Puncture 'Em All: Stronger Updatable Encryption with No-Directional Key Updates

Daniel Slamanig and Christoph Striecks

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

Abstract. Updatable encryption (UE; CRYPTO 2013) is a symmetric encryption primitive that 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. In existing UE constructions, the update token thereby allows bi-directional updates of keys *and* ciphertexts, which leads to undesired information leakage and rather involved security models. A recent work by Jiang (ASIACRYPT 2020) shows that in the currently strongest UE model due to Boyd et al. (CRYPTO 2020), UE with bi-directional key and ciphertext updates implies schemes with uni-directional ones. While this might suggests that uni-directionality does not add security, we show that this rather stems from a defective security model and in an adequate model uni-directionality is indeed stronger. Irrespective of this fact, even uni-directional UE schemes still do not capture the intuitive security requirements expected from UE. To overcome this leakage problem and obtain natural security guarantees, UE schemes with so-called *no-directional* key updates are necessary, i.e., where tokens can solely update ciphertexts and only in one direction. However, it stayed unclear whether such UE schemes can be constructed and this tasks is presented as a challenging open problem in both aforementioned works.
In this work, we resolve these issues and present the first UE constructions with uni- and even no-directional key updates. We show that such UE schemes can be constructed in the standard model via the notion of dual system groups from the standard d-Lin assumption in prime-order bilinear groups. Our approach of constructing UE significantly departs from previous ones and in particular views UE from the perspective of puncturable encryption (Green and Miers, S&P 2015). Towards constructing UE, as an stepping stone, we introduce a variant of puncturable encryption that additionally support puncturing of ciphertexts. This turns out to be a useful abstraction on our way to construct UE and may be of independent interest.

Keywords: Updatable Encryption, Puncturable Encryption, Dual System 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 (Gen, Enc, Dec) for key-generation, encryption, and decryption. Time is discretized in so-called epochs and Gen produces an initial secret key K_1 (for epoch 1). Additionally, there is an algorithm Next which takes

[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

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 arguably more desirable, UE schemes can be ciphertext-independent [LT18, KLR19, BDGJ20, Jia20] 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.

Determinism in UE. 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 is advisable.

Directionality in UE. Intuitively, one would expect that in UE the only functionality of an update token is to update ciphertexts from one to the next epoch. Interestingly, however, in all known UE schemes update tokens can also be used to downgrade ciphertexts and, more importantly, they also allow to upgrade and downgrade keys. To see why key upgrades are a problem, for instance assume that all ciphertexts have been updated 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 $i+1$. Unfortunately, existing UE schemes allow key upgrades, e.g., to use Δ_{i+1} to upgrade the key K_i to K_{i+1} and decrypt all the future data. Moreover, they also allow key downgrades. Those are are more subtle, but as we discuss soon, they also have unwanted side-effects. Besides being counterintuitive, these issues result in security models that have to deal with these inferred information, have to explicitly exclude such cases from constituting an attack, and, hence, are quite involved and unnatural.

The discussion of directionality in UE was raised by Lehmann and Tackmann [LT18]. In particular, they introduced the notions of so-called uni-directional and bi-directional updates, which capture whether tokens can be used to update ciphertexts and keys in either both directions or only into the "future" respectively. While uni-directional UE schemes intuitively seem preferable, Jiang [Jia20] recently showed that UE with bi-directional key and ciphertext updates implies UE with uni-directional key and ciphertext updates in the current state-of-the-art UE model of Boyd et al. [BDGJ20]. At first, this result seems surprising, but as we will discuss this rather stems from a defective model of UE. To prevent attacks like the one mentioned above and to capture the "right intuition" of the security of UE, one requires even stronger UE schemes with so-called *no-directional* key updates. In such schemes, update tokens are not helpful to up- or downgrade keys. Indeed, as also discussed by Jiang [Jia20], such hypothetical UE schemes with no-directional key updates are strictly stronger than UE with uni-directional key and ciphertext updates already in the model due to Boyd et al. [BDGJ20]. However, as we will argue, in the state-of-the-art UE model, no-directional schemes only benefit from preventing key upgrades and it requires a strengthened UE model (that additionally considers expiry epochs for ciphertexts) such that preventing key downgrades, i.e., already the use of uni-directional UE schemes, indeed provides stronger and more natural security guarantees for the practical use of UE schemes.

Since existing models do not seem to capture the right intuition regarding uni-directional or even no-directional key updates, in this work we ask:

What is a suitable notion of security for UE schemes?

Apart from yielding the right intuition of security and providing stronger guarantees in practice, the construction of UE schemes with strong properties has so far been elusive and in this work we further ask:

Is it possible to construct practical and provably secure UE schemes with uni- or even no-directional key updates (in such stronger model)?

1.1 Our Results

We observe that all the previous models and, thus, in particular the state-of-the-art UE model of Boyd et al. [BDGJ20] are too weak as they cannot take advantage of uni-directional key updates. This means that the achieved forward-security as well as post-compromise security guarantees are weak and enable attacks that seem relevant in practice. Consequently, we revisit the security of UE schemes and present a stronger model that is capable of capturing the guarantees provided by UE with uni- and also no-directional key updates, yielding progress towards answering the first question. (We want to mention again that Jiang [Jia20] showed that UE with no-directional key updates is already strictly stronger in the model of Boyd et al.) As our main contribution, we then answer the second question affirmatively and provide the first provably secure constructions of UE with uni- and even no-directional key updates. We therefore introduce a primitive dubbed Ciphertext Puncturable Encryption (CPE) which we believe provides an easier intuition towards our UE constructions and then construct UE from CPE. We show how to construct CPE from the powerful dual system paradigm [Wat09, LW10, LW11, OT12, CW13, HKS15, CGKW18, GW20], yielding an instantiation under the well known d-Lin assumption in prime-order bilinear groups adapting the work of Chen and Wee [CW13]. While CPE is more powerful than what is required for UE (mainly as CPE is a public-key primitive), 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 PE primitive and might be of independent interest. In Figure 1, we provide a brief comparison of our UE constructions with other IND-UE-CPA-secure UE schemes that are only secure in weaker models.[4]

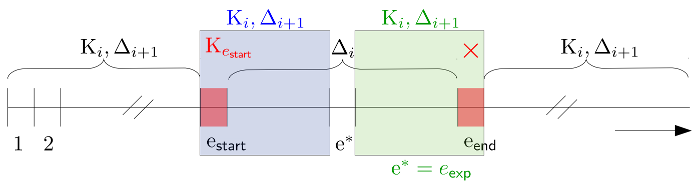
	key-size	ct-size	tok-size	IND-UE-CPA	model	dir. (key)	dir. (ct)	Assumption
SHINE [BDGJ20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	det †	IC	bi	bi	DDH
RISE [LT18]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	rand †	SM	bi	bi	DDH
Jiang [Jia20]	$\mathbf{O}(1)$	$\mathbf{O}(1)$	$\mathbf{O}(1)$	rand †	SM	bi	bi	DLWE
Ours (uni)	$\mathbf{O}(\log^2 n)$	$\mathbf{O}(\log^2 n)$	$\mathbf{O}(1)$	rand	SM	uni	uni	SXDH
Ours (no)	$\mathbf{O}(n)$	$\mathbf{O}(n \log n)$	$\mathbf{O}(1)$	rand	SM	no	uni	SXDH

Fig. 1. Overview of (weakly) IND-UE-CPA-secure ciphertext-independent UE schemes. With n we denote the maximum number of updates. (See that for our uni-directional UE, we can allow an unbounded number of updates by setting $n = 2^\lambda$ with security parameter λ.) † denotes IND-UE-CPA security in a weaker model. With det and rand, we define deterministic and randomized ciphertext updates, respectively. IC and SM stand for Ideal Cipher and Standard Model, respectively.

On limitations of existing UE models. We will now use Figure 2 to illustrate the most important aspects of the IND-UE-CPA notion of Boyd et al. [BDGJ20] (which strengthens and simplifies previous models [LT18, KLR19, Jia20]) by means of the maximum information available to an adversary. Let K_i and Δ_{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 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 2). In between, all update tokens are revealed and prior to e_{start} and after e_{end} the adversary can obtain all keys and update tokens. (Jiang and Boyd et al. capture trivial wins via so-called

[4] We note that while our uni-directional construction is reasonably practical, our no-directional construction is rather a feasibility result. But we hope that this work serves as an inspiration for future work on UE in a strong and realistic model particularly towards constructions with better practical efficiency.

leakage profiles where such firewalls have to exist for their bi- and uni-directional UE schemes, also to prove security.) Now the critical restrictions are that in e_{start} *only* the key is revealed, as otherwise in bi- or uni-directional UE schemes 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.



$\mathrm{K}_i, \Delta_{i+1}$
$\mathrm{K}_i, \Delta_{i+1}$
$\mathrm{K}_{e_{\mathsf{start}}}$
Δ_i
$\mathrm{K}_i, \Delta_{i+1}$
$\mathrm{K}_i, \Delta_{i+1}$
1
2
$\mathrm{e}_{\mathsf{start}}$
e^*
$\mathrm{e}_{\mathsf{end}}$
$\mathrm{e}^* = e_{\mathsf{exp}}$

Fig. 2. Information an adversary is allowed to obtain in the IND-UE-CPA experiment of Boyd et al. [BDGJ20]. Strengthening of the model allows to remove the left firewall (blue box) and expiry dates allow to hand out all keys and tokens after the challenge epoch e^* (green box) for no-directional UE schemes (for uni-directional schemes only the green box applies).

Let us now look at no-directionality and in particular the case where tokens do not allow upgrading of keys (for the time being let us ignore the uni-directional feature, i.e., downgrading may be possible). 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 schemes we can remove the firewall at e_{start} entirely and hand out all keys and tokens up to e^* to the adversary, which is indicated by the blue box in Figure 2. This change makes the IND-UE-CPA stronger and more natural and is our first change to the model by Boyd et al. [BDGJ20]. (Jiang [Jia20] already provided a detailed discussion for uni- and no-directional UE and its leakage profiles with winning conditions in such a model, but we want to make the model even more compact and intuitive.) Note that this gives 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 that have been produced in epochs $j > i$.

Now we turn our attention to downgrading of keys. Here we have to look at the epochs after the challenge epoch e^*. 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 model by Boyd et al. [BDGJ20]. 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 ciphertext 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 [Jia20] or Boyd et al. [BDGJ20] (or our strengthening).

Now the above mentioned attack is mitigated for all keys that are leaked after the expiry date of a ciphertext as long as the UE scheme is at least uni-directional, i.e., prevents key downgrades. To see this, note that in bi-directional schemes 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 uni-directional UE schemes and, thus, particularly in no-directional schemes. Latter, as discussed above, in addition also provide the stronger post-compromise guarantees, making no-directional UE schemes with expiry epochs the most desirable goal.

Towards stronger constructions of UE. 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, DN18] (henceforth UM-PREs). We recall that a UM-PRE scheme is a public-key encryption scheme, where given secret key sk_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 uni- or no-directional UE schemes.

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 [DN18] 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 [DN18]. 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 and at that point it seems more promising to take a different path.

On chosen-ciphertext security for uni- and no-directional 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 which does not rely on schemes with deterministic updates. Namely, Klooß et al. [KLR19] present a construction of an RCCA secure bi-directional UE scheme that relies on a Naor-Yung (NY) like transform together with suitable signature scheme and a malleable NIZK proof system [CKLM12]. We currently do not see a suitable approach to provide (R)CCA security for stronger uni- and no-directional UE schemes and in particular our concrete instantiations, but consider this is important future research question.

1.2 Our Techniques

For our UE constructions, we significantly depart from previous work and view UE from the perspective of Puncturable Encryption (PE) [GM15]. 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 has found numerous applications [GM15, CHN$^+$16, CRRV17, BMO17, DKL$^+$18, GHJL17, DJSS18], has been extended in several ways [WCW$^+$19, DKL$^+$18, DRSS19] and the design of novel PE schemes is still ongoing [SSS$^+$20, SDLP20]. 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.

From puncturing to UE. Roughly, 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 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. The challenging issue from a UE perspective is now that one needs to prevent “unpuncturing,” i.e., even if secret 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 secret keys to new secret keys and thus ciphertexts (no-directionality of key updates in the language of UE).

Intuition behind CPE. We now extend the functionality of puncturing from secret keys only to also allow puncturing of ciphertexts via the notion of ciphertext puncturable encryption (CPE). A CPE scheme can abstractly be viewed as a 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 PE: keys as in PE are initially associated to the entire tag space $\mathcal{T}$ (which is here considered to be an ordered set of potentially exponential size). Ciphertexts in PE are computed w.r.t. a set of tags explicitly provided to the Enc algorithm, whereas in CPE we can create them w.r.t a potentially exponentially sized 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 secret and public key (somewhat akin to key-updatable PKE [JS18, JMM19] or KEMs [PR18]). In addition, the key puncturing on tag t produces a puncturing token Δ_t. Ciphertext puncturing is no fully public and stand-alone operation, but requires the puncturing token Δ_t, to keep ciphertexts and public keys in synchronization. The idea is that a ciphertext produced under a public key (punctured on set of tags $\vec{t}$) using this token can be punctured on tag t and results in a new ciphertext under the new public key punctured on tags $\vec{t} \cup \{t\}$. Thereby, key puncturings can happen on any not necessarily ordered sequence of tags iteratively and ciphertexts can be punctured using the corresponding tokens in an arbitrary order. The semantics of CPE is now that a secret key punctured on $\vec{t}$ can decrypt any ciphertext w.r.t. the corresponding public key as long as the ciphertext still caries some non-punctured tag $t \geq t_i$ for all $t_i \in \vec{t}$. 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 completely punctured on all tags $t_i \leq t_{\mathsf{exp}}$, a ciphertext puncturing fails and decryption is no longer possible.

How to instantiate CPE. From the work of Günther et al. [GHJL17], we know that we can instantiate puncturable encryption (PE) in bilinear groups and in particular using hierarchical identity-based encryption (HIBE) [HL02, GS02, BBG05]. Since CPE is a strictly stronger primitive and, particularly, allows delegation of ciphertexts, basing it on plain HIBE techniques does not work. The reason is that in a HIBE it is not foreseen to delegate ciphertexts after encryption, which we however need for CPE. Nevertheless, we can utilize underlying building blocks used to prove adaptive security of HIBEs, namely dual system groups [CW13, CW14b]. We want to note that adaptive security, i.e., where the adversary can adaptively submit tags to puncture keys and ciphertexts, is inherently backed into our security model for CPE.

Indeed, the dual system paradigm originally developed by Waters [Wat09] provides a rich foundation when one wants to prove adaptive security under simple assumptions in the standard model. In general, the dual system paradigm proved to be a versatile tool to achieve adaptive security in advanced encryption schemes not only for HIBEs [LW10, LW11, CW14b], but also for attribute-based encryption (ABE) [OT12, CGKW18, GWW19, GW20], tightly secure IBEs [CW13, HKS15], and more. We will use Dual System Groups (DSGs) due to Chen and Wee [CW13, CW14b] as a dual system abstraction which can be instantiated from the standard d-Lin assumption in prime-order bilinear groups in the standard model. Interestingly, the DSG approach provides us with elements we need for delegating ciphertext elements (a feature not needed before but available in DSG). This observations extends the use of DSGs to further application domains such as puncturable and updatable encryption.

Recall that a HIBE organizes identities in a tree, where identities at some level can delegate secret keys to its descendant entities, but cannot decrypt ciphertexts intended for other (hierarchical) identities. 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 leaf nodes labeled by the concatenation of the labels of all nodes from the root to the leaf. Consequently, for a height-λ tree, we have an ordered tag space $\mathcal{T}$ of 2^{λ} elements. The natural approach followed in [GHJL17] is that puncturing of a prefix $pt \in \{0,1\}^{\ell}$, $\ell \leq \lambda$, in the tree results in deriving HIBE keys for identities which are not ancestors or descendants of the corresponding node and deleting all other keys, i.e., truncating the tree accordingly. Note that this allows to derive keys for all non-punctured leaves (tags). To construct a CPE scheme, we implicitly arrange tags of the CPE scheme associated to the public and secret keys *or* ciphertexts in a complete binary tree as above and the root of the tree is associated with keys $(pk_{\varepsilon}, sk_{\varepsilon})$ or ciphertext $C_{pk_{\varepsilon},t_{\mathsf{exp}}}$ of the CPE, respectively. Now, in a naive construction encrypting to a public key punctured on $\vec{t}$ would yield a ciphertext that represents the collection of independent HIBE ciphertexts to all the shortest prefixes allowing to derive secret keys for all unpunctured tags, i.e., tags in $\mathcal{T} \setminus \vec{t}$. If encryption takes an additional expiry-tag t_{exp}, then the tree is pruned in a way that no ciphertext element for any tag $t' > t_{\mathsf{exp}}$

can be derived anymore (and, hence, decryption fails). So far, however, leaking old keys would immediately "undo" the puncturing operations as secret keys could simply be delegated (in terms of the HIBE functionality) to allow decryption of punctured ciphertexts.

