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

Daniel Slamanig and Christoph Striecks

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

Abstract. Updatable encryption (UE) allows a third party to periodically rotate encryption keys from one epoch to another without the need to download, decrypt, re-encrypt and upload already encrypted data by a client. Updating those outsourced ciphertexts is carried out via the use of so-called update tokens which in turn are generated during key rotation and can be sent (publicly) to the third party. The arguably most efficient variant of UE is *ciphertext-independent* UE as the key rotation does not depend on the outsourced ciphertexts which makes it particularly interesting in scenarios where access to (information of the) ciphertexts is not possible during key rotation. Available security notions for UE cannot guarantee any form of *forward security* (i.e., old ciphertexts are in danger after key leakage). Counter-intuitively, forward security would violate correctness, as ciphertexts should be updatable ad-infinitum given the update token. In this work, we investigate if we can have at least some form of "controlled" forward security to mitigate the following shortcoming: an adversary would record available information (i.e., some ciphertexts, all update tokens) and simply would wait for a *single* key leakage to decrypt all data ever encrypted. Our threefold contribution is as follows:

a) First, we introduce an epoch-based UE CPA security notion to allow fine-grained updatability. It covers the concept of *expiry epochs*, i.e., ciphertexts can lose the ability of being updatable via a token after a certain epoch has passed. This captures the above mentioned shortcoming as the encrypting party can decide how long a ciphertext can be updatable (and, hence, decryptable).

b) Second, we introduce a novel approach of constructing UE which significantly departs from previous ones and in particular views UE from the perspective of puncturable encryption (Green and Miers, S&P'15). We define *tag-inverse* puncturable encryption as a new variant that generalizes UE and may be of independent interest.

c) Lastly, we present and prove secure the first UE scheme with the aforementioned properties. It is constructed via tag-inverse puncturable encryption and instantiated from standard assumptions. As it turned out, constructing such puncturing schemes is not straightforward and we require adapted proof techniques. Surprisingly, as a special case, this yields the first backwards-leak UE scheme with sub-linear ciphertexts from standard assumptions (an open problem posted in two recent works by Jiang Galteland and Pan & Miao et al., PKC'23).

Keywords: Updatable Encryption · Puncturable Encryption · Dual-System Groups.

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

* This is the extended version of [SS23] which appears in the Proceedings of the 21st Theory of Cryptography Conference (TCC 2023), Taipei, Taiwan, November 29 - December 2, 2023.

Google[1], Microsoft[2] or Amazon[3] recommend this practice and sometimes it is even mandated by regulations [Bar16, PCI22]. This raises the immediate question of how to efficiently update already outsourced encrypted data to new keys. An obvious solution for key rotation is to download the data, decrypt it locally under the old key, re-encrypt it under a new key and upload it again. Unfortunately, this imposes a significant overhead and soon becomes impractical.

Updatable Encryption. At CRYPTO 2013, Boneh, Lewi, Montgomery, and Raghunathan [BLMR13] proposed the concept of updatable encryption (UE). UE is a symmetric encryption primitive that addresses the above problem by allowing to update ciphertexts to new keys without the requirement for decryption by means of a so-called update token. UE schemes can be *ciphertext-dependent* [BLMR13, EPRS17, BEKS20, CLT20] where the key rotation depends on the specific ciphertext to be updated and, thus, to compute the update token, a part of every ciphertext needs to be downloaded. Or, and from an efficiency point more desirable, quite a number of recent works deal with UE schemes that are *ciphertext-independent* [LT18, KLR19, BDGJ20, Jia20, Nis22, GP23, MPW23] such that the key rotation is independent of any information of the ciphertexts in the system.

Particularly, we strive for scenarios where the key-rotating party may not have access to ciphertexts directly.[4] Hence, to be as generic as possible, in the remainder of this work, we focus on UE schemes with ciphertext-independent updates and will simply call them UE schemes. Such a UE scheme consists of the usual algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for key generation, encryption and decryption. Time is discretized in so-called epochs and Gen produces an initial secret key K_1 (for epoch 1). Additionally, there is an algorithm RotKey which takes a key K_e and outputs a next-epoch key K_{e+1} along with a so-called update token Δ_{e+1}. This update token can be used by a (semi-trusted) third party to update ciphertexts under key K_e for epoch e to ciphertexts for epoch $e+1$ under key K_{e+1} via an algorithm Upd.

Motivating Stronger Guarantees. When looking at state-of-the-art UE security models[5] [Nis22, GP23, MPW23], it can be observed that they do not capture forward security due the following shortcoming. Namely, an adversary would record available information (i.e., ciphertexts $(C_e)_e$, all update tokens $(\Delta_{e+1})_e$) in the lifetime of the system and simply would wait for a *single* key leakage $K_{e'}$ in epoch e' with $e' > e$. Such single key leakage allows to completely break confidentiality of all ciphertexts captured before.

While such a behavior is demanded by the correctness of current UE definitions (i.e., ciphertexts should always be updatable by a token to the current epoch and decryptable), it is a natural question if such a coarse-grained approach is necessary. Indeed, if we want to mitigate such type of shortcoming, we must introduce a more fine-grained adjustment of the updatability of ciphertexts. However, since we do not want to allow access to any (information of the) ciphertexts during key rotation, the token cannot carry the information needed which ciphertext should be updated and which not. Hence, the only possibility we see is via the encryption phase where information to limit the updatability can be embedded.

Post-compromise security for UE (i.e., leaking the current key does not endanger new ciphertexts), in contrast, is already well understood and constructions in the currently strongest security model are available [Jia20, Nis22, GP23, MPW23]. Unfortunately, even having strong guarantees such as the ones

[1] https://cloud.google.com/kms/docs/keyrotation

[2] https://docs.microsoft.com/en-us/azure/storage/blobs/security-recommendations

[3] https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html

[4] One can think of a user that holds encrypted sensitive data with a current key from some key management system and gets offline for some time before the ciphertexts should be decrypted again. However, during the user's offline time, key rotations might be executed. This issue was also mentioned during a talk at RWC 2023 on Google's crypto agility concerning key rotation [KPMS23].

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

provided by Nishimaki [Nis22], the orthogonal security property of *forward security* (i.e., leaking the current key does not endanger *old* ciphertexts) cannot be met by any of the known UE models and constructions.

Extended UE Security with Expiry Epochs. As an important security feature, forward security was already considered in many cryptographic contexts such as interactive key-exchange protocols (e.g., TLS 1.3, QUIC, hybrid KE, or ratcheting) [Gün90, DvOW92, DDG+20, BRS23, RSS23], public-key encryption [CHK03, Gro21], digital signatures [BM99, DGNW20], search on encrypted data [BMO17], mobile Cloud backups [DCM20], proxy cryptography [DKL+18], new approaches to Tor [LGM+20], and distributed key management [DRSS21], among others.

We believe that forward security yields an important feature for UE in practice as well and should be inherently considered in full in the UE regime.[6] Google's key management system for instance sets the maximum age for key-wrapping ciphertexts to 90 days[7], which shows that ciphertexts *should not* be made available forever in real-world systems to mitigate the risk after key leakages. While forward security can be achieved in *ciphertext-dependent* UE in a fine-grained way (as access to parts of the ciphertexts is allowed during key rotation), we want to stress that the situation for ciphertext-independent UE is precarious as the key rotation does not have access to any ciphertext information.

As we show in this work, restricting the update capabilities during the encryption phase yields fine-grained updatability. In particular, we introduce the concept of expiry epochs such that for every ciphertext, one can decide during encryption time how long updates should yield decryptable ciphertexts, i.e., encryption in epoch e is performed as $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_e, M, e_{\mathsf{exp}})$ and when epoch e_{exp} is reached, a ciphertext cannot longer be updated into a decryptable ciphertext. Note that an update token should still work for all ciphertexts that have an expiry epoch in the future. Also, by virtually never letting ciphertexts expire, i.e., using $e_{\mathsf{exp}} = 2^\lambda$ for all encryptions with $\lambda \in \mathbb{N}$ being the security parameter, we are essentially back in the currently strongest models [Nis22, GP23, MPW23].

