difference, this excludes attacks that track ciphertexts or determine the existence of ciphertexts at a certain point in time already via the model. As this additional leakage can only increase the number of potential attacks, the use of randomized updates seems advisable.

The motivating questions. Intuitively, one would expect that in UE the only functionality of an update token is to update ciphertexts in the forward-direction from one to the next epoch. Interestingly, however, in all known UE schemes with the exception of a recent UE scheme from indistinguishability obfuscation by Nishimaki [Nis21][4], update tokens can also be used to at least update keys in the backward direction. We will go through the different possibilities of key updates

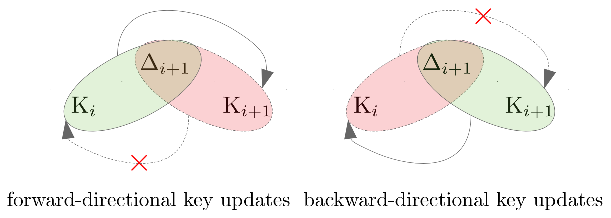
Δ_{i+1}
K_i
K_{i+1}
forward-directional key updates
Δ_{i+1}
K_i
K_{i+1}
backward-directional key updates

Fig. 1. UE schemes can by construction leak future keys (forward-directional key updates) or past keys (backward-directional key updates). Schemes that have bi-directional key updates leak in both directions, the future and the past. When considering ciphertexts instead of keys, forward-directionality is required per correctness. But schemes may additionally have backward-directional ciphertext updates and are then said to provide bi-directional ciphertext updates.

in a step-by-step fashion now (cf. Figure 1). We start with UE schemes that allow only key updates in the forward-direction. To see why such updates are a problem, for instance assume that all ciphertexts have been updated from epoch i to epoch $i+1$. In UE, one assumes that when this happens all old ciphertexts from epoch i are then deleted. Now, if an adversary manages to compromise some (old) key K_i and update token Δ_{i+1}, this should intuitively not endanger confidentiality of ciphertexts in epoch $i+1$. However, if an UE scheme allows key updates in the forward-direction, one can use Δ_{i+1} to upgrade the key K_i to K_{i+1} and then decrypt all the future data. Moreover, an UE scheme may allow to downgrade keys from epoch $i+1$ to epoch i. The associated issues are more subtle, but as we discuss in Section 2, such UE schemes also have unwanted side-effects. Lastly, we can have UE schemes with ciphertext updates in backward direction. Ignoring forward-directional key updates for a moment, this is also not desirable as adversaries that retrieve some old secret key K_i in epoch i and the token Δ_{i+1} can also downgrade ciphertexts from epoch $i+1$ to epoch i and simply decrypt. Besides being counter-intuitive, these issues result in security models that have to

[4] This has been proposed in concurrent and independent work and will be discussed below.

deal with these inferred information to explicitly exclude such cases from constituting an attack, and, hence, such models are quite involved as they have to track those leakages.

The discussion of directionality in UE was raised by Lehmann and Tackmann [LT18]. To prevent attacks like the ones mentioned above and to capture the "right intuition" of the security of UE, one requires UE schemes with so-called *no-directional* key and forward-directional ciphertext updates. In such schemes, update tokens are not helpful to update keys in either direction and ciphertexts can only be updated into the future. Indeed, as discussed by Jiang [Jia20] and Nishimaki [Nis21], such UE schemes with no-directional key updates are strictly stronger than UE schemes with at least backward-directional key updates[5] already in the model of Jiang [Jia20] extending the one due to Boyd et al. [BDGJ20]. However, the Jiang model is quite involved due to rather complex leakage-profile paradigm to capture all the "unnatural" attacks above coming from UE schemes with at least backward-directional key updates. In this work we ask:

What is a clean and strong security notion for UE schemes?

On the way to answer the above question, we additionally introduce the notion of expiry epochs for UE and argue that this feature makes UE even more natural (since ciphertexts can now expire) to prevent certain types of attacks that seem to be relevant in practice.

Apart from providing strong security guarantees in the sense of indistinguishabilty under chosen plaintext attacks (EE-IND-UE-CPA)[6], the construction of UE schemes with no-directional key updates from standard assumptions has so far been elusive and in this work we further ask:

Is it possible to construct provably secure UE schemes with no-directional key updates (in our strong model) from standard assumptions?

1.1 Our contribution

Our contribution is three-fold:

i) We revisit the IND-UE-CPA security of UE schemes and present a cleaner and simpler model that is capable of capturing the guarantees provided by UE with no-directional key updates, yielding progress towards answering the first question. In particular, our model does not need the complex leakage-profile paradigm or the insulated regions (IR) approach which made existing UE models rather cumbersome. Our model explicitly focuses on no-directional key updates and, consequently, we can give the adversary all updates tokens and all secret keys except for the target-end epoch and the target epochs, respectively, to prevent trivial wins. This massively reduces oracle calls and checks for trivial wins due to leaking tokens or keys. We furthermore introduce expiry epochs as a feature of UE that extends and generalizes the security model for no-directional key updates due to Jiang [Jia20]. By letting ciphertexts expire, we capture the "wait for one key leakage and update all ciphertexts captured in past epochs to the leaking-key epoch"-type of attacks which we consider as a relevant attack in current ciphertext independent UE schemes.

ii) As a second contribution, we introduce a novel primitive dubbed Ciphertext Puncturable Encryption (CPE) which we believe provides an easier intuition towards our UE constructions with no-directional key updates and then build such a UE scheme from any CPE scheme. Furthermore, we show a concrete asymptotically efficient construction of such a scheme. While CPE is slightly more powerful than what is required for UE with no-directional key updates (mainly as CPE has tags from a certain tag space that can be adaptively queried), interestingly, we were not able to construct such UE schemes from tools that are weaker than those required to instantiate CPE. Moreover, we believe that the ciphertext puncturing in CPE will further increase the applicability of the already very useful puncturable-encryption primitive and might be of independent interest.

[5] Jiang [Jia20] speaks of uni-directional UE schemes which we refer to ones that allow to upgrade keys but prevent downgrades. Recently, Nishimaki [Nis21] showed that one can consider directionality more nuanced as there can be schemes that prevent forward but allow backward-directional key updates.

[6] As Nishimaki [Nis21], we only deal with a CPA-like UE notion starting from the model of Jiang. Currently, indistinguishabilty under chosen-ciphertext attacks (IND-UE-CCA) seems to be hard to achieve with our constructional approach and we leave it as interesting open problem. See paragraph on CCA security for more information.

iii) Finally, we present the first UE scheme with no-directional key updates and expiry epochs via ii). Concretely, we instantiate it from the standard d-Lin assumption (where for $d = 1$ we get UE with no-directional key updates from SXDH) in prime-order bilinear groups using the well-known dual system paradigm. In Figure 2, we provide a brief comparison of our UE constructions with other UE schemes.

Schemes	key-sizes	ct-sizes	tok-sizes	security	model	dir. (key)	dir. (ct)	assumption
BMLR+ [BLMR13, 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
SHINE0 [BDGJ20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	d-IND-UE-CCA	IC	bi	bi	DDH
MirrorSHINE [BDGJ20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	d-IND-UE-CCA	IC	bi	bi	DDH
OCBSHINE [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	DLWE
Nishimaki [Nis21]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	backw	forw	LWE
Nishimaki [Nis21]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	r-IND-UE-CPA	SM	no	forw	IO, OWF
Ours	$\mathbf{O}(\log^2 n)$	$\mathbf{O}(\log^2 p)$	$\mathbf{O}(1)$	EE-IND-UE-CPA	SM	no	forw	SXDH

Fig. 2. Overview of ciphertext-independent UE schemes with at least some form of weak IND-UE-CPA security due to [BDGJ20]. With n we denote the total number of epochs and with $p = 2^\ell \leq n$ we denote the expiry epoch of a ciphertext, with some $\ell \in \mathbb{N}$ with $\ell \leq \lambda$. 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 with expiry epochs. IC and SM stand for Ideal Cipher and Standard Model. KH-PRF, DDH, (D)LWE, IO, OWF and SXDH stand for key-homomorphic pseudo-random function, decisional Diffie-Hellman, (decisional) learning with errors, indistinguishability obfuscation, one-way function, and symmetric extended Diffie-Hellman, respectively. The version of the concurrent and independent work of Nishimaki [Nis21] was retrieved from ePrint on 16th of December 2021.

Independent concurrent work. In an independent and concurrent work, Nishimaki [Nis21] also studies the construction of no-directional UE schemes. Besides providing the already mentioned separation of UE schemes with bi- and backward-directional key updates in the model of Jiang [Jia20], he also provides constructions of UE schemes with backward and no-directional key updates. Compared to our no-directional scheme (see Figure 2), which can be instantiated from standard tools, his construction relies on indistinguishability obfuscation (IO). While his schemes have parameters sizes independent of the system parameters, despite recent progress in IO [JLS20, GJLS21] towards constructions under simple-to-state or even standard assumptions, their practical efficiency is still elusive. Furthermore, our construction techniques clearly differ since he deals with IO-based (and lattice-based) constructions.

More related work. Besides the already mentioned work on ciphertext independent and ciphertext dependent UE schemes [BLMR13, EPRS17, LT18, KLR19, BDGJ20, BEKS20, CLT20, Jia20], recently there has also been increased interest in studying UE in composable frameworks such as the universal composability (UC) framework [JKR19], the framework of constructive cryptography [dVR21, FMM21], and UE for onion routing [KHRS21]. Furthermore, Sehrawat and Desmedt [SD19] introduced bi-homomorphic lattice-based pseudo-random functions (PRFs) which are PRFs that are key-homomorphic (as used for the first UE schemes in [BLMR13]) and additionally partially-input homomorphic. As an application, they present UE schemes with unidirectional ciphertext but bidirectional key updates. While this is an interesting step towards UE schemes with unidirectional ciphertext and key updates, their scheme suffers from the drawback that for each ciphertext, the data owner needs to sample a random data-block identifier which needs to be privately shared with the server. Consequently, the state of the data owner (i.e., the secret key) is linear in the number of ciphertexts (which is not the case for our constructions).

A primitive that seems closely related to UE is uni-directional proxy re-encryption (PRE) formalized by Ateniese et al. [AFGH05] and in particular multi-hop variants thereof [Gen09, CCL+14, PRSV17, DN21] (henceforth UM-PREs). We recall that a UM-PRE scheme is a public-key encryption scheme, where given secret key $_A$ and public key pk_B one can compute a re-key $rk_{A\to B}$ that translates ciphertexts under pk_A to ones under pk_B and this process can be applied multiple times.

While this intuitively seems to match UE when converted to it in the obvious way, there are subtleties that need to be discussed when aiming for UE schemes with strong guarantees. UM-PRE schemes and their established CPA security notions [AFGH05, Coh19, FKKP19] do not require that original and re-encrypted ciphertexts are of the same form and thus indistinguishable. And indeed, lattice-based constructions [PRSV17] or generic constructions from any PKE and garbled circuits [DN21] do trivially yield distinguishable ciphertexts due to the growing noise and the linear growth of ciphertexts respectively. What remains from known constructions is UM-PRE constructed from fully homomorphic encryption (FHE) using bootstrapping [Gen09], obfuscators for re-encryption (realized via FHE with bootstrapping) [CCL$^+$14] and constructions from any PKE and probabilistic indistinguishability obfuscation [DN21]. Consequently, the state-of-the-art in UM-PRE, even when ignoring that the security notions do not support the required features, i.e., indistinguishability of ciphertexts or expiry epochs in a black-box way, seems no fruitful avenue towards reasonably practical UE solutions with no-directional key updates and at that point it seems more promising to take a different path.

On stronger (i.e., chosen-ciphertext) security for UE. Klooß et al. in [KLR19] initiated the study of (replayable) chosen ciphertext security for UE schemes and since then (R)CCA security has been treated in subsequent works in the ciphertext-dependent [BDGJ20, BEKS20, CLT20] and ciphertext-independent [BDGJ20] setting. There is only one approach to CCA security for ciphertext-independent UE schemes Klooß et al. in [KLR19] which does not rely on schemes with deterministic updates. We currently do not see a suitable approach to provide (R)CCA security for stronger UE schemes with no-directional key updates and in particular for our concrete instantiations, but consider this an interesting question for future research. Hence, as it is also done in the concurrent and independent work of Nishimaki [Nis21], in this work, we solely deal with chosen-plaintext security. This already turned out to be non-trivial to achieve.

2 Motivation and Overview of Our Techniques

In this section, we motivate our techniques concerning the contribution and give an overview of i) our extended UE security model with expiring epochs, ii) UE from a puncturable-encryption perspective, and iii) instantiating ciphertext puncturable encryption from standard assumptions.

2.1 Extended and simpler security model with expiry epochs

We will now use Figure 3 to illustrate the most important aspects of the CPA notion of Jiang [Jia20] (extending Boyd et al. [BDGJ20]) for UE schemes with bi/forward-directional key and ciphertext updates (which strengthen and simplify previous models [LT18, KLR19]) by means of the maximum information available to an adversary.

Let K_i and $\varDelta_{i+1}$ be the key and the token for epoch i, 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 $\tilde{e} < e^*$. The key concept in common UE security models (particularly, in [LT18, BDGJ20, Jia20, Nis21]) is that of a firewall which prevents trivial wins and there is a firewall e_{start} before and e_{end} after the challenge epoch e^* (indicated by red boxes in Figure 3). In between, all update tokens can be revealed and prior to e_{start} and after e_{end} the adversary can obtain all keys and update tokens.[7] Previous work captures trivial wins via so-called leakage profiles where such firewalls have to be present to be able to prove security. 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^*, and in e_{end} *neither* the key *nor* the update token are revealed. Otherwise, due to 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 upgrading. Here, we have to look at the epochs before the challenge epoch e^*, as tokens should not even allow upgrades of keys. We observe for such UE schemes, we can remove the firewall at e_{start} entirely and hand out all keys and tokens up to e^* to the adversary.

[7] There can be more so-called isolated regions (IRs) that are bounded by firewalls, but we only focus on the one containing the challenge epoch (cf. [BDGJ20]).

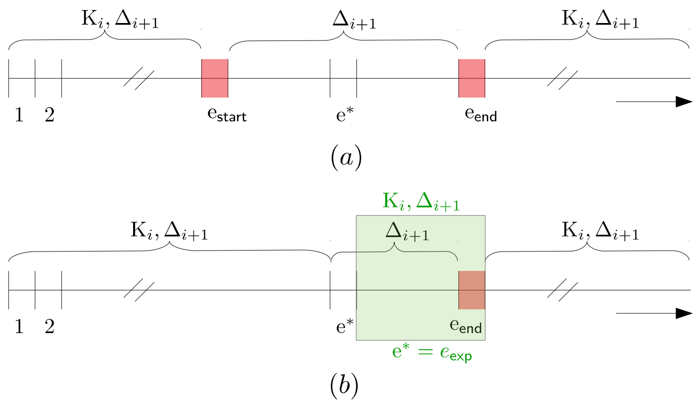

Fig. 3. (a) Example of information an adversary is allowed to obtain in known CPA notions [LT18, BDGJ20, Jia20] for UE schemes with bi-directional key and ciphertext updates. (b) Example of information an adversary is allowed to obtain in our EE-IND-UE-CPA experiment for UE schemes with no-directional key updates. We are able to remove the left firewall e_{start} and 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 challenge epoch e^* obviously.