To now showcase our techniques to avoid such problems, we find it illustrative to consider a concrete example and start from the basic Chen-Wee HIBE [CW14b] (which closely mirrors the Boneh-Boyen-Goh HIBE [BBG05]) for simplicity in a symmetric bilinear group setting with pairing $e : \mathbb{G} \times \mathbb{G} \to G_T$. This helps to show on the way where intuitive approaches fail. In such a HIBE (leaving out details for ease of understanding), one has a main public mpk, main secret key msk, hierarchical secret keys $sk_{\vec{t}=(t_1,t_2,\dots,t_\ell)}$, and ciphertexts $C_{\vec{t}=(t_1,t_2,\dots,t_\ell)}$ as follows:

$$mpk = (g, g_1, \dots, g_\lambda, e(g, g^\alpha)) \qquad msk = g^\alpha$$

$$sk_{\vec{t}} = (g^s, g^\alpha \cdot \prod_{i=1}^{\ell} g_i^{t_i \cdot s}, g_{\ell+1}^s, \dots, g_\lambda^s), \qquad C_{\vec{t}} = (g^r, \prod_{i=1}^{\ell} g_i^{t_i \cdot r}, e(g, g^\alpha)^r \cdot M),$$

for uniformly random exponents α, r, s and group elements $g, g_1, \dots, g_\lambda$. A HIBE ciphertext can be decrypted if the secret key is associated to an ancestor of the tag-hierarchy in the ciphertext where randomness cancels out and the blinding term $e(g, g^\alpha)^r$ is retrieved. Using the HIBE key delegation, one can restrict the secret keys further for any new tag $t_{\ell+1}$, i.e.,

$$sk_{t_1,\dots,t_{\ell+1}} = (g^s, g^\alpha \cdot \prod_{i=1}^{\ell} g_i^{t_i \cdot s} \cdot \boxed{g_{\ell+1}^{t_{\ell+1} \cdot s}}, g_{\ell+2}^s, \dots, g_\lambda^s),$$

where the delegated key is not able to decrypt the above ciphertext $C_{t_1,\dots,t_\ell}$ anymore. In the CPE sense, we will use this feature to provide puncturing of the key (and preventing key unpuncturing).

However, we now face the issue that the ciphertext cannot be delegated or punctured. One way to mitigate this is to add further elements $(g_{\ell+1}^r, \dots, g_\lambda^r)$ to the HIBE ciphertext that allows delegation of ciphertexts similar to the ones for HIBE keys, i.e.,

$$C_{t_1,\dots,t_\ell} = (g^r, \prod_{i=1}^{\ell} g_i^{t_i \cdot r}, \boxed{g_{\ell+1}^r, \dots, g_\lambda^r}, e(g, g^\alpha)^r \cdot M).$$

See that the delegated secret key can now decrypt the ciphertext again; a functionality we want to achieve for CPE. (We can simply "delegate" the ciphertext such that it matches the delegated secret key again.) Now, we have delegation capabilities for keys *and* ciphertexts.

However, although the delegation or puncturing cannot be undone, old secret keys can still decrypt the newly delegated ciphertext, a problematic issue for CPE. Hence, we have to introduce an additional mechanism that "shifts" the $msk = g^\alpha$ *and* the ciphertext to a new $msk' = g^\alpha \cdot g^{\alpha'}$. This can be done by sampling a uniform exponent α' and "add" this to the α-component of the secret key, i.e.,

$$sk_{t_1,\dots,t_{\ell+1}} = (g^s, g^\alpha \cdot \boxed{g^{\alpha'}} \cdot \prod_{i=1}^{\ell} g_i^{t_i \cdot s} \cdot g_i^{t_{\ell+1} \cdot s}, g_{\ell+2}^s, \dots, g_\lambda^s).$$

Now, we have to introduce this shift as well to the ciphertext which can be done by having an "update token" $g^{\alpha'}$ which can be paired with the randomness part of the ciphertext which, together with ciphertext delegation, yields

$$C_{t_1,\dots,t_{\ell+1}} = (g^r, \prod_{i=1}^{\ell} g_i^{t_i \cdot r} \cdot \boxed{g_{\ell+1}^{t_{\ell+1} \cdot r}}, g_{\ell+2}^r, \dots, g_\lambda^r, e(g, g^\alpha)^r \cdot \boxed{e(g^r, g^{\alpha'})} \cdot M),$$

which yields consistently updated-and-delegated secret keys and ciphertexts. See that now an old secret key *cannot* decrypt the updated and delegated ciphertext since the α-components do not match (and, in particular, α' is sampled uniformly at random). This will be the basis for preventing unpuncturing in CPE and particular for our uni-directional UE later on.

Now, we want to consider an even stronger notion. The reader might have noticed that if someone has access to the "update token" $g^{\alpha'}$, old keys can decrypt the updated-and-delegated ciphertext again (by shifting the secret key with such an update token). To mitigate this, we need to blind the token such that updates work for ciphertexts but *not* for keys. We will add specific uniform elements $(h_1, \ldots, h_\lambda)$, which we call shadow elements, to the public key. They will be used for the update tokens and the ciphertexts only. The main public key *mpk* and token Δ now have the form

$$mpk = (g, g_1, \ldots, g_\lambda, e(g, g^\alpha), \boxed{h_1, \ldots, h_\lambda}) \text{ and } \Delta = (\boxed{g^{s'}}, g^{\alpha'} \cdot \boxed{\prod_{i=1}^{\ell} h_i^{s'}})$$

for fresh randomness s' independent of the randomness of the secret key and only dependent on the number of already done updates $\ell \leq \lambda$. To make the ciphertext update work, we need to add further elements to the ciphertext that share the ciphertext random coins r to "unblind" the token during ciphertext update and delegation. A ciphertext has now the form

$$C_{t_1,\ldots,t_\ell} = (g^r, \prod_{i=1}^{\ell} g_i^{t_i \cdot r}, g_{\ell+1}^r, \ldots, g_\lambda^r, \boxed{\prod_{i=1}^{\ell} h_i^r, h_{\ell+1}^r, \ldots, h_\lambda^r}, e(g, g^\alpha)^r \cdot M).$$

The delegate-and-update mechanism for ciphertexts will work as follows. First, "unblind" the term $e(g, g^{\alpha'})^r$ using the token Δ and the added ciphertext components which can be done by

$$e(g, g^{\alpha'})^r = \frac{e(g^r, g^{\alpha'} \cdot \prod_{i=1}^{\ell} h_i^{s'})}{e(\prod_{i=1}^{\ell} h_i^r, g^{s'})}.$$

This blinding term can be used to shift the main public key in the ciphertext (and we also delegate to prevent unpuncturing as before), i.e.,

$$C_{t_1,\ldots,t_{\ell+1}} = (g^r, \prod_{i=1}^{\ell+1} g_i^{t_i \cdot r}, g_{\ell+2}^r, \ldots, g_\lambda^r, \prod_{i=1}^{\ell} h_i^r \cdot \boxed{h_{\ell+1}^r}, h_{\ell+2}^r, \ldots, h_\lambda^r, e(g, g^\alpha)^r \cdot \boxed{e(g, g^{\alpha'})^r}) \cdot M).$$

Such a delegated-and-updated ciphertext can now be decrypted with associated delegated-and-updated secret key *but preventing* that an old key is able to use the update token to gain any information about the new ciphertext. The critical feature here is that the puncture token and the updated ciphertext *do not* share the $h_{\ell+1}$-element and, hence, the token cannot be mix-and-matched with the key in a way that would be useful for decryption. (Intuitively, this can be seen analogously to a secret key in a HIBE that cannot decrypt higher-level ciphertexts, but with switched functionality here.) We will use this feature to prevent puncturing keys via update tokens in CPE and particular for our no-directional UE.

We left out some important operation in our showcase example above, firstly, re-randomization. This can be carried out straightforwardly (as also done in the Chen-Wee HIBE for keys) and we extend this to re-randomizing ciphertexts accordingly. Secondly, our HIBE-extension is quite inefficient due to parameter sizes linear in the number of puncturing/updates. To mitigate this problem, we will use established techniques (cf. [CHK03, GHJL17]) to reduce the parameter sizes to logarithmic in the number of puncturing for our weak form of UE later on (see also example below). However, for our fully secure CPE scheme (and no-directional UE scheme later on), we are bound to the number of puncturing and parameters have linear size. We leave it as one main open problem to reduce parameter sizes for no-directional UE schemes.

Showcasing CPE using the tree approach. We will now use a showcase example for our CPE scheme using the tree approach to support exponentially many puncturable tags. In our scheme, the secret key $sk_{t_1}$and the ciphertext C_{t_1} have several components as depicted in Figure 3. Our initial public key pk_ε will be a HIBE public key consisting of group elements and particularly $e(g, g^{\alpha_\varepsilon})$. (The punctured public key pk_{t_0} holds $e(g, g^{\alpha_{t_0}})$.) The secret keys at the nodes $sk_{t_1}^{01}$ are HIBE secret keys specific for that node, as outlined above. The ciphertext components $(C_{t_1}^{01}, SC_{t_1}^{01})$ are delegatable and updateable HIBE ciphertexts as above, where the shadow component SC explicitly holds the update-related elements $h_1, \ldots, h_\lambda$. We now discuss what happens if such keys and ciphertexts will be punctured on a different tag t_3. We will $\boxed{\text{box}}$ the relevant elements only available for our strongest CPE notion, yielding no-directional UE. Omitting the $\boxed{\text{boxed}}$ elements,

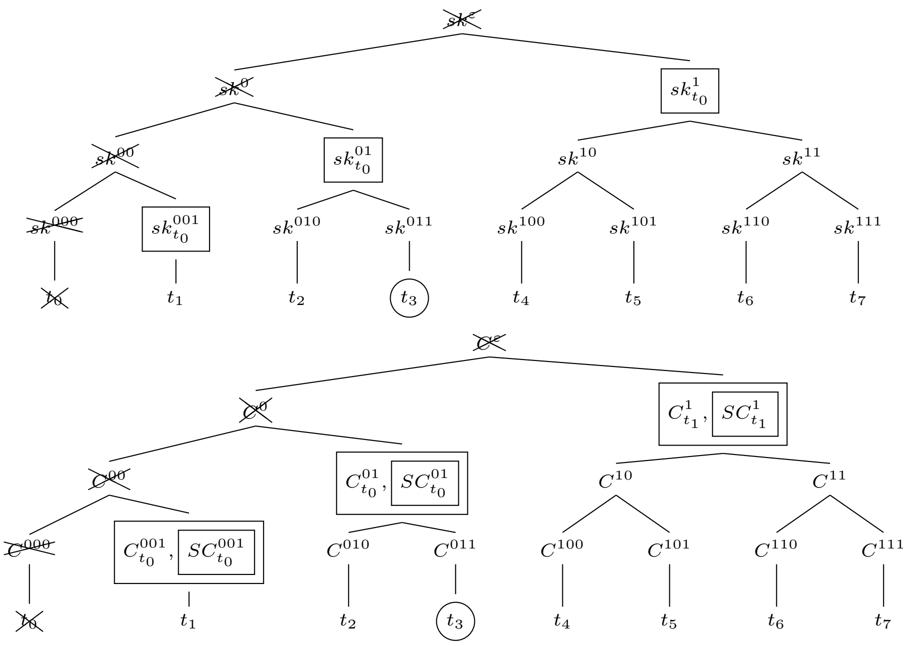


Fig. 3. Example of a CPE secret key and ciphertext that have been punctured on t_0. The secret key sk_{t_0} has the $\boxed{\text{boxed}}$ elements $(sk^{001}_{t_0}, sk^{01}_{t_0}, sk^{1}_{t_0})$. The ciphertext C_{t_0,t_7} with expiry tag t_7 also has the respective $\boxed{\text{boxed}}$ elements $((C^{001}_{t_0}, SC^{001}), (C^{01}_{t_0}, SC^{01}), (C^{1}_{t_0}, SC^{1}))$ where SC-component denote the shadow ciphertext parts that have to be updated for each puncturing. As given in the running text, we circle t_3 as the tag to be punctured on next. SC are the corresponding shadow components in the ciphertext to maintain the overall puncturing status.

we can use our CPE to construct uni-directional UE. To puncture t_3, compute punctured secret key and a puncture token as:

$$sk^{010}_{t_0,t_3} = (g^s, g^{\alpha_{t_0}} \cdot g^{\alpha_{t_3}} \cdot (g_1^{(0)} g_2^{(01)} g_3^{(010)})^s)$$

$$pk_{t_0,t_3} = e(g, g^{\alpha_{t_0}}) \cdot e(g, g^{\alpha_{t_3}}) \qquad \Delta_{t_3} = (\boxed{g^{s'}}, g^{\alpha_{t_3}} \boxed{\cdot (h_1 h_2)^{s'}})$$

Now, using Δ_{t_3}, puncturing the ciphertext works as described above in the HIBE example where we touch each shadow component:

$$C^{010}_{t_0,t_3} = (g^r, (g_1^{(0)} g_2^{(01)} g_3^{(010)})^r, e(g^r, g^{\alpha_{t_0}}) \cdot e(g^r, g^{\alpha_{t_3}}) \cdot M)$$

$$\boxed{SC^{010}_{t_0,t_3}} = ((h_1 h_2 h_3)^r, h_4^r, \ldots, h_\lambda^s) \qquad \boxed{SC^{001}_{t_0,t_3}} = ((h_1 h_2 h_3)^{r'}, (h_4)^{r'}, \ldots, (h_\lambda)^{r'})$$

$$\boxed{SC^{1}_{t_0,t_3}} = ((h_1 h_2 h_3)^{r''}, (h_4)^{r''}, \ldots, (h_\lambda)^{r''}),$$

where $e(g^r, g^{\alpha_{t_3}})$ can be computed via $SC^{01}_{t_0}$ and Δ_{t_3} as shown in the HIBE example above. Furthermore, re-randomization is done afterwards. Again, this can be carried out straightforwardly (since we only use adapted HIBE operations).

We want to mention that our CPE construction (i.e., leaving out the boxed elements above) in a weaker CPE model has linear keys and ciphertexts in the number of punctures yielding uni-directional UE with logarithmic sizes in the weaker UE model due to that epochs are punctured sequentially. Furthermore, incorporating the boxed elements, our CPE construction has linear-size public-keys, secret keys, and ciphertexts. Such CPE construction is also secure in our (standard) CPE model which yields no-directional UE also in our (stronger) UE model.

UE from CPE. To construct UE, let $\mathsf{CPE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{KPunc}, \mathsf{CPunc}, \mathsf{Dec})$ be a CPE scheme and we view the tag space $\mathcal{T} = \{1, 2, \ldots\}$ of CPE as a polynomially bounded ordered set of epochs. Now, the CPE scheme is used as a symmetric encryption scheme (the respective public key is kept secret and included into the secret key) and 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 secret key K_1 is derived from the public-secret key pair $(pk_\varepsilon, sk_\varepsilon)$ of the CPE scheme and every update of the key from epoch e to $e+1$ represents a puncturing of the secret key on the respective epoch number $e+1$ (viewed as tag), setting the key K_{e+1} to $(pk_{\vec{t}}, sk_{\vec{t}})$ with $\vec{t} = (1, \ldots, e+1)$. The update token Δ_{e+1} for ciphertexts from period e to $e+1$ represents the CPE-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 public key $pk_{\vec{t}}$ (with $\vec{t} = (1, \ldots, e+1)$) of the CPE (which is part of the key K_{e+1}).

Outline of the paper. Section 2 recalls some preliminaries. In Section 3, we present our strengthened and simplified security model starting from the UE framework of Boyd et al. [BDGJ20] to reflect uni- and no-directional key updates. In Section 4, we introduce Ciphertext Puncturable Encryption (CPE) along with our security model. Then, in Section 5, 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 discuss a concrete instantiation of CPE under the d-Lin assumption in prime-order bilinear groups.

2 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

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

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.[5]

3 Updatable Encryption

In this section, we recap updatable encryption (UE) with the enhancement of expiring ciphertexts and give a stronger as well as simpler UE security model. We build on the UE security model given

[5] The original definition of d-LIN requires $a_1, \ldots, a_d, s_{d+1} \leftarrow \mathbb{Z}_p$; however, as in [CW14b, Remark 12], we allow for a negligible difference of $(d+1)/p$ in the $\mathsf{Adv}^{\mathsf{d-LIN}}_{\mathsf{G},D}$ function.

by Boyd et al. [BDGJ20] and enhance its security guarantees while providing simplicity.[6] Notably, dealing with uni- and no-directional key updates for our UE definition allows to simplify the UE security model significantly by avoiding the use of the rather complicated leakage-profile paradigm present in all prior UE models (which comes into play when considering weaker UE schemes that do not provide uni-/no-directionality).

The main idea of UE with expiring ciphertexts 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 secret key K_1. With this secret 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. Note that in the Boyd et al. [BDGJ20] model, we would simply have no expiration and $e_{\mathsf{exp}} = \infty$. 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 $C_{1,e_{\mathsf{exp}}}$ is "consistent" with K_2. Correctness guarantees that decryption of $C_{1,e_{\mathsf{exp}}}$ yields $M = \mathsf{Dec}(K_2, C_{2,e_{\mathsf{exp}}})$ as intended if the ciphertext is not expired already. More formally, we will now define UE and its correctness property.

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

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

$\mathsf{Next}(K_e)$*: on input key* K_e*, 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}*, 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} *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 all* $K_1 \leftarrow \mathsf{Gen}(\lambda)$*, for all* $e \in [\lfloor e(\lambda) \rfloor]$[7]*, for all* $(k_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$*, for all* $M \in \mathcal{M}$*, for all expiry epochs* $e_{\mathsf{exp}} \geq e$*, for all* $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$*, for all* $C_{e+1,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_{e+1}, C_e)$*, we have that* $M = \mathsf{Dec}(k_{e'}, C_{e',e_{\mathsf{exp}}})$ *holds, for all* $e' \leq e+1$ *and* $e' \leq e_{\mathsf{exp}}$*.*

3.1 Strengthening the Security Model

For no-directional and also already for our form of uni-directional UE, we need to enhance the currently standard notion of UE security. Namely, to allow expiring ciphertexts, we need to deal with problem that keys can be downgrade using the previous update token. This is cannot be captured by the current IND-UE-CPA model due to Boyd et al. [BDGJ20].