This conceptually simple modification has an interesting effect. Namely, to meet our proposed UE security notion, we at least require the UE scheme to solely allow ciphertext updates via the token in the forward direction. So far, such UE schemes are only known to exist by relying on indistinguishability obfuscation [Nis22]. Our notion even requires more and is particularly not implied by [Nis22] which makes the task of constructing such a UE scheme non-trivial (there exists no such UE scheme so far). Moreover, since UE is inspired mainly by practice, we want constructions from standard assumptions and where key, ciphertext and token sizes are as compact as possible, but certainly sub-linear in the maximum number of possible epochs. While compactness can be achieved in weaker models or from non-standard assumptions [BDGJ20, Jia20, Nis22], this important feature turned out to be non-trivial in our model. Notably, the strongest UE schemes from standard assumptions [GP23, MPW23] have linear-size ciphertexts and constructing schemes with sub-linear ciphertext expansion was posted as a significant open problem in the aforementioned works.

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

[6] Prior works offer only a very weak form of forward security by restricting access to tokens artificially.

[7] https://cloud.google.com/docs/security/key-management-deep-dive/resources/google-cloud-kms-deep-dive.pdf, Sec. 4.2

In UE, rotating keys from one epoch to the next is abstractly reminiscent of puncturing when viewing tags as epochs. Loosely speaking, puncturing a key on an epoch e would make all ciphertexts in epoch e inaccessible (if all ciphertexts are "tagged" on e). However, this would not yield a useful UE scheme and one needs a mechanism to transform ciphertexts to the next epoch $e+1$.

From a forward-security point of view, we want that for some ciphertexts, the punctured key should not work while for some other ciphertexts, the punctured key should indeed work. Interestingly, we can make the puncturing more fine-grained by tagging ciphertexts not only on the current epoch e but allow also each ciphertext to be associated with a unique tag.

Such a view has an interesting effect. Namely, the crucial point is that puncturing would take the current epoch key as input, but also tags for ciphertexts for which the key should *not* be punctured. Notably, see that such a feature partly *inverts* the view of plain PE. The output of such a puncturing algorithm would be the punctured key (as in plain PE), but also some information which ciphertext should be excluded from puncturing (see that we do not want to allow access to any ciphertext during puncturing).

To transport such information, the concept of update tokens for ciphertexts must be introduced. Consequently, via such tokens, only ciphertexts that are not intended to be punctured in the key or are not expired[8] yet will be decryptable after key puncturing while for all other ciphertexts, the key will be punctured and, hence, decryption will fail. Indeed, such a puncturable view abstractly yields a UE scheme with "controlled" forward-security guarantees.

1.1 Our Contribution

Briefly summarized, our contribution is as follows:

a) First, we simplify and extend the state-of-the-art UE chosen-plaintext security models [Nis22, GP23, MPW23] to capture the guarantees provided by UE schemes that restrict the function of update tokens to ciphertext updates in the forward direction only. Importantly, we introduce *expiry epochs* as a fine-grained updatability feature. By letting ciphertexts expire, we can mitigate the "record now, leak later" attack discussed above. Moreover, we show that our notion implies the most recent chosen-plaintext UE notion due to [Nis22, GP23].

b) Second, we introduce a novel primitive dubbed Tag-Inverse Puncturable Encryption (TIPE) which we believe provides an easier intuition towards UE with "controlled" forward-security guarantees. We prove that the TIPE notion implies our UE notion. In particular, we believe that the tag-inverse puncturing in TIPE will further increase the applicability of the already very useful concept of puncturable encryption [GM15] and might be of independent interest.

c) Lastly, we construct a TIPE scheme that is secure in our model and thus yields the first UE scheme with such strong properties. Moreover, its security is based on the standard d-Lin assumption (where for $d = 1$ we get SXDH) in prime-order bilinear groups using the well-known dual-system paradigm [Wat09, Lew12, CW13, GCTC16]. Indeed, to overcome the hurdles towards TIPE with such strong properties, we require novel construction and adapted proof techniques. Noteworthy, our UE scheme enjoys sub-linear key and ciphertext sizes (yielding the first backward-leak UE scheme with such properties as a special case and answering an open problem from [MPW23, GP23] in the affirmative).

Modelling Key Security Features with Expiry Epochs. Post-compromise security (PCS) is an essential security guarantee in UE and the currently strongest schemes under standard assumptions are known to achieve it [Nis22, GP23, MPW23]. PCS loosely speaking means: once an old key leaks, future ciphertexts are not in danger even in the presence of tokens. This is due to the fact that tokens can be neither used to update keys in the forward direction nor for updating ciphertexts in the backward

[8] We introduce expiry epoch analogously to our UE model.

direction. We achieve PCS as we are building on the strongest prior (game-based) chosen-plaintext-secure model.

Moreover, the introduction of expiry epochs allows us to achieve forward security (FS) when keys are leaked beyond that expiry epoch of a ciphertext — where by FS we mean, again loosely speaking: once a key leaks, expired ciphertexts are not in danger even in presence of tokens. Recall that by correctness of UE schemes *without* expiry epochs, such strong form of FS *cannot* be met and is indeed not foreseen in any prior model (i.e., once a key leaks, old ciphertexts are immediately in danger when access to all tokens is granted).

In contrast to prior models, we particularly consider the attack that an adversary can use a token Δ_e to update a key K_e to K_{e-1} in the backward direction (i.e., yielding a key that is consistent with epoch-$(e-1)$ ciphertexts and which would break FS). Indeed, the currently strongest UE schemes [Nis22, GP23, MPW23] allow for such an attack. In contrast, in our model, we aim for mitigating such a leakage. Say, we have a ciphertext that expires in epoch e^*. The token in the expiry epoch Δ_{e^*+1} should not be of help to update a key K_{e^*+1} to an expiry-epoch key K_{e^*} in the backward direction and we explicitly provide the adversary with capabilities to query such a token.

Noteworthy, when setting the expiry epoch of each ciphertext to 2^λ (for security parameter λ), we achieve the same security guarantees as [Nis22, GP23, MPW23] which we formally show. Moreover, we can even show that our notion implies a simple and natural "ciphertext indistinguishability" notion where challenge ciphertexts with different expiry epochs are indistinguishable in the same challenge epoch.

Tag-Inverse Puncturable Encryption as an Abstraction of UE. In a nutshell, TIPE can be viewed as a symmetric PE scheme $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{Dec})$ with an additional algorithm ExPunc to control which ciphertexts are excluded for key puncturing. Such a scheme is associated to a polynomial sized set of epochs, i.e., $(1, \ldots, n)$, as well as an unbounded ciphertext-tag space $\mathcal{T}$. KPunc sequentially punctures keys on epochs, i.e., removes the ability to decrypt ciphertexts tagged under them step by step. In addition, KPunc can take a set of tags $\mathcal{S} \subseteq \mathcal{T}$ (or a special "for-all" tag $\forall$) and outputs an update token, which can then be used to exclude ciphertexts carrying tags in $\mathcal{S}$ from puncturing (in case of $\forall$, all ciphertext can be excluded). In TIPE, ciphertexts are not only computed w.r.t. a tag $t \in \mathcal{T}$, but additionally take an "expiry-tag" $e_{\mathsf{exp}} \in [n]$. If epoch e_{exp} is reached, then for ciphertexts carrying such a tag, the key is implicitly punctured and such a ciphertext cannot be excluded from puncturing anymore.