This change gives rise to what Nishimaki in [Nis21] calls UE with backward-directional key updates, i.e., downgrading of keys is possible, but upgrading of keys is not possible. 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. 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 (key-rotations) a newer key leaks, it will still be possible to decrypt an old ciphertext by simply updating it to the respective epoch.

Note that due to correctness, even UE schemes that *do not* allow downgrades of keys do not help to protect against this threat. To prevent these types of attacks and provide strong forward-security guarantees, we introduce an enhancement yielding a generalization of the CPA model with no-directional key updates by Jiang.

Namely, 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 $C_{i,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_i, M, e_{\mathsf{exp}})$ 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 date in the future. Also, by setting no expiry date for ciphertexts, i.e., $e_{\mathsf{exp}} = \infty$, we are back in the model of Jiang (without leakage profiles).

Jiang [Jia20] already provided a detailed discussion for UE with no-directional key updates and its leakage profiles with winning conditions in her model, but we want to make the common model even more compact and intuitive, particularly, by removing the complex leakage-profile paradigm. Note that this gives strong post-compromise guarantees without relying on artificial model restrictions (i.e., the left firewall), as now a leaked key, i.e, K_i, even when everything is available to the adversary does not endanger ciphertexts produced in epochs $j > i$ due the no-directional key update feature.

With this concept, the above mentioned key-downgrade attack is mitigated for all keys that are leaked after the expiry date of a ciphertext as long as the key updates are at most in the forward-direction, i.e., key downgrades are prevented. To see this, note that in UE schemes with backward-directional key updates even with expiry date, the attack is not prevented as one could simply downgrade the leaked key back into the respective epoch of the ciphertext. But this is not possible with forward-directional key updates and, thus, particularly for no-directional key updates. Having no-directional key updates in addition also provides the stronger post-compromise guarantees as discussed above, making no-directional UE schemes with expiry epochs a desirable goal.

How forward and post-compromise security is modeled. We briefly recall the forward and post-compromise security (FS and PCS) guarantees. As already implicit in Jiang [Jia20], via forward-directionality in key-updates, we achieve strong PCS security. By requiring in our model that a current-epoch key cannot be downgraded and tokens cannot be used to downgrade ciphertexts, together with expiry epochs allows us to achieve strong FS in particular when downgrading goes below the expiry epoch.

2.2 UE from a puncturable-encryption perspective

For our UE constructions, we significantly depart from previous work and view 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 encryption primitive with an additional puncturing algorithm that takes a secret key and a tag t as input, and produces an updated secret key. This updated (punctured) secret key is able to decrypt all ciphertexts *except* those tagged with t and (updated) secret keys can be iteratively punctured on distinct tags. PE found numerous applications from non-interactive messaging [GM15], watermarking [CHN+16], chosen-ciphertext secure FHE [CRRV17], searchable encryption [BMO17], forward-secure proxy re-encryption [DKL+18] to forward-secret zero round-trip time (0-RTT) key-exchange protocols [GHJL17, DJSS18, DGJ+21] and has been extended in several ways [WCW+19, DKL+18] while the design of novel PE schemes is still ongoing [SSS+20, SDLP20, DRSS21]. Despite being slightly different in their concrete formulation (e.g., allowing single or multiple tags per ciphertext), existing PE schemes all provide the same puncturing functionality as discussed above. Notable, we define CPE as a symmetric-key primitive extending the definitions of PE used in prior work [SYL+18, AGJ19, AGJ21, BDdK+21].

We observe the following similarity between UE with no-directional key updates and (symmetric) PE. The core guarantee in a UE scheme is that newer (as well as updated) ciphertexts can no longer be decrypted by older keys. This is abstractly reminiscent of puncturing when we view tags as epochs, i.e., associate ciphertexts to all epochs and puncture keys on epochs, such that they no longer can be used for decryption. As in UE, one however has to update ciphertexts *and* keys, puncturing needs to happen on both in a synchronized way and 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 update ciphertexts. Realizing this feature augments the known PE functionalities as we will discuss. The challenging issue from a UE perspective is now that one needs to prevent "unpuncturing," i.e., even if keys and update information leak, it should not be possible to remove tags from newer ciphertexts (downgrade ciphertexts in the language of UE) as well as use the update information to synchronize old keys to new keys and thus ciphertexts (no-directionality of key updates in the language of UE).

We now extend the functionality of puncturing from keys only as in PE to also allow puncturing of ciphertexts via the notion of ciphertext puncturable encryption (CPE). A CPE scheme can abstractly be viewed as a (symmetric) PE scheme $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{Dec})$ with an additional algorithm CPunc to puncture ciphertexts. However, there are some significant differences compared to conventional non-interactive PE (particularly, puncturings on keys and ciphertexts now happen in a synchronized manner via a puncture token).

More concretely, keys in CPE are initially associated to the entire tag space $\mathcal{T}$ (which is here considered to be an ordered set of polynomial size in the security parameter). As in PE, in CPE ciphertexts are computed w.r.t. a tag set $\mathcal{T}' \subseteq \mathcal{T}$. This happens via Enc by either taking no tag at all, which means that ciphertexts carry all tags in $\mathcal{T}$. Or one provides an "expiry-tag" t_{exp} which means that a ciphertext carries all tags $t \in \mathcal{T}$ with $t \leq t_{\mathsf{exp}}$. When it comes to puncturing, keys can be incrementally punctured on tags, whereas in contrast to conventional PE, this puncturing results in an updated key. Consequently, also the ciphertext puncturing needs to output a ciphertext towards the new punctured key. Otherwise, if one does not synchronize the key and ciphertext puncturing process, old keys might be able to still decrypt punctured ciphertexts, a property we want to avoid (and is related to achieve strong post-compromise and forward security).

The key puncturing on tag t produces a puncturing token Δ_t. Ciphertext puncturing is not a fully public and stand-alone operation, but requires the puncturing token Δ_t to keep ciphertexts and keys in synchronization. The idea is that a ciphertext produced under a key (punctured on set of tags $\mathbf{t}$) using this token can be punctured on tag t and results in a new ciphertext under the new key punctured on tags $\mathbf{t} \cup \{t\}$. Thereby, key puncturings can happen on any not necessarily ordered sequence of tags iteratively and ciphertexts can be punctured using the corresponding puncture tokens in an arbitrary order. The semantics of CPE is now that a key $K_{\mathbf{t}}$ punctured on $\mathbf{t}$ can only decrypt ciphertexts as long as the ciphertext still caries some non-punctured tag $t \notin \vec{t}$. Observe that this allows to put an expiry-tag on ciphertexts by creating a ciphertext with respect to some tag t_{exp}; as soon as the key and the ciphertext are punctured on *all* tags $t_i \leq t_{\mathsf{exp}}$, a ciphertext puncturing fails. In Figure 4, we give a brief overview of the differences between (symmetric) PE and CPE puncturing.

$$\begin{tikzcd}
\text{PE:} & K_{\mathbf{t}} \arrow[r] & K_{\mathbf{t}\cup\{t\}} & C_t & \\
\text{CPE:} & K_{\mathbf{t}} \arrow[r] & K_{\mathbf{t}\cup\{t\}} & C_{\mathbf{t}} \arrow[r, "\Delta_t"] & C_{\mathbf{t}\cup\{t\}}
\end{tikzcd}$$

Fig. 4. Overview on how puncturing takes place in PE and CPE. In PE, C_t is not punctured while in CPE, $C_{\mathbf{t}}$ can be punctured via Δ_t while Δ_t cannot unpuncture $C_{\mathbf{t}\cup\{t\}}$. Furthermore, Δ_t cannot transform $K_{\mathbf{t}}$ or $K_{\mathbf{t}\cup\{t\}}$ in any direction. $K_{\mathbf{t}\cup\{t\}}$ and $C_{\mathbf{t}\cup\{t\}}$ have to be in sync for successful decryption.

To construct UE with no-directional key updates from CPE, let $\mathsf{CPE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{KPunc}, \mathsf{CPunc}, \mathsf{Dec})$ be a CPE scheme and we view the tag space $\mathcal{T} = \{1, 2, \ldots, n\}$ of CPE as a polynomially bounded ordered set of epochs with n being polynomially bounded in the security parameter. Now, we compute all ciphertexts with respect to the entire tag space $\mathcal{T}$ (or expiry-tag t_{exp} in Enc when provided). The intuition of the construction is as follows: our initial key K_1 is derived from the key key punctured on tag 1 of the CPE scheme and every update of the key from epoch e to $e+1$ represents a puncturing of the key on the epoch number $e+1$ (viewed as tag), setting the key K_{e+1} to $K_{\mathbf{t}}$ with $\mathbf{t} = \{1, \ldots, e+1\}$. The update token Δ_{e+1} for ciphertexts from period e to $e+1$ represents the CPE puncture token Δ_{e+1} and updating ciphertexts from e to $e+1$ corresponds to puncturing ciphertexts on tag $e+1$. Encrypting then amounts to using the encryption algorithm of the CPE using the respective key $K_{\mathbf{t}}$ (with $\mathbf{t} = \{1, \ldots, e+1)\}$ of the CPE. Correctness and security guarantees carry over as we will show.

2.3 CPE construction from standard assumptions

In this section, we showcase our CPE construction by combining Lewko's approach for unbounded hierarchical identity-based encryption (HIBE) [Lew12] with the binary-tree encryption (BTE) approach due to Canetti, Halevi, and Katz [CHK03] via the dual-system paradigm in the composite-order setting. Later we will present a prime-order instantiation of our combined approach using (extended) dual system groups [CW13, GCTC16]. The hardest part in our UE-from-CPE construction is to provide key and ciphertext sizes sub-linear in the security parameter.[8] This can be achieved since UE is essentially a sequence version of CPE where we will traverse a binary tree (which holds keys and ciphertext elements) in pre-order yielding the desired properties for UE.

As the name suggests, the BTE approach organizes binary identities $\{0,1\}^\ell$ (which we view as tags) in a binary tree, where prefix identities/tags at some level $< \ell$ can delegate keys to its descendant entities, but cannot decrypt ciphertexts intended for other (hierarchical) identities/tags. Say, a key K^0 can be used to derive keys K^{00}, K^{01} for both descendant identities/tags 00 and 01, but neither K^{00} nor K^{01} can decrypt ciphertexts for K^0. We will extend this approach to have an analog delegation for ciphertext tags by maintaining a binary tree for each ciphertext as well.

Now, let us consider a complete binary tree (labeled ϵ at the root) and the usual labeling of left and right child with 0 and 1 respectively, yielding leafs labeled by some tags $\{0,1\}^\ell$. Consequently, for a height-ℓ binary tree, we have an ordered tag space $\mathcal{T}$ of 2^ℓ elements by starting the ordering with left-most leaf. The natural approach following [GHJL17] is that puncturing of a tag $t \in \{0,1\}^\ell$ in the tree results in deriving BTE keys for prefix tags which are not ancestors of the corresponding leaf and deleting all other keys, i.e., truncating the tree accordingly. Note that this allows to derive keys for all non-punctured leaves aka tags. We observe that we can do the delegation on ciphertexts as well which is reminiscent of Lee et al. [LCL+13]. Unfortunately, their scheme has no feature on delegating keys for tags and we do not see how to straightforwardly synchronize both in their setting

[8] Since UE is mainly motivated by practice, constructing UE scheme with sub-linear size parameters seems to be the goal (particularly, when comparing the construction with the simple download, decrypt, encrypt under new key, and upload approach).

while achieving our strong CPE security requirements particularly towards forward security *and* post-compromise security.

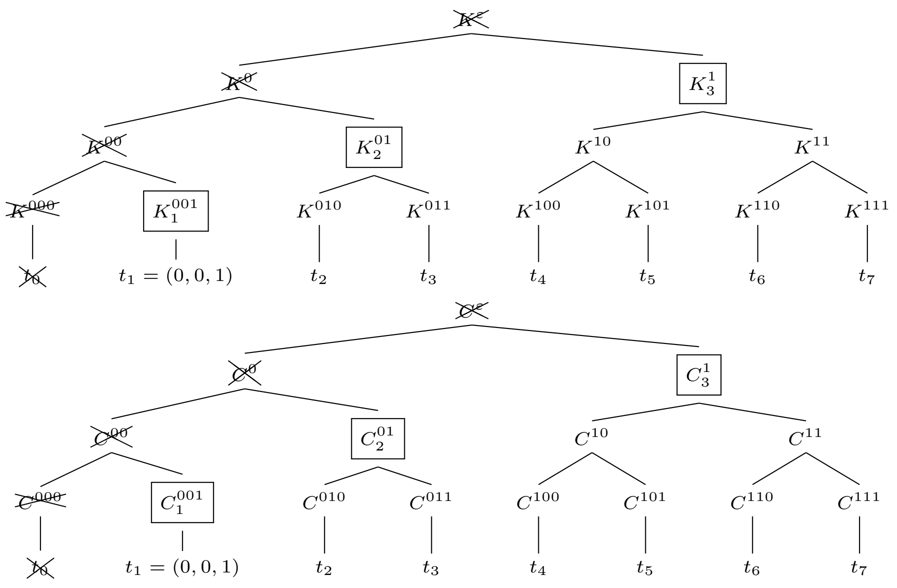

Fig. 5. Example of a CPE key $K_{t_0} = (K^{1}_{t_0}, K^{01}_{t_0}, K^{001}_{t_0})$ and ciphertext $C_{t_0} = (C^{1}_{t_0}, C^{01}_{t_0}, C^{001}_{t_0}, e(g, g^{\alpha_{t_0}})^r \cdot M)$ that have been punctured on t_0. Only the $\boxed{\text{boxed}}$ elements have to be stored and the remaining elements can be derive from those.

To construct a CPE scheme, we implicitly arrange tags for ciphertexts in a binary tree as well, analogously to keys. The root of the key delegation tree is associated with the initial key K_ε while the root of the ciphertext delegation tree is $C_{\varepsilon,t_{\mathsf{exp}}}$. In Figure 5, we give an example of how the ciphertexts and keys are constructed and where the tag t_0 is already punctured.