Our adapted IND-UE-CPA notion enhances the model due to Boyd et al. [BDGJ20]. We want to stress that our notion does neither need the concept of leakage profiles or any bookkeeping except for honestly generated ciphertexts and can be seen as one step closer to a more natural UE indistinguishability notion. We further constitute that maintaining leakage profiles to check adversarial winning conditions yields an overhead for UE security models and all prior UE solutions have to deal with it.[8]

We particularly consider the attack that an adversary can downgrade keys using tokens which is not reflected in known UE security models. This is because one could build an UE scheme that has *no* forwards-directional key updates but can have backwards-directional key downgrades and this

[6] The Boyd et al. model generalizes and strengthens the UE model due to Lehmann and Tackmann [LT18] while Jiang [Jia20] also incorporates UE schemes with no- and uni-directionality for that model.

[7] For any polynomial integer e in the security parameter λ.

[8] We stress that all known UE schemes have bi-directional key and ciphertext updates and no UE scheme with no-directional key updates was proposed yet.

would still be IND-UE-CPA secure in the model of Boyd et al. Since we want ciphertexts that can expire, token in the expiring epoch should not be of help to downgrade a key from the next epoch. Indeed, this extra feature of expiring ciphertexts separates bi- and uni-directional UE schemes in our model. Note that although a no-directional UE scheme would already yield stronger security in the Boyd et al. model (by having stronger guarantees before the challenge epoch), which was shown by Jiang [Jia20], uni- and bi-directional UE schemes are equivalent in such a model. We enhance the security model of Boyd et al. in the discussed direction also capturing UE schemes with no-directional key updates.

Security against chosen-plaintext attacks. This security notion ensures that fresh and updated ciphertexts are indistinghuishable, even if the adversary has access to keys and update tokens. We say that an UE scheme is weakly UE-IND-CPA-secure if any PPT adversary A succeeds in the following experiment only with probability negligibly larger than $1/2$. This weakly secure experiment closely mirrors the model due to Boyd etc al. [BDGJ20] where we discuss relations below. We want to mention that the weakly secure notion prevents the adversary from gaining access to the update token or the secret key prior to the challenge epoch.[9]

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 $e \in [\lfloor e(\lambda) \rfloor - 1]$. The adversary first outputs the target epoch e^*, the "firewall" epochs $e_{\mathsf{start}}, e_{\mathsf{end}}$, as well as a ciphertext expiry epoch e_{exp}. Furthermore, the experiment tosses a coin $b \leftarrow \{0,1\}$ and returns b if the the target is not within the firewall boundaries, i.e., $e_{\mathsf{start}} \geq e^*$ or $e_{\mathsf{end}} \leq e^*$, 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 $e_{\mathsf{start}} + 1, \ldots, \min(e_{\mathsf{end}}, e_{\mathsf{exp}})$ where the adversary has access to the challenge ciphertext from e^* onwards. Moreover, it receives *all* update tokens except for the epochs e_{start} and e_{end} (the latter only if $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$). That is if $e_{\mathsf{exp}} < e_{\mathsf{end}}$, then receiving *all* update tokens can be allowed for UE schemes with uni-/no-directional key updates, but not for weaker bi-directional UE schemes. Furthermore, the adversary has access to an Enc'-oracle which is defined in Figure 4.

$\mathsf{Enc}'(e, M, e_{\mathsf{exp}})$: on input message 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 $\mathcal{L} := \mathcal{L} \cup (e, C_{e,e_{\mathsf{exp}}})$, where $\mathcal{L}$ is an initially empty set of queried ciphertexts.[a]

[a] We assume that all messages in $\mathcal{M}$ have the same length which can be enforced by padding the original messages up to a fixed length.

Fig. 4. Encryption oracle in the IND-UE-CPA experiment.

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. If the target ciphertext was not queried in any epoch prior to the challenge epoch, then the experiment outputs bit b.[10] 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 $b = b'$.

Remark. The adversary is allowed to be able to update the challenge ciphertext until epoch e_{end} without being able to trivially decrypting it. (See that if the challenge ciphertext expires before e_{end}, then update will not work anymore.) This particularly also mirrors the firewalling technique due to [BDGJ20] where the adversary is only allowed to receive updates tokens but no keys for such epochs. Particularly, unlike in the previously known UE security models, we can safely hand over the keys and tokens as discussed due to our *selective* flavor we discuss below. Hence, no leakage-profile tracking is necessary. Furthermore, see that if $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$, then our weak IND-UE-CPA model is asymptotically equivalent to the Boyd et al. model (cf. paragraph below).

Figure 5 depicts the experiment and we define weak IND-UE-CPA security.

[9] This artificial restriction on the adversary is also the case for all known UE models in the literature.

[10] This enforces that we only deal with honestly generated ciphertexts that can be tracked as in previous UE models, particularly as in [BDGJ20].

Definition 2 (Weak IND-UE-CPA security). *An UE scheme* UE *is weakly IND-UE-CPA-secure iff for any PPT adversary A, the advantage functions*

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

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

```
Experiment $\mathsf{Exp}^{\text{ind-ue-cpa}}_{\mathsf{UE},A}(\lambda)$
  $K_1 \leftarrow \mathsf{Gen}(\lambda), (K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{Next}(K_e)$, for all $e \in [\lfloor e(\lambda) \rfloor - 1], \mathcal{L} := \emptyset$
  $(e^*, e_{\mathsf{start}}, e_{\mathsf{end}}, e_{\mathsf{exp}}, \mathtt{st}) \leftarrow A(\lambda), b \leftarrow \{0,1\}$
  if $e_{\mathsf{start}} \geq e^*$ or $e_{\mathsf{end}} \leq e^*$ or $e_{\mathsf{exp}} < e^*$ then return $b$
  if $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$ then
    $(\vec{K}_e, \vec{\Delta}_{e+1}) := (K_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}, \ldots, e^* - 1, e^*, \ldots, e_{\mathsf{end}}\}}, (\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor - 1] \setminus \{e_{\mathsf{start}}, e_{\mathsf{end}}\}})$
  else
    $(\vec{K}_e, \vec{\Delta}_{e+1}) := (K_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}, \ldots, e^* - 1, e^*, \ldots, e_{\mathsf{exp}}\}}, (\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor - 1] \setminus \{e_{\mathsf{start}}\}})$
  $(M^*, C^*_{\tilde{e}, e_{\mathsf{exp}}}, \mathtt{st}) \leftarrow A^{\mathsf{Enc}'}(\vec{K}_e, \vec{\Delta}_{e+1}, \mathtt{st})$
  if $(\tilde{e}, C^*_{\tilde{e}, e_{\mathsf{exp}}}) \notin \mathcal{L}$ then return $b$
  $C^*_{e, e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_e, C^*_{e-1, e_{\mathsf{exp}}})$, for all $e = \tilde{e} + 1, \ldots, e^*$
  $C^*_0 \leftarrow \mathsf{Enc}(K_{e^*}, M^*, e_{\mathsf{exp}}), C^*_1 := C^*_{e^*, e_{\mathsf{exp}}}$
  $b' \leftarrow A^{\mathsf{Enc}'}(C^*_b, \mathtt{st})$
  if $b = b'$ then return 1 else return 0
```

Fig. 5. The weak IND-UE-CPA security notion for UE.

On the selectively chosen target epoch and firewall epochs. In Figure 5, we let the adversary *selectively* output the target epoch e^*, the firewall start and end epochs e_{start} and e_{end}, respectively, *before* seeing any keys, tokens, or ciphertexts of the UE scheme. This is done to ease the description of the IND-UE-CPA model and is asymptotically equivalent to the common *adaptive* notions in [LT18, BDGJ20, Jia20].

It is important to note that in these notions, corruptions of tokens and keys can only be done in the current epoch, i.e,. the adversary has already moved to this epoch via Next. We argue that any successful adversary in adaptive model can be transformed in a successful selective adversary with a reduction loss polynomial in the security parameter. Such simple reduction would guess the target epoch e^*, the firewall epochs $e_{\mathsf{start}}, e_{\mathsf{end}}$, and the expiry-epoch e_{exp} for the adaptive adversary while outputting $e^*, e_{\mathsf{start}}, e_{\mathsf{start}}, e_{\mathsf{exp}}$ to its selective IND-UE-CPA experiment. The 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 IND-UE-CPA as well as requests for further keys and updates until $\lfloor e(\lambda) \rfloor$. If the adaptive adversary outputs its guess, then the reduction returns this guess to the selective 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 the selective IND-UE-CPA sense and is runs in PPT. Since $\lfloor e(\lambda) \rfloor$ is polynomially bounded, we have that the success probability of the selective adversary is bounded by $\varepsilon / e(\lambda)^4$ which is significant.

4 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 secret keys only. The distinguishing feature of CPE is that puncturing on secret keys is now synchronized with ciphertext puncturings. 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.

A CPE scheme has the syntax of a PE scheme, i.e., consisting of the PPT algorithms Gen, Enc, Dec, KPunc, but has an additional *ciphertext-puncturing* algorithm CPunc and puncturing now works differently. Ciphertexts can be punctured on any sequence of tags and a key can decrypt a ciphertext as long as there are still tags (from an ordered tagspace $\{t_1, t_2, t_3, \ldots\}$) on which *neither* the key sk *nor* the ciphertext C have been punctured, e.g., if sk is punctured on t_1 and C is punctured on t_1, t_2, then decryption succeeds since neither sk nor C was punctured on t_3 for example. This may sound counterintuitive, but ciphertext puncturing is not a fully public operation and requires some puncture token which is obtained from the respective secret key puncturing. Hence, we have a "coupled" synchronization between key-puncturing and ciphertext puncturing via a puncture token, e.g., Δ_{t_3} that is used for ciphertext puncturing on tag t_3 to make it consistent with puncturings on secret keys. Furthermore, in plain PE, where ciphertexts can be tagged with multiple tags, even if the secret key is punctured on a single one of these tags, decryption *must not* work anymore which is also different for CPE.

A simple translation of key puncturing of PE to ciphertext puncturing in CPE as described above 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 secret key and is able to add tags from an exponentially large tag space to that secret key which translate to ciphertext puncturing in the sense that the ciphertext is now valid for exponentially many secret keys when crafted. To mitigate this, we introduce an "expiry tag" for ciphertexts where after puncturing with tags larger than this tag, the corresponding secret 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 $t_{\mathsf{exp}} = \max(\mathcal{T})$. 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* $\mathcal{T}$ *and message space* $\mathcal{M}$ *consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{CPunc}, \mathsf{Dec})$*:*

$\mathsf{Gen}(\lambda)$: key generation, on input security parameter λ, outputs public and secret key pair $(pk_\varepsilon, sk_\varepsilon)$. (We assume that $\mathcal{M}$ and $\mathcal{T}$ is implicitly given in pk_ε and that ε denotes the empty set.)

$\mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: key puncturing, on input $(pk_{\mathbf{t}}, sk_{\mathbf{t}})$ and tag $t \in \mathcal{T}$, outputs an updated public key $pk_{\mathbf{t}\cup\{t\}}$, an updated secret key $sk_{\mathbf{t}\cup\{t\}}$, and a puncture token Δ_t or error $\perp$.

$\mathsf{Enc}(pk_{\mathbf{t}}, M, t_{\mathsf{exp}})$: encryption, on input $pk_{\mathbf{t}}$, $M \in \mathcal{M}$, and end tag $t_{\mathsf{exp}} \in \mathcal{T}$, outputs a ciphertext $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$.

$\mathsf{CPunc}(C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}, \Delta_t)$: ciphertext puncturing, on input $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$ and puncture token Δ_t, outputs a punctured ciphertext $C_{pk_{\mathbf{t}\cup\{t\}}, t_{\mathsf{exp}}}$ or error $\perp$.

$\mathsf{Dec}(sk_{\mathbf{t}}, C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}})$: decryption, on input secret key $sk_{\mathbf{t}}$ and ciphertext $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$, outputs M or error $\perp$.

Correctness. Correctness ensures that for all updated punctured key pairs, the ciphertexts under the corresponding public key are successfully decryptable as long as not all tags smaller or equal than the end tag of those ciphertexts are punctured from the secret key. More concretely, for all $\lambda \in \mathbb{N}$, for all $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(\lambda)$, for all $\mathbf{t} \in (\mathcal{T})^{\mathbf{O}(\lambda)} \cup \{\varepsilon\}$, for all $t \in \mathcal{T}$, for all $(pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_t) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$, for all $M \in \mathcal{M}$, for all $t_{\mathsf{exp}} \in \mathcal{T}$ with $t_{\mathsf{exp}} \geq t$, for all $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(pk_{\mathbf{t}}, M, t_{\mathsf{exp}})$, for all $C_{pk_{\mathbf{t}\cup\{t\}}, t_{\mathsf{exp}}} \leftarrow \mathsf{CPunc}(C_{pk_{\mathbf{t}}, t_{t_{\mathsf{exp}}}}, \Delta_t)$, we require that $\mathsf{Dec}(sk_{\mathbf{t}\cup\{t\}}, C_{pk_{\mathbf{t}\cup\{t\}}, t_{\mathsf{exp}}}) = M$ 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 tokens can potentially be used to also puncture keys or even unpuncture keys/ciphertexts. In that vein, we define an indistinguishability-based notion, dubbed IND-CPE-CPA, which guarantees that freshly generated ciphertexts cannot be distinguished from punctured ones for the same set of public keys. In particular, we have to make sure that update tokens cannot be used to puncture keys or unpuncture ciphertexts. Hence, we only want to allow puncturing of ciphertexts, but keys should not be eligible to being punctured/unpunctured via tokens. Furthermore, since we are in the public-key setting, we have to deal with the problem that the adversary can potentially submit a (malformed) ciphertext of its choice it wants to be challenge upon. To mitigate this problem (and proceed with a sound proof), we introduce a ciphertext verification algorithm (see Definition 5) to test valid ciphertexts in the domain of Enc using only publicly available parameters.

IND-CPE-CPA security. The notion ensures that fresh ciphertexts from Enc and punctured ciphertexts from CPunc are indistinguishable given that the have the same expiry tag. We say that a CPE scheme is IND-CPE-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 public and secret key pair $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(\lambda)$. Those keys are not punctured as indicated by ε and will be punctured on tags via KPunc' during the experiment. Let $(pk_{\mathbf{t}}, sk_{\mathbf{t}})$ be the public and secret keys 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 has to first commit to a target tag t^*, expiry tag t^*_{exp}, and a "window" tag t^*_{win} (see discussion below). With this window tag, we want allow the adversary to puncture the challenge ciphertext without being able to trivially decrypting it while t^*_{exp} denotes the expiry tag of the target ciphertext. During the experiment, the adversary then has fully adaptive access to a $\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)$-oracle as given in Figure 6.

```
$\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: on input $t$, if $(\cdot,\cdot,\cdot,\Delta_t) \in \mathcal{L}$, return $\perp$. Otherwise, compute
    $(pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_t) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$, set $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t}\cup\{t\}, pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_t)\}$
    and $(pk_{\mathbf{t}}, sk_{\mathbf{t}}) := (pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}})$.
      – If $t \in \{t^*, \ldots, \min(t^*_{\mathsf{win}}, t^*_{\mathsf{exp}})\}$, return $(pk_{\mathbf{t}}, \Delta_t)$,
      – else return $(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \Delta_t)$.
```

Fig. 6. Key puncture oracle in the IND-CPE-CPA experiment.

At some point, the adversary outputs a target message M^*, a target ciphertext $C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}$. The experiment checks that target ciphertext $C^*_{pk_{\mathbf{t}\setminus\{t\}}, t^*_{\mathsf{exp}}}$ is a valid ciphertext using the public verifiability algorithm Ver (in the sense of Definition 5). If so and the KPunc oracle was not yet called on t^*, it proceeds with puncturing the current key pair on the target tag t^*. Furthermore, the experiment tosses a coin b. If $b = 0$, then compute a fresh encryption C_0 of the target message M under the update key pair using Enc and expiry tag t^*_{exp}; otherwise, if $b = 1$, then compute punctured ciphertext C_1 using CPunc. The adversary eventually outputs a guess b' where the experiment returns 1 if $b' = b$. Figure 7 depicts the experiment.

Remark. We require that the adversary outputs a target tag t^*, expiry tag t^*_{exp}, and a window tag t^*_{win} before it has access to the KPunc'-oracle. See that the adversary still is able to *adaptively* puncture keys and the challenge tag set $\mathbf{t} \setminus \{t^*\}$ still contains adaptively chosen tags. This small restriction helps us to keep the description of the oracle clear and concise. For a fully adaptive CPE model, we can let the adversary adaptively output t^*, t^*_{exp}, and t^*_{win}, and check the winning condition in the end if it did not receive keys for $t^*, \ldots, \min(t^*_{\mathsf{exp}}, t^*_{\mathsf{win}})$ after querying KPunc'. Moreover, choosing target tag(s) non-adaptively is also the case for other prior PE schemes (cf. [GHJL17]).