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

Indeed, we can see that UE is essentially a special variant of TIPE. For UE, it is sufficient to set $\forall$ as input to key puncturing KPunc and fix the ciphertext-tag set of TIPE to a singleton $\mathcal{T} = \{t\}$ for any arbitrary tag t. Particularly, we use key puncturing for rotating to the next UE key (i.e., "puncturing" on the TIPE epoch e) and exclude puncturing for ciphertexts via a token, i.e., ciphertexts can be updated to the next epoch (where ciphertexts with expired epochs are punctured implicitly). Encryption and decryption directly map to UE's encryption and decryption functionality, respectively. Moreover, we believe that TIPE provides an interesting abstraction for protected outsourced file storage with forward security and fine-grained secure shredding of files (in the vein of puncturable key wrapping [BGP22], but augmented with efficient key rotation).

Idea of Instantiating TIPE. The main construction idea is the following. The first ingredient is a special encoding mapping epochs to encoded binary epochs. See that such binary epochs have only a length of λ while allowing 2^λ epochs. One can think of it as a binary-tree encoding as discussed

in detail in [DGNW20, Sec. 4.2] where nodes are labeled as epochs, e.g., epochs $e_0 = (0,0,0), e_1 = (0,0,1), \ldots, e_7 = (1,1,1)$ encoding 2^3 epochs. However, this is not sufficient and we need a second ingredient, namely, group elements from the dual-system groups [LW10, LW11, CW13], to support such an encoding in the final TIPE scheme.

We use a special variant of dual-system groups (DSGs) due to Gong et al. [GCTC16] (which is based on [CW13]) and that was used to build an unbounded hierarchical identity-based encryption (HIBE) scheme [Lew11]. Indeed, our scheme is closely related to unbounded HIBEs, but we need more features which an HIBE cannot guarantee. (Particularly, ciphertext updates are not foreseen in HIBEs.) Fortunately, we observe that the above DSG from [GCTC16] has more to offer than implying unbounded HIBEs, which was not known before. Interestingly, we can even use a relaxed version of their DSG and leave out unnecessary features whereby we did not add anything to their syntax, correctness or security. As a consequence, we can safely assume that our relaxed DSG variant is implied by the full DSG variant from [GCTC16] and we give a concrete prime-order instantiation from the standard d-Lin assumption in the standard model in Appendix A.

Starting from the initial work by Waters [Wat09], the richness of the dual-system paradigm was demonstrated in several prior works (e.g., [LW10, LW11, OT12, CW13, HKS15, AHY15, GCD+16, GCTC16, GWW19, GW20, DKW23]). The abstraction concept of [GCTC16] is particularly useful as it provides us with functionalities that are also essential in the TIPE (and, hence, UE) paradigm where ciphertext updates only work in the forward direction. This connection is new and enriches the applications of the dual-system paradigm. We will now use such a connection for our TIPE construction.

To construct a TIPE scheme from DSGs with the encoding from [DGNW20], we implicitly encode epochs in keys in a complete binary tree, i.e., the nodes represent a prefix bit representation of the epoch and, hence, the root of the tree is associated with key K^{ε} (an initial key used to bootstrap *all* epoch keys). In its basic form, this is not new and reminiscent of prior works, e.g., work on forward secure public-key encryption [CHK03, GHJL17, DJSS18, DKL+18]. However, we need to add ciphertext updates as well and enhance it to incorporate tokens.

Fortunately, the DSG approach allows us to also encode epochs in ciphertexts in a complete binary tree similarly to keys, i.e., the nodes represent a prefix bit representation of epochs and, hence, the root of the tree is associated with ciphertext $C^{\varepsilon}_{t,e_{\mathsf{exp}}}$ (an initial ciphertext). The more the epochs advance, depending on the configuration of the tree, the more ciphertext elements are required. See Fig. 1 for an illustrative example how keys and ciphertext are constructed on the intuitive level.

As a ciphertext can have many elements, the main hurdle is the common randomness that blinds the message part where such a randomness has to be "associated" to all ciphertext parts (otherwise decryption will not work). Surprisingly, we can use techniques from [GCTC16] where ciphertexts have "local" randomnesses and one "global" randomness, where the latter is used to hide the message part and the local randomness are required to mitigate mix-and-match attacks.

The correctness guarantees are now that a ciphertext for epoch $e_0 = (0,0,0)$ (encoded using [DGNW20]) can be decrypted by a key in epoch e_0, but not with keys in later epochs.[9] When a key is updated to $e_1 = (0,0,1)$, the node containing key elements from $K^{000}_{e_1}$ is discarded and all other key elements will have a uniform "linear shift" in the exponents of their group elements. (Such a shift can be seen as switching the master secret key in an HIBE and we use the same shift for all key elements.)

Moreover, an update token is generated and incorporates the linear shift (as we have to transport this information to the ciphertexts). Such a token will not work on any key components, but can be used to "lift" a ciphertext in epoch e_0 to a valid ciphertext in epoch e_1. The tokens work only on ciphertext element C^{000}_{t,e_0} which results in a "shifting" element that can be used to update all other ciphertext elements. The shift operation ensures that the secret key and the ciphertext are in sync again. After the shifting is done, C^{000}_{t,e_0} is discarded. This results in a ciphertext that cannot be decrypted by prior-epoch key elements as the token does not work on the updated ciphertexts to "undo" the shifting operation.

[9] Also later-epochs keys cannot decrypt prior-epoch ciphertexts.

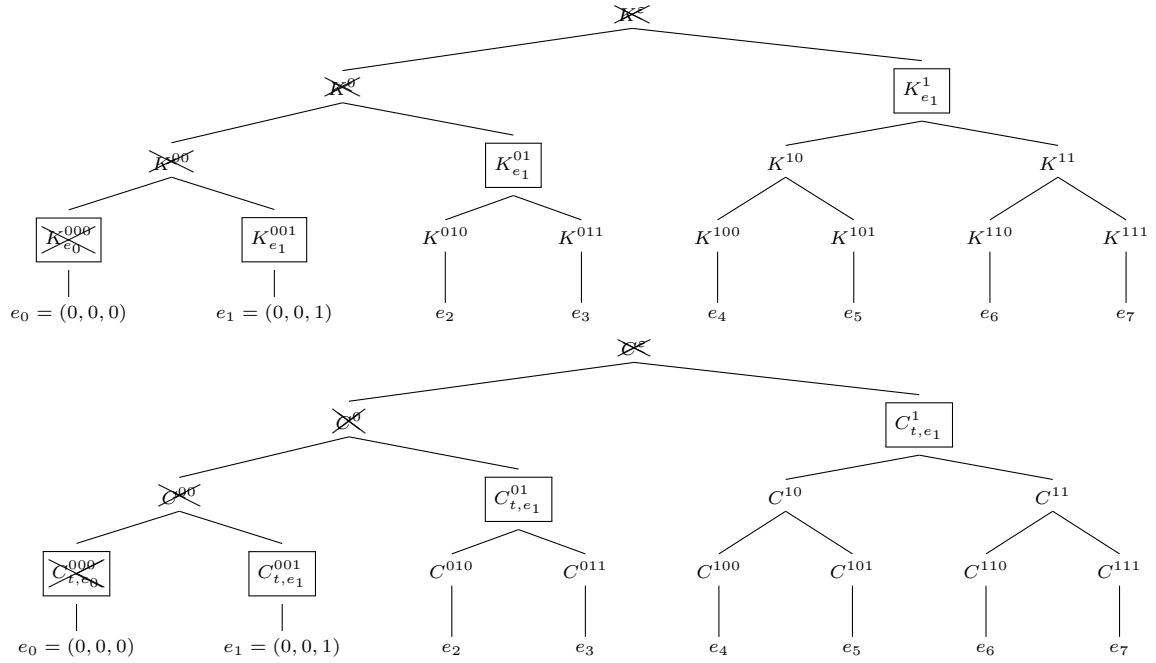

Fig. 1. Example of a TIPE key $K_{e_1} = (K^1_{e_1}, K^{01}_{e_1}, K^{001}_{e_1})$ that has been punctured on epoch $e_0 = (0,0,0)$. Moreover, a token tailored to t and epoch e_0 is generated during puncturing and can be used to update the ciphertext C_{t,e_0} to $C_{t,e_1} = (C^1_{e_1}, C^{01}_{e_1}, C^{001}_{e_1})$. Only the boxed elements have to be stored and the remaining elements lower in the tree can be derived from those.