In our construction, keys and ciphertexts are a certain combination of the keys and ciphertexts in Lewko's approach enhanced with the techniques due to Canetti, Halevi, and Katz' (CHK) BTE work. As said, we will only showcase our results on the very high level and the composite-order setting here, but in the actual construction later, we will use the prime-order instantiation of the combined approach. Lewko's approach works in the bi-linear setting, where $e : \mathbb{G} \times \mathbb{G} \to G_T$ denotes the pairing, for groups $\mathbb{G}, \mathbb{G}_T$ of composite-order. Moreover, we lift Lewko's approach to the CHK setting where now the keys and ciphertexts are built as

$$K_{\mathbf{t}} = (g, v, (g_{0,i}, g_{1,i})_i, e(g,g)^{\alpha_{\mathbf{t}}}, (K_i^{t_i})_i) \text{ and } C_{\mathbf{t}} = \left((C_i^{t_i})_i, e(g, g^{\alpha_{\mathbf{t}}})^r \cdot M\right), \text{ for}$$

$$K_i^{t_i} = (K_{i,0}, K_{i,1}, K_{i,2}, \ldots, K_{i,2\ell}) = \left(g^{\alpha_{\mathbf{t}}}, g^{s_i}, v^{\alpha_{\mathbf{t}}} \cdot \prod_{j=1}^{\ell'=|t_i|} g^{s_i}_{t_{i,j},j}, \begin{matrix} g^{s_i}_{0,\ell'+1}, \ldots, g^{s_i}_{0,\ell} \\ g^{s_i}_{1,\ell'+1}, \ldots, g^{s_i}_{1,\ell} \end{matrix} \right),$$

$$C_i^{t_i} = (C_{i,0}, C_{i,1}, C_{i,2}, \ldots, C_{i,2\ell}) = \left(g^{r} \cdot v^{w_i}, g^{w_i}, \prod_{j=1}^{\ell'=|t_i|} g^{w_i}_{t_{i,j},j}, \begin{matrix} g^{w_i}_{0,\ell'+1}, \ldots, g^{w_i}_{0,\ell} \\ g^{w_i}_{1,\ell'+1}, \ldots, g^{w_i}_{1,\ell} \end{matrix} \right),$$

for global randomness r, local randomnesses s_i, w_i, and prefix tags $t_i = (t_{i,1}, \ldots, t_{i,\ell'}) \in \{0,1\}^{\ell' \leq \ell}$ (according to the tree configuration). Furthermore, $g, v, g_{0,i}, g_{1,i}$ are uniform group elements from $\mathbb{G}$ and M denotes the message. Decryption works as in Lewko's approach and we want to show the puncturing part here (which is very similar to decryption). The puncture token is

$$\Delta_t = (\Delta_0, \Delta_1, \Delta_2) = \left(g^{\alpha_t}, g^{s'}, v^{\alpha_t} \cdot \prod_{j=1}^{|t|} g^{s'}_{j,t_j}, e(g, g^{\alpha_t}) \right),$$

for uniform exponent α_t (which will act as a uniform linear shift in exponent), uniform randomness s' and some unpuctured tag $t \in \{0,1\}^\ell$. This puncture token can be used to retrieve the element $e(g, g^{\alpha_t})^r$ by "decrypting" C_i^t (which is derived from some $C_i^{t_i}$ since t was not punctured yet) with

$\varDelta_t$:

$$e(g, g^{\alpha_t})^r = \frac{e(C_0, \varDelta_0) \cdot e(C_2, \varDelta_1)}{e(C_1, \varDelta_2)} = \frac{e(g^r \cdot \cancel{v^{w_i}}, g^{\alpha_t}) \cdot \cancel{e(\prod_{j=1}^{|t|} g_{t_j,j}^{w_i}, g^{s'})}}{\cancel{e(g^{w_i}, v^{\alpha_t} \prod_{j=1}^{|t|} g_{t_j,j}^{s'})}}.$$

The $e(g, g^{\alpha_t})$ element from $\varDelta_t$ and $e(g, g^{\alpha_t})^r$ element just computed can now be used to update the $\alpha_{\mathbf{t}}$-component in $K_{\mathbf{t}}$ and $C_{\mathbf{t}}$ (via properties of the pairing), respectively, yielding

$$K_{\mathbf{t}\cup\{t\}} = (g, v, (g_{0,i}, g_{1,i})_i, \boxed{e(g, g^{\alpha_{\mathbf{t}}+\alpha_t})}, (K_i^{t'_i})_i)$$
$$C_{\mathbf{t}\cup\{t\}} = \left((C_i^{t'_i})_i, \boxed{e(g, g^{\alpha_{\mathbf{t}}+\alpha_t})^r} \cdot M\right),$$

where $(K_i^{t'_i})_i$ and $(C_i^{t'_i})_i$ contain delegated key and ciphertext elements for tags t'_i using the methodology explained above and in Figure 5. This yields a consistent key and ciphertext for punctured tags $\mathbf{t} \cup t$. The tags $(t'_i)_i$ correspond to the prefixes of all unpunctured tags as in Figure 5.

The crucial feature here is that the puncture token $\varDelta_t$ and the key $K_{\mathbf{t}\cup\{t\}}$ cannot be used to produce a valid key $K_{\mathbf{t}}$ (and, hence, key unpuncturing is prevented). This is due to that the node in the key tree corresponding to t is delegated and cannot be recovered anymore.

Similarly, the puncture token $\varDelta_t$ and the key $K_{\mathbf{t}}$ cannot be used to produce a valid key $K_{\mathbf{t}\cup t}$. This is due to the fact that $\varDelta_t$ is tailored towards a specific tag t and, hence, the key parts (not associated to t) cannot be updated to a valid key that excludes t. This is also the reason why you cannot downgrade the ciphertext $C_{\mathbf{t}\cup t}$ to $C_{\mathbf{t}}$ via the puncture token $\varDelta_t$ since the tag t was already punctured in the $C_{\mathbf{t}\cup t}$-associated tree configuration.

Furthermore, we want to emphasize that re-randomization is necessary to prevent mix-and-match attacks. Keys and ciphertexts can be publicly re-randomized straightforwardly.

We want to emphasize that the depicted scheme above serves only as a showcase of our CPE techniques. Our concrete construction is in the prime-order setting (due to efficiency reasons). In this sequel, we borrow ideas from Chen and Wee [CW13] as well as Gong et al. [GCTC16] where latter proved the Lewko approach [Lew12] secure in the prime-order setting. Fortunately, as we will argue in the respective section, a slight simplification of the Gong et al. security properties already suffices to prove our CPE secure. However, we have to carefully examine the proof since our CPE is not exactly comparable to Lewko's approach.

Alternative viewpoint. We can view our approach in terms of forward-secure cryptography (originating from works due to Günther [Gün90], Diffie et al. [DvW92], and Anderson [And00] on forward-secure key-exchange and signatures). In particular, forward-secure public-key encryption (fs-PKE) due to Canetti et al. [CHK03] is closely related to parts of our approach (essentially the BTE approach). In fs-PKE, keys are associated to a time t_k allowing to decrypt ciphertexts encrypted for such time. While keys can be non-interactively updated to any time $t'_k > t_k$, ciphertexts cannot be updated. Lee et al. [LCL+13] introduced the concept of self-updatable encryption (SUE). In a SUE scheme a ciphertext and a secret key (generated via a master secret key) are associated with a time t_c and t_k respectively. A user can decrypt a ciphertext with time t_c using a secret key with time t_k if $t_k \geq t_c$. Additionally, ciphertexts can be updated by anyone to any time $t'_c > t_c$. However, keys in SUE cannot be updated and we do not see how to straightforwardly equip their approach with key updating. (The main reason is the shared global randomness we need across the ciphertext tree and which is only present in Lewko's HIBE-based approach.) We want to mention that both works also apply the Binary-Tree Encryption (BTE) approach [CHK03]. While puncturable encryption (PE) closely mirrors fs-PKE (since keys can be punctured), our approach extends fs-PKE and SUE to allow updating secret keys *and* ciphertexts consistently via puncture tokens in CPE yielding new application areas such as UE.

Outline of the paper. In Section 3, we present preliminaries. In Section 4, we present our strengthened and simplified security model starting from the UE framework of Boyd et al. [BDGJ20] to reflect no-directional key updates and a simpler security model for such a feature. In Section 5, we introduce Ciphertext Puncturable Encryption (CPE) along with our security model. Then, in Section 5.2, we proceed to show how to instantiate UE with no-directional key updates from any CPE scheme satisfying our security notion. Finally, in Section 6, we present our construction of CPE from the dual system paradigm along with a discussion for our rationale of this choice. Moreover, we give a concrete instantiation of CPE under the d-Lin assumption in prime-order bilinear groups.

3 Preliminaries

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 $\forall c \exists k_0 \forall \lambda \geq k_0 : |f(\lambda)| < 1/\lambda^c$). We may write $q = q(\lambda)$ if we mean that the value q depends polynomially on λ. We write sets in bold font, e.g., $\mathbf{v} = \{v_1, \ldots, v_n\}$ for a set of size $n \in \mathbb{N}$ and with components $v_1, \ldots, v_n$. (We may also write $\mathbf{v} = (v_i)_{i \in [n]}$ or even $\mathbf{v} = (v_i)_i$ in this case.) 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 of order n'. A *pairing* $e : \mathbb{G} \times \mathbb{H} \rightarrow G_T$ 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, given security parameter λ and integer n', generates the tuple $(\mathbb{G}, \mathbb{H}, G_T, N, g, h, (g_{p_i})_{i \in [n']}, e)$, for a pairing $e : \mathbb{G} \times \mathbb{H} \rightarrow G_T$, for composite-order groups $\mathbb{G}, \mathbb{H}, G_T$, all of known group order $N = p_1 \cdots p_{n'}$, generators $g_{\mathbb{G}}, h_{\mathbb{H}}, (g_{p_i})_{i \in [n']}$, and for $\Theta(\lambda)$-bit primes $(p_i)_i$. As a special case, let $\mathsf{SG}(\lambda, n')$ be a group generator similar to G except that it outputs $(\mathbb{G}, G_T, N, g, (g_{p_i})_{i \in [n']}, e)$, for symmetric pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow G_T$.

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

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

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.[9]

4 Updatable Encryption with No-Directional Key Updates and Expiry Epochs

In this section, we recap updatable encryption with no-directional key updates and the enhancement of expiry epochs. We build on the recent UE models [BDGJ20, Jia20] dealing solely with the natural form of updatability namely security for UE schemes with no-directional key updates and forward-directional ciphertext updates. This allows to simplify the security model for UE with no-directional key updates significantly by avoiding the use of the rather complicated leakage-profile paradigms (which come into play when considering weaker UE schemes that do not provide no-directionality of keys).

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 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)*:*

[9] 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.

$\mathsf{Gen}(\lambda)$: *on input security parameter* λ, *the key generation algorithm outputs an initial key* K_1.
$\mathsf{Next}(K_e)$: *on input key* K_e *for* $e \in [e(\lambda) - 1]$[10], *the key update algorithm outputs an updated key* K_{e+1} *for 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}, *for* $e, e_{\mathsf{exp}} \in [e(\lambda)]$, *encryption outputs a ciphertext* $C_{e,e_{\mathsf{exp}}}$.
$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$: *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. *Correctness ensures that an update of a valid ciphertext* $C_{e,e_{\mathsf{exp}}}$ *(via* Δ_{e+1}*) from epoch* e *to* $e+1$ *yields a valid ciphertext* $C_{e+1,e_{\mathsf{exp}}}$ *that can be decrypted under the epoch key* K_{e+1} *which is derived from via* $\mathsf{Next}(K_e)$ *if the ciphertext is not already expired, i.e.,* $e_{\mathsf{exp}} \geq e+1$ *must hold for correct decryption in epoch* $e+1$.

More formally, for all $\lambda \in \mathbb{N}$, *for* $K_1 \leftarrow \mathsf{Gen}(\lambda)$, *for all* $e \in [e(\lambda) - 1]$, *for all*

$$(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e),$$

for all $M \in \mathcal{M}$, *for all* $e_{\mathsf{exp}} \in [e(\lambda)]$, *for all* $e' \in [e(\lambda)]$, *for all*

$$C_{e',e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_{e'}, M, e_{\mathsf{exp}}),$$

we require that $M = \mathsf{Dec}(K_{e'}, C_{e',e_{\mathsf{exp}}})$ *holds if* $e_{\mathsf{exp}} \geq e'$.

Moreover, for all $j \in \{0, 1, \ldots\}$, *for all*

$$C'_{j+1,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_{e+1}, C'_{j,e_{\mathsf{exp}}})$$

where $C'_{0,e_{\mathsf{exp}}} = C_{e,e_{\mathsf{exp}}}$, *we have that* $M = \mathsf{Dec}(K_{e'}, C'_{e',e_{\mathsf{exp}}})$ *holds if* $e_{\mathsf{exp}} \geq e'$.

Security notion. We particularly consider the attack that an adversary can downgrade keys using tokens. This is because one could build an UE scheme that has *no* forward-directional but can have backward-directional key updates and this would still be secure in the model of Boyd et al. and Jiang (cf. [Nis21]). Since we want ciphertexts that can expire, token in the expiry epoch should not be of help to downgrade a key from the next epoch.

We will dub our CPA security notion with expiry epochs EE-IND-UE-CPA. Essentially, it ensures that fresh and updated ciphertexts are indistinguishable even if the adversary has access to keys and update tokens. (In particular, we want a strong and natural CPA notion capturing all attacks we described.) We say that an UE scheme is EE-IND-UE-CPA-secure if any PPT adversary A succeeds in the following experiment only with probability negligibly larger than $1/2$.

The experiment starts by computing the initial secret key $K_1 \leftarrow \mathsf{Gen}(\lambda)$ and computes all secret keys and update tokens by iteratively invoking $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$, for $e \in [\lfloor e(\lambda) \rfloor - 1]$. During the experiment, the adversary has access to an encryption oracle Enc'. The adversary first outputs the target epoch e^*, the "firewall" epoch e_{end}, as well as a target ciphertext's expiry epoch e_{exp}. Furthermore, the experiment tosses a coin $b \leftarrow \{0,1\}$ and returns b if the the target epoch is not within the firewall boundaries, i.e., $e_{\mathsf{end}} \leq e^*$, or if the challenge ciphertext expires before the target epoch, i.e., $e_{\mathsf{exp}} < e^*$.

During the experiments, the adversary receives *all* keys except for the epochs $e^*, \ldots, \min(e_{\mathsf{end}}, e_{\mathsf{exp}})$ where the adversary has access to the challenge ciphertext from e^* onwards (to avoid trivial wins). Moreover, it receives *all* update tokens with one exception, namely if $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$, then it does not receive the token for e_{end}.