Definition 4 (IND-CPE-CPA security). *A CPE scheme* CPE *is IND-CPE-CPA-secure iff for any valid PPT adversary A the advantage functions*

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

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}$ *is defined as follows:*

```
**Experiment** $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda)$
  $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(\lambda)$, $b \leftarrow \{0,1\}$, $\mathcal{L} := \emptyset$
  $(t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}, \mathsf{st}) \leftarrow A(\lambda)$
  if $t^*_{\mathsf{win}} \leq t^*$ or $t^*_{\mathsf{exp}} \leq t^*$, then return $b$
  $(M^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}, \mathsf{st}) \leftarrow A^{\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)}(\lambda, \mathsf{st})$
  if $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}) = 0$ or $(pk_{\mathbf{t}\setminus\{t^*\}}, \cdot, \cdot, \cdot) \notin \mathcal{L}$, then return $b$
  if $(\cdot,\cdot,\cdot,\Delta_{t^*}) \notin \mathcal{L}$, then
    $(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}\setminus\{t^*\}}, sk_{\mathbf{t}\setminus\{t^*\}}, t^*)$
  $C_0 \leftarrow \mathsf{Enc}(pk_{\mathbf{t}}, M^*, t^*_{\mathsf{exp}})$, $C_1 \leftarrow \mathsf{CPunc}(C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}, \Delta_{t^*})$
  $b' \leftarrow A^{\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)}(C_b, pk_{\mathbf{t}}, \Delta_{t^*}, \mathsf{st})$
  if $b = b'$ then return 1 else return 0
```

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

Furthermore, we introduce a ciphertext verification algorithm for CPE to test valid ciphertexts in the domain of Enc using only publicly available parameters. Intuitively, this allows us to check which tags have been punctured in the ciphertext and if the ciphertext is encrypted under the corresponding public key.

Definition 5 (Public verifiability of CPE ciphertexts). *We define a verification algorithm* $\mathsf{Ver}(pk_{\mathbf{t}}, C_{pk_{\mathbf{t}},t_{\mathsf{exp}}})$*, which on input a public key* $pk_{\mathbf{t}}$ *and the ciphertext* $C_{pk_{\mathbf{t}},t_{\mathsf{exp}}}$*, outputs 1 if and only if* $C_{pk_{\mathbf{t}},t_{\mathsf{exp}}}$ *is a valid ciphertext in the image of* $\mathsf{Enc}(pk_{\mathbf{t}}, \cdot, t_{\mathsf{exp}})$*.*

We also provide a weakened security notion of CPE (called restricted IND-CPE-CPA), which is useful to construct uni-directional (i.e., weak IND-UE-CPA secure) UE schemes. We defer this definition to Appendix A.1.

5 UE with No-Directional Key Updates from CPE

In this section, we construct updatable encryption (UE) with no-directional key updates from ciphertext puncturable encryption (CPE). The transformation is rather straightforward since UE is essentially a sequenced version of CPE, i.e., in the speak of CPE, we only have polynomially many tags which we map to epochs. 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 IND-UE-CPA security.

$\mathsf{Gen}(\lambda)$: compute $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{CPE.Gen}(\lambda)$ and $(pk_1, sk_1, \Delta_1) \leftarrow \mathsf{CPE.KPunc}(pk_\varepsilon, sk_\varepsilon, 1)$, and return $K_1 := (pk_1, sk_1)$.
$\mathsf{Next}(K_e)$: for $K_e =: (pk_e, sk_e)$, return $(K_{e+1} := (pk_{e+1}, sk_{e+1}), \Delta_{e+1}) \leftarrow \mathsf{CPE.KPunc}(pk_e, sk_e, e+1)$.
$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: for $K_e =: (pk_e, sk_e)$, return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(pk_e, M, e_{\mathsf{exp}})$.
$\mathsf{Update}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})$: return $\mathsf{CPE.CPunc}(C_{e,e_{\mathsf{exp}}}, \Delta_{e+1})$.
$\mathsf{Dec}(K_e, C_e)$: for $K_e =: (pk_e, sk_e)$, return $\mathsf{CPE.Dec}(sk_e, C_{e,e_{\mathsf{exp}}})$.

Fig. 8. Construction of UE from CPE.

For correctness, see that this directly translates from the CPE scheme, i.e., the ciphertext that was computed by Enc and updated via Update can be decrypted by Dec if the keys match and the ciphertext is not expired. We now turn to IND-UE-CPA security.

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

Proof. We show the Theorem by constructing a PPT distinguisher D in the IND-CPE-CPA security experiment with CPE as defined in Figure 7 from any successful PPT adversary A in the IND-UE-CPA security with UE as defined in Figure 5. The distinguisher D starts A and receives $e^*, e_{\mathsf{start}}, e_{\mathsf{end}}, e_{\mathsf{exp}}$. (If the A-input does not have the right distribution as defined in Figure 5, then output $b \leftarrow \{0,1\}$.) It then outputs $(t^* := e^*, t^*_{\mathsf{win}} := e_{\mathsf{end}}, t^*_{\mathsf{exp}} := e_{\mathsf{exp}})$ to its IND-CPE-CPA challenger. Then, D queries $(pk_e, (sk_e), \Delta_e) \leftarrow \mathsf{KPunc}(pk_{e-1}, sk_{e-1}, e)$ (with $\varepsilon := e-1$), for all $e \in [\lfloor e(\lambda) \rfloor]$, from its IND-CPE-CPA challenger (while D *does not* receive secret keys sk_e for $e = e^*, \ldots, \min(e_{\mathsf{end}}, e^*_{\mathsf{exp}})$. D sets $\Delta_1 := \bot$. If $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$, then D sends $(pk_e, sk_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e^*, \ldots, e_{\mathsf{end}}\}}$ and $(\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{end}}\}}$ to A. Otherwise, i.e., if $e_{\mathsf{exp}} < e_{\mathsf{end}}$, then D sends $(pk_e, sk_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e^*, \ldots, e_{\mathsf{exp}}\}}$ and $(\Delta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor]}$ to A. Encryption queries in epoch $e \in [\lfloor e(\lambda) \rfloor]$ to $\mathsf{Enc}'(M)$ for expiry tag e'_{exp} are answered as follows: return $C_{e,e'_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(pk_e, M, e'_{\mathsf{exp}})$ 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 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 $e = \tilde{e}+1, \ldots, e^*$. D forwards $(M^*, C^*_{e^*-1,e_{\mathsf{exp}}}, e^*)$ to its IND-CPE-CPA challenger. D receives $(C_b, pk_{e^*}, \Delta_{e^*})$. D forwards C_b to A. Encryption queries with expiry epoch e'_{exp} in all epochs $e \in [e(\lambda)]$ to $\mathsf{Enc}'(M)$ are answered as $C_e \leftarrow \mathsf{CPE.Enc}(pk_e, M, e'_{\mathsf{exp}})$.

Eventually, A outputs a guess b' which is forwarded to D's challenger. D is able to provide a consistent view for A for keys $(K_e)_e = (pk_e, sk_e)_e$. 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 IND-UE-CPA security experiment with UE, then D is a successful PPT adversary in the IND-CPE-CPA security experiment with CPE which shows the Theorem. $\square$

Uni-Directional UE from CPE. Analogously to the above result, a CPE scheme satisfying a weakened notion of security (so called restricted IND-CPE-CPA, cf. Appendix A.1) yields a uni-directional UE scheme (i.e., a UE scheme that satisfies weak IND-UE-CPA security). As already mentioned in Section 1.2, the concrete construction is obtained by a simplification of our main construction in the next section. For completeness, we present the construction and the proof of security in Appendix A.2.

6 CPE from d-Lin in the Standard Model

We recall that due to Günther et al. [GHJL17], we know that we can obtain PE in bilinear groups and in particular using hierarchical any identity-based encryption (HIBE) scheme [HL02, GS02, BBG05]. However, for CPE which also allows "delegation" of ciphertexts, basing it on plain HIBE techniques does not work. But as we have discussed in Section 1.2, we can utilize underlying building blocks used to prove adaptive security of HIBEs, namely the dual system group (DSG) abstraction due to Chen and Wee [CW13, CW14b] 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 delegating ciphertext elements, a feature not needed before but available in DSG. This particularly broadens the use of DSGs to further application domains such as puncturable and updatable encryption.

6.1 Dual system groups

The concept of Dual system groups (DSGs) [CW13, CW14b] is a rich abstraction of Waters' dual system paradigm [Wat09] and particularly useful to prove HIBEs adaptively secure. The main observation for our intentions is that plain DSG also provides delegation elements for ciphertexts. In their HIBEs-from-DSG work [CW14b], Chen and Wee simply did not need such elements (as HIBEs do not provide such a feature), but as it turns out, those elements are of central interest for us to allow delegation of ciphertexts. To look ahead, a token will be a fresh HIBE key (determined by the number of prior updates) under a uniform main secret key with no delegation elements and will only work for updating the intended ciphertext which yields the desired functionality (since such tokens cannot update keys).

We will now recap DSGs. In our DSG variant, we additionally need to introduces a $\mathbb{G}$-subgroup property for re-randomizing ciphertexts. Our DSG DSG consists of the five PPT algorithms (SampP, SampG, SampH, $\widehat{\mathsf{SampG}}$, $\widehat{\mathsf{SampH}}$) (where differences to the Chen-Wee DSG are given by $\underline{\text{underlining}}$):

$\mathsf{SampP}(\lambda, n)$: 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 $\widehat{h}_0$ (see $\widehat{\mathsf{SampH}}$), 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 $\mathbf{h} = (h_0, \ldots, h_n) \in \mathbb{H}^{n+1}$.

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

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

Correctness of DSG. 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 g_0 which is the first output of $\mathsf{SampG}(pp; s)$, for all $h \in \mathbb{H}$, we have $m(h)^s = e(g_0, h)$.

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

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 have

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

for all $j \in [n]$.

Security of DSG. 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 g_0 as the first output of $\mathsf{SampG}(pp)$, we have $e(g_0, \widehat{h}) = 1$.)

Non-degeneracy. For all $\widehat{g}_0$ which is the first output of $\widehat{\mathsf{SampG}}(pp, sp)$ and for $\widehat{h}_0$ which is the first output of $\widehat{\mathsf{SampH}}(pp, sp)$ where $\widehat{h}$ lies in the group of $\widehat{h}_0$, it holds that $e(\widehat{g}_0, \widehat{h})^\alpha$ is uniformly distributed over the generators of a non-trivial subgroup of G_T, for $\alpha \leftarrow \mathbb{Z}_N$.

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

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

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

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

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

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

DSG in prime-order groups. In comparison to the Chen-Wee DSG, for correctness, we change the $\mathbb{G}$-subgroup property. Concerning security, nothing is changed. As we will argue in Subsection 6.4, their prime-order instantiation already allows such simple modification. For completeness, we provide the concrete prime-order instantiation in Subsection 6.4. We conclude:

Corollary 1. DSG *in Subsection 6.4 is a DSG under the d-Lin assumption in the standard model.*

6.2 Constructing CPE from DSG

To construct a CPE scheme from DSGs, we implicitly arrange tags of the CPE scheme associated to the public and secret keys or ciphertexts in a complete binary tree, i.e., the nodes represent a prefix bit representation of the tag and, hence, the root of the tree is associated with keys $(pk_\varepsilon, sk_\varepsilon)$ or ciphertext $C_{pk_\varepsilon, t_{\mathsf{exp}}}$ of the CPE (as also discussed in the introduction with Figure 3), respectively. We define additional PPT helper algorithms KTrunc and CTrunc within a CPE scheme to prune the tree to output punctured public and secret keys, puncture tokens, and punctured ciphertexts that corresponds to a given set of tags. This is reminiscent of prior works, e.g., [CHK03, GHJL17, DJSS18, DKL+18], but we need to add ciphertext puncturing and enhance it to incorporate tokens.

The main algorithms we use for the tree pruning are KTrunc for keys and CTrunc for ciphertexts that correspond to the KPunc and CPunc algorithms, respectively. Encryption is done according to an associated already punctured tag set where the ciphertext parts are built according to the pruned tree. Furthermore, and this is the important part, an entire shadow components will also be associated to each ciphertext part during encryption.

KTrunc, CTrunc and further helper algorithms. We define the two tree-pruning algorithms KTrunc and CTrunc. Moreover, we need further helper PPT algorithms to make the description more modular, i.e., key delegation KDel and re-randomization $\mathsf{KRerand}$ as well as ciphertext delegation CDel and re-randomization $\mathsf{CRerand}$.

Intuition of KTrunc. Essentially, KTrunc takes the current tree configuration as provided in the public and secret keys (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. KTrunc first finds all elements from the root to the associated leaf of tag t. (Since those elements can be used to derive a secret key for tag t.) It delegates the key elements on that path such that no ancestor elements for t are available anymore and keeps the other key elements.

Furthermore, a puncture token is generated according to the number of already happened punctures, i.e., $|\mathbf{t}|$. Such puncture token incorporates a shift in the current public and secret keys. The result is a pruned tree that excludes secret-key material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The concrete PPT algorithms works as follows:

$\mathsf{KTrunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: on input keys $(pp, m(msk_{\mathbf{t}})) := pk_{\mathbf{t}}$ and $(sk_{\mathbf{t},1}, \ldots, sk_{\mathbf{t},m}) := sk_{\mathbf{t}}$, for some integer $m \in \mathbf{O}(\lambda)$, output punctured public key, punctured secret key, and token according to $t = (t_1, \ldots, t_\lambda)$ as follows:

1a. let $sk_{\mathbf{t},i}$ be the secret key part associated to the unique node which is associated to a prefix of t. (Such unique element always exists, otherwise t would have been punctured already.) Derive delegated secret keys hanging from the path to t by iteratively calling KDel on all prefixes of t starting from the node associated to $sk_{\mathbf{t},i}$ and set $sk'_{\mathbf{t}} := (sk'_{\mathbf{t},\leq m}, sk'_{\mathbf{t},m+1}, sk'_{\mathbf{t},m+2}, \ldots)$, where $sk'_{\mathbf{t},\leq m}$ is the same as $sk_{\mathbf{t}}$, but without $sk_{\mathbf{t},i}$, and $sk'_{\mathbf{t},m+1}, sk'_{\mathbf{t},m+2}, \ldots$ are those derived delegated keys via KDel hanging from the path to t; else,

1b. if there exist a leaf associated to a t-secret key $sk_{\mathbf{t},i}$, for $i \in [m]$, then set $sk'_{\mathbf{t}} := sk'_{\mathbf{t},\leq m}$, where $sk'_{\mathbf{t},\leq m}$ is the same as $sk_{\mathbf{t}}$, but without the leaf-associated secret key $sk_{\mathbf{t},i}$.

2. Sample $\delta \leftarrow \mathbb{H}$, for all $sk'_{\mathbf{t}}$-elements $sk'_{\mathbf{t},i} =: (h, msk_{\mathbf{t}} \cdot h_0 \cdots, \ldots)$, compute $sk'_{\mathbf{t}\cup\{t\},i} := (h, msk_{\mathbf{t}} \cdot \delta \cdot h_0 \cdots, \ldots)$.

3. For each node prefix tag $pt_i = (pt_{i,1}, \ldots, pt_{i,m'})$ for prefix length $m' \leq \lambda$ associated to $sk'_{\mathbf{t}\cup\{t\},i}$, re-randomize all elements $sk'_{\mathbf{t}\cup\{t\},i}$ via $sk''_{\mathbf{t}\cup\{t\},i} \leftarrow \mathsf{KRerand}(sk'_{\mathbf{t}\cup\{t\},i}, pt_i)$ and set $sk'_{\mathbf{t}\cup\{t\}} := (sk''_{\mathbf{t}\cup\{t\},i})_i$.

4. Sample $(h_0, \ldots, h_{\lambda+1}, \ldots, h_{2\lambda}) \leftarrow \mathsf{SampH}(pp)$, and compute $\Delta_{\mathbf{t}\cup\{t\}} := (h_0, \delta \cdot \prod_{i=1}^{\ell} h_{\lambda+i})$, for $\ell = |\mathbf{t}|$.

5. Output $(pk_{\mathbf{t}\cup\{t\}}, sk'_{\mathbf{t}\cup\{t\}}, \Delta_{\mathbf{t}\cup\{t\}})$, for $pk_{\mathbf{t}\cup\{t\}} := (pp, m(msk_{\mathbf{t}}) \cdot m(\delta)) = (pp, m(msk_{\mathbf{t}} \cdot \delta))$.

Intuition of CTrunc. Essentially, CTrunc works similarly to KPunc and takes the current tree configuration as provided in the public keys (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 $\Delta_{\mathbf{t}\cup\{t\}}$. 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 that excludes ciphertext material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The PPT algorithm works as follows:

$\mathsf{CTrunc}(C_{pk_{\mathbf{t}},t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$: on input ciphertext $C_{pk_{\mathbf{t}},t_{\mathsf{exp}}} =: ((C_{pk_{\mathbf{t}},1}, SC_{pk_{\mathbf{t}},1}), \ldots, (C_{pk_{\mathbf{t}},m}, SC_{pk_{\mathbf{t}},m}))$, for some integer $m \in \mathbf{O}(\lambda)$, and $\Delta_{\mathbf{t}} =: (h_0, \delta \cdot \prod_{i=1}^{\ell} h_{\lambda+i})$ with $\ell := |\mathbf{t}| - 1 < \lambda$, output a punctured ciphertext as follows:

1. For all $C_{pk_{\mathbf{t}},i} = (g_0, \ldots, e(g_0, msk_{\mathbf{t}}) \cdot M)$ and $SC_{pk_{\mathbf{t}},i}$ with $SC_{pk_{\mathbf{t}},i} = (\prod_{j=1}^{\ell} g_{\lambda+j}, g_{\lambda+\ell+1}, \ldots, g_{2\lambda})$, compute

$$e(g_0, \delta) := \frac{e(g_0, \delta \cdot \prod_{j=1}^{\ell} h_{\lambda+j})}{e(\prod_{j=1}^{\ell} g_{\lambda+j}, h_0)} \tag{1}$$

and set $C'_{pk_{\mathbf{t}},i} := (g_0, \ldots, e(g_0, msk_{\mathbf{t}}) \cdot e(g_0, \delta) \cdot M) = (g_0, \ldots, e(g_0, msk_{\mathbf{t}} \cdot \delta) \cdot M)$. (See that (1) holds due to the associativity properties of DSG.)