This is enforced by carefully tailoring the token to epoch e_0 (i.e., only working when C^{000}_{t,e_0} is present), as well as to the tag t (which excludes that the token works on any other ciphertext in epoch e_0 if the tags are different to t). Particularly, since C^{000}_{t,e_0} is not contained in C_{t,e_1} anymore, such a token cannot be used to update the ciphertext from C_{t,e_1} to C_{t,e_0}. Together with expiry epochs (that can be integrated straightforwardly thanks to using the binary tree and a simple pruning of the tree accordingly), this yields the desired TIPE properties.

Concerning security, the proof methodology makes use of the dual-system paradigm where keys and ciphertexts can be of two forms, i.e., normal or semi-functional. Any combination of both will decrypt correctly as long as the ciphertext and the key are not both semi-functional. In a first step, the challenge ciphertext is made semi-functional via introducing semi-functional components into such a ciphertext. We stress that such a semi-functional ciphertext can be decrypted by key elements coming from the normal distribution, but will fail with high probability for semi-functional key components. This is exactly what we will use in the remainder of the proof where we carefully introduce uniform randomness into each key and token components which have key or token elements associated to the challenge ciphertext, respectively. This is done as in the usual dual-system paradigm but adapted to our setting where we can embed uniform randomness into such components and their associated tags or epochs encodings are not "prefixes" of the challenge ciphertext tag or epoch encoding.

1.2 Preliminaries and Outline

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 $\forall c \in \mathbb{N}\ \exists k_0\ \forall \lambda \geq k_0 : |f(\lambda)| < 1/\lambda^c$). We write $\mathbf{v} = (v_i)_{i\in[n]}$, for $n \in \mathbb{N}$. We may also write vectors in bold fonts which depends on the context, i.e., we use a component-wise multiplication of vectors, i.e., $\mathbf{v}\cdot\mathbf{v}' = (v_1, \ldots, v_n)\cdot(v'_1, \ldots, v'_n) = (v_1\cdot v'_1, \ldots, v_n\cdot v'_n)$.

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

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

Outline of the Paper. In Sec. 2, we present our security model with expiry epochs and discuss relations to previous models. In Sec. 3, we introduce Tag-Inverse Puncturable Encryption (TIPE), show how we can instantiate UE from TIPE and present a concrete TIPE construction. Moreover, we briefly discuss other applications of TIPE.

2 Updatable Encryption with Expiry Epochs

We define UE with expiry epochs in ciphertexts which allow ciphertexts to being excluded from updates. We build on the recent UE models [Nis22, GP23, MPW23]. The main idea of UE with expiry epochs is the following. On the very high level, all operations are bound to discrete epochs $1, 2, \ldots$ where keys and ciphertexts as well as so-called update tokens are associated to. System setup Gen creates a first-epoch symmetric key K_1. As an illustration, with this key, one can create a first-epoch ciphertext $C_{1,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_1, M, e_{\mathsf{exp}})$, for some message M and expiry epoch $e_{\mathsf{exp}} > 1$, and, e.g., outsource $C_{1,e_{\mathsf{exp}}}$ to some semi-trusted third-party. With probabilistic algorithm RotKey, K_1 can be updated (or, rotated) to K_2 while also an update token Δ_2 is generated. With Δ_2, a semi-trusted third-party can update $C_{1,e_{\mathsf{exp}}}$ to $C_{2,e_{\mathsf{exp}}} \leftarrow \mathsf{Upd}(\Delta_2, C_{1,e_{\mathsf{exp}}})$ such that $C_{2,e_{\mathsf{exp}}}$ is "consistent" with K_2. Correctness guarantees that decryption of $C_{2,e_{\mathsf{exp}}}$ yields $M = \mathsf{Dec}(K_2, C_{2,e_{\mathsf{exp}}})$ as intended (and so on, if the ciphertext is not expired already). More formally:

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

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

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

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

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

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

Correctness. We require that an honestly generated epoch-j ciphertext $C_{j,e_{\mathsf{exp}}}$ (obtained via $\mathsf{Enc}(K_j, M, e_{\mathsf{exp}})$) is decryptable to M if the expiry epoch is not reached yet. Moreover, an honest update of a valid ciphertext $C_{j,e_{\mathsf{exp}}}$ (via Δ_{j+1}) from epoch j to $j+1$ yields a valid ciphertext $C_{j+1,e_{\mathsf{exp}}}$ that can be decrypted under the epoch key K_{j+1} (obtained via $\mathsf{RotKey}(K_j)$) if the ciphertext is not already expired. (See that we do not give any correctness guarantees beyond the expiry epochs.)

More formally, for all $\lambda \in \mathbb{N}$, for all $e \in [\lfloor \mathsf{poly}(\lambda) \rfloor]$, for $K_1 \leftarrow \mathsf{Gen}(\lambda)$, for all $i \in \{1, \ldots, e\}$, for all $(K_{i+1}, \Delta_{i+1}) \leftarrow \mathsf{RotKey}(K_i)$, for all $M \in \mathcal{M}$, for all $e_{\mathsf{exp}} \in \mathbb{N}$, for all $j \in \{1, \ldots, e+1\}$, for all

$C_{j,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_j, M, e_{\mathsf{exp}})$, we require that $M = \mathsf{Dec}(K_j, C_{j,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq j$. Moreover, for all $j \in \{1, \ldots, e\}$, for all $C_{j,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_j, M, e_{\mathsf{exp}})$, for all $i \in \{j, \ldots, e\}$, for $C'_{j,e_{\mathsf{exp}}} := C_{j,e_{\mathsf{exp}}}$, for all $C'_{i+1,e_{\mathsf{exp}}} \leftarrow \mathsf{Upd}(\Delta_{i+1}, C'_{i,e_{\mathsf{exp}}})$, we require that $M = \mathsf{Dec}(K_{e+1}, C'_{e+1,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq e+1$.

Intuition of Our Security Notion. The notion is an extension of prior-work notions [Nis22, GP23] which we augment with the introduction of expiry epochs. (We later show that our notion implies such prior-work notions.) The distinguishing feature of our model is that we allow the adversary to query the update token in an epoch which corresponds to the challenge-ciphertext expiry epoch e_{exp} *and* allow that the key in $e_{\mathsf{exp}}+1$ can be retrieved. In models without expiry epochs, leaking the epoch-e challenge ciphertext and an epoch-e token together with a key in epoch $e+1$ would yield a trivial win (per definition of correctness).