At some point, the adversary outputs a target message and a target ciphertext for any prior-challenge epoch $\tilde{e} < e^*$ with associated expiry epoch e_{exp}. If the target ciphertext was not queried in any epoch prior to the challenge epoch, then the experiment outputs bit b.[11] If the bit $b = 0$, then the experiment encrypts the message which yields a fresh challenge ciphertext with expiry tag e_{exp} for target epoch e^*; otherwise, the experiments updates the adversarial target ciphertext to the current target epoch e^*. The resulting challenge ciphertext is send to the adversary and it eventually outputs a guess b' and succeeds if $b = b'$. More formally:

[10] For any polynomial integer e in the security parameter λ.
[11] This enforces that we only deal with honestly generated ciphertexts that can be tracked as in previous UE models, particularly as in [LT18, BDGJ20, Jia20].

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

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

is negligible in λ, where $\mathsf{Exp}^{\text{ee-ind-ue-cpa}}_{\mathsf{UE},A}$ is defined as in Figure 6.

```
Experiment $\mathsf{Exp}^{\text{ee-ind-ue-cpa}}_{\mathsf{UE},A}(\lambda)$                Explanation
  $K_1 \leftarrow \mathsf{Gen}(\lambda)$                                   // Initial key
  for $e \in [\lfloor e(\lambda)\rfloor - 1]$ do
    $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$                  // All keys and tokens
  $\mathcal{L} := \emptyset, b \leftarrow \{0,1\}$                          // Initialize list and bit
  $(e^*, e_{\mathsf{end}}, e_{\mathsf{exp}}, \mathtt{st}) \leftarrow A^{\mathsf{Enc}'}(\lambda)$                 // Target, end, and expiry epochs
  if $e_{\mathsf{end}} \le e^*$ or $e_{\mathsf{exp}} < e^*$
    then return $b$                                       // Abort if not correctly distributed
  $\vec{K}_e := (K_e)_{e \in [\lfloor e(\lambda)\rfloor] \setminus \{e^*,\dots,\min(e_{\mathsf{end}},e_{\mathsf{exp}})\}}$   // All keys
  if $e_{\mathsf{exp}} \ge e_{\mathsf{end}}$ then
    $\vec{\Delta}_{e+1} := (\Delta_{e+1})_{e \in [\lfloor e(\lambda)\rfloor - 1] \setminus \{e_{\mathsf{end}}\}}$     // All tokens except the one for $e_{\mathsf{end}}$
  else
    $\vec{\Delta}_{e+1} := (\Delta_{e+1})_{e \in [\lfloor e(\lambda)\rfloor - 1]}$                // All tokens
  $(M^*, C^*_{\tilde{e},\tilde{e}_{\mathsf{exp}}}, \mathtt{st}) \leftarrow A^{\mathsf{Enc}'}(\vec{K}_e, \vec{\Delta}_{e+1}, \mathtt{st})$        // Run $A$ on keys and tokens
  if $(\tilde{e}, C^*_{\tilde{e},\tilde{e}_{\mathsf{exp}}}) \notin \mathcal{L} \vee \tilde{e} \ge e^* \vee \tilde{e}_{\mathsf{exp}} \ne e_{\mathsf{exp}}$
    then return $b$                                       // Abort if $C^*_{\tilde{e},\tilde{e}_{\mathsf{exp}}}$ is invalid
  for $e = \tilde{e}+1, \dots, e^*$ do
    $C^*_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_e, C^*_{e-1,e_{\mathsf{exp}}})$            // Update to pre-target epoch
  $C^*_0 \leftarrow \mathsf{Enc}(K_{e^*}, M^*, e_{\mathsf{exp}})$                     // Fresh ciphertext
  $C^*_1 := C^*_{e^*,e_{\mathsf{exp}}}$                                    // Updated ciphertext
  $b' \leftarrow A^{\mathsf{Enc}'}(C^*_b, \mathtt{st})$                              // Run $A$ on challenge ciphertext
  if $b = b'$ then return 1 else return 0                 // Check winning condition
```

Encryption oracle
$\mathsf{Enc}'(e, M, e_{\mathsf{exp}})$: on input epoch $e \in [\lfloor e(\lambda)\rfloor]$, message $M \in \mathcal{M}$, and expiry epoch e_{exp}, returns ciphertext $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$ and sets $\mathcal{L} := \mathcal{L} \cup (e, C_{e,e_{\mathsf{exp}}})$.

Fig. 6. The EE-IND-UE-CPA security notion for UE.

On the adversarially chosen target, firewall end, and expiry epochs. We let the adversary output the target epoch e^*, the firewall end epoch e_{end}, and the expiry epoch e_{exp} before seeing any keys and tokens (however, it can query ciphertexts). This is done to simplify the description of the model and is asymptotically equivalent to the adaptive notions in [LT18, BDGJ20, Jia20] due to the fact that epochs evolve sequentially and, hence, only polynomially many epochs and a *single* target epoch exist. We argue that any successful adversary in adaptive UE setting can be transformed in a successful adversary in our UE setting generically with a reduction loss polynomial in the security parameter.

Such simple reduction would guess the target epoch e^*, the firewall epoch e_{end}, and the expiry-epoch e_{exp} for the adaptive UE adversary while outputting $e^*, e_{\mathsf{start}}, e_{\mathsf{exp}}$ to our challenger. This challenger would generate all keys and tokens and send the respective values to the reduction which are used to answer the oracle queries by the adaptive adversary. If the adaptive adversary decides to query a challenge in e^*, we forward the challenge ciphertext to the adaptive adversary. Note that the reduction is also capable of answering requested update tokens until e_{end} for our setting as well as requests for further keys and update tokens until $\lfloor e(\lambda)\rfloor$. Encryption queries from the adaptive adversary can be answered directly since the reduction has all keys for all epochs.

If the adaptive UE adversary outputs its guess, then the reduction returns this guess to the our challenger. In any other cases, the reduction outputs a random bit $b' \leftarrow \{0,1\}$. Now, if the adaptive adversary is successful with probability ε, then the reduction yields a successful adversary in our sense and it runs in PPT. Since $\lfloor e(\lambda)\rfloor$ is polynomially bounded, we have that the success probability of the reduction is bounded by $\varepsilon/e(\lambda)^3$ which is significant.

How forward and post-compromise security is modeled. As already implicit in Jiang [Jia20], via forward-directionality in key-updates, we achieve strong post-compromise security. By requiring in our model that a current-epoch key cannot be downgraded and update tokens cannot be used to downgrade ciphertexts, together with expiry epochs allows us to achieve strong forward security in particular when downgrading goes below the expiry epoch.

5 Updatable Encryption with No-Directional Key Updates and Expiry Epochs from Ciphertext Puncturable Encryption

In this section, we introduce a novel primitive dubbed Ciphertext Puncturable Encryption (CPE), provide a CPE security model, and show how one can construct UE with no-directional key updates and expiry epochs from CPE. Essentially, CPE views our strong variant of UE from a puncturable-encryption perspective.

5.1 (Symmetric) ciphertext puncturable encryption

Ciphertext Puncturable Encryption (CPE) augments plain Puncturable Encryption (PE) [GM15, GHJL17, DJSS18, DKL+18, SSS+20, SDLP20] to also support "puncturing" of ciphertexts instead of puncturing keys only. We recall that despite being slightly different in their concrete formulation (e.g., allowing single or multiple tags per ciphertext), existing PE schemes all provide the same basic idea in their functionality, i.e., that they allow to puncture (secret) keys in a way that they can no longer decrypt certain ciphertexts. The distinguishing feature of our CPE is that puncturing on keys is *synchronized* with ciphertext puncturings. Notable, we define CPE as a symmetric-key primitive extending the definitions of PE used in prior work [SYL+18, AGJ19, AGJ21, BDdK+21].

A CPE scheme has the syntax of a symmetric PE scheme, i.e., consisting of the PPT algorithms Gen, Enc, Dec, KPunc, but has an additional *ciphertext-puncturing* algorithm CPunc. In plain symmetric PE, only keys are punctured on any sequence of tags. In our PE variant, ciphertexts can be additionally punctured on any sequence of tags. The correctness now guarantees that a key can decrypt a ciphertext as long as there is at least one tag from an ordered tagspace $\mathcal{T} = \{t_1, t_2, t_3, \ldots\}$ on which neither the key K nor the ciphertext C have been punctured. For example, if K is punctured on tag t_1 and C is punctured on t_1, t_2, then decryption succeeds since neither K nor C was punctured on t_3 for example. This may sound counter-intuitive (essentially because as it seems that one can always decrypt if the tag space is large enough), but as we will see next via introducing expiry tags it exactly serves our desired purpose (and is indeed different to plain PE).

More precisely, a simple translation of key puncturing of plain PE to ciphertext puncturing in CPE introduces the problem that ciphertexts can be decrypted at any time, particularly, if the tag space is of exponential size. Essentially, in PE one starts with an unpunctured key and is able to add tags from an exponentially large tag space to that key which translate to ciphertext puncturing in the sense that the ciphertext is now valid for many keys when crafted. To mitigate this, we introduce an "expiry tag" for ciphertexts such that after all tags smaller or equal to this tag are punctured in the ciphertext, the corresponding key cannot decrypt this tagged ciphertext anymore. Note that below we use the convention that if no expiry tag $t_{\mathsf{exp}} < \max(\mathcal{T})$ is desired for Enc, we simply treat the supplied tag as $t_{\mathsf{exp}} = \max(\mathcal{T})$.

Ciphertext puncturing is not a fully public operation but requires some puncture token which is obtained from the respective key puncturing. Hence, we have a "coupled" synchronization between key and ciphertext puncturing via a puncture token, e.g., Δ_t that is used for ciphertext puncturing on tag t to make it consistent with puncturings on secret keys.

We now define CPE with its correctness and security notions.

Definition 3 (Ciphertext Puncturable Encryption). *A Ciphertext Puncturable Encryption (CPE) scheme* CPE *with ordered tag space* $\mathcal{T}$ *of polynomial size in the security parameter* λ *and message space* $\mathcal{M}$ *consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{CPunc}, \mathsf{Dec})$*:*

$\mathsf{Gen}(\lambda)$: on input security parameter λ, outputs initial key K.

$\mathsf{KPunc}(K, t)$: on input key K and tag $t \in \mathcal{T}$, outputs a punctured key K' and a puncture token Δ_t.

$\mathsf{Enc}(K, M, t_{\mathsf{exp}})$: on input key K, message $M \in \mathcal{M}$, and expiry tag $t_{\mathsf{exp}} \in \mathcal{T}$, outputs a ciphertext C.

$\mathsf{CPunc}(C, \Delta_t)$: on input ciphertext C and puncture token Δ_t, outputs a punctured ciphertext C'.

$\mathsf{Dec}(K, C)$: on input key K and ciphertext C, outputs message $M \in \mathcal{M}$ or $\perp$.

Correctness. Correctness ensures that for all punctured keys, the ciphertexts under the corresponding key are successfully decryptable as long as not all tags smaller or equal than the expiry tag of those ciphertexts are punctured from the key.

More concretely, for all $\lambda, n \in \mathbb{N}$, for all $K_\varepsilon \leftarrow \mathsf{Gen}(\lambda)$, for all $\{t_j\}_{j\in[n]} \in (\mathcal{T})^n$ and any (arbitrary interleaved) sequence of invocations of

$$(K_j, \varDelta_{t_j}) \leftarrow \mathsf{KPunc}(K_{j-1}, t_j),$$

where $j \in [n]$ and $K_0 := K_\varepsilon$, for all $M \in \mathcal{M}$, for all $\{t_{\mathsf{exp},i}\}_{i\in[n]} \in (\mathcal{T})^n$ with $t_{\mathsf{exp},i} \geq \max(t_1, \ldots, t_n)$, for all $C_i \leftarrow \mathsf{Enc}(K_i, M, t_{\mathsf{exp},i})$, we require that $\mathsf{Dec}(K_i, C_i) = M$, for $i \in [n] \cup \{0\}$ holds.

Furthermore, for any (arbitrary interleaved) sequence of invocations of

$$C'_j \leftarrow \mathsf{CPunc}(C'_{j-1}, \varDelta_{t_j}),$$

where $j \in \{j', j'+1, \ldots, n\}$ for $j' \leftarrow [n] \cup \{0\}$, $C'_{j-1} := C_{j-1}$, we require that $\mathsf{Dec}(K_j, C'_j) = M$ holds.

Intuition of our security notions for CPE. Compared to plain PE, we now use puncture tokens not present in PE which introduces higher burdens on the security guarantees. See that such puncture tokens can potentially be used to also puncture keys or even unpuncture keys and/or ciphertexts. In that vein, we define an indistinguishability-based notion, dubbed $\mathsf{IND\text{-}CPE\text{-}CPA}$, which guarantees that freshly generated ciphertexts cannot be distinguished from punctured ones. Of course, the adversary must not trivially win the game, hence, we also require that no tags up to the expiry tag of the challenge ciphertext are left in keys and ciphertexts. In particular, we only want to allow puncturing of ciphertexts via the puncture token, but keys must not be eligible to being punctured/unpunctured via tokens. This carried out via a key-puncturing oracle which the adversary can query to retrieve punctured keys and tokens.

$\mathsf{IND\text{-}CPE\text{-}CPA}$ security. We say that a CPE scheme is $\mathsf{IND\text{-}CPE\text{-}CPA}$-secure if any PPT adversary succeeds in the following experiment only with probability negligibly larger that $1/2$.

The experiment starts by computing the initial key $K_\varepsilon \leftarrow \mathsf{Gen}(\lambda)$. The key is not punctured as indicated by ε and will be punctured on tags via an oracle KPunc' during the experiment. During the entire experiment, the adversary has access to encryption oracle Enc'. Let $K_{\mathbf{t}}$ be the key associated to the (currently largest) punctured-tag set $\mathbf{t}$. As in previous PE schemes, we define the notion in a tag-selective setting, i.e., the adversary in the initialization phase has to first commit to a target tag t^*, expiry tag t^*_{exp}, and a "window" tag t^*_{win}. With this window tag, we want to allow the adversary to puncture the challenge ciphertext (on ordered tags $t^*, \ldots, t^*_{\mathsf{win}}$) without being able to trivially decrypt it; t^*_{exp} denotes the expiry tag of the target and challenge ciphertexts. After the initialization phase, the adversary then has fully adaptive access to a KPunc'-oracle. Notably, the adversary retrieves K_ε in the initialization phase.

At some point, the adversary outputs a target message M^* and a target ciphertext $C^*_{\mathbf{t},t^*_{\mathsf{exp}}}$ that was not punctured on t^* yet. The experiment iteratively proceeds with puncturing the current key on all tags $t_j \leq t^*$ to retrieve $(K_j, \varDelta_{t_j})$. Furthermore, the experiment tosses a coin b. If $b = 0$, then it computes a fresh encryption C_0 of the target message M^* under the latest punctured key using Enc and expiry tag t^*_{exp}; otherwise, if $b = 1$, then it computes a punctured ciphertext C_1 using CPunc with $\varDelta_{t^*}$. The adversary eventually outputs a guess b' where the experiment returns 1 if $b' = b$. Figure 7 depicts the experiment.