2a. For all $C'_{pk_{\mathbf{t}},i}$, derive delegated ciphertexts hanging from the path to t by iteratively calling CDel on all prefixes of t starting from the node associated to $C'_{pk_{\mathbf{t}},i}$ and set $C'_{pk_{\mathbf{t}},t'} := (C'_{pk_{\mathbf{t}},\leq m}, C'_{pk_{\mathbf{t}},m+1}, C'_{pk_{\mathbf{t}},m+2}, \ldots)$, where $C'_{pk_{\mathbf{t}},\leq m}$ is the same as $C'_{pk_{\mathbf{t}},t'}$, but without $C_{pk_{\mathbf{t}},i}$, and $C'_{pk_{\mathbf{t}},m+1}, C'_{pk_{\mathbf{t}},m+2}, \ldots$ are those derived delegated keys via CDel hanging from the path to t; furthermore, for all $SC_{pk_{\mathbf{t}},i}$, proceed similarly, i.e., derive delegated shadow ciphertext parts hanging from the path to t by iteratively calling SCDel on all prefixes of t starting from the node associated to $SC_{pk_{\mathbf{t}},i,j}$ and set $SC'_{pk_{\mathbf{t}},i} := (SC'_{pk_{\mathbf{t}},i,\leq m'}, SC'_{pk_{\mathbf{t}},i,m'+1}, SC'_{pk_{\mathbf{t}},i,m'+2}, \ldots)$, where $SC'_{pk_{\mathbf{t}},i,\leq m'}$ is the same as $SC_{pk_{\mathbf{t}},i}$, but without $SC_{pk_{\mathbf{t}},i,j}$, and $SC'_{pk_{\mathbf{t}},i,m'+1}$, $SC''_{pk_{\mathbf{t}},i,m'+2}, \ldots$ are those derived delegated shadow ciphertext parts via SCDel hanging from the path to t. Else,

2b. if there exist a leaf associated to a t-ciphertext $C'_{pk_{\mathbf{t}},i}$ and $SC'_{pk_{\mathbf{t}},i}$, for $i \in [m]$, then set $C''_{pk_{\mathbf{t}}} := C'_{pk_{\mathbf{t}},\leq m}$ and $SC''_{pk_{\mathbf{t}},i} := SC'_{pk_{\mathbf{t}},i,\leq m}$, where $C'_{pk_{\mathbf{t}},\leq m}$ and $SC'_{pk_{\mathbf{t}},i,\leq m}$ are the same as $C_{pk_{\mathbf{t}}}$ and $SC_{pk_{\mathbf{t}}}$, but without the leaf associated ciphertext $C_{pk_{\mathbf{t}},i}$ and $SC_{pk_{\mathbf{t}},i,j}$, respectively.

3. For each node prefix tag $pt_i = (pt_{i,1}, \ldots, pt_{i,m'})$ with prefix length $m' \leq \lambda$ associated to $C'_{pk_{\mathbf{t}\cup\{t\}},i}$, re-randomize all elements $C'_{pk_{\mathbf{t}\cup\{t\}},i}$ by computing $C''_{pk_{\mathbf{t}\cup\{t\}},i} \leftarrow \mathsf{CRerand}(C'_{pk_{\mathbf{t}\cup\{t\}},i}, pt_i)$ and set $C'_{pk_{\mathbf{t}\cup\{t\}},t_{\mathsf{exp}}} := (C''_{pk_{\mathbf{t}\cup\{t\}},i})_i$.

4. Output $C''_{pk_{\mathbf{t}},t_{\mathsf{exp}}}$.

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

$\mathsf{KDel}(sk_{\mathbf{t},i}, pt)$: on input secret key $sk_{\mathbf{t},i} =: (h_0, msk_{\mathbf{t}} \cdots, h_\ell)$ for prefix $pt' = (pt_1, \ldots, pt_{\ell-1})$ and (prefix) tag $pt = (pt', pt_\ell)$, output

$$sk'_{\mathbf{t},i} := (h_0, msk_{\mathbf{t}} \cdot h_\varepsilon \cdot \prod_{j=1}^{\ell-1} h_j^{pt_j} h_\ell^{pt_\ell}, h_{\ell+1}, \ldots, h_\lambda).$$

$\mathsf{KRerand}(sk_{\mathbf{t},i}, pt)$: on input secret key $sk_{\mathbf{t},i} =: (h_0, msk_{\mathbf{t}} \cdots, h_\ell, \ldots, h_\lambda)$ for prefix tag $pt' = (pt_1, \ldots, pt_\ell)$, for $(h'_0, h'_\varepsilon, h'_1, \ldots, h'_{2\lambda}) \leftarrow \mathsf{SampH}(pp)$, output

$$sk'_{\mathbf{t},i} := (h_0 \cdot h'_0, msk_{\mathbf{t}} \cdot h_\varepsilon h'_\varepsilon \cdot \prod_{j=1}^{\ell} h_j^{pt_j} \cdot \prod_{j=1}^{\ell} (h'_j)^{pt_j}, h_{\ell+1} \cdot h'_{\ell+1}, \ldots, h_\lambda \cdot h'_\lambda).$$

$\mathsf{CDel}(C_{pk_{\mathbf{t}},i}, pt)$: on input ciphertext

$$C_{pk_{\mathbf{t}},i} =: (g_0, g_\varepsilon \cdots, g_\ell, \ldots, g_\lambda, e(g, msk_{\mathbf{t}}) \cdot M)$$

for prefix $pt = (pt_1, \ldots, pt_{\ell-1})$, and (prefix) tag $pt = (pt', pt_\ell)$, output

$$C'_{pk_{\mathbf{t}},i} := (g_0, g_\varepsilon \cdot \prod_{j=1}^{\ell-1} g_j^{pt_j} g_\ell^{pt_\ell}, g_{\ell+1}, \ldots, e(g, msk_{\mathbf{t}}) \cdot M).$$

$\mathsf{SCDel}(SC_{pk_{\mathbf{t}},i}, pt)$: on input $SC_{pk_{\mathbf{t}},i} =: (\prod_{j=1}^{\ell-1} g_{\lambda+j}, g_{\lambda+\ell}, \ldots, g_{2\lambda})$ output

$$SC'_{pk_{\mathbf{t}},i} := (\prod_{j=1}^{\ell} g_{\lambda+j}, g_{\lambda+\ell+1}, \ldots, g_{2\lambda}).$$

$\mathsf{CRerand}((C_{pk_{\mathbf{t}},i}, SC_{pk_{\mathbf{t}},i}), pt)$: on input ciphertext $C_{pk_{\mathbf{t}},i} =: (g_0, g_\varepsilon \cdots, g_\ell, \ldots, g_\lambda, e(g_0, msk_{\mathbf{t}}) \cdot M)$ and shadow ciphertext part $SC_{pk_{\mathbf{t}},i} =: (\prod_{j=1}^{\ell} g_{\lambda+j}, g_{\lambda+\ell+1}, \ldots, , g_{2\lambda})$, for prefix tag $pt = (pt_1, \ldots, pt_\ell)$ and $\ell' := |\mathbf{t}| - 1$, for $(g'_0, g'_\varepsilon, g'_1, \ldots, g'_{2\lambda}) \leftarrow \mathsf{SampG}(pp)$, output

$$\begin{aligned} C'_{pk_{\mathbf{t}},i} &:= (g'_0 g_0, g_\varepsilon g'_\varepsilon \cdot \prod_{j=1}^{\ell} g_j^{pt_j} \cdot \prod_{j=1}^{\ell} (g'_j)^{pt_j}, g_{\ell+1} g'_{\ell+1}, \ldots, g_\lambda g'_\lambda, \\ &\quad e(g_0, msk_{\mathbf{t}}) \cdot e(g'_0, msk_{\mathbf{t}}) \cdot M) \\ SC'_{pk_{\mathbf{t}},i} &:= (\prod_{j=1}^{\ell'} (g_{\lambda+j}) \cdot \prod_{j=1}^{\ell'} (g'_{\lambda+j}), g_{\lambda+\ell+1} g'_{\lambda+\ell+1}, \ldots, g_{2\lambda} g'_{2\lambda}). \end{aligned}$$

6.3 CPE construction

Let $\mathsf{DSG} = (\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$ be a DSG 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 $\mathcal{T} := \mathbb{Z}_N$ (determined in pk_ε after running SampP in Gen). The construction of our CPE scheme CPE is given in Figure 9.

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, 2\lambda + 1)$, secret key $sk_\varepsilon := (h_0, h_\varepsilon \cdot msk_\varepsilon, h_1, \ldots, h_\lambda)$, for $msk_\varepsilon \leftarrow \mathbb{H}$, for $(h_0, h_\varepsilon, \ldots, h_{2\lambda}) \leftarrow \mathsf{SampH}(pp)$, and public key $pk_\varepsilon := (pp, m(msk_\varepsilon))$, and return $(pk_\varepsilon, sk_\varepsilon)$.[a]

$\mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: on input $(pk_{\mathbf{t}}, sk_{\mathbf{t}})$ and tag $t \in \mathcal{T}$, outputs a public key, punctured secret key, and update token $(pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_{\mathbf{t}\cup\{t\}}) \leftarrow \mathsf{KTrunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$.

$\mathsf{Enc}(pk_{\mathbf{t}}, M, t_{\mathsf{exp}})$: on input public key $pk_{\mathbf{t}} =: (pp, m(msk_{\mathbf{t}}))$, message $M \in \mathcal{M}$, and expiry tag $t_{\mathsf{exp}} \leq 2^{\lambda'}, \lambda' \in [\lambda]$, for $(g_i, g_{i,\varepsilon}, g_{i,1}, \ldots, g_{i,2\lambda}) \leftarrow \mathsf{SampG}(pp; s)$, for $s \leftarrow \mathbb{Z}_N$, find all tag prefixes $pt_i = (pt_{i,1}, \ldots, pt_{i,m})$, $m \in \mathbf{O}(\lambda)$, according to the binary tree that exclude $(\mathbf{t}, \ldots, t_{\mathsf{exp}})$, for $\ell := |\mathbf{t}|$, and compute

$$C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}, i} := (g_{i,0}, g_{i,\varepsilon} \prod_{j=1}^{m} g_{i,j}^{pt_{i,j}}, g_{i,m+1} \ldots, g_{i,\lambda'}, m(msk_{\mathbf{t}})^s \cdot M, SC_{pk_{\mathbf{t}},i}),$$

with $SC_{pk_{\mathbf{t}},i} := (\prod_{j=1}^{\ell} g_{i,\lambda+j}, g_{i,\lambda+\ell+1}, \ldots, g_{i,2\lambda})$. Output $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}} := (C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}, i})_i$.

$\mathsf{CPunc}(C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$: on input ciphertext $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$, puncture token $\Delta_{\mathbf{t}\cup\{t\}}$, outputs $C_{pk_{\mathbf{t}\cup\{t\}}, t_{\mathsf{exp}}} \leftarrow \mathsf{CTrunc}(C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$.

$\mathsf{Dec}(sk_{\mathbf{t}}, C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}})$: on input secret key $sk_{\mathbf{t}} =: (sk_{\mathbf{t},1}, \ldots)$ and ciphertext $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}} =: (C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}, 1}, \ldots)$ find the smallest tag $t \in \mathcal{T}$ that has not been punctures in either $sk_{\mathbf{t}}$ and $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$. (Such tag must exists.) Find the corresponding secret-key element $sk_{\mathbf{t},i}$ and the ciphertext element $C_{pk_{\mathbf{t}}, \mathbf{t}, i}$ that are associated to a prefix of $t = (t_1, \ldots, t_\lambda)$. (This is either a leaf-tag itself or a prefix with length $\lambda - 1$.) Compute $(h_0, msk_{\mathbf{t}} \cdot h_\varepsilon \cdot \prod_{j=1}^{\lambda} h_j^{t_j}, \ldots) \leftarrow \mathsf{KDel}(sk_{\mathbf{t},i}, t)$ and $(g_0, g_\varepsilon \cdot \prod_{j=1}^{\lambda} g_j^{t_j}, \ldots, e(g, msk_{\mathbf{t}}) \cdot M) \leftarrow \mathsf{CDel}(C_{\mathbf{t}, t_{\mathsf{exp}}, i}, t)$, and output

$$M := \frac{e(h_0, g_\varepsilon \cdot \prod_{j=1}^{\lambda} g_j^{t_j})}{e(g_0, msk_{\mathbf{t}} \cdot h_\varepsilon \cdot \prod_{j=1}^{\lambda} h_j^{t_j})} \cdot e(g, msk_{\mathbf{t}}) \cdot M.$$

[a] Note that sp will only be used in the security proof.

Fig. 9. CPE from DSG.

Public verifiability of ciphertexts. Intuitively, we check which tags have already been punctured in the ciphertext and if the ciphertext encrypts under the corresponding public key. This can be done with public parameters only utilizing the pairing such that ciphertext components (containing backed-in tag and shadow information) and public parameters can be paired to see if the result matches under the correct public key. More concretely, For each prefix tag $pt = (pt_1, \ldots, pt_{m'})$, for each ciphertext part

$$C_{pk_{\mathbf{t}},i} = (g_0, g_\varepsilon \cdot \prod_{j=1}^{m'} g_j^{pt_j}, g_{m'+1}, \ldots, g_{\lambda'}, e(g_0, msk_{\mathbf{t}}) \cdot M, SC_{pk_{\mathbf{t}},i}),$$

in $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$, with $SC'_{pk_{\mathbf{t}},i} =: (g_{\lambda+1} \cdots, \ldots, g_{2\lambda})$ see that if

$$e(g_0, h_0 \prod_{j=1}^{m'} h_j^{pt_j} \prod_{j=m'+1}^{\lambda'} h_j \prod_{j'=\lambda+1}^{2\lambda} h_j) = e(g_\varepsilon \prod_{j=1}^{m'} g_j^{pt_j} \prod_{j=m'+1}^{\lambda'} g_j \prod_{j=\lambda+1}^{2\lambda} g_j, h_0),$$

for $(h_0, \ldots, h_{2\lambda}) \leftarrow \mathsf{SampH}(pp)$, holds for some $t_{\mathsf{exp}} \leq 2^{\lambda'}$, for $\lambda' \in [\lambda]$, $C'_{\mathbf{t},i}$ is valid ciphertext part and, hence, $C_{pk_{\mathbf{t}}, t_{\mathsf{exp}}}$ is a valid ciphertext under $pk_{\mathbf{t}}$. This due to the DSG's associativity property.

Correctness of CPE**.** Correctness holds due to DSG's associativity and projective properties (cf. Dec in Figure 9).

IND-CPE-CPA security of CPE**.** Our proof strategy is very similar to the proof strategy framework of [CW14a]. We define auxiliary PPT algorithms:

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

$$\widehat{\mathsf{Enc}}(msk_\varepsilon, M, t_{\mathsf{exp}} := 2^{\lambda'}; \mathbf{g}\widehat{\mathbf{g}}) = (g_0\widehat{g}_0, g_\varepsilon\widehat{g}_\varepsilon, \ldots, g_{\lambda'}\widehat{g}_{\lambda'},\\ g_{\lambda+1}, \ldots, g_{2\lambda}, e(g_0\widehat{g}_0, msk_\varepsilon)\cdot M),$$

for $(g_0, g_\varepsilon, \ldots, g_\lambda) \leftarrow \mathsf{SampG}(pp)$ and $\widehat{\mathbf{g}} = (\widehat{g}_0, \widehat{g}_\varepsilon, \ldots, \widehat{g}_{2\lambda}) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.

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

$$\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}}) = (h_0\widehat{h}_0, msk_\varepsilon \cdot h_\varepsilon\widehat{h}_\varepsilon, \ldots, h_\lambda\widehat{h}_\lambda),$$

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

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

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

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

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

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{CPE},A}(\lambda) \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},\mathsf{G},D_1}(\lambda, 2\lambda+1)\\ + \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D_2}(\lambda, 2\lambda+1) + \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D_3}(\lambda, 2\lambda+1). \quad (2)$$

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, D decrypts and re-encrypts again starting from a ciphertext for mpk_ε. The change is conceptional. Also we re-write how the first secret key is derived.
Game 2. The challenge ciphertext is pseudo-normal.
Game 3. The secret keys are pseudo-normal.
Game 4. The secret keys are semi-functional pseudo-normal.
Game 5. The secret keys are semi-functional.
Game 6. The challenge ciphertext message is a uniform G_T-element.

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

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

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 $C^*_{pk_{\mathbf{t}\setminus\{t^*\}},t^*_{\mathsf{exp}}}$ (after positive verification via Ver) as input to CPunc to compute the challenge ciphertext, D decrypts $M \leftarrow \mathsf{Dec}(sk_{\mathbf{t}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}},t^*_{\mathsf{exp}}})$, re-encrypts again by computing $C_{pk_{\mathbf{t}_{i+1}}} \leftarrow \mathsf{CPunc}(C_{pk_{\mathbf{t}_i}}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$ and $C_{pk_\varepsilon} \leftarrow \mathsf{Enc}(pk_\varepsilon, M, t_{\mathsf{exp}})$, for all $t_i \in \mathbf{t}$, for $\mathbf{t}$ the largest set in $\mathcal{L}$ (after inserting the target tag t^*).