More concretely, in each epoch, the adversary has access to several oracles as given in Fig. 2. Thereby, we assume that any (expiry-)epoch information is explicitly retrievable from the keys, tokens, and ciphertexts. The adversary is allowed to query honestly generated ciphertexts (with chosen expiry epochs) via Enc'. RotKey' triggers rotation of the current key to the next epoch. Upd' is an oracle that updates chosen but honestly generated ciphertexts to the current epoch (depending on the expiry epoch). $\mathsf{Corrupt}$ lets the adversary retrieve either the token or the key for a specific epoch. Chall takes a chosen message and a chosen but honestly generated ciphertext, and outputs a challenge ciphertext (depending on a uniform bit b). $\mathsf{GetUpdC}^*$ returns the current challenge ciphertext. At the end of the experiment, we require that the adversary did not retrieve keys to trivially decrypt the challenge ciphertext or retrieves an update token to update the challenge ciphertext to an epoch where it has a key. If such validity checks pass and the adversary guessed b correctly, then the adversary wins the experiment. Otherwise, we say that a UE scheme is $\mathsf{EE\text{-}IND\text{-}UE\text{-}CPA}$ secure as such a notion essentially ensures that fresh and updated ciphertexts are indistinguishable even if the adversary has access the aforementioned oracles adaptively. We define:

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

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

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}$ *is defined as in Fig. 2.*

2.1 Relation to Other UE Security Notions

Our security notions implies IND-UE-CPA security as defined in [GP23, Nis22]. Moreover, we can even show that our notion implies a simple and natural ciphertext indistinguishability notion for chosen messages and (public) expiry epochs where challenge ciphertexts with different expiry epochs are indistinguishable with respect to chosen messages in the same challenge epoch.

EE-IND-UE-CPA Implies IND-UE-CPA for UE with $e_{\mathsf{exp}} = 2^\lambda$. The security notions of IND-UE-CPA as given in [GP23, Nis22] are closely related to ours with the exception that ours allows expiry epochs. In the following, we show that our notions imply the prior CPA notion for an UE scheme with $e_{\mathsf{exp}} = 2^\lambda$ for large enough $\lambda \in \mathbb{N}$ via a simple reduction. Before that, we recap the IND-UE-CPA notion.

We adapt the most recent IND-UE-CPA notion from [GP23] (which is also used in [Nis22]) and left out unnecessary details such as deterministic updates, CCA, and some sets for the leakage profiles. Indeed, to show our implication, we require that any successful IND-UE-CPA adversary yields a successful EE-IND-UE-CPA adversary. The contrary does not hold. The reason is that in such a hypothetical reduction, we cannot simulate encryption queries and a valid challenge with different expiry epochs, and certainly cannot provide the token $\Delta_{e^*_{\mathsf{exp}}+1}$ to the EE-IND-UE-CPA adversary as this would yield a trivial win via the non-validity of the adversary in the IND-UE-CPA security experiment. Notably, here the currently strongest notion of IND-UE-CPA is presented, i.e., security for backward-directional key updates.

Experiment $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{RotKey}',\mathsf{Upd}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUpdC}^*}(\lambda)$
if A is not valid, then return $b'' \leftarrow \{0,1\}$
if $b = b'$, then return 1 else return 0

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

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

Fig. 2. Our EE-IND-UE-CPA security notion for UE schemes with expiry epochs.

Definition 3 ([GP23]). *A UE scheme* UE *is* $\mathsf{IND\text{-}UE\text{-}CPA}$*-secure iff for any PPT adversary* A*, the advantage function*

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

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

In [Nis22, GP23], efficient UE schemes were presented that fulfill the IND-UE-CPA notion. Those schemes allow key updates in the backwards direction. We briefly want to mention that any potential UE scheme that allows key updates in the backward direction *cannot* be EE-IND-UE-CPA-secure. The reason is that our notion allows the adversary to query the update token $\Delta_{e_{\mathsf{exp}}+1}$ in the expiry epoch and the key $K_{e_{\mathsf{exp}}+1}$ right after the expiry epoch of the challenge ciphertext. In a backward-directional key update setting, the adversary can take $K_{e_{\mathsf{exp}}+1}$ and $\Delta_{e_{\mathsf{exp}}+1}$ to derive $K_{e_{\mathsf{exp}}}$ which results in a adversarial win in our notion. Hence, this yields a trivial UE scheme separation between IND-UE-CPA and EE-IND-UE-CPA security as we strictly have to rely on UE schemes where tokens cannot update secret keys in any direction.

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

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

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

Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}', \mathsf{RotKey}', \mathsf{Upd}', \mathsf{Corrupt}, \mathsf{Chall}, \mathsf{GetUpdC}^*}(\lambda)$
if A is not valid, then return $b'' \leftarrow \{0,1\}$
if $b = b'$, then return 1 else return 0

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

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

Fig. 3. The IND-UE-CPA security notion for UE schemes.

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

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

Ciphertext Indistinguishability for Chosen Messages under (Public) Expiry Epochs. Next, we introduce a simple and natural relaxation of our EE-IND-UE-CPA security notion, which we dub EE-IND-UE-ENC, to show security of ciphertexts for chosen messages under different expiry epochs but in the same challenge epoch. See that such a property is not immediately achieved in our stronger EE-IND-UE-CPA notion since there we guarantee indistinguishability of fresh and updated ciphertexts that have the same expiry epoch in the same challenge epoch. However, as we show, our EE-IND-UE-CPA implies the EE-IND-UE-ENC notion.

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

The intuition is that in our EE-IND-UE-CPA notion, the adversary is capable of querying encryptions (through Enc') for different expiry epochs in the challenge epoch. While this intuitively yields ciphertext

indistinguishability for different expiry epochs, we want to make it more formal next. The key feature is that the adversary queries the challenge oracle Chall with two messages and two expiry epochs of its choice. In the challenge phase, it has to distinguish for which expiry epoch which message was encrypted, where the choice in the challenge ciphertexts is determined by a uniformly random bit b which the adversary has to guess to succeed in the notion.

Definition 4. *A UE scheme* UE *is* $\mathsf{EE\text{-}IND\text{-}UE\text{-}ENC}$*-secure iff for any PPT adversary* A*, the advantage function*

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

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

Experiment $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}enc}}_{\mathsf{UE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}', \mathsf{RotKey}', \mathsf{Upd}', \mathsf{Corrupt}, \mathsf{Chall}, \mathsf{GetUpdC}^*}(\lambda)$
if A is not valid, then return $b'' \leftarrow \{0,1\}$
if $b = b'$, then return 1 else return 0

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

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

Fig. 4. The EE-IND-UE-ENC security notion for UE schemes with expiry epochs.

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

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

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

If A queries Chall with $(M^*_0, M^*_1, e^*_{\mathsf{exp},0}, e^*_{\mathsf{exp},1})$, D queries its Chall-oracle with $M^*_0, M^*_1, e^*_{\mathsf{exp},0}$ and its Enc'-oracle with $(M^*_b, e^*_{\mathsf{exp},1})$, for uniform $b \leftarrow \{0,1\}$. D retrieves $C^*_{e,0}$ from Chall and $C^*_{e,1}$ from Enc' which D both forwards to A as the answer of A's Chall-oracle. In half of the cases, this yields a consistent contribution for A's challenge ciphertexts.

In the challenge phase, RotKey' switches the epochs and lets the EE-IND-UE-CPA challenger updates the challenge ciphertext $C^*_{e,0}$ while $C^*_{e,1}$ is updated via the Upd'-oracle (which is possible since $C^*_{e,1}$ is a

valid ciphertext queried from D's Enc'-oracle). If A queries the updated challenge ciphertexts, D returns those.

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

3 UE from a Puncturable-Encryption Perspective

We introduce a novel primitive dubbed Tag-Inverse Puncturable Encryption (TIPE), provide a TIPE security model and show how to construct UE with expiry epochs from TIPE. Finally, we give an instantiation of TIPE under standard assumptions. TIPE essentially views UE from the perspective of puncturable encryption [GM15] and generalizes UE. We believe that TIPE has application beyond UE.