Definition 4 ($\mathsf{IND\text{-}CPE\text{-}CPA}$ security). *A CPE scheme* CPE *is* $\mathsf{IND\text{-}CPE\text{-}CPA}$*-secure iff for any valid 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 follows:*

5.2 UE with no-directional key updates and expiring ciphertexts from CPE

In this section, we construct updatable encryption with no-directional key updates and expiry epochs from ciphertext puncturable encryption. We can see that UE is essentially a sequenced

```
Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda)$          Explanation
  $K_\varepsilon \leftarrow \mathsf{Gen}(\lambda)$                                      // Initial key
  $b \leftarrow \{0,1\}$, $\mathcal{L} := \emptyset$, $\mathbf{t} := \emptyset$
  $(t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}, \mathtt{st}) \leftarrow A^{\mathsf{Enc}'}(\lambda, K_\varepsilon)$        // Retrieve target, window, and expiry tags
  if $t^*_{\mathsf{win}} \leq t^*$ or $t^*_{\mathsf{exp}} < t^*$                        // Abort if not correctly distributed
    then return $b$
  $(M^*, C^*_{\mathbf{t}',t'_{\mathsf{exp}}}, \mathtt{st}) \leftarrow A^{\mathsf{Enc}',\mathsf{KPunc}'}(\mathtt{st})$        // Retrieve targets
  if $(\mathbf{t}, C^*_{\mathbf{t}',t'_{\mathsf{exp}}}) \notin \mathcal{L}$ or $t'_{\mathsf{exp}} \neq t^*_{\mathsf{exp}}$
    then return $b$                                            // Abort if not correctly distributed
  $K_0 := K_{\mathbf{t}}, C_0 := C^*_{\mathbf{t}',t'_{\mathsf{exp}}}$
  For all $(t_1, t_2, \ldots, t_n)$ with $t_j \notin \mathbf{t}$ and $t_j \leq t^*$,   // Puncture all tags $\leq t^*$ to avoid trivial wins
    $(K_{t_j}, \Delta_{t_j}) \leftarrow \mathsf{KPunc}(K_{t_{j-1}}, t_j)$
    $C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \Delta_{t_j})$
    $K_{\mathbf{t}} := K_{t_j}, \mathbf{t} := \mathbf{t} \cup \{t_j\}$
  $C_0 \leftarrow \mathsf{Enc}(K_{\mathbf{t}}, M^*, t^*_{\mathsf{exp}})$                   // Fresh challenge ciphertext
  $C_1 := C_n$                                                 // Punctured challenge ciphertext
  $b' \leftarrow A^{\mathsf{Enc}',\mathsf{KPunc}'}(C_b, (K_{t_j})_{t_j \neq t^*}, (\Delta_{t_j})_j, \mathtt{st})$     // Run $A$ on challenge
  if $b = b'$ then return 1 else return 0                      // Check winning condition
```

Encryption oracle

$\mathsf{Enc}'(M, t_{\mathsf{exp}})$: on input M and t_{exp}, compute $C_{\mathbf{t},t_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_{\mathbf{t}}, M, t_{\mathsf{exp}})$, set $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t}, C_{\mathbf{t},t_{\mathsf{exp}}})\}$. Return $C_{\mathbf{t},t_{\mathsf{exp}}}$.

Key puncturing oracle

$\mathsf{KPunc}'(t)$: on input t, if $t \in \mathbf{t}$, return $\perp$. Otherwise, compute $(K'_{\mathbf{t}\cup\{t\}}, \Delta'_t) \leftarrow \mathsf{KPunc}(K_{\mathbf{t}}, t)$, set $K_{\mathbf{t}} := K'_{\mathbf{t}\cup\{t\}}$ and $\mathbf{t} := \mathbf{t} \cup \{t\}$.
- If $t \in \{t^*, \ldots, \min(t^*_{\mathsf{exp}}, t^*_{\mathsf{win}})\}$ then set $K'_{\mathbf{t}} := \perp$, else set $K'_{\mathbf{t}} := K_{\mathbf{t}}$.
- If $t^*_{\mathsf{exp}} \geq t^*_{\mathsf{win}}$ and $t = t^*_{\mathsf{win}}$ then set $\Delta'_t := \perp$.
- Return $(K'_{\mathbf{t}}, \Delta'_t)$.

Fig. 7. The IND-CPE-CPA security notion for CPE.

version of CPE, i.e., in the speak of CPE, we only have ordered tags which we map to epochs sequentially. Particularly, we use key puncturing for generating the next UE key (i.e., puncturing the old-epoch key) and use ciphertext puncturing for updating ciphertexts (i.e., puncturing old-epoch ciphertexts). Encryption and decryption directly map to UE's encryption and decryption, respectively. More concretely, let $\mathsf{CPE} = (\mathsf{CPE.Gen}, \mathsf{CPE.KPunc}, \mathsf{CPE.Enc}, \mathsf{CPE.CPunc}, \mathsf{CPE.Dec})$ with message space $\mathcal{M}_{\mathsf{CPE}}$ and tag space $\mathcal{T} = \{1, 2, \ldots, \lfloor e(\lambda) \rfloor\}$ be a CPE scheme. We present our UE scheme $\mathsf{UE} = (\mathsf{Gen}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} := \mathcal{M}_{\mathsf{CPE}}$ in Figure 8 and further show correctness as well as EE-IND-UE-CPA security.

$\mathsf{Gen}(\lambda)$: compute $K_\varepsilon \leftarrow \mathsf{CPE.Gen}(\lambda)$ and $(K_1, \Delta_1) \leftarrow \mathsf{CPE.KPunc}(K_\varepsilon, 1)$, and return K_1.

$\mathsf{Next}(K_e)$: return
$$(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{CPE.KPunc}(K_e, e+1).$$
$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: return
$$C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(K_e, M, e_{\mathsf{exp}}).$$
$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$: return
$$C_{e+1,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.CPunc}(C_{e,e_{\mathsf{exp}}}, \Delta_{e+1}).$$
$\mathsf{Dec}(K_e, C_e)$: return
$$M := \mathsf{CPE.Dec}(K_e, C_{e,e_{\mathsf{exp}}}).$$

Fig. 8. Construction of UE from CPE.

For correctness, 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 match and the ciphertext is not expired.

Theorem 1. *If* CPE *is* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *secure, then* UE *is* $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ *secure. Concretely, for any PPT adversary A there is a distinguisher D in the* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *security experiment, such that* $\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},D}(\lambda) \geq \mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$.

Proof. We show the Theorem by constructing a PPT distinguisher D in the $\mathsf{IND\text{-}CPE\text{-}CPA}$ security experiment with CPE as defined in Figure 7 from any successful PPT adversary A in the $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ security with UE as defined in Figure 6.

Description of reduction. D starts A and receives $(e^*, e_{\mathsf{end}}, e_{\mathsf{exp}})$. If the A-input does not have the right distribution as defined in Figure 6, then output $b \leftarrow \{0,1\}$. D then outputs

$$(t^* := e^*, t^*_{\mathsf{win}} := e_{\mathsf{end}}, t^*_{\mathsf{exp}} := e_{\mathsf{exp}})$$

to its $\mathsf{IND\text{-}CPE\text{-}CPA}$ challenger. Then, D queries its key-puncture oracle $(\{K_e, \bot\}, \{\Delta_e, \bot\}) \leftarrow \mathsf{KPunc}'(K_{e-1}, e)$ (with $K_0 = K_\varepsilon$), for all $e \in [\lfloor e(\lambda) \rfloor]$, from its $\mathsf{IND\text{-}CPE\text{-}CPA}$ challenger. See that D *does not* receive keys K_e for $e = e^*, \ldots, \min(e_{\mathsf{end}}, e_{\mathsf{exp}})$ and no update token $\Delta_{e_{\mathsf{end}}+1}$ for e_{end} if $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$. D sets $\Delta_1 := \bot$. If $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$, then D sends

$$\vec{K}_e = (K_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e^*, \ldots, e_{\mathsf{end}}\}} \text{ and } \vec{\Delta}_{e+1} = (\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{end}}\}}$$

to A. Otherwise, i.e., if $e_{\mathsf{exp}} < e_{\mathsf{end}}$, then D sends to A:

$$\vec{K}_e = (K_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e^*, \ldots, e_{\mathsf{exp}}\}} \text{ and } \vec{\Delta}_{e+1} = (\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor]}.$$

During the reduction, encryption queries in epoch $e \in [\lfloor e(\lambda) \rfloor]$ to $\mathsf{Enc}'(M, e'_{\mathsf{exp}})$ with expiry epoch e'_{exp} are forwarded to D's encryption oracle which returns $C_{e,e'_{\mathsf{exp}}} \leftarrow \mathsf{Enc}'(M, e'_{\mathsf{exp}})$ and set $\mathcal{L} := \mathcal{L} \cup \{(e, C_{e,e'_{\mathsf{exp}}})\}$. D receives $(M^*, C^*_{\tilde{e},e_{\mathsf{exp}}})$ from A and checks if $(\tilde{e}, C^*_{\tilde{e},e_{\mathsf{exp}}}) \in \mathcal{L}$ and returns $b \leftarrow \{0,1\}$ if not. If $\tilde{e} < e^* - 1$, then iteratively run

$$C^*_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(C^*_{e-1,e_{\mathsf{exp}}}, \Delta_e),$$

for $e = \tilde{e} + 1, \ldots, e^* - 1$. D forwards $(M^*, C^*_{e^*-1,e_{\mathsf{exp}}})$ to its $\mathsf{IND\text{-}CPE\text{-}CPA}$ challenger. D receives $(C_b, \cdot, \Delta_{e^*})$ and forwards C_b to A. Eventually, A outputs a guess b' which is forwarded to D's challenger.

Analysis of reduction. The $\mathsf{IND\text{-}CPE\text{-}CPA}$ distinguisher D is able to provide a consistent view for $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ adversary A for keys $(K_e)_e$. Enc'-answers also yield a consistent view for A, for all $e \in [\lfloor e(\lambda) \rfloor]$. Now, if A is a successful PPT adversary in the $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ security experiment with UE, then D is a successful PPT adversary in the $\mathsf{IND\text{-}CPE\text{-}CPA}$ security experiment with CPE which shows the Theorem. □

6 CPE from *d*-Lin from Standard Assumptions

We recall that due to Günther et al. [GHJL17], we know that we can obtain Puncturable Encryption (PE) in bilinear groups and in particular from any hierarchical any identity-based encryption (HIBE) scheme [HL02, GS02, BBG05]. However, for our CPE construction which also allows "delegation" of ciphertexts, basing it on plain HIBE (or BTE as a relaxation) techniques does not work. But as we have discussed in Section 2, we can utilize underlying building blocks used to prove strong security of HIBEs, namely the dual system group (DSG) abstraction due to Chen-Wee and Gong et al. [CW13, GCTC16] which can be instantiated from the standard d-Lin assumption in prime-order bilinear groups in the standard model. Interestingly, the dual-system approach provides us with elements we need for puncturing ciphertext elements and to prove security quite conveniently. The richness of such dual-system proof paradigm was demonstrated in several prior works already (e.g., [Wat09, LW10, LW11, OT12, CW13, HKS15, AHY15, GCD+16, GCTC16, GWW19, GW20]).

The concept of (Extended) Dual System Groups (EDSGs) [CW13, GCTC16] is particularly useful to prove BTE/HIBE schemes strongly secure. The main observation for our intentions is that plain EDSG also provides elements that we can use for puncturing ciphertexts. In their HIBEs-from-EDSG work [GCTC16], Gong et al. simply did not need such elements (as HIBEs/BTEs do not provide such a feature), but as it turns out, those elements are of central interest for us to allow puncturing of ciphertexts and beyond. To look ahead, a puncture token will be a uniform BTE key (determined by the tag to be punctured on) under a uniform main secret key without delegating features and will only work for updating the intended ciphertext which yields the desired functionality (since such tokens cannot update keys).

On a specific note it is important to say that we only have to deal with *one* challenge ciphertext (that can be punctured with respective tags adaptively) and *one* challenge secret key (that also can be punctured with respective tags adaptively) which allows us to simplify the underlying dual-system instantiation in the sense that we do not need the "full power" thereof. This has the benefit that the proof can be adapted quite straightforwardly to our setting (nevertheless, we carefully examine it in this work). All in all, this broadens the use of the dual-system paradigm to further application domains such as puncturable and updatable encryption.

6.1 Dual system groups

We will now recap EDSGs and, in particular, we only need a relaxed version of it and not all features of the EDSG from [GCTC16]. We want emphasize that we only left out not necessary 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 EDSG version of [GCTC16]. Our relaxed EDSG EDSG consists of the PPT algorithms $(\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \mathsf{SampS}, \mathsf{SampK}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$:

$\mathsf{SampP}(\lambda, n)$: parameter sampling, given security parameter λ and parameter $n \in \mathbb{N}$, samples $(\mathbb{G}, \mathbb{H}, G_T, N, (g_{p_i})_{i\in[n']}, e) \leftarrow \mathsf{G}(\lambda, n')$, for n' determined in SampP, and outputs public parameters $pp = (\mathbb{G}, \mathbb{H}, G_T, N, e, m, pars)$ and secret parameters $sp = (\widehat{h}, \widehat{pars})$, where $m : \mathbb{H} \to \mathbb{G}_T$ is a linear map, $\widehat{h} \neq 1$ is an element of the group generated by h_s (see $\widehat{\mathsf{SampH}}$) and g_s is an element of the group generated by $\widehat{g} \neq 1$ (see $\widehat{\mathsf{SampG}}$), and $pars, \widehat{pars}$ may contain arbitrary information.

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

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

$\mathsf{SampS}(pp)$: given pp, outputs $S \in \mathbb{G}$.

$\mathsf{SampK}(pp)$: given pp, outputs $K \in \mathbb{H}$.

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

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

Correctness of EDSG. For correctness, for all $\lambda \in \mathbb{N}$, for all integers $n = n(\lambda) > 1$, for all pp, where pp is the first output of $\mathsf{SampP}(\lambda, n)$, we require:

Projective. For all $s \leftarrow \mathbb{Z}_N^*$, for all $h \in \mathbb{H}$, we have $m(h)^s = e(\mathsf{SampS}(pp; s), h)$.

Orthogonality. For all $(h_0, \ldots, h_n) \in \mathbb{H}^{n+1} \leftarrow \mathsf{SampH}(pp)$ and $S \leftarrow \mathsf{SampS}(pp)$, we require $e(S, h_i) = 1$, for all $i \in [n]$. For all $(g_0, \ldots, g_n) \in \mathbb{H}^{n+1} \leftarrow \mathsf{SampH}(pp)$ and $K \leftarrow \mathsf{SampS}(pp)$, we require $e(g_0, K) = 1$.