This change cannot be noticed by A since the distributions of $C_{pk_{\mathbf{t}\setminus\{t^*\}}}$ and $C^*_{\mathbf{t}\setminus\{t^*\},t^*_{\mathsf{exp}}}$ are perfectly indistinguishable due to the re-randomization properties of $\mathbb{G}$-elements as output by $\mathsf{SampG}(pp)$ and the public verifiability of $C^*_{\mathbf{t}\setminus\{t^*\},t^*_{\mathsf{exp}}}$ using Ver. See that if Ver outputs 1, the G_T-element that blinds the message is fixed under the respective public key $pk_{\mathbf{t}\setminus\{t^*\}}$. Furthermore, we write $\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$ to derive the first secret key which is only a re-write in different form to make the input of SampH explicit.

	$\widehat{\mathsf{Enc}}$-output for challenge ciphertext	$\overline{\mathsf{Ext}}$-output for secret keys	Assumption
Game 0	Not used	Not used	-
Game 1	$\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g})$	$\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$	-
Game 2	$\underline{\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g}\widehat{\mathbf{g}})}$	As in Game 1	LS
Game 3	As in Game 2	$\overline{\mathsf{Ext}}\underline{(msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})}$	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	$\underline{\widehat{\mathsf{Enc}}(\widehat{h} \cdot msk'_\varepsilon, R; \mathbf{g}\widehat{\mathbf{g}})}$, 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 secret keys, for $\alpha \leftarrow \mathbb{Z}_N$, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, for $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and 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 2 (Game 1 to Game 2). *Under LS of* DSG*, Game 1 and Game 2 are computationally indistinguishable. Concretely, for any PPT adversary A in the IND-CPE-CPA security experiment with* CPE *there is a distinguisher D on LS such that*

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

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 $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.

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

D samples $msk_\varepsilon \leftarrow \mathbb{H}$ and sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, $sk_\varepsilon := \overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$, for $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers $\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$-queries, for $\mathbf{t}$ the currently largest set in $\mathcal{L}$, A-chosen $t \in \mathcal{T}$, and depending on A-provided tags $t^*, \ldots, \min(t^*_{\mathsf{win}}, t^*_{\mathsf{exp}})$.

When A outputs the target message and ciphertext $(M_0^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}) = 0$ holds; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t^*_{\mathsf{exp}}; \mathbf{T})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{\mathbf{t}_i \cup \{t_i\}})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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 secret keys, also the further puncturings of the challenge ciphertext stay unnoticed in the view of A due to LS.

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

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D}(\lambda, 2\lambda + 1). \tag{5}$$

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 $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

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

First, D samples $msk'_\varepsilon \leftarrow \mathbb{H}$, sets $msk_\varepsilon := msk'_\varepsilon \cdot \widehat{h}$, sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, sets

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

and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers KPunc'-queries (depending on $t^*, \ldots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$), for $\mathbf{t}$ the currently largest set in $\mathcal{L}$ and A-chosen $t \in \mathcal{T}$.

When A outputs the target message and ciphertext $(M^*_0, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}) = 0$ holds; otherwise, D decrypts message $M^*_1 := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M^*_b; \mathbf{g}\widehat{\mathbf{g}})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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. Particular, see that A only receives puncture tokens for $t^*, \ldots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$. Furthermore, see that msk and msk' are identically distributed and we have that $m(msk) = m(msk')$ holds due to orthogonality of DSG. Hence, (5) follows.

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

$$|\Pr[S_3] - \Pr[S_4]| = 0. \tag{6}$$

Proof. In Game 3, we have pseudo-normal secret keys while in Game 3 we have pseudo-normal semi-functional secret keys. We set $sk_\varepsilon := \overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})$, 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 4 in [CW14b]. Essentially, we use the parameter-hiding property of DSG to information-theoretically embed $(\widehat{h})^\alpha$. The results in pseudo-normal semi-functional keys. This even holds after the adversary sees the (punctured) challenge ciphertext for all α as shown in [CW14b] since due to non-degeneracy, we have that $(\widehat{h})^\alpha$ can be replaced by some suitable $(\widehat{h}_0)^{\alpha'}$, for $\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots) \leftarrow \mathsf{SampH}(pp, sp)$ and suitable $\alpha' \in \mathbb{Z}_N$. Hence, (6) follows. In particular, see that $m(msk_\varepsilon) = m(\widehat{h} \cdot msk_\varepsilon)$ and, hence, the adversary does not receive any further information on $\widehat{h}$ through the public keys.

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

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D}(\lambda, 2\lambda + 1). \tag{7}$$

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 $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $(\mathbf{h}) \leftarrow \mathsf{SampH}(pp)$, $\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

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

First, D samples $msk'_\varepsilon \leftarrow \mathbb{H}$, sets $msk_\varepsilon := msk'_\varepsilon \cdot \widehat{h}$, sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, sets

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

for uniform $\alpha \leftarrow \mathbb{Z}^*_{\mathsf{ord}(\mathbb{H})}$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \bot)\}$. D starts A with λ and, during the entire experiment, answers KPunc'-queries (depending on A-provided tags $t^*, \dots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$), for $\mathbf{t}$ the currently largest set in $\mathcal{L}$ and A-chosen $t \in \mathcal{T}$.

When A outputs the target message and ciphertext $(M_0^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}) = 0$ holds; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g}\widehat{\mathbf{g}})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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. Particular, see that A only receives puncture tokens for $t^*, \dots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$. Hence, (7) follows.

Lemma 6 (Game 5 to Game 6). *Game 5 and Game 6 are perfectly indistinguishable due to* DSG*'s non-degeneracy property. Concretely, for any PPT adversary A on the IND-CPE-CPA security of* CPE*, it holds that*

$$|\Pr[S_{A,5}] - \Pr[S_{A,6}]| = 0. \tag{8}$$

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 DSG's non-degeneracy property for this change. Concretely, for Game-5 challenge ciphertext

$$\widehat{\mathsf{Enc}}(\widehat{h} \cdot msk'_\varepsilon, M_b^*, t_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}}) := (g_0\widehat{g}_0, \dots, e(g_0\widehat{g}_0, \widehat{h} \cdot msk'_\varepsilon) \cdot M_b^*)$$

for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, for $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, note that $e(\widehat{g}_0, \widehat{h})$, is uniformly distributed in a nontrivial subgroup $G'_T \subset G_T$ due to the non-degeneracy property of DSG.

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 and (9) follows.

Taking (3), (4), (5), (6), (7), (8), and (9) together, shows (2). □

6.4 Concrete Instantiation under the *d*-Lin Assumption

For completeness, we provide the concrete DSG instantiation of Chen and Wee [CW14b] and argue that our slight change can be instantiated straightforwardly in prime-order groups under the d-Lin assumptions.

We use the matrix-in-the-exponent notation [CW14b]; namely, for matrices $\mathbf{A} = (a_{i,j})_{i,j} \leftarrow (\mathbb{Z}_p^{k\times k})$ and $\mathbf{B} = (b_{i,j})_{i,j} \leftarrow (\mathbb{Z}_p^{k\times k})$, we have $e(g^{\mathbf{A}}, g^{\mathbf{B}}) = e(g,g)^{\mathbf{A}^\top\mathbf{B}}$. The DSG construction (adapted mostly verbatim from [CW14b]) 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 $\mathbb{G} := \mathbb{G}_1^{d+1}, \mathbb{H} := \mathbb{G}_2^{d+1}, G_T := G'_T, e := e', g := g_1, h := g_2$. Furthermore, sample matrices $\mathbf{B} \leftarrow \mathsf{GL}_{d+1}(\mathbb{Z}_p)$ and $\mathbf{A}_1, \dots, \mathbf{A}_n \leftarrow \mathbb{Z}_p^{d+1\times d+1}$, set $\mathbf{B}^* := (\mathbf{B}^{-1})^\top$, and sample random full-rank diagonal matrix $\mathbf{R} \in \mathbb{Z}_p^{(d+1)\times(d+1)}$ whose bottom-right entry is 1. Then, set

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

for all $i \in [n]$, and function $m(h^{\mathbf{v}}) := e(g^{\mathbf{D}}, h^{\mathbf{v}})$ $(= e(g,h)^{\mathbf{D}^\top \mathbf{v}})$, for all $\mathbf{v} \in \mathbb{Z}_p^{d+1}$. Define $\widehat{h} := h_2^{f^*}$ and output

$$pp := (\mathbb{G}, \mathbb{H}, G_T, p, g, h, g_T, e, m, g^{\mathbf{D}}, g^{\mathbf{D}_1}, \ldots, g^{\mathbf{D}_n}, h^{\mathbf{D}^*}, h^{\mathbf{D}_1^*}, \ldots, h^{\mathbf{D}_n^*})$$
$$sp := (g^{\mathbf{f}}, g^{\mathbf{f}_1}, \ldots, g^{\mathbf{f}_n}, h^{\mathbf{f}^*}, h^{\mathbf{f}_1^*}, \ldots, h^{\mathbf{f}_n^*}).$$

$\mathbf{g} \leftarrow \mathsf{SampG}(pp)$: sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output $\mathbf{g} := (g^{\mathbf{Ds}}, g^{\mathbf{D}_1\mathbf{s}}, \ldots, g^{\mathbf{D}_n\mathbf{s}}) \in \mathbb{G}^{n+1}$.
$\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$: sample $\widehat{\mathbf{s}} \leftarrow \mathbb{Z}_p^*$ and output $\widehat{\mathbf{g}} := (g^{\widehat{\mathbf{s}}\mathbf{f}}, g^{\widehat{\mathbf{s}}\mathbf{f}_1}, \ldots, g^{\widehat{\mathbf{s}}\mathbf{f}_n}) \in \mathbb{G}^{n+1}$.
$\mathbf{h} \leftarrow \mathsf{SampH}(pp)$: sample $\mathbf{r} \leftarrow \mathbb{Z}_p^d$ and output $\mathbf{h} := (h^{\mathbf{D}^*\mathbf{r}}, h^{\mathbf{D}_1^*\mathbf{r}}, \ldots, h^{\mathbf{D}_n^*\mathbf{r}}) \in \mathbb{H}^{n+1}$.
$\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp)$: sample $\widehat{\mathbf{r}} \leftarrow \mathbb{Z}_p^*$ and output $\widehat{\mathbf{h}} := (h^{\widehat{r}\mathbf{f}^*}, h^{\widehat{r}\mathbf{f}_1^*}, \ldots, h^{\widehat{r}\mathbf{f}_n^*}) \in \mathbb{H}^{n+1}$.

Correctness. See that the output of SampG is uniformly distributed in $\mathbb{G}^{n+1}$ which follows from the fact that $\mathbb{Z}_p^d$ is a additive group, and hence, also the $\mathbb{G}$-subgroup property holds straightforwardly.

Security. All security claims carry over from [CW14b] since no changes in the assumptions or distributions were made for our slight DSG adaptation.

Acknowledgements. 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.

[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.

[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.

[CGKW18] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Heidelberg, April / May 2018.

[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.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Heidelberg, April 2012.

[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.

[CW14a] Jie Chen and Hoeteck Wee. Doubly spatial encryption from DBDH. Cryptology ePrint Archive, Report 2014/199, 2014. http://eprint.iacr.org/2014/199.

[CW14b] Jie Chen and Hoeteck Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014. http://eprint.iacr.org/2014/265.

[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.

[DN18] Nico Döttling and Ryo Nishimaki. Universal proxy re-encryption. "Cryptology ePrint Archive, Report 2018/840", to appear at PKC 2021, 2018.

[DRSS19] David Derler, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. I want to forget: Fine-grained encryption with full forward secrecy in the distributed setting. Cryptology ePrint Archive, Report 2019/912, 2019. https://eprint.iacr.org/2019/912.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

A UE with Uni-Directional Key Updates in the Weak IND-UE-CPA Model

In this section, we give UE with uni-directional key updates in our weak IND-UE-CPA model by slightly adapting the IND-CPE-CPA notion such that this notion implies weak IND-UE-CPA security. Furthermore, we give a construction of such a CPE RCPE and sketch the differences to the stronger CPE construction. Interestingly, we only have to change the way puncture tokens are constructed. This has the effect that we achieve a much more efficient UE construction with $\mathbf{O}(\log^2 n)$ keys and ciphertexts as well as constant token-size with *unbounded* number of updates (for maximum number n of possible puncturings).

A.1 Restricted IND-CPE-CPA Model

We have to restrict the IND-CPE-CPA model only in one aspect, namely, that the adversary is restricted to output a tag t^*_{start} from the ordered tag space such that the adversary now does not get keys for tags $t^*_{\mathsf{start}}, \ldots, t^*, \ldots, \min(t^*_{\mathsf{exp}}, t^*_{\mathsf{win}})$ instead of $t^*, \ldots, t^*_{\mathsf{win}}$ as in the full IND-CPE-CPA model. This is done by restricting the KPunc'-oracle to tags not in $t^*_{\mathsf{start}}, \ldots, t^*, \ldots, \min(t^*_{\mathsf{exp}}, t^*_{\mathsf{win}})$ and only outputting public-keys and tokens but no keys for such tags. The restricted IND-CPE-CPA model is as follows. The CPE correctness definition and the public-verifiability property stay the same.

Restricted IND-CPE-CPA security. The notion ensures that fresh ciphertexts from Enc and punctured ciphertexts from CPunc are indistinguishable. We say that a CPE scheme is restricted IND-CPE-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 public and secret key pair $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(\lambda)$. Those keys are not punctured as indicated by ε and will be punctured on tags via our restricted KPunc' during the experiment as given in Figure 10. Let $(pk_{\mathbf{t}}, sk_{\mathbf{t}})$ be the public and secret keys associated to the (currently largest) punctured-tag set $\mathbf{t}$. During the experiment, the adversary then has fully adaptive access to a $\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)$-oracle as given

$\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: on input t, if $(\cdot,\cdot,\cdot,\Delta_t) \in \mathcal{L}$, return $\perp$. Otherwise, compute $(pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_t) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$, set $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t}\cup\{t\}, pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_t)\}$ and $(pk_{\mathbf{t}}, sk_{\mathbf{t}}) := (pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}})$.
- If $t \in \{t^*_{\mathsf{start}}, \ldots, t^*, \ldots, \min(t^*_{\mathsf{win}}, t^*_{\mathsf{exp}})\}$, return $(pk_{\mathbf{t}}, \Delta_t)$,
- else return $(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \Delta_t)$.

Fig. 10. Key puncture oracle in the restricted IND-CPE-CPA experiment.

At some point, the adversary outputs a target message M^*, a target ciphertext $C^*_{t^*_{\mathsf{exp}}}$. The experiment checks that target ciphertext $C^*_{t^*_{\mathsf{exp}}}$ is a valid ciphertext using the public verifiability algorithm Ver. If so and the KPunc' oracle was not yet called on t^*, it proceeds with puncturing the current key pair on the target tag t^*. Furthermore, the experiment tosses a coin b. If $b = 0$, then compute a fresh encryption C_0 of the target message M under the update key pair using Enc and expiry tag t^*_{exp}; otherwise, if $b = 1$ compute punctured ciphertext C_1 using CPunc. The adversary eventually outputs a guess b' where the experiment returns 1 if $b' = b$. Figure 11 depicts the experiment.

Definition 6 (Restricted IND-CPE-CPA security). *A CPE scheme* CPE *is restricted IND-CPE-CPA-secure iff for any valid PPT adversary A the advantage functions*

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

is negligible in λ, where $\mathsf{Exp}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{RCPE},A}$ *is defined as follows:*

A.2 Constructing CPE in the Restricted IND-CPE-CPA Security Model

The CPE construction under the restricted IND-CPE-CPA security model will be almost the same except that the puncture token will simpler. Hence, also public keys and ciphertexts will contain

```
Experiment $\mathsf{Exp}^{\text{ind-cpe-cpa}}_{\mathsf{RCPE},A}(\lambda)$
  $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(\lambda)$, $b \leftarrow \{0,1\}$
  $(t^*, t^*_{\mathsf{win}}, t^*_{\mathsf{exp}}, , t^*_{\mathsf{start}}\mathtt{st}) \leftarrow A(\lambda)$
  if $t^*_{\mathsf{win}} \leq t^*$ or $t^*_{\mathsf{exp}} \leq t^*$ or $t^*_{\mathsf{start}} > t^*$, then return $b$
  $(M^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}, \mathtt{st}) \leftarrow A^{\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)}(\lambda, \mathtt{st})$
  if $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}) = 0$ or $pk_{\mathbf{t}\setminus\{t^*\}} \notin \mathcal{K}$, then return $b$
  if $(\cdot,\cdot,\cdot,\varDelta_{t^*}) \notin \mathcal{L}$, then
    $(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \varDelta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}\setminus\{t^*\}}, sk_{\mathbf{t}\setminus\{t^*\}}, t^*)$
  $C_0 \leftarrow \mathsf{Enc}(pk_{\mathbf{t}}, M^*, t^*_{\mathsf{exp}}), C_1 \leftarrow \mathsf{CPunc}(C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}, \varDelta_{t^*})$
  $b' \leftarrow A^{\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, \cdot)}(C_b, pk_{\mathbf{t}}, \varDelta_{t^*}, \mathtt{st})$
  if $b = b'$ then return 1 else return 0
```

Fig. 11. The restricted IND-CPE-CPA security notion for RCPE.

less group elements as in our stronger CPE construction (i.e., less elements from the DSG will be needed). In particular, we do not need the concept of shadow components anymore since we do not have to blind the puncture token. KPunc will the same as before, but will not blind the δ_t with elements from SampH and directly output δ_t. For completeness, we recap the construction now and argue that the security proof will proceed with almost the same strategy. Furthermore, correctness and public verifiability is also achieved.

KTrunc, CTrunc and further helper algorithms for RCPE. As for CPE, we define the two tree-pruning algorithms KTrunc and CTrunc. Furthermore, we need two helper PPT algorithms, i.e., key delegation KDel and re-randomization KRerand as well as ciphertext delegation CDel and re-randomization CRerand.