3.1 Tag-Inverse Puncturable Encryption

The main intuition of TIPE 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 update tokens are associated to. Moreover, as in plain PE, ciphertexts are attached with a tag coming from an exponentially large tag space $\mathcal{T}$. System setup Gen creates a first-epoch symmetric key K_1. To show the idea, with such a key, one can create a first-epoch ciphertext $C_{1,t,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(K_1, t, M, e_{\mathsf{exp}})$, for some message M, tag $t \in \mathcal{T}$, and expiry epoch $e_{\mathsf{exp}} > 1$. With probabilistic algorithm KPunc, K_1 can be updated to K_2 depending on a tag set $\mathcal{S} \subseteq \mathcal{T}$ (i.e., for which ciphertexts the key should *not* be punctured) while also update tokens $(\Delta_{2,t})_{t\in\mathcal{S}\cup\{\forall\}}$ are generated. (We allow even a special symbol $\forall$ in the token $\Delta_{2,\forall}$ which indicates that the key is not punctured on *any* ciphertext.) With $(\Delta_{2,t})_{t\in\mathcal{S}\cup\{\forall\}}$, a semi-trusted third-party can update $C_{1,t,e_{\mathsf{exp}}}$ to $C_{2,t,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(\Delta_{2,t'}, C_{1,t,e_{\mathsf{exp}}})$ if $t \in \{t', \forall\}$ such that $C_{2,t,e_{\mathsf{exp}}}$ is "consistent" with K_2. Correctness guarantees that decryption of $C_{2,t,e_{\mathsf{exp}}}$ yields $M = \mathsf{Dec}(K_2, C_{2,t,e_{\mathsf{exp}}})$ as intended (and so on, if the ciphertext is not expired already). More formally:

Definition 5. *A Tag-Inverse Puncturable Encryption (TIPE) scheme* TIPE *for epochs* $(1, \ldots, \mathsf{poly}(\lambda))$*, ciphertext-tag space* $\mathcal{T}$*, and message space* $\mathcal{M}$ *consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{ExPunc}, \mathsf{Dec})$*:*

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

$\mathsf{KPunc}(K_e, \mathcal{S})$: on input key K_e for epoch e and ciphertext tags $\mathcal{S} \subseteq \mathcal{T}$ or $\mathcal{S} = \forall$, if $\mathcal{S} \subseteq \mathcal{T}$, key puncturing outputs a punctured key K_{e+1} and tag-specific tokens $(\Delta_{e+1,t})_{t\in\mathcal{S}}$; otherwise, if $\mathcal{S} = \forall$, outputs a punctured key K_{e+1} and (universal) token $\Delta_{e+1,\forall}$.

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

$\mathsf{ExPunc}(\Delta_{e+1,t'}, C_{e,t,e_{\mathsf{exp}}})$: on input token $\Delta_{e+1,t'}$ and ciphertext $C_{e,t,e_{\mathsf{exp}}}$, the exclude-from-puncturing algorithm outputs a ciphertext $C_{e+1,t,e_{\mathsf{exp}}}$ if $t' \in \{t, \forall\}$ and $e < e_{\mathsf{exp}}$; otherwise outputs $\perp$.

$\mathsf{Dec}(K_e, C_{e',t,e_{\mathsf{exp}}})$: on input key K_e and ciphertext $C_{e,t,e_{\mathsf{exp}}}$, decryption outputs message $M \in \mathcal{M}$ if $e = e'$; otherwise outputs $\perp$.

Correctness. We require that an honestly generated epoch-j ciphertext $C_{j,t,e_{\mathsf{exp}}}$ (obtained via $\mathsf{Enc}(K_j, t, M, e_{\mathsf{exp}})$) is decryptable to M if the expiry epoch is not reached yet. Moreover, an honest exclusion from being punctured for a valid ciphertext $C_{j,t,e_{\mathsf{exp}}}$ (via $\Delta_{j+1,\mathcal{S}}$ with $t \in \mathcal{S}$ or $\mathcal{S} = \forall$) from epoch j to $j+1$ yields a valid ciphertext $C_{j+1,t,e_{\mathsf{exp}}}$ that can be decrypted under the epoch key K_{j+1} (obtained via $\mathsf{KPunc}(K_j, \mathcal{S})$) if the ciphertext is not already expired. (See that we do not give any correctness guarantees beyond the expiry epochs.)

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

Intuition of Our Security Notion. We define IND-TIPE-CPA which guarantees that freshly generated ciphertexts cannot be distinguished from ones that are excluded from puncturing similarly to UE, but we give more power to the adversary as it is allowed to even query more tokens. Thereby, we assume that any (expiry-)epoch information is explicitly retrievable from the keys, tokens, and ciphertexts. Similarly, we assume that any tag information is explicitly retrievable from the tokens and ciphertexts.

To emphasize the main difference to our UE model, consider some epoch e' after the challenge epoch where the adversary queried a key $K_{e'}$ and the challenge ciphertext is not expired, i.e., $e' \leq e_{\mathsf{exp}}$. In our UE definition, the adversary is not allowed to query a token $\Delta_{e'}$ while in TIPE, we allow querying tokens $(\Delta_{e',t})_{t \in \mathcal{S}}$ that do not incorporate update capabilities for the challenge tag t^*, i.e., $t^* \notin \mathcal{S}$. Moreover, a token $\Delta_{e',\forall}$ is obviously not allowed to be queried in epoch e'. Those are the essential differences to our UE notion.

Moreover, similarly to our UE notion, we assume that any (expiry-)epoch and tag information is explicitly retrievable from the keys, tokens, and ciphertexts. We define:

Definition 6. *A TIPE scheme* TIPE *is* IND-TIPE-CPA*-secure iff for any PPT adversary A, the advantage function*

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

is negligible in λ, where $\mathsf{Exp}^{\mathsf{ind\text{-}tipe\text{-}cpa}}_{\mathsf{TIPE},A}$ is defined as in Fig. 5.

3.2 Generic UE Construction from TIPE

In the following, let $\mathsf{TIPE} = (\mathsf{TIPE.Gen}, \mathsf{TIPE.KPunc}, \mathsf{TIPE.Enc}, \mathsf{TIPE.ExPunc}, \mathsf{TIPE.Dec})$ be a TIPE with tag space $\mathcal{T} = \{t\}$ and message space $\mathcal{M}_{\mathsf{TIPE}}$. We construct a UE scheme $\mathsf{UE} = (\mathsf{Gen}, \mathsf{RotKey}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ with message space $\mathcal{M} := \mathcal{M}_{\mathsf{TIPE}}$. The main intuition here is that we only need a single tag in $\mathcal{T}$ and each ciphertext is generated with such a tag. Moreover, the key rotation in UE generates the next UE key and a token that works for *any* tag, particularly for the single tag in $\mathcal{T}$. We construct:

$\underline{\mathsf{Gen}(\lambda)}$: return $K_1 \leftarrow \mathsf{TIPE.Gen}(\lambda)$.

$\underline{\mathsf{RotKey}(K_e)}$: return $(K_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{TIPE.KPunc}(K_e, \forall)$.

$\underline{\mathsf{Enc}(K_e, M, e_{\mathsf{exp}})}$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{TIPE.Enc}(K_e, t, M, e_{\mathsf{exp}})$, for $t \in \mathcal{T}$.

$\underline{\mathsf{Upd}(\Delta_{e+1}, C_{e,e_{\mathsf{exp}}})}$ return $C_{e+1,e_{\mathsf{exp}}} \leftarrow \mathsf{TIPE.ExPunc}(C_{e,e_{\mathsf{exp}}}, \Delta_{e+1})$.

$\underline{\mathsf{Dec}(K_e, C_{e,e_{\mathsf{exp}}})}$: return $M := \mathsf{TIPE.Dec}(K_e, C_{e,e_{\mathsf{exp}}})$.

Correctness. See that this directly translates from the TIPE scheme, i.e., the ciphertexts that were computed by Enc and/or updated via Upd can be decrypted by Dec if the keys are in the same epoch and the ciphertext is not expired.