Associativity. For all $(g_0, \ldots, g_n) \leftarrow \mathsf{SampG}(pp)$ and for all $(h_0, \ldots, h_n) \leftarrow \mathsf{SampH}(pp)$, we require

$$e(g_0, h_j) = e(g_j, h_0),$$

for all $j \in [n]$.

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

Security of EDSG. For security, for all $\lambda \in \mathbb{N}$, for all integers $n = n(\lambda) > 1$, for all $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, n)$, we require:

Orthogonality. For m specified in pp, for $\widehat{h}$ specified in sp, we require $m(\widehat{h}) = 1$. (Note that by the projective property, for S as the output of $\mathsf{SampS}(pp)$, we have $e(S, \widehat{h}) = 1$.)

Non-degeneracy. For any h_s which is the second-to-last output of $\widehat{\mathsf{SampH}}(pp, sp)$, $\widehat{h}$ lies in a subgroup group of h_s. For any g_s which is the last output of $\widehat{\mathsf{SampG}}(pp, sp)$, g_s lies in a subgroup group of $\widehat{g}$.

Left-subgroup indistinguishability (LS). For any PPT adversary D, we have that the function

$$\mathsf{Adv}^{\mathsf{ls}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, n) := |\Pr[D(pp, \mathbf{g}) = 1] - \Pr[D(pp, \mathbf{g}\widehat{\mathbf{g}}) = 1]|$$

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

Right-subgroup indistinguishability (RS). For any PPT adversary D, we have that the function

$$\mathsf{Adv}^{\mathrm{rs}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, n) := |\Pr\left[D(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{h}) = 1\right] - \Pr\left[D(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{h}\widehat{\mathbf{h}}) = 1\right]|$$

is negligible in λ, where $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and $(\widehat{\mathbf{h}}, \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, for $\widehat{h}$ specified in 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, where $(\widehat{\mathbf{g}} = (\widehat{g}_0, \dots, \widehat{g}_n), g_s) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \dots, \widehat{h}_n), h_s, h_a) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $\widehat{\mathbf{g}}' = (1, g_s^{\gamma_1}, \dots, g_s^{\gamma_n})$, and $\widehat{\mathbf{h}}' = (1, h_s^{\gamma_1}, \dots, h_s^{\gamma_n})$, for $\gamma_1, \dots, \gamma_n \leftarrow \mathbb{Z}_N$ and $\widehat{g}, \widehat{h}$ specified in sp.

Computational non-degeneracy (ND). For any PPT adversary D, we have that the function

$$\mathsf{Adv}^{\mathrm{nd}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, n) := |\Pr\left[D(pp, \mathbf{S} \cdot \mathbf{g}\widehat{\mathbf{g}}, K \cdot \widehat{h}^{\alpha}, e(S, K)) = 1\right] - \Pr\left[D(pp, \mathbf{S} \cdot \mathbf{g}\widehat{\mathbf{g}}, K \cdot \widehat{h}^{\alpha}, R = 1\right]|$$

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

Construction of EDSG in prime-order groups. In comparison to the Gong et al. EDSG [GCTC16], we only omitted the 2nd LS and the asymmetric parameter-hiding properties to have a more concise presentation (and we do not need such properties in our instantiation due to the fact that we do not deal with many independent ciphertext instances sharing the same public key, a property Gong et al. need). For completeness, we provide their concrete prime-order instantiation which we denote as EDSG below inSubsection 6.4.We conclude:

Corollary 1. EDSG *is an EDSG under the d-Lin assumption.*

6.2 Helper algorithms

To construct a ciphertext puncturable encryption scheme from extended dual system groups, we implicitly arrange tags of the CPE scheme associated to the key or ciphertext in a complete binary tree, i.e., the nodes represent a prefix bit representation of the tags and the root of the tree is associated with the initial key K_ε or ciphertext $C_{\varepsilon, t_{\mathsf{exp}}}$ of the CPE (as also discussed in the introduction with Figure 5).

We define additional PPT helper algorithms KTrunc and CTrunc within a CPE scheme to prune the tree to output a punctured key-token pair and a punctured ciphertext, respectively, which in turn will correspond to CPE's KPunc and CPunc algorithms, respectively. Encryption is done according to an (already punctured) key where the ciphertext parts are built according to the pruned tree.

Intuition of KTrunc. Essentially, KTrunc takes the current binary-tree configuration as provided in the key (i.e., which tags are already punctured and, hence, how the tree is pruned for such tags analogously to Figure 5 in the motivation section). It receives an input tag t that will be punctured. KTrunc first finds all nodes from the root to the associated leaf of tag t. (Since those nodes can be used to derive a secret key for tag t.) It delegates the key nodes on that path such that no ancestor nodes for t are available anymore and keeps the other key nodes. Furthermore, a puncture token is generated for t. Such puncture token can only be used for ciphertexts that *do not* have t punctured already. The result is a pruned tree configuration that excludes key material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The concrete PPT algorithm works as follows:

$\mathsf{KTrunc}(K_{\mathbf{t}}, t)$: on input key $K_{\mathbf{t}} = (pp, m(msk_{\mathbf{t}}), K_{\mathbf{t},1}, \dots, K_{\mathbf{t},j})$, for some integer $j \in \mathbf{O}(\lambda)$, output punctured key-token pair according to $t = (t_1, \dots, t_\ell) \in \{0,1\}^\ell$ as follows:

1 Find the unique inner or leaf node key element $K_{\mathbf{t},i}$ which is associated to a unique prefix of t. Such unique node always exists, otherwise t would have been punctured already.

2a. If $K_{\mathbf{t},i}$ is associated to an inner node, derive key elements hanging from the path to t by iteratively calling KDel on all prefixes of t starting from the node associated to $K_{\mathbf{t},i}$ and set $K'_{\mathbf{t}} := (pp, m(msk_{\mathbf{t}}), K'_{\mathbf{t},\leq j}, K'_{\mathbf{t},j+1}, K'_{\mathbf{t},j+2}, \dots)$, where $K'_{\mathbf{t},\leq j}$ is the same as in $K_{\mathbf{t}}$, but without $K_{\mathbf{t},i}$, and $K'_{\mathbf{t},j+1}, K'_{\mathbf{t},j+2}, \dots$ are those derived delegated key elements via KDel hanging from the path to t.

2b. If $K_{\mathbf{t},i}$ is associated to a leaf node, then set $K'_{\mathbf{t}} := (pp, m(msk_{\mathbf{t}}), K'_{\mathbf{t},\leq j})$, where $K'_{\mathbf{t},\leq j}$ is the same as in $K_{\mathbf{t}}$, but without the leaf-node associated key element $K_{\mathbf{t},i}$.
3. Sample $\delta \leftarrow \mathsf{SampK}(pp)$ and for all $K'_{\mathbf{t}}$-elements $K'_{\mathbf{t},i} =: (h_0, msk_{\mathbf{t}} \cdots, \ldots)$, compute $K'_{\mathbf{t}\cup\{t\},i} := (h_0, msk_{\mathbf{t}} \cdot \delta \cdots, \ldots)$ and set $K'_{\mathbf{t}\cup\{t\}} := (pp, m(msk_{\mathbf{t}} \cdot \delta), (K'_{\mathbf{t}\cup\{t\},i})_i)$
4. Sample $(h_0, \ldots, h_{2\ell}) \leftarrow \mathsf{SampH}(pp)$ and compute $\Delta_t := (h_0, \delta \cdot \prod_{i=1}^{\ell} h_{2i-t_i}, m(\delta \cdot msk_{\mathbf{t}}))$.
5. Re-randomize $K'_{\mathbf{t}\cup\{t\}}$ to obtain $K''_{\mathbf{t}\cup\{t\}} \leftarrow \mathsf{KRerand}(K'_{\mathbf{t}\cup\{t\}})$.
5. Output $(K''_{\mathbf{t}\cup\{t\}}, \Delta_t)$.

Intuition of CTrunc. Essentially, CTrunc works analogously to KPunc, but for ciphertexts. It takes the current tree configuration as provided in the ciphertext (i.e., which tags are already punctured and, hence, how the tree is pruned for such tags). It further receives an input tag t that will be punctured with the help of a puncture token Δ_t. CTrunc first finds all elements from the root to the associated leaf of tag t. (Since those elements can be used to derive a potentially decryptable ciphertext element for tag t.) It delegates the ciphertext elements on that path such that no ancestor elements for t are available anymore and keeps the other ciphertext elements. The result is a pruned tree configuration that excludes ciphertext material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The PPT algorithm works as follows:

$\mathsf{CTrunc}(C_{\mathbf{t},t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$: on input ciphertext $C_{\mathbf{t},t_{\mathsf{exp}}} = ((C_{\mathbf{t},i})_{i\in[j]}, \ldots, m(msk_{\mathbf{t}})^s \cdot M))$, for some integer $j \in \mathbf{O}(\lambda)$, and $\Delta_t = (\Delta_0, \Delta_1) = (h_0, \delta \cdot \prod_{i=1}^{\ell} h_{2i-t_i}, \cdot msk_{\mathbf{t}})$ output a punctured ciphertext as follows:

1 Find the unique inner or leaf node ciphertext element $C_{\mathbf{t},i}$ which is associated to a unique prefix of t. Such unique node always exists, otherwise t would have been punctured already.
2a. If $C_{\mathbf{t},i}$ is associated to an inner node, derive delegated ciphertexts hanging from the path to t by iteratively calling CDel on all prefixes of t starting from the node associated to $C_{\mathbf{t},i}$ and set $C'_{\mathbf{t}} := (C'_{\mathbf{t},\leq j}, C'_{\mathbf{t},j+1}, C'_{\mathbf{t},j+2}, \ldots)$, where $C'_{\mathbf{t},\leq j}$ is the same as $C_{\mathbf{t},t_{\mathsf{exp}}}$, but without $C_{\mathbf{t},i}$, and $C'_{\mathbf{t},j+1}, C'_{\mathbf{t},j+2}, \ldots$ are those ciphertext elements derived via CDel hanging from the path to t.
2b. If $C_{\mathbf{t},i}$ is associated to a leaf node, then set $C'_{\mathbf{t}} := C'_{\mathbf{t},\leq j}$, where $C'_{\mathbf{t},\leq j}$ is the same as $C_{\mathbf{t},t_{\mathsf{exp}}}$, but without the leaf associated ciphertext $C_{\mathbf{t},i}$.
3. Let $C_{\mathbf{t},i} = (C_0, C_1, \ldots) = (S \cdot g_0, \prod_{j=1}^{\ell} g_{2j-t_j}, \ldots)$ be the ciphertext associated to t found in the previous step. Compute

$$e(S, \delta) := \frac{e(C_0, \Delta_1)}{e(C_1, \Delta_0)} = \frac{e(S \cdot g_0, \delta \cdot \prod_{j=1}^{\ell} h_{2j-t_j})}{e(\prod_{j=1}^{\ell} g_{2j-t_j}, h_0)} \tag{1}$$

and $m(msk_{\mathbf{t}})^s \cdot M \cdot e(S, \delta) \cdot M) = e(S, msk_{\mathbf{t}}) \cdot e(S, \delta) \cdot M) = m(msk_{\mathbf{t}} \cdot \delta)^s \cdot M$.
4. Set $C'_{\mathbf{t}\cup\{t\},t_{\mathsf{exp}}} := ((C''_{\mathbf{t}\cup\{t\},i})_i, e(S', msk_{\mathbf{t}} \cdot \delta) \cdot M)$, for fresh $S' = S \cdot S''$ with $S'' \leftarrow \mathsf{SampS}(pp; s'')$ using $m(\delta \cdot msk_{\mathbf{t}})^{s''} = e(S'', \delta \cdot msk_{\mathbf{t}})$ for uniform $s'' \leftarrow \mathbb{Z}_N$. Furthermore, use S'' to re-randomize the first components of $C''_{\mathbf{t}\cup\{t\},i})_i$ to derive $(C''_{\mathbf{t}\cup\{t\},t_{\mathsf{exp}}} := (C''_{\mathbf{t}\cup\{t\},i})_i, e(S', msk_{\mathbf{t}} \cdot \delta) \cdot M)$.
5. Re-randomize all elements in $C''_{\mathbf{t}\cup\{t\},t_{\mathsf{exp}}}$ by computing $C'''_{\mathbf{t}\cup\{t\}} \leftarrow \mathsf{CRerand}(C''_{\mathbf{t}\cup\{t\},t_{\mathsf{exp}},i})$.
6. Output $C'''_{\mathbf{t}\cup\{t\},t_{\mathsf{exp}}}$.

Intuition of KDel, $\mathsf{KRerand}$, CDel, SCDel and $\mathsf{CRerand}$. Essentially, KDel delegates secret key material as done in binary-tree encryption key delegation where $\mathsf{KRerand}$ re-randomizes the key material. CDel delegates ciphertext material as done in the ciphertext delegation for normal ciphertext (as discussed in the motivation section) where $\mathsf{CRerand}$ re-randomizes the ciphertext material. The concrete PPT algorithms work as follows:

$\mathsf{KDel}(K_{\mathbf{t},i}, t_{\ell'})$: on input key $K_{\mathbf{t},i} = (K_0, K, \ldots, K_{2\ell})$ and bit $t_{\ell'}$, output

$$(K_0, K \cdot K_{2\ell'-t_{\ell'}}, K_{2\ell'+1}, \ldots, K_{2\ell}).$$

$\mathsf{KRerand}(K_{\mathbf{t}})$: on input key $K_{\mathbf{t}} = (pp, m(msk_{\mathbf{t}}), (K_{\mathbf{t},i})_i)$ with

$$K_{\mathbf{t},i} = (K_{i,0}, K_i, K_{i,2\ell'+1}, \ldots, K_{i,2\ell}),$$

for all i, sample fresh $(h_0, h_1, \ldots, h_{2\ell}) \leftarrow \mathsf{SampH}(pp)$, and compute

$$K'_{\mathbf{t},i} := (K_{i,0} \cdot h_0, K_i \cdot \prod_{j=1}^{\ell'} h'_{2j-t_j}, K_{i,2\ell'+1} \cdot h_{i,2\ell'+1}, \ldots, K_{i,2\ell} \cdot h_{2\ell}),$$

where $t = (t_1, \ldots, t_{\ell'})$ is the key element associated prefix. Output $(pp, m(msk_{\mathbf{t}}), (K'_{\mathbf{t},i})_i)$.