Intuition of KTrunc. Essentially, KTrunc takes the current tree configuration as provided in the public and secret keys (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. KTrunc first finds all elements from the root to the associated leaf of tag t. (Since those elements can be used to derive a secret key for tag t.) It delegates the key elements on that path such that no ancestor elements for t are available anymore and keeps the other key elements. The result is a pruned tree that excludes secret-key material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The concrete PPT algorithms works as follows:

$\mathsf{KTrunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$: on input keys $(pp, m(msk_{\mathbf{t}})) := pk_{\mathbf{t}}$ and $(sk_{\mathbf{t},1}, \ldots, sk_{\mathbf{t},m}) := sk_{\mathbf{t}}$, for some integer $m \in \mathbf{O}(\lambda)$, output punctured public key, punctured secret key, and token according to $t = (t_1, \ldots, t_\lambda)$ as follows:

1a. let $sk_{\mathbf{t},i}$ be the secret key part associated to the unique node which is associated to a prefix of t. (Such unique element always exists, otherwise t would have been punctured already.) Derive delegated secret keys hanging from the path to t by iteratively calling KDel on all prefixes of t starting from the node associated to $sk_{\mathbf{t},i}$ and set $sk'_{\mathbf{t}} := (sk'_{\mathbf{t},\leq m}, sk'_{\mathbf{t},m+1}, sk'_{\mathbf{t},m+2}, \ldots)$, where $sk'_{\mathbf{t},\leq m}$ is the same as $sk_{\mathbf{t}}$, but without $sk_{\mathbf{t},i}$, and $sk'_{\mathbf{t},m+1}, sk'_{\mathbf{t},m+2}, \ldots$ are those derived delegated keys via KDel hanging from the path to t; else,
1b. if there exist a leaf associated to a t-secret key $sk_{\mathbf{t},i}$, for $i \in [m]$, then set $sk'_{\mathbf{t}} := sk'_{\mathbf{t},\leq m}$, where $sk'_{\mathbf{t},\leq m}$ is the same as $sk_{\mathbf{t}}$, but without the leaf-associated secret key $sk_{\mathbf{t},i}$.
2. Sample $\delta \leftarrow \mathbb{H}$, for all $sk'_{\mathbf{t}}$-elements $sk'_{\mathbf{t},i} =: (h, msk_{\mathbf{t}} \cdot h_0 \cdots, \ldots)$, compute $sk'_{\mathbf{t}\cup\{t\},i} := (h, msk_{\mathbf{t}} \cdot \delta \cdot h_0 \cdots, \ldots)$.
3. For each node prefix tag $pt_i = (pt_{i,1}, \ldots, pt_{i,m'})$ for prefix length $m' \leq \lambda$ associated to $sk'_{\mathbf{t}\cup\{t\},i}$, re-randomize all elements $sk'_{\mathbf{t}\cup\{t\},i}$ via $sk''_{\mathbf{t}\cup\{t\},i} \leftarrow \mathsf{KRerand}(sk'_{\mathbf{t}\cup\{t\},i}, pt_i)$ and set $sk'_{\mathbf{t}\cup\{t\}} := (sk''_{\mathbf{t}\cup\{t\},i})_i$.
4. Set $\varDelta_{\mathbf{t}\cup\{t\}} := \delta$.
5. Output $(pk_{\mathbf{t}\cup\{t\}}, sk'_{\mathbf{t}\cup\{t\}}, \varDelta_{\mathbf{t}\cup\{t\}})$, for $pk_{\mathbf{t}\cup\{t\}} := (pp, m(msk_{\mathbf{t}}) \cdot m(\delta)) = (pp, m(msk_{\mathbf{t}} \cdot \delta))$.

Intuition of CTrunc. Essentially, CTrunc (works similarly to KPunc) and takes the current tree configuration as provided in the public keys (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 $\varDelta_{\mathbf{t}\cup\{t\}}$. CTrunc first finds all elements from the root to the associated leaf of tag t. (Since those elements can be used to derive a decryptable ciphertext elements 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 that excludes ciphertext material for t for the new set of punctured tags $\mathbf{t} \cup \{t\}$. The concrete PPT algorithms works as follows:

$\mathsf{CTrunc}(C_{pk_{\mathbf{t}},t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$: on input ciphertext

$$C_{pk_{\mathbf{t}},t_{\mathsf{exp}}} =: ((C_{pk_{\mathbf{t}},1}, SC_{pk_{\mathbf{t}},1}), \ldots, (C_{pk_{\mathbf{t}},m}),$$

for some integer $m \in \mathbf{O}(\lambda)$, and $\Delta_{\mathbf{t}} =: \delta$ with $\ell := |\mathbf{t}| - 1 < \lambda$, output a punctured ciphertext as follows:

1. Set $C'_{pk_{\mathbf{t}},i} := (g_0, \ldots, e(g_0, msk_{\mathbf{t}}) \cdot e(g_0, \delta) \cdot M) = (g_0, \ldots, e(g_0, msk_{\mathbf{t}} \cdot \delta) \cdot M)$.

2a. For all $C'_{pk_{\mathbf{t}},i}$, derive delegated ciphertexts hanging from the path to t by iteratively calling CDel on all prefixes of t starting from the node associated to $C'_{pk_{\mathbf{t}},i}$ and set $C'_{pk_{\mathbf{t}},t'} := (C'_{pk_{\mathbf{t}},\leq m}, C'_{pk_{\mathbf{t}},m+1}, C'_{pk_{\mathbf{t}},m+2}, \ldots)$, where $C'_{pk_{\mathbf{t}},\leq m}$ is the same as $C'_{pk_{\mathbf{t}},t'}$, but without $C_{pk_{\mathbf{t}},i}$, and $C'_{pk_{\mathbf{t}},m+1}, C'_{pk_{\mathbf{t}},m+2}, \ldots$ are those derived delegated keys via CDel hanging from the path to t.

2b. if there exist a leaf associated to a t-ciphertext $C'_{pk_{\mathbf{t}},i}$, for $i \in [m]$, then set $C''_{pk_{\mathbf{t}}} := C'_{pk_{\mathbf{t}},\leq m}$, where $C'_{pk_{\mathbf{t}},\leq m}$ is the same as $C_{pk_{\mathbf{t}}}$, but without the leaf associated ciphertext $C_{pk_{\mathbf{t}},i}$.

3. For each node prefix tag $pt_i = (pt_{i,1}, \ldots, pt_{i,m'})$ with prefix length $m' \leq \lambda$ associated to $C''_{pk_{\mathbf{t}\cup\{t\}},i}$, re-randomize all elements $C''_{pk_{\mathbf{t}\cup\{t\}},i}$ by computing $C'''_{pk_{\mathbf{t}\cup\{t\}},i} \leftarrow \mathsf{CRerand}(C''_{pk_{\mathbf{t}\cup\{t\}},i}, pt_i)$ and set $C'_{pk_{\mathbf{t}\cup\{t\}},t_{\mathsf{exp}}} := (C'''_{pk_{\mathbf{t}\cup\{t\}},i})_i$.

4. Output $C''_{pk_{\mathbf{t}},t_{\mathsf{exp}}}$.

Intuition of KDel, $\mathsf{KRerand}$, CDel, and $\mathsf{CRerand}$. Essentially, KDel delegates secret key material as done in HIBE key delegation where $\mathsf{KRerand}$ re-randomizes the key material. CDel delegates ciphertext material as done in HIBE ciphertext delegation (as discussed in the introduction) where $\mathsf{CRerand}$ re-randomizes the ciphertext material. The concrete PPT algorithms works as follows:

$\mathsf{KDel}(sk_{\mathbf{t},i}, pt)$: on input secret key $sk_{\mathbf{t},i} =: (h_0, msk_{\mathbf{t}} \cdots, h_\ell)$ for prefix $pt' = (pt_1, \ldots, pt_{\ell-1})$ and (prefix) tag $pt = (pt', pt_\ell)$, output

$$sk'_{\mathbf{t},i} := (h_0, msk_{\mathbf{t}} \cdot h_\varepsilon \cdot \prod_{j=1}^{\ell-1} h_j^{pt_j} h_\ell^{pt_\ell}, h_{\ell+1}, \ldots, h_\lambda).$$

$\mathsf{KRerand}(sk_{\mathbf{t},i}, pt)$: on input secret key $sk_{\mathbf{t},i} =: (h_0, msk_{\mathbf{t}} \cdots, h_\ell, \ldots, h_\lambda)$ for prefix tag $pt' = (pt_1, \ldots, pt_\ell)$, for $(h'_0, h'_\varepsilon, h'_1, \ldots, h'_{2\lambda}) \leftarrow \mathsf{SampH}(pp)$, output

$$sk'_{\mathbf{t},i} := (h_0 \cdot h'_0, msk_{\mathbf{t}} \cdot h_\varepsilon h'_\varepsilon \cdot \prod_{j=1}^{\ell} h_j^{pt_j} \cdot \prod_{j=1}^{\ell} (h'_j)^{pt_j}, h_{\ell+1} \cdot h'_{\ell+1}, \ldots, h_\lambda \cdot h'_\lambda).$$

$\mathsf{CDel}(C_{pk_{\mathbf{t}},i}, pt)$: on input ciphertext

$$C_{pk_{\mathbf{t}},i} =: (g_0, g_\varepsilon \cdots, g_\ell, \ldots, g_\lambda, e(g, msk_{\mathbf{t}}) \cdot M)$$

for prefix $pt = (pt_1, \ldots, pt_{\ell-1})$, and (prefix) tag $pt = (pt', pt_\ell)$, output

$$C'_{pk_{\mathbf{t}},i} := (g_0, g_\varepsilon \cdot \prod_{j=1}^{\ell-1} g_j^{pt_j} g_\ell^{pt_\ell}, g_{\ell+1}, \ldots, e(g, msk_{\mathbf{t}}) \cdot M).$$

$\mathsf{CRerand}((C_{pk_{\mathbf{t}},i}, pt)$: on input ciphertext

$$C_{pk_{\mathbf{t}},i} =: (g_0, g_\varepsilon \cdots, g_\ell, \ldots, g_\lambda, e(g_0, msk_{\mathbf{t}}) \cdot M),$$

for prefix tag $pt = (pt_1, \ldots, pt_\ell)$ and $\ell' := |\mathbf{t}| - 1$, for $(g'_0, g'_\varepsilon, g'_1, \ldots, g'_\lambda) \leftarrow \mathsf{SampG}(pp)$, output

$$C'_{pk_{\mathbf{t}},i} := (g'_0 g_0, g_\varepsilon g'_\varepsilon \cdot \prod_{j=1}^{\ell} g_j^{pt_j} \cdot \prod_{j=1}^{\ell} (g'_j)^{pt_j}, g_{\ell+1} g'_{\ell+1}, \ldots, g_\lambda g'_\lambda,$$
$$e(g_0, msk_{\mathbf{t}}) \cdot e(g'_0, msk_{\mathbf{t}}) \cdot M).$$

Let $\mathsf{DSG} = (\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$ be a DSG scheme. We will construct a CPE scheme $\mathsf{RCPE} = (\mathsf{Gen}, \mathsf{KPunc}, \mathsf{CPunc}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} := G_T$ and tag space $\mathcal{T} := \mathbb{Z}_N$ (determined in pk_ε after running SampP in Gen). The construction of our CPE scheme RCPE from DSG is given in Figure 12.

$\mathsf{Gen}(\lambda)$: compute $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, \lambda+1)$, secret key $sk_\varepsilon := (h_0, h_\varepsilon \cdot msk_\varepsilon, h_1, \ldots, h_\lambda)$, for $msk_\varepsilon \leftarrow \mathbb{H}$, for $(h_0, h_\varepsilon, \ldots, h_\lambda) \leftarrow \mathsf{SampH}(pp)$, and public key $pk_\varepsilon := (pp, m(msk_\varepsilon))$, and return $(pk_\varepsilon, sk_\varepsilon)$.[a]

$\mathsf{KPunc}(pk_\mathbf{t}, sk_\mathbf{t}, t)$: on input $(pk_\mathbf{t}, sk_\mathbf{t})$ and tag $t \in \mathcal{T}$, outputs a public key, punctured secret key, and update token $(pk_{\mathbf{t}\cup\{t\}}, sk_{\mathbf{t}\cup\{t\}}, \Delta_{\mathbf{t}\cup\{t\}}) \leftarrow \mathsf{KTrunc}(pk_\mathbf{t}, sk_\mathbf{t}, t)$.

$\mathsf{Enc}(pk_\mathbf{t}, M, t_{\mathsf{exp}})$: on input public key $pk_\mathbf{t} =: (pp, m(msk_\mathbf{t}))$, message $M \in \mathcal{M}$, and expiry tag $t_{\mathsf{exp}} \leq 2^{\lambda'}, \lambda' \in [\lambda]$, for $(g_i, g_{i,\varepsilon}, g_{i,1}, \ldots, g_{i,\lambda}) \leftarrow \mathsf{SampG}(pp; s)$, for $s \leftarrow \mathbb{Z}_N$, find all tag prefixes $pt_i = (pt_{i,1}, \ldots, pt_{i,m})$, $m \in \mathbf{O}(\lambda)$, according to the binary tree that exclude $(\mathbf{t}, \ldots, t_{\mathsf{exp}})$, for $\ell := |\mathbf{t}|$, and compute

$$C_{pk_\mathbf{t}, t_{\mathsf{exp}}, i} := (g_{i,0}, g_{i,\varepsilon} \prod_{j=1}^{m} g_{i,j}^{pt_{i,j}}, g_{i,m+1} \ldots, g_{i,\lambda'}, m(msk_\mathbf{t})^s \cdot M),$$

Output $C_{pk_\mathbf{t}, t_{\mathsf{exp}}} := (C_{pk_\mathbf{t}, t_{\mathsf{exp}}, i})_i$.

$\mathsf{CPunc}(C_{pk_\mathbf{t}, t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$: on input ciphertext $C_{pk_\mathbf{t}, t_{\mathsf{exp}}}$, puncture token $\Delta_{\mathbf{t}\cup\{t\}}$, outputs $C_{pk_{\mathbf{t}\cup\{t\}}, t_{\mathsf{exp}}} \leftarrow \mathsf{CTrunc}(C_{pk_\mathbf{t}, t_{\mathsf{exp}}}, \Delta_{\mathbf{t}\cup\{t\}})$.

$\mathsf{Dec}(sk_\mathbf{t}, C_{pk_\mathbf{t}, t_{\mathsf{exp}}})$: on input secret key $sk_\mathbf{t} =: (sk_{\mathbf{t},1}, \ldots)$ and ciphertext $C_{pk_\mathbf{t}, t_{\mathsf{exp}}} =: (C_{pk_\mathbf{t}, t_{\mathsf{exp}}, 1}, \ldots)$ find the smallest tag $t \in \mathcal{T}$ that has not been punctures in either $sk_\mathbf{t}$ and $C_{pk_\mathbf{t}, t_{\mathsf{exp}}}$. (Such tag must exists.) Find the corresponding secret-key element $sk_{\mathbf{t},i}$ and the ciphertext element $C_{pk_\mathbf{t}, \mathbf{t}, i}$ that are associated to a prefix of $t = (t_1, \ldots, t_\lambda)$. (This is either a leaf-tag itself or a prefix with length $\lambda - 1$.) Compute $(h_0, msk_\mathbf{t} \cdot h_\varepsilon \cdot \prod_{j=1}^{\lambda} h_j^{t_j}, \ldots) \leftarrow \mathsf{KDel}(sk_{\mathbf{t},i}, t)$ and $(g_0, g_\varepsilon \cdot \prod_{j=1}^{\lambda} g_j^{t_j}, \ldots, e(g, msk_\mathbf{t}) \cdot M) \leftarrow \mathsf{CDel}(C_{\mathbf{t}, t_{\mathsf{exp}}, i}, t)$, and output

$$M := \frac{e(h_0, g_\varepsilon \cdot \prod_{j=1}^{\lambda} g_j^{t_j})}{e(g_0, msk_\mathbf{t} \cdot h_\varepsilon \cdot \prod_{j=1}^{\lambda} h_j^{t_j})} \cdot e(g, msk_\mathbf{t}) \cdot M.$$

[a] Note that sp will only be used in the security proof.

Fig. 12. RCPE from DSG.

Public verifiability of ciphertexts. As done for CPE, this holde due to the DSG's associativity property.

Correctness of RCPE. Correctness holds similarly to correctness of CPE, this due to DSG's associativity and projective properties (cf. Dec in Figure 12).

IND-CPE-CPA security of RCPE. Our proof strategy is essentially the same as the proof strategy for CPE adapted to the new t^*_{start} and how puncture tokens are generated. For completeness, we recap this here.

Theorem 3. *If* DSG *is a DSG scheme, then* RCPE *is restricted IND-CPE-CPA-secure. Concretely, for any PPT adversary A there are distinguishers D_1 on LS and D_2, D_3 on RS, respectively,*

$$\begin{aligned} \mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{RCPE},A}(\lambda) \leq\ & \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},\mathsf{G},D_1}(\lambda, 2\lambda+1) \\ & + \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D_2}(\lambda, 2\lambda+1) + \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D_3}(\lambda, 2\lambda+1). \end{aligned} \tag{10}$$

Proof. We show the restricted IND-CPE-CPA security of RCPE 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 give an overview how the challenge ciphertexts and the secret keys are derived in Table 2.

Game 0. The restricted IND-CPE-CPA experiment.

Game 1 Instead of directly using the ciphertext input by A, D decrypts and re-encrypts again starting from a ciphertext for mpk_ε. The change is conceptional. Also we re-write how the first secret key is derived.

Game 2. The challenge ciphertext is pseudo-normal.