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

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

Proof. We show the theorem by constructing a PPT distinguisher D in the IND-TIPE-CPA security experiment with TIPE from any successful PPT adversary A in the EE-IND-UE-CPA security with UE.

Experiment $\mathsf{Exp}^{\mathsf{ind\text{-}tipe\text{-}cpa}}_{\mathsf{TIPE},A}(\lambda)$
$K_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 0, \mathcal{S} = \emptyset, c = 0, \Delta_{1,\forall} = \bot$
$\mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow A^{\mathsf{Enc}',\mathsf{KPunc}',\mathsf{ExPunc}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUnpuncC}^*}(\lambda)$
if A is not valid, then return $b'' \leftarrow \{0,1\}$
if $b = b'$, then return 1 else return 0

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

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

Fig. 5. Our IND-TIPE-CPA security notion for TIPE.

D runs $A(\lambda)$. Let $\mathsf{TIPE.Enc}', \mathsf{KPunc}, \mathsf{ExPunc}, \mathsf{TIPE.Corrupt}, \mathsf{TIPE.Chall}, \mathsf{TIPE.GetUnpuncC}^*$ be the TIPE oracles. Let A's oracles be as follows:
$\underline{\mathsf{Enc}'(M, e_{\mathsf{exp}})}$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{TIPE.Enc}'(M, t, e_{\mathsf{exp}})$, for $t \in \mathcal{T}$.
$\underline{\mathsf{RotKey}'}$: run $\mathsf{KPunc}(\forall)$.
$\underline{\mathsf{Upd}'(C_{e-1,e_{\mathsf{exp}}})}$: return $C_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{ExPunc}(C_{e-1,e_{\mathsf{exp}}})$.
$\underline{\mathsf{Corrupt}(\mathsf{inp}, e')}$: return the result of $\mathsf{TIPE.Corrupt}(\mathsf{inp}, e')$. (This is either a key or a token depending on inp.)
$\underline{\mathsf{Chall}(M, C_{e-1,e_{\mathsf{exp}}})}$: return $C^*_{e,b} \leftarrow \mathsf{TIPE.Chall}(M, C_{e-1,e_{\mathsf{exp}}})$.
$\underline{\mathsf{GetUpdC}^*}$: return $C^*_{e,b} \leftarrow \mathsf{TIPE.GetUnpuncC}^*$.

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

3.3 TIPE from Standard Assumptions

We show how to construct TIPE from standard assumption. Before that, we introduce encoding of epochs and the dual-system group paradigm. Both are important ingredients to our construction and discussed in the introduction on the high-level. In the following, we formally provide such ingredients and give a proof from standard assumptions.

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

$$e(\mathbf{t}) = 1 + \sum_{i=1}^{|\mathbf{t}|} (1 + (2^{\lambda - i} - 1)(t_i - 1)).^{10}$$

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

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

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

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

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

Dual System Groups. Our (relaxed) DSG DSG based on [GCTC16] consists of the PPT algorithms $(\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \mathsf{SampS}, \mathsf{SampK}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$:

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

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

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

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

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

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

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

$\mathsf{SampG}, \mathsf{SampS}, \mathsf{SampH}$ and SampK sample from a "normal" distribution (used for correctness) while $\widehat{\mathsf{SampG}}$ and $\widehat{\mathsf{SampH}}$ sample from a "semi-functional" distribution (used in the security proof). When proving UE security, we can switch UE ciphertexts and keys to semi-functional ones. The essence of dual system is then carried out, namely, semi-functional ciphertexts and keys are incompatible meaning that we can derive at a stage where the UE ciphertexts carry a uniformly random group element and indistinguishability can be shown.

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

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

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

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

[10] In the introduction, we used the tag set $\{0,1\}^\lambda$ for illustrating purposes; due to technical reasons, the tag set $\{1,2\}^\lambda$ is actually required.

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

Security. For all $\lambda, n \in \mathbb{N}$, for all $(pp, sp) \leftarrow \mathsf{SampP}(\lambda, n)$:
Orthogonality. $m(\widehat{h}) = 1$.
Non-degeneracy. $\widehat{h}$ lies in a subgroup of h_s, g_s lies in a subgroup of $\widehat{g}$.
Left-subgroup indistinguishability (LS). For any PPT D, $\mathsf{Adv}^{\mathrm{ls}}_{\mathsf{DSG},D}(\lambda, n) :=$

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

is negligible in λ, for $\mathbf{g} \leftarrow \mathsf{SampG}(pp)$ and $(\widehat{\mathbf{g}}, \cdot) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$.
Right-subgroup indistinguishability (RS). For any PPT D, $\mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, n) :=$

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

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

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

are identically distributed, for $(\widehat{\mathbf{g}} = (\widehat{g}_0, \ldots, \widehat{g}_n), g_s) \leftarrow \widehat{\mathsf{SampG}}(pp, sp)$, $(\widehat{\mathbf{h}} = (\widehat{h}_0, \ldots, \widehat{h}_n), h_s, h_a) \leftarrow \widehat{\mathsf{SampH}}(pp, sp)$, $\widehat{\mathbf{g}}' = (1, g_s^{\gamma_1}, \ldots, g_s^{\gamma_n})$, and $\widehat{\mathbf{h}}' = (1, h_s^{\gamma_1}, \ldots, h_s^{\gamma_n})$, for $\gamma_1, \ldots, \gamma_n \leftarrow \mathbb{Z}_N$.
Computational non-degeneracy (ND). For any PPT D, $\mathsf{Adv}^{\mathrm{nd}}_{\mathsf{DSG},D}(\lambda, n) :=$

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

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

Intuition of Our Construction. Beyond what is already discussed in the introduction on the intuition of our construction, we give some more technical details in the following. We assume that tags in the tag set $\mathcal{T}$ are integers in $\mathbb{Z}_N^*$ for simplicity. During key generation, we use $\lambda + 1$ as input to the DSG parameter-sampling algorithm SampP as this will give us one additional element for embedding the ciphertext tag during encryption. The initial key K_1 consists of DSG group elements (that are sampled from the normal distribution) with a distinguished element k_1 that can be seen as acting similarly to a master secret key in an (unbounded) HIBE [GCTC16]. (K_1 corresponds essentially to the root of the binary tree for the key.)

Key puncturing KPunc prunes the tree as follows: if a node corresponding to the current epoch is not a leaf node, then it computes the children of such a node. One child gets associated with 1 and the other one with 2, which maps directly to the encoding. Then, the key material of the parent node is discarded; otherwise, if such a node is a leaf node, then the key material for this node is discarded. A re-randomization step for all elements forms the resulting key for epoch $e + 1$. The corresponding token is computed around a distinguished element δ (depending on $\mathcal{S}$) which is blinded with epoch (and tag) dependent group elements from the DSG's normal distribution; it acts a linear shift for k_e.

Encryption Enc, depending on the current tree configuration and the expiry epoch, computes the ciphertext material depending on such a configuration where each group element is similarly computed to a simple form of an unbounded HIBE ciphertext in [GCTC16] but enhanced with further DSG group elements. Those elements are crucial and used in ExPunc. The message M itself is blinded by $m(k_e)^s$, where the uniform s is also embedded as "global" randomness in ciphertext elements corresponding to the tree configuration (see element S).

During ExPunc, the linear shift $m(\delta)^s$ is computed only if ciphertext material corresponding to the token is available (i.e., if such a token — potentially depending also on the tag — matches ciphertext material in the current epoch). Such a shift can then be used to update the blinding group element for the message to the next epoch. Moreover, the ciphertext tree is pruned analogously to how key material is delegated and discarded in KPunc. Particularly, ciphertext material for the node associated to the then-old epoch is discarded. Decryption recovers the message if the epochs of the key and unexpired ciphertext match.