$\mathsf{CDel}(C_{\mathbf{t},i}, t_{\ell'})$: on input ciphertext $C_{\mathbf{t},i} =: (C_0, C, \cdots, C_{2\ell'-1}, C_{2\ell'}, \ldots, C_{2\ell})$ for bit $t_{\ell'}$, output

$$(C_0, C \cdot C_{2\ell'-t_{\ell'}}, C_{2\ell'+1}, \ldots, C_{2\ell}).$$

$\mathsf{CRerand}(C_{\mathbf{t},t_{\mathsf{exp}}})$: on input ciphertext $C_{\mathbf{t},t_{\mathsf{exp}}} = ((C_{\mathbf{t},i})_i, m(msk_{\mathbf{t}}) \cdot M)$ with

$$C_{\mathbf{t},i} = (C_0, C, \cdots, C_{2\ell'+1}, \ldots, g_{2\ell}),$$

for prefix $(t_1, \ldots, t_{\ell'})$, for all i, sample $(g_0, g_1, \ldots, g_{2\ell}) \leftarrow \mathsf{SampG}(pp)$, and compute

$$C'_{\mathbf{t},i} := (C_0 \cdot g_0, C \cdot \prod_{j=1}^{\ell'} g_{2j-pt_j}, C_{2\ell'+1} g_{2\ell'+1}, \ldots, C_{2\ell} g_{2\ell}).$$

Output $C_{\mathbf{t},t_{\mathsf{exp}}} = ((C'_{\mathbf{t},i})_i, m(msk_{\mathbf{t}}) \cdot M)$.

6.3 CPE construction

Let $\mathsf{EDSG} = (\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \mathsf{SampS}, \mathsf{SampK}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$ be an EDSG scheme. We will construct a CPE scheme $\mathsf{CPE} = (\mathsf{Gen}, \mathsf{KPunc}, \mathsf{CPunc}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} := G_T$ and tag space $\mathcal{T} := \{0,1\}^\ell$ (for $\ell = \ell(\lambda)$ determined in K_ε after running SampP in Gen). The construction of our CPE scheme CPE is given in Figure 9.

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, 2\ell)$, sample $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$ and $(h_0, \ldots, h_{2\ell}) \leftarrow \mathsf{SampH}(pp)$, and return key

$$K_\varepsilon = (pp, m(msk_\varepsilon), h_0, msk_\varepsilon, h_1, \ldots, h_{2\ell}).$$

$\mathsf{KPunc}(K_{\mathbf{t}}, t)$: output punctured key and token (depending on puncture tag $t \in \mathcal{T}$)

$$(K_{\mathbf{t} \cup \{t\}}, \Delta_t) \leftarrow \mathsf{KTrunc}(K_{\mathbf{t}}, t).$$

$\mathsf{Enc}(K_{\mathbf{t}}, M, t_{\mathsf{exp}})$: for key $K_{\mathbf{t}} = (pp, m(msk_{\mathbf{t}}), (K_{i,\mathbf{t}})_i)$, run $(g_i, g_{i,1}, \ldots, g_{i,2\ell}) \leftarrow \mathsf{SampG}(pp)$, $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N$, find all unique tag prefixes $t_i = (t_{i,1}, \ldots, t_{i,j_i})$ associated to inner or lead nodes according to the configuration of the tree that can be used to derive tags from $(\mathcal{T} \setminus \{\mathbf{t}\})$ but no tags after t_{exp}. For all i, compute

$$C_{\mathbf{t},t_{\mathsf{exp}},i} = (S \cdot g_{i,0}, \prod_{j=1}^{j_i} g_{i,2j-t_{i,j}}, g_{i,j_i+1} \ldots, g_{i,2\ell}),$$

and output ciphertext

$$C_{\mathbf{t},t_{\mathsf{exp}}} = ((C_{\mathbf{t},t_{\mathsf{exp}},i})_i, m(msk_{\mathbf{t}})^s \cdot M).$$

$\mathsf{CPunc}(C_{\mathbf{t},t_{\mathsf{exp}}}, \Delta_t)$: output punctured ciphertext (depending on Δ_t)

$$C_{\mathbf{t} \cup \{t\}, t_{\mathsf{exp}}} \leftarrow \mathsf{CTrunc}(C_{\mathbf{t},t_{\mathsf{exp}}}, \Delta_t).$$

$\mathsf{Dec}(K_{\mathbf{t}}, C_{\mathbf{t},t_{\mathsf{exp}}})$: on input key $K_{\mathbf{t}} = (pp, m(msk_{\mathbf{t}}), (K_{\mathbf{t},i})_i)$ and ciphertext $C_{\mathbf{t},t_{\mathsf{exp}}} = ((C_{\mathbf{t},t_{\mathsf{exp}},i})_i, C)$ use $K_{\mathbf{t},1} = (K_0, K_1, \ldots)$ and $C_{\mathbf{t},t_{\mathsf{exp}},1} = (C_0, C_1, \ldots)$, and output

$$M := \frac{e(C_0, K)}{e(C_1, K_1) \cdot C}.$$

Fig. 9. CPE from EDSG.

Correctness of CPE**.** See that Dec outputs

$$M := \frac{e(C_0, K)}{e(C_1, K_1) \cdot C} = \frac{e(S \cdot g_0, msk_{\mathbf{t}} \cdot \prod_{j=1}^{j_1} h_{2j-t_i})}{e(\prod_{j=1}^{j_1} g_{2j-t_i}, h_0) \cdot m(msk_{\mathbf{t}})^s} \cdot M,$$

for some $t = (t_1, \ldots, t_{j_1})$. Particularly, see that correctness holds due to EDSG's associativity (i.e., $e(g_0, h_i) = e(g_i, h_0)$), orthogonality (i.e., $e(S, h_i) = 1$ and $e(g_0, K) = 1$), and projective (i.e., $m(msk_{\mathbf{t}})^s = e(\mathsf{SampS}(pp; s), msk_{\mathbf{t}})$) properties (cf. Dec in Figure 9).

Furthermore, see that in the above construction one can puncture keys and ciphertext with tags from $\mathcal{T}$ in arbitrary order; the resulting key decrypts the corresponding ciphertext correctly as puncturing of keys and ciphertexts stay in-sync.

IND-CPE-CPA security of CPE. Our proof strategy is very similar to the dual system proof strategy framework of [CW13, GCTC16]. We define auxiliary PPT algorithms:

A pseudo-normal ciphertext $\widehat{C}_{\varepsilon, t_{\mathsf{exp}}}$ is generated via

$$\widehat{\mathsf{Enc}}(msk_\varepsilon, M, t_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}}) = (Sg_0\widehat{g}_0, \prod_{i=1}^{\ell-\ell'} g_{2i-t_i}\widehat{g}_{2i-t_i}, \ldots, g_{2\ell}\widehat{g}_{2\ell}, e(S, msk_\varepsilon) \cdot M),$$

with $t_{\mathsf{exp}} = 2^{\ell'}$, for tag prefix $t_i \in \{0,1\}^{\leq \ell-\ell'}$ for tags smaller or equal than t_{exp} (which excludes tag prefixes for tags greater than t_{exp}), $\mathbf{g} = (g_0, \ldots, g_{2\ell}) \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_{2\ell}), \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $S \leftarrow \mathsf{SampS}(pp)$, and $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$.

A pseudo-normal key $\widehat{K}_\varepsilon$ is generated via

$$\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}}) = (pp, m(msk_\varepsilon), h_0\widehat{h}_0, msk_\varepsilon, \ldots, h_{2\ell}\widehat{h}_{2\ell}),$$

for $\mathbf{h} = (h_0, \ldots, h_{2\ell}) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_{2\ell}), \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, and $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$.

A semi-functional (pseudo-normal) key $\widehat{K}_\varepsilon$ is generated via

$$\overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}}) = (pp, m(msk_\varepsilon), h_0\widehat{h}_0, (\widehat{h})^\alpha \cdot msk_\varepsilon, \ldots, h_\ell\widehat{h}_\ell),$$

for $\alpha \leftarrow \mathbb{Z}_N$, $\mathbf{h} = (h_0, \ldots, h_{2\ell}) \leftarrow \mathsf{SampH}(pp)$ and $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_{2\ell}), \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, and $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$. If $\widehat{\mathbf{h}}$ is not present, the key is called semi-functional.

Theorem 2. *If* EDSG *is an EDSG scheme, then* CPE *is* IND-CPE-CPA*-secure. Concretely, for any PPT adversary A there are distinguishers D_1 on LS, D_2, D_3 on RS, D_4 on ND, respectively,*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda) \leq {} & \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{EDSG},\mathsf{G},D_1}(\lambda, 2\ell) + \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{EDSG},\mathsf{G},D_2}(\lambda, 2\ell) \\ & \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{EDSG},\mathsf{G},D_3}(\lambda, 2\ell) + \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{EDSG},\mathsf{G},D_4}(\lambda, 2\ell). \end{aligned} \quad (2)$$

Proof. We show the IND-CPE-CPA security of CPE for any PPT adversary A in a sequence of games where we successively change the games until we arrive at a game where A has only negligible advantage (i.e., success probability of 1/2). Let $S_{A,j}$ be the event that A succeeds in Game j. We want to explicitly mention that the key puncturing oracle KPunc′ works as defined in the security experiment for CPE. We give an overview how the challenge ciphertexts and the secret keys are derived in Table 1.

Game 0. The IND-CPE-CPA experiment.
Game 1 Instead of directly using the ciphertext input by A, the distinguisher decrypts and re-encrypts again starting from a ciphertext for $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$. The change is conceptional. Also we re-write how the initial key is derived.
Game 2. The challenge ciphertext is pseudo-normal.
Game 3. The keys are pseudo-normal.
Game 4. The keys are semi-functional pseudo-normal.
Game 5. The keys are semi-functional.
Game 6. The challenge ciphertext message is a uniform G_T-element.

	Challenge ciphertext	Keys	Assumption
Game 0	Not used	Not used	-
Game 1	$\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t_{\mathsf{exp}}; \mathbf{g})$	$\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$	-
Game 2	$\underline{\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}})}$	As in Game 1	LS
Game 3	As in Game 2	$\underline{\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})}$	RS
Game 4	As in Game 2	$\underline{\overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})}$	Parameter hiding
Game 5	As in Game 2	$\underline{\overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h})}$	RS
Game 6	$\underline{\widehat{\mathsf{Enc}}(msk_\varepsilon, R, t_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}})}$, for $R \leftarrow G_T$	As in Game 5	Non-degeneracy

Table 1. Output of $\widehat{\mathsf{Enc}}$ and $\overline{\mathsf{Ext}}$ to generate (challenge) ciphertexts and keys, for $\alpha \leftarrow \mathbb{Z}_N$, $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, for $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, for $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and $\widehat{\mathbf{h}} \leftarrow \mathsf{SampH}(pp, sp)$. The differences between games are given by underlining.

Lemma 1 (Game 0 to Game 1). *For Game 0 and Game 1 are perfectly indistinguishable, i.e.,*

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

Proof. The is a conceptional change in the security experiment and, hence, does not change the view of A. Instead of using the A-provided ciphertext $C^*_{\mathbf{t}',t'_{\mathsf{exp}}}$ as input to CPunc to compute the challenge ciphertext, D decrypts $M \leftarrow \mathsf{Dec}(K_{\mathbf{t}}, C^*_{\mathbf{t}',t'_{\mathsf{exp}}})$ (after checking that $\mathbf{t} = \mathbf{t}'$ and $t'_{\mathsf{exp}} = t^*_{\mathsf{exp}}$), re-encrypts again by computing

$$C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \varDelta_{t_j}),$$

for $C_0 \leftarrow \mathsf{Enc}(K_\varepsilon, M, t^*_{\mathsf{exp}})$, for all $t_j \in \mathbf{t}''$, for $\mathbf{t}''$ the largest set in $\mathcal{L}$ (after puncturing key $K_{\mathbf{t}}$ with all tags smaller or equal to t^*_{exp} not already included in $\mathbf{t}$).

This change cannot be noticed by A since the now crafted challenge ciphertext has the same distribution as $C^*_{\mathbf{t}'',t'_{\mathsf{exp}}}$ (starting puncturing from $C^*_{\mathbf{t}',t'_{\mathsf{exp}}}$) due to the perfect re-randomization properties of $\mathbb{G}$-elements as output by $\mathsf{SampG}(pp)$. Furthermore, we write $\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$ to derive the first key which is only a re-write in different form to make the input of SampH explicit.

Lemma 2 (Game 1 to Game 2). *Under LS of* EDSG*, Game 1 and Game 2 are computationally indistinguishable. Concretely, for any PPT adversary A in the* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *security experiment with* CPE *there is a distinguisher D on LS such that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, 2\ell). \tag{4}$$

Proof. In Game 1, the challenge ciphertext is normal in the sense of CPE while in Game 2, the challenge ciphertext is pseudo-normal.

Description. The challenge input is provided as $(pp, \mathbf{T})$, where $\mathbf{T}$ is either $\mathbf{g}$ or $\mathbf{g}\widehat{\mathbf{g}}$, for $pp = (\mathbb{G}, \mathbb{H}, G_T, N, e, pars)$, $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, and $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.

Internally, D keeps track of all keys and tokens queried to KPunc' via initially empty set $\mathcal{L}$ (depending on $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$). During the experiment, let $\mathbf{t}$ the currently largest tag set in $\mathcal{L}$.

D samples $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$ and sets $K_\varepsilon := \overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$, for $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, K_\varepsilon, \bot)\}$. D starts A with λ and, during the entire experiment, answers Enc' and KPunc' queries depending on A-provided tags $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$. During the experiment, D can punctures keys $(K_{\mathbf{t}\cup\{t\}}, \varDelta_t) \leftarrow \mathsf{KPunc}(K_{\mathbf{t}}, t)$ for any $t \in \mathcal{T}$. Those keys are used to answer Enc' oracle queries.

When A outputs the target message and ciphertext $(M_0^*, C^*_{\mathbf{t}',t'_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if $C^*_{\mathbf{t}',t'_{\mathsf{exp}}}$ was not queried before or $\mathbf{t}' \neq \mathbf{t}, t'_{\mathsf{exp}} \neq t^*_{\mathsf{exp}}$; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(K_{\mathbf{t}}, C^*_{\mathbf{t}',t'_{\mathsf{exp}}})$.

Furthermore, A punctures keys via KPunc for all tags smaller or equal to t^*. D computes

$$C_0 \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t^*_{\mathsf{exp}}; \mathbf{T})$$

and $C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \varDelta_{t_j})$, for all t_j in (currently largest) $\mathbf{t}$ in $\mathcal{L}$ and all j. D sends to A:

$$(C_{\mathbf{t}}, (K_{t_j})_{t_j \neq t^*}, (\varDelta_{t_j})_j).$$

Eventually, A outputs a guess b'. D outputs 1 if $b' = b$, else outputs 0.

Analysis. If $\mathbf{T} = \mathbf{g}$, then the challenge ciphertext is distributed identically as in Game 1. Otherwise, i.e., if $\mathbf{T} = \mathbf{g}\widehat{\mathbf{g}}$, then the challenge ciphertext is distributed identically as in Game 2. Hence, (4) follows. We want to mention that A only receives tokens to puncture ciphertexts up to $\min(t^*_{\mathsf{win}}, t^*_{\mathsf{exp}})$. Since A does not receive corresponding keys, also the further puncturings of the challenge ciphertext stay unnoticed in the view of A due to LS. Furthermore, due to $\mathbb{G}$ subgroup property, punctured challenge ciphertexts do not reveal any further information.