Game 3. The secret keys are pseudo-normal.

Game 4. The secret keys are semi-functional pseudo-normal.
Game 5. The secret keys are semi-functional.
Game 6. The challenge ciphertext message is a uniform G_T-element.

	$\widehat{\mathsf{Enc}}$-output for challenge ciphertext	$\overline{\mathsf{Ext}}$-output for secret keys	Assumption
Game 0	Not used	Not used	-
Game 1	$\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g})$	$\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$	-
Game 2	$\underline{\widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g}\widehat{\mathbf{g}})}$	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	$\underline{\widehat{\mathsf{Enc}}(\widehat{h} \cdot msk'_\varepsilon, R; \mathbf{g}\widehat{\mathbf{g}})}$, for $R \leftarrow G_T$	As in Game 5	Non-degeneracy

Table 2. Output of $\widehat{\mathsf{Enc}}$ and $\overline{\mathsf{Ext}}$ to generate (challenge) ciphertexts and secret keys, for $\alpha \leftarrow \mathbb{Z}_N$, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, for $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and 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 8 (Game 0 to Game 1). *Game 0 and Game 1 are perfectly indistinguishable, i.e.,*

$$|\Pr[S_{A,0}] - \Pr[S_{A,1}]| = 0. \tag{11}$$

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 $C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}}$ (after positive verification via Ver) as input to CPunc to compute the challenge ciphertext, D decrypts $M \leftarrow \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, re-encrypts again by computing $C_{pk_{\mathbf{t}_{i+1}}} \leftarrow \mathsf{CPunc}(C_{pk_{\mathbf{t}_i}}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$ and $C_{pk_\varepsilon} \leftarrow \mathsf{Enc}(pk_\varepsilon, M, t^*_{\mathsf{exp}})$, for all $t_i \in \mathbf{t}$, for $\mathbf{t}$ the largest set in $\mathcal{L}$ (after inserting the target tag t^*).

This change cannot be noticed by A since the distributions of $C_{pk_{\mathbf{t}\setminus\{t^*\}}}$ and $C^*_{\mathbf{t}\setminus\{t^*\}, t^*_{\mathsf{exp}}}$ are perfectly indistinguishable due to the re-randomization properties of $\mathbb{G}$-elements as output by $\mathsf{SampG}(pp)$ and the public verifiability of $C^*_{\mathbf{t}\setminus\{t^*\}, t^*_{\mathsf{exp}}}$ using Ver. See that if Ver outputs 1, the G_T-element that blinds the message is fixed under the respective public key $pk_{\mathbf{t}\setminus\{t^*\}}$. Furthermore, we write $\overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$ to derive the first secret key which is only a re-write in different form to make the input of SampH explicit.

Lemma 9 (Game 1 to Game 2). *Under LS of* DSG*, Game 1 and Game 2 are computationally indistinguishable. Concretely, for any PPT adversary A in the restricted IND-CPE-CPA security experiment with* RCPE *there is a distinguisher D on LS such that*

$$|\Pr[S_{A,1}] - \Pr[S_{A,2}]| \leq \mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},\mathsf{G},D}(\lambda, 2\lambda + 1). \tag{12}$$

Proof. In Game 1, the challenge ciphertext is normal in the sense of RCPE 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 $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.

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

D samples $msk_\varepsilon \leftarrow \mathbb{H}$ and sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, $sk_\varepsilon := \overline{\mathsf{Ext}}(msk_\varepsilon; \mathbf{h})$, for $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers $\mathsf{KPunc}'(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t)$-queries, for $\mathbf{t}$ the currently largest set in $\mathcal{L}$, A-chosen $t \in \mathcal{T}$, and depending on $t^*_{\mathsf{start}}, \ldots, \min(t^*_{\mathsf{win}}, t^*_{\mathsf{exp}})$.

When A outputs $(M_0^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*})$ $= 0$ holds; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*, t^*_{\mathsf{exp}}; \mathbf{T})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{\mathbf{t}_i\cup\{t_i\}})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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, (12) 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 secret keys, also the further puncturings of the challenge ciphertext stay unnoticed in the view of A due to LS.

Lemma 10 (Game 2 to Game 3). *Under RS of* DSG*, Game 2 and Game 3 are computationally indistinguishable. Concretely, for any PPT adversary A in the restricted IND-CPE-CPA security experiment with* RCPE*, there is a distinguisher D on RS such that*

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D}(\lambda, 2\lambda + 1). \tag{13}$$

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 $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $\mathbf{h} \leftarrow \mathsf{SampH}(pp)$, $\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

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

First, D samples $msk'_\varepsilon \leftarrow \mathbb{H}$, sets $msk\varepsilon := \widehat{h} \cdot msk'_\varepsilon$, sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, sets

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

and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers KPunc'-queries (depending on $t^*_{\mathsf{start}}, \ldots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$), for $\mathbf{t}$ the currently largest set in $\mathcal{L}$ and A-chosen $t \in \mathcal{T}$.

When A outputs $(M_0^*, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*})$ $= 0$ holds; otherwise, D decrypts message $M_1^* := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M_b^*; \mathbf{g}\widehat{\mathbf{g}})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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. Particular, see that A only receives puncture tokens for $t^*_{\mathsf{start}}, \ldots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$. furthermore, see that msk and msk' are identically distributed and, in particular, $m(msk_\varepsilon) = m(msk'_\varepsilon)$ holds due to DSG's orthogonality property. Hence, (13) follows.

Lemma 11 (Game 3 to Game 4). *We have*

$$|\Pr[S_3] - \Pr[S_4]| = 0. \tag{14}$$

Proof. In Game 3, we have pseudo-normal secret keys while in Game 3 we have pseudo-normal semi-functional secret keys. We set $sk_\varepsilon := \overline{\mathsf{Ext}}((\widehat{h})^\alpha \cdot msk_\varepsilon; \mathbf{h}\widehat{\mathbf{h}})$, 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 4 in [CW14b]. Essentially, we use the parameter-hiding property of DSG to information-theoretically embed $(\widehat{h})^\alpha$. The results in pseudo-normal semi-functional keys. This even holds after the adversary sees the (punctured) challenge ciphertext for all α as shown in [CW14b] since due to non-degeneracy, we have that $(\widehat{h})^{\alpha}$ can be replaced by some suitable $(\widehat{h}_0)^{\alpha'}$, for $\widehat{\mathbf{h}} = (\widehat{h}_0, \dots) \leftarrow \mathsf{SampH}(pp, sp)$ and suitable $\alpha' \in \mathbb{Z}_N$. Hence, (14) follows. In particular, see that $m(msk_\varepsilon) = m(\widehat{h} \cdot msk_\varepsilon)$ and, hence, the adversary does not receive any further information on $\widehat{h}$ through the public keys.

Lemma 12 (Game 4 to Game 5). *Under RS of* DSG *hold, Game 4 and Game 5 are computationally indistinguishable. Concretely, for any PPT adversary A in the restricted IND-CPE-CPA security experiment with* RCPE*, there is a distinguisher D on RS such that*

$$|\Pr[S_4] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},\mathsf{G},D}(\lambda, 2\lambda+1). \qquad (15)$$

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 $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, and $(\mathbf{h}) \leftarrow \mathsf{SampH}(pp)$, $\widehat{\mathbf{h}} \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$.

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

First, D samples $msk'_\varepsilon \leftarrow \mathbb{H}$, sets $msk\varepsilon := \widehat{h} \cdot msk'_\varepsilon$, sets $pk_\varepsilon := (pp, m(msk_\varepsilon))$, sets

$$sk_\varepsilon := \overline{\mathsf{Ext}}((\widehat{h})^{\alpha} \cdot msk'_\varepsilon; \mathbf{T}),$$

for uniform $\alpha \leftarrow \mathbb{Z}_{\mathsf{ord}(\mathbb{H})}$, and sets $\mathcal{L} := \mathcal{L} \cup \{(\{\varepsilon\}, pk_\varepsilon, sk_\varepsilon, \perp)\}$. D starts A with λ and, during the entire experiment, answers KPunc'-queries (depending on $t^*_{\mathsf{start}}, \dots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$), for $\mathbf{t}$ the currently largest set in $\mathcal{L}$ and A-chosen $t \in \mathcal{T}$.

When A outputs $(M^*_0, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$, D outputs $b \leftarrow \{0,1\}$ if verification $\mathsf{Ver}(pk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*})$ $= 0$ holds; otherwise, D decrypts message $M^*_1 := \mathsf{Dec}(sk_{\mathbf{t}\setminus\{t^*\}}, C^*_{pk_{\mathbf{t}\setminus\{t^*\}}, t^*_{\mathsf{exp}}})$. Furthermore, if not already done via KPunc', D punctures keys via $(pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}) \leftarrow \mathsf{KPunc}(pk_{\mathbf{t}}, sk_{\mathbf{t}}, t^*)$ and updates the list $\mathcal{L} := \mathcal{L} \cup \{(\mathbf{t} \cup \{t^*\}, pk_{\mathbf{t}\cup\{t^*\}}, sk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*})$.

D computes $C_\varepsilon \leftarrow \widehat{\mathsf{Enc}}(msk_\varepsilon, M^*_b; \mathbf{g}\widehat{\mathbf{g}})$ and $C_{\mathbf{t}_{i+1}} \leftarrow \mathsf{CPunc}(C_{\mathbf{t}_i}, \Delta_{t_i})$, for $\mathbf{t}_0 := \varepsilon$, for all $t_i \in \mathbf{t} \cup \{t^*\}$. D sends to A:

$$(C_{\mathbf{t}\cup\{t^*\}}, pk_{\mathbf{t}\cup\{t^*\}}, \Delta_{t^*}).$$

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. Particular, see that A only receives puncture tokens for $t^*, \dots, \min(t^*_{\mathsf{win}}, t_{\mathsf{exp}}*)$. Hence, (15) follows.

Lemma 13 (Game 5 to Game 6). *Game 5 and Game 6 are statistically indistinguishable. Concretely, for any PPT adversary A on the IND-CPE-CPA security of* CPE*, it holds that*

$$|\Pr[S_{A,5}] - \Pr[S_{A,6}]| = 0. \qquad (16)$$

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 DSG's non-degeneracy property for this change. Concretely, for Game-5 challenge ciphertext

$$\widehat{\mathsf{Enc}}(\widehat{h} \cdot msk'_\varepsilon, M^*_b, t_{\mathsf{exp}}; \mathbf{g}\widehat{\mathbf{g}}) := (g_0\widehat{g}_0, \dots, e(g_0\widehat{g}_0, \widehat{h} \cdot msk_\varepsilon) \cdot M^*_b)$$

for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$, for $\widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, note that $e(\widehat{g}_0, \widehat{h})$, is uniformly distributed in a nontrivial subgroup $G'_T \subset G_T$ due to the non-degeneracy property of DSG.

Lemma 14 (Game 6). *For any PPT adversary A in the restricted IND-CPE-CPA security experiment with* RCPE*, it holds that*

$$\Pr[S_{A,6}] = 1/2. \qquad (17)$$

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 and (17) follows.

Taking (11), (12), (13), (14), (15), (16), and (17) together, shows (10). □

A.3 Weakly secure UE from restrictedly secure CPE

In this section, we construct updatable encryption (UE) with uni-directional key updates from restrictedly secure ciphertext-puncturing encryption (CPE). The transformation is rather straightforward and the same as given in given in Section 5. More concretely, let $\mathsf{RCPE} = (\mathsf{RCPE.Gen}, \mathsf{RCPE.KPunc}, \mathsf{RCPE.Enc}, , \mathsf{RCPE.CPunc}, \mathsf{RCPE.Dec})$ with message space $\mathcal{M}_{\mathsf{RCPE}}$ and tag space $\mathcal{T} = \{1, 2, \ldots, \lfloor e(\lambda) \rfloor\}$ be a restrictedly secure CPE scheme. We present our weakly secure UE scheme $\mathsf{UE} = (\mathsf{Gen}, \mathsf{Next}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} := \mathcal{M}_{\mathsf{RCPE}}$ in Figure 13 for completeness and further show correctness as well as weak IND-UE-CPA security.

$\mathsf{Gen}(\lambda)$: compute $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{CPE.Gen}(\lambda)$ and $(pk_1, sk_1, \varDelta_1) \leftarrow \mathsf{CPE.KPunc}(pk_\varepsilon, sk_\varepsilon, 1)$, and return $K_1 := (pk_1, sk_1)$.
$\mathsf{Next}(K_e)$: for $K_e =: (pk_e, sk_e)$, return $(K_{e+1} := (pk_{e+1}, sk_{e+1}), \varDelta_{e+1}) \leftarrow \mathsf{CPE.KPunc}(pk_e, sk_e, e + 1)$.
$\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$: for $K_e =: (pk_e, sk_e)$, compute $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(pk_e, M, e_{\mathsf{exp}})$.
$\mathsf{Update}(\varDelta_{e+1}, C_{e,e_{\mathsf{exp}}})$: return $\mathsf{CPE.CPunc}(C_{e,e_{\mathsf{exp}}}, \varDelta_{e+1})$.
$\mathsf{Dec}(K_e, C_e)$: for $K_e =: (pk_e, sk_e)$, return $\mathsf{CPE.Dec}(sk_e, C_{e,e_{\mathsf{exp}}})$.

Fig. 13. Construction of weakly IND-UE-CPA secure UE from restrictedly IND-CPE-CPA secure CPE.

For correctness, see that this directly translates from the CPE scheme, i.e., the ciphertext that was computed by Enc and updated via Update can be decrypted by Dec if the keys match and the ciphertext is not expired. We now turn to weak IND-UE-CPA security.

Theorem 4. *If* RCPE *is restrictedly IND-CPE-CPA secure, then* UE *is weakly IND-UE-CPA secure. Concretely, for any PPT adversary A there is a distinguisher D in the restricted IND-CPE-CPA security experiment, such that* $\mathsf{Adv}^{\mathsf{ind\text{-}cpe\text{-}cpa}}_{\mathsf{RCPE},D}(\lambda) \geq \mathsf{Adv}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$.

Proof. The difference in this proof compared to Theorem 1 is that utilize the starting epoch e^*_{start} which we will map to the starting tag t^*_{start} of the RCPE. Anything else stays the same, we highlight the changes, and recap the proof here for completeness.

We show the Theorem by constructing a PPT distinguisher D in the restricted IND-CPE-CPA security experiment with RCPE as defined in Figure 11 from any successful PPT adversary A in the weak IND-UE-CPA security experiment with UE as defined in Figure 5. The distinguisher D starts A and receives $e^*, e_{\mathsf{start}}, e_{\mathsf{end}}, e^*_{\mathsf{exp}}$. (If the A-input does not have the right distribution as defined in Figure 5, then output $b \leftarrow \{0, 1\}$.) It then outputs $(t^* := e^*, t^*_{\mathsf{win}} := e_{\mathsf{end}}, t^*_{\mathsf{exp}} := e_{\mathsf{exp}}, t^*_{\mathsf{start}} := e_{\mathsf{start}})$ to its IND-CPE-CPA challenger. Then, D queries $(pk_e, (sk_e), \varDelta_e) \leftarrow \mathsf{KPunc}(pk_{e-1}, sk_{e-1}, e)$ (with $\varepsilon := e - 1$), for all $e \in [\lfloor e(\lambda) \rfloor]$, from its restricted IND-CPE-CPA challenger (while D *does not* receive secret keys sk_e for $e = e_{\mathsf{start}}, \ldots, e^*, \ldots, \min(e_{\mathsf{end}}, e^*_{\mathsf{exp}})$). D sets $\varDelta_1 := \perp$. If $e_{\mathsf{exp}} \geq e_{\mathsf{end}}$, then D sends $(pk_e, sk_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}, \ldots, e^*, \ldots, e_{\mathsf{end}}\}}$ and $(\varDelta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}, e_{\mathsf{end}}\}}$ to A. Otherwise, i.e., if $e_{\mathsf{exp}} < e_{\mathsf{end}}$, then D sends $(pk_e, sk_e)_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}, \ldots, e^*, \ldots, e_{\mathsf{exp}}\}}$ and $(\varDelta_{e+1})_{e \in [\lfloor e(\lambda) \rfloor] \setminus \{e_{\mathsf{start}}\}}$ to A. Encryption queries in epoch $e \in [\lfloor e(\lambda) \rfloor]$ to $\mathsf{Enc}'(M)$ for expiry tag e'_{exp} are answered as follows: return $C_{e,e'_{\mathsf{exp}}} \leftarrow \mathsf{CPE.Enc}(pk_e, M, e'_{\mathsf{exp}})$ 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 if not. If $\tilde{e} < e^* - 1$, then iteratively run $C^*_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(C^*_{e-1,e_{\mathsf{exp}}}, \varDelta_e)$, for $e = \tilde{e}+1, \ldots, e^*$. D forwards $(M^*, C^*_{e^*-1,e_{\mathsf{exp}}}, e^*)$ to its restricted IND-CPE-CPA challenger. D receives $(C_b, pk_{e^*}, \varDelta_{e^*})$. D forwards C_b to A. Encryption queries with expiry epoch e'_{exp} in all epochs $e \in [e(\lambda)]$ to $\mathsf{Enc}'(M)$ are answered as $C_e \leftarrow \mathsf{CPE.Enc}(pk_e, M, e'_{\mathsf{exp}})$. Eventually, A outputs a guess b' which is forwarded to D's challenger. D is able to provide a consistent view for A for keys $(K_e)_e = (pk_e, sk_e)_e$. 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 weak IND-UE-CPA security experiment with UE, then D is a successful PPT adversary in the restricted IND-CPE-CPA security experiment with RCPE which shows the Theorem. □