Lemma 3 (Game 2 to Game 3). *Under RS of* EDSG*, Game 2 and Game 3 are computationally indistinguishable. Concretely, for any PPT adversary A in the* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *security experiment with* CPE*, there is a distinguisher D on RS such that*

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, 2\ell). \quad (5)$$

Proof. In Game 2, we have normal secret keys while in Game 3 we have pseudo-normal secret keys.
Description. The challenge input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T}$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for pp as before, for $\widehat{h}$ specified in sp, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}}, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

Internally, D keeps track of all keys and tokens queried to KPunc' via initially empty set $\mathcal{L}$ (depending on $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$). During the experiment, let $\mathbf{t}$ the currently largest tag set in $\mathcal{L}$.

First, D samples $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$, computes

$$K_\varepsilon := \overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{T}),$$

and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, K_\varepsilon, \bot)\}$. D starts A with λ and, during the entire experiment, answers Enc' and KPunc' queries depending on A-provided tags $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$. During the experiment, D can punctures keys $(K_{\mathbf{t}\cup\{t\}}, \Delta_t) \leftarrow \mathsf{KPunc}(K_{\mathbf{t}}, t)$ for any $t \in \mathcal{T}$. Those keys are used to answer Enc' oracle queries.

When A outputs the target message and ciphertext $(M^*_0, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if $C^*_{\mathbf{t}', t'_{\mathsf{exp}}}$ was not queried before or $\mathbf{t}' \neq \mathbf{t}, t'_{\mathsf{exp}} \neq t^*_{\mathsf{exp}}$; otherwise, D decrypts message $M^*_1 := \mathsf{Dec}(K_{\mathbf{t}}, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$.

Furthermore, A punctures keys via KPunc for all tags smaller or equal to t^*. D computes

$$C_0 \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M^*_b, t^*_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}})$$

and $C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \Delta_{t_j})$, for all t_j in (currently largest) $\mathbf{t}$ in $\mathcal{L}$ and all j. D sends to A:

$$(C_{\mathbf{t}}, (K_{t_j})_{t_j \neq t^*}, (\Delta_{t_j})_j).$$

Eventually, A outputs a guess b'. D outputs 1 if $b' = b$, else outputs 0.

Analysis. If $\mathbf{T} = \mathbf{h}$, then the secret keys are distributed identically as in Game 2. Otherwise, i.e., if $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the secret keys are distributed identically as in Game 3. Since A does not receive corresponding keys, also the further puncturings of the challenge ciphertext stay unnoticed in the view of A due to LS. Furthermore, due to $\mathbb{G}$ subgroup property, punctured challenge ciphertexts do not reveal any further information.

Lemma 4 (Game 3 to Game 4). *We have*

$$|\Pr[S_3] - \Pr[S_4]| = 0. \quad (6)$$

Proof. In Game 3, we have pseudo-normal secret keys while in Game 3 we have pseudo-normal semi-functional secret keys. We set $K_\varepsilon := \overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})$, for uniform $\alpha \leftarrow \mathbb{Z}_N$, $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and $\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$. This is reminiscent of Lemma 11 in [GCTC16]. Essentially, we use the symmetric parameter-hiding property of EDSG to information-theoretically embed $(\widehat{h})^\alpha$. The results in a pseudo-normal semi-functional key. As shown in [GCTC16], due to non-degeneracy, we have that $(\widehat{h})^\alpha$ can be replaced by some suitable $(h_s)^{\alpha'}$, for $(\cdot, h_s, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$ and suitable $\alpha' \in \mathbb{Z}_N$.

Lemma 5 (Game 4 to Game 5). *Under RS of* EDSG *hold, Game 4 and Game 5 are computationally indistinguishable. Concretely, for any PPT adversary A in the* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *security experiment with* CPE*, there is a distinguisher D on RS such that*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, 2\ell). \quad (7)$$

Proof. In Game 4 we have pseudo-normal semi-functional secret keys while in Game 5 we have semi-functional secret keys.

Description. The challenge input is provided as $(pp, \widehat{h}, \mathbf{g}\widehat{\mathbf{g}}, \mathbf{T})$, where $\mathbf{T}$ is either $\mathbf{h}$ or $\mathbf{h}\widehat{\mathbf{h}}$, for pp as before, for $\widehat{h}$ specified in sp, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}}, \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

Internally, D keeps track of all keys and tokens queried to KPunc' via initially empty set $\mathcal{L}$ (depending on $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$). During the experiment, let $\mathbf{t}$ the currently largest tag set in $\mathcal{L}$.

First, D samples $msk_\varepsilon \leftarrow \mathsf{SampK}(pp)$, computes

$$K_\varepsilon := \overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{T}),$$

for uniform $\alpha \leftarrow \mathbb{Z}^*_{\mathsf{ord}(\mathbb{H})}$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, K_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers Enc' and KPunc' queries depending on A-provided tags $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$. During the experiment, D can punctures keys $(K_{\mathbf{t}\cup\{t\}}, \varDelta_t) \leftarrow \mathsf{KPunc}(K_{\mathbf{t}}, t)$ for any $t \in \mathcal{T}$. Those keys are used to answer Enc' oracle queries.

When A outputs the target message and ciphertext $(M_0^*, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if $C^*_{\mathbf{t}', t'_{\mathsf{exp}}}$ was not queried before or $\mathbf{t}' \neq \mathbf{t}, t'_{\mathsf{exp}} \neq t^*_{\mathsf{exp}}$; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(K_{\mathbf{t}}, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$.

Furthermore, A punctures keys via KPunc for all tags smaller or equal to t^*. D computes

$$C_0 \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t^*_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}})$$

and $C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \varDelta_{t_j})$, for all t_j in (currently largest) $\mathbf{t}$ in $\mathcal{L}$ and all j. D sends to A:

$$(C_{\mathbf{t}}, (K_{t_j})_{t_j \neq t^*}, (\varDelta_{t_j})_j).$$

Eventually, A outputs a guess b'. D outputs 1 if $b' = b$, else outputs 0.

Analysis. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the secret keys are distributed identically as in Game 4. Otherwise, i.e., if $\mathbf{T} = \mathbf{h}$, then the secret keys are distributed identically as in Game 5. Since A does not receive corresponding keys, also the further puncturings of the challenge ciphertext stay unnoticed in the view of A due to LS. Furthermore, due to $\mathbb{G}$ subgroup property, punctured challenge ciphertexts do not reveal any further information.

Lemma 6 (Game 5 to Game 6). *Under ND of* EDSG *hold, Game 5 and Game 6 are computationally indistinguishable. Concretely, for any PPT adversary A in the* $\mathsf{IND\text{-}CPE\text{-}CPA}$ *security experiment with* CPE*, there is a distinguisher D on ND such that*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathrm{nd}}_{\mathsf{EDSG},\mathsf{G},D}(\lambda, 2\ell). \tag{8}$$

Proof. In Game 6, we replace the challenge message M_b^*, for $b \in \{0,1\}$, with a (fresh) uniformly random G_T-element. We argue with EDSG's non-degeneracy property for this change.

Description. The challenge 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 pp as before, for $\widehat{h}$ specified in sp, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $(\mathbf{h}) \leftarrow \mathsf{SampH}(pp)$, $(\widehat{\mathbf{h}}, \cdot, \cdot) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $S \leftarrow \mathsf{SampS}(pp)$, $\mathbf{S} = (1, 0, \ldots)$, $K \leftarrow \mathsf{SampK}(pp)$.

Internally, D keeps track of all keys and tokens queried to KPunc' via initially empty set $\mathcal{L}$ (depending on $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$). During the experiment, let $\mathbf{t}$ the currently largest tag set in $\mathcal{L}$.

D sets

$$K_\varepsilon := \overline{\mathsf{Ext}}(K \cdot \widehat{h}^\alpha; \mathbf{h}),$$

for $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, K_\varepsilon, \perp)\}$. D starts A with (λ, K_ε) and, during the entire experiment, answers Enc' and KPunc' queries depending on A-provided tags $t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}$. During the experiment, D can punctures keys $(K_{\mathbf{t}\cup\{t\}}, \varDelta_t) \leftarrow \mathsf{KPunc}(K_{\mathbf{t}}, t)$ for any A-chosen $t \in \mathcal{T}$. Those keys are used to answer Enc' oracle queries.

When A outputs the target message and ciphertext $(M_0^*, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if $C^*_{\mathbf{t}', t'_{\mathsf{exp}}}$ was not queried before or $\mathbf{t}' \neq \mathbf{t}, t'_{\mathsf{exp}} \neq t^*_{\mathsf{exp}}$; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(K_{\mathbf{t}}, C^*_{\mathbf{t}', t'_{\mathsf{exp}}})$. Furthermore, A punctures keys via KPunc for all tags smaller or equal to t^*. D computes initial ciphertext

$$C_0 = (S \cdot g_0\widehat{g}_0, g_1\widehat{g}_1, \ldots, g_{2\ell}\widehat{g}_{2\ell}, \mathbf{T} \cdot M_b^*)$$

which is iteratively punctured with all tags from $t \in \mathbf{t}$ via $C_j \leftarrow \mathsf{CPunc}(C_{j-1}, \Delta_t)$. Afterwards, D punctures all tags t_j smaller or equal to t^* (for positive integer $j > 0$) via

$$(K_{t_j}, \Delta_{t_j}) \leftarrow \mathsf{KPunc}(K_{t_{j-1}}, t_j)$$

with $t_0 = \mathbf{t}$, and sends to A:

$$(C_{\mathbf{t}}, (K_{t_j})_{t_j \neq t^*}, (\Delta_{t_j})_j).$$

Eventually, A outputs a guess b'. D outputs 1 if $b' = b$, else outputs 0.

Analysis. If $\mathbf{T} = e(S, K)$, then the challenge ciphertext is distributed identically as in Game 5. Otherwise, i.e., if $\mathbf{T} = R$, then the challenge ciphertext is distributed identically as in Game 6. Particular, see that A only receives puncture tokens for $t^*, \ldots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$. Furthermore, by the orthogonality property of EDSG, we have $m(\widehat{h}) = 1$.

Lemma 7 (Game 6). *For any PPT adversary A in the* IND-CPE-CPA *security experiment with* CPE*, it holds that*

$$\Pr[S_{A,6}] = 1/2. \tag{9}$$

Proof. In Game 6, for (uniform) $b \in \{0, 1\}$, we provide A with a challenge ciphertext 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 (3), (4), (5), (6), (7), (8), and (9) together, shows (2). $\square$

On parameter sizes. See that we are able to optimize parameter sizes of our CPE construction if tags are punctured in order. Similarly to [CHK03] and [BBG05], we then achieve parameter sizes of $\mathbf{O}(\log^2 n)$ for keys and $\mathbf{O}(\log^2 p)$ ciphertexts where n is the total number of epochs and $p = 2^\ell \leq n$ is defined by the expiry tag. Thereby, the tree is traversed in pre-order as describe in [CHK03, Section 3.3].

6.4 Concrete instantiation under the d-Lin assumption

For completeness, we provide the concrete EDSG 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. The EDSG 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 $\widehat{\mathbf{g}} := ((g^{\widehat{s}\mathbf{f}_0}, g^{\widehat{s}\mathbf{f}}), (g^{\mathbf{0}}, g^{\widehat{\mathbf{s}}\mathbf{f}_1}), \ldots, (g^{\mathbf{0}}, g^{\widehat{\mathbf{s}}\mathbf{f}_n}))$ 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.

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 by the Austrian Science Fund (FWF) and netidee SCIENCE grant P31621-N38 (PROFET).

References

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS 2005*. The Internet Society, February 2005.

[AGJ19] Nimrod Aviram, Kai Gellert, and Tibor Jager. Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 117–150. Springer, Heidelberg, May 2019.

[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.

[And00] Ross Anderson. Two remarks on public-key cryptology. Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS 1997, Zurich, Switzerland, April 1–4, 1997, September 2000.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[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, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 681–710. Springer, 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.

[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.

[CCL+14] Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 95–112. Springer, Heidelberg, March 2014.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Heidelberg, May 2003.

[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.

[Coh19] Aloni Cohen. What about bob? The inadequacy of CPA security for proxy reencryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 287–316. Springer, Heidelberg, April 2019.

[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.

[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.

[DN21] Nico Döttling and Ryo Nishimaki. Universal proxy re-encryption. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 512–542. Springer, Heidelberg, May 2021.

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

[dVR21] 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.

[DvW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, June 1992.

[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.

[FKKP19] Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 317–346. Springer, Heidelberg, April 2019.

[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.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[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.

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021*

- *40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 97–126. Springer, 2021.

[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.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, Heidelberg, December 2002.

[Gün90] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37. Springer, Heidelberg, April 1990.

[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.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481. Springer, Heidelberg, April / May 2002.

[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.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 379–393. ACM Press, November 2019.

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. https://eprint.iacr.org/2020/1003.

[JMM19] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 159–188. Springer, Heidelberg, May 2019.

[JS18] Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[KHRS21] Christiane Kuhn, Dennis Hofheinz, Andy Rupp, and Thorsten Strufe. Onion routing with replies. Cryptology ePrint Archive, Report 2021/1178, 2021. https://ia.cr/2021/1178.

[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.

[LCL$^+$13] Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 235–254. Springer, Heidelberg, December 2013.

[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.

[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.

[Nis21] Ryo Nishimaki. The direction of updatable encryption does matter. Cryptology ePrint Archive, Report 2021/221, 2021. https://eprint.iacr.org/2021/221, version 05-12-2021.

[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.

[PCI16] PCI SSC. Data security standard. https://www.pcisecuritystandards.org/, 2016.

[PR18] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2018.

[PRSV17] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.*, 20(4):14:1–14:31, 2017.

[SD19] Vipin Singh Sehrawat and Yvo Desmedt. Bi-homomorphic lattice-based prfs and unidirectional updatable encryption. *CoRR*, abs/1908.09032, 2019.

[SDLP20] Willy Susilo, Dung Hoang Duong, Huy Quoc Le, and Josef Pieprzyk. Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 107–127. Springer, Heidelberg, September 2020.

[SSS$^+$20] Shifeng Sun, Amin Sakzad, Ron Steinfeld, Joseph K. Liu, and Dawu Gu. Public-key puncturable encryption: Modular and compact constructions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 309–338. Springer, Heidelberg, May 2020.

[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.

[WCW$^+$19] Jianghong Wei, Xiaofeng Chen, Jianfeng Wang, Xuexian Hu, and Jianfeng Ma. Forward-secure puncturable identity-based encryption for securing cloud emails. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part II*, volume 11736 of *LNCS*, pages 134–150. Springer, Heidelberg, September 2019.