Our Construction. Let $\mathsf{DSG} = (\mathsf{SampP}, \mathsf{SampG}, \mathsf{SampH}, \mathsf{SampS}, \mathsf{SampK}, \widehat{\mathsf{SampG}}, \widehat{\mathsf{SampH}})$ be DSG. We construct a TIPE scheme $\mathsf{TIPE} = (\mathsf{Gen}, \mathsf{KPunc}, \mathsf{Enc}, \mathsf{ExPunc}, \mathsf{Dec})$ with tag space $\mathcal{T} = \mathbb{Z}_N^*$ (determined during Gen) and message space $\mathcal{M}$:

$\underline{\mathsf{Gen}(\lambda)}$: compute $(pp, sp) \leftarrow \mathsf{SampP}(\lambda + 1)$, set $\mathcal{T} = \mathbb{Z}_N^*$, sample $(h_0, \ldots, h_{\lambda+1}) \leftarrow \mathsf{SampH}(pp)$, $k_1 \leftarrow \mathsf{SampK}(pp)$ and return $K_1 = (\{K_1'\}, m(k_1), pp)$, with $K_1' = (h_0, k_1 \cdot h_1, \ldots, h_\lambda)$.

$\underline{\mathsf{KPunc}(K_e, \mathcal{S})}$: for $K_e = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(k_e), pp)$, if $|\mathbf{t}(e)| < \lambda - 1$, then find

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

compute

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

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

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

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

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

$\underline{\mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})}$: if $e > e_{\mathsf{exp}}$, then return $\perp$. For $K_e = (\ldots, m(k_e), pp)$, sample $S \leftarrow \mathsf{SampS}(pp; s)$, for $s \leftarrow \mathbb{Z}_N^*$, and $(g_{\mathbf{u},0}, \ldots, g_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}} \leftarrow \mathsf{SampG}(pp)$; for all $\mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}$, return

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

$\underline{\mathsf{ExPunc}(\Delta_{e+1,t'}, C_{e,t,e_{\mathsf{exp}}})}$: if $e \geq e_{\mathsf{exp}}$ or $t' \notin \{t, \forall\}$, then return $\perp$. For

$$\Delta_{e+1,t} = (D_0, D_1, m(k_{e+1})) \text{ and } C_{e,t,e_{\mathsf{exp}}} = (\{C'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(k_e)^s \cdot M),$$

if $|\mathbf{t}(e)| < \lambda - 1$, then find $C'_{\mathbf{t}(e)} = (S_0, S_1, S_{|\mathbf{t}(e)|+1}, \ldots, S_\lambda)$ and compute

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

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

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

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

$\underline{\mathsf{Dec}(K_e, C_{e,t,e_{\mathsf{exp}}})}$: if $e > e_{\mathsf{exp}}$, then return $\perp$. For

$$K_e = (\{K'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, m(k_e), pp) \text{ and } C_{e,t,e_{\mathsf{exp}}} = (\{C'_{\mathbf{u}} : \mathbf{u} \in \Gamma_{\mathbf{t}(e)}\}, S_T),$$

find $K'_{\mathbf{t}(e)} = (T_0, T_1, \ldots)$ and $C'_{\mathbf{t}(e)} = (S_0, S_1, \ldots)$, and return

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

Correctness. We show:

1) Correct decryption of ciphertexts:

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

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

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

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

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

Theorem 2. *If* DSG *is a DSG scheme, then* TIPE *is* IND-TIPE-CPA*-secure. Concretely, for any PPT adversary A, it holds:*

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

with q number of tag-based token queries.

Proof. As a brief proof outline, the sequence of games is as follows:

Game 0. The IND-TIPE-CPA experiment.
Game 1. Fresh encryption of $C^*_{e,t,1}$ in Chall-oracle by using Dec and Enc instead of Upd. This is a conceptional change. Here, we use the fact that fresh ciphertexts and updated ciphertexts are indistinguishable.
Game 2. The challenge ciphertext(s) are semi-functional (via left-subgroup indistinguishability (LS)). This is a property that comes from the underlying DSG assumption and is a common hybrid step in proving constructions secure in the dual-system paradigm.
Game 3.i.0. The i-th token or key query is pseudo-normal (via right-subgroup indistinguishability (RS)). This is a common next step in many dual-system proofs to prepare the embedding of uniform elements. All queries up to $i-1$ carry a uniform element $(\widehat{h})^\alpha$.
Game 3.i.1. The i-th token or key query is pseudo-normal semi-functional (via parameter-hiding). This is further common step in many dual-system proofs to prepare the embedding of a uniform element. All queries up to $i-1$ carry a uniform element $(\widehat{h})^\alpha$.
Game 3.i.2. The i-th token or key query now carries a uniform element $(\widehat{h})^\alpha$; all queries up to $i-1$ carry a uniform element $(\widehat{h})^\alpha$.
Game 3.i.3. The i-th token or key query is semi-functional (via right-subgroup indistinguishability (RS)). This is a common next step in many dual-system proofs to finalize the embedding of a uniform element. All queries up to i carry a uniform element $(\widehat{h})^\alpha$.
Game 4. The message in the challenge ciphertext(s) is a uniform G_T-element (via computational non-degeneracy (ND)). In this game, the queried tokens and keys are independent of the challenge ciphertext(s), the message is independent of the bit b, and security readily follows.

Let $S_{A,j}$ be the event that A succeeds in Game j. We map the symbol $\forall$ to the integer 0 for ease of proof exposition. Let $S_{A,j}$ be the event that A succeeds in Game j. We highlight changes boxed.

Lemma 3 (Game 0 to Game 1). *For any PPT adversary A, it holds:*

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

Proof. We change the behavior of the challenge oracle such that we use a fresh encryption instead of updating the ciphertext provided by A. The change is agnostic of the tag (and independent of the tokens) and the Chall oracle in Game 1 is as follows:
$\underline{\mathsf{Chall}(M, C_{e-1,t,e_{\mathsf{exp}}})}$: if $\mathsf{phase} = 1$, or if $t \notin \mathcal{S}$ and $\mathcal{S} \neq \forall$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,t,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. If $b = 0$, set $C^*_{e,t,0} \leftarrow \mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})$, else compute $\boxed{M := \mathsf{Dec}(K_{e-1}, C_{e-1,t,e_{\mathsf{exp}}})}$ and $\boxed{C^*_{e,t,1} \leftarrow \mathsf{Enc}(K_e, t, M, e_{\mathsf{exp}})}$. Set $\mathcal{C}^* = \mathcal{C}^* \cup (e, C^*_{e,t,b})$, $e^* = e$, $t^* = t$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $C^*_{e,t,b}$.

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

Lemma 4 (Game 1 to Game 2). *For any PPT adversary A there is a distinguisher D on LS such that*

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

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

$\underline{\mathsf{Chall}(M, C_{e-1,t,e_{\mathsf{exp}}})}$: if $\mathsf{phase} = 1$, or if $t \notin \mathcal{S}$ and $\mathcal{S} \neq \forall$, return $\perp$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, C_{e-1,t,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\perp$. With $\mathbf{u} = (\mathbf{u}_1, \ldots) \in \Gamma_{\mathbf{t}(e)} \setminus \Gamma_{\mathbf{t}(e_{\mathsf{exp}}+1)}$, set

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

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

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

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

Lemma 5 (Game 2 to Game 3.1.0). *For any PPT adversary A, it holds:*

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

Proof. This is a conceptional change. Essentially we consistently re-randomize the token and key elements. The KPunc'-oracle in Game 3.0.0 is as follows (where we alter K'_{e+1} after calling KPunc):
$\underline{\mathsf{KPunc}'(\mathcal{S}')}$: run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{KPunc}(K_e, \mathcal{S}')$. For $K'_{e+1} = (\{(T_{\mathbf{u},0}, T_{\mathbf{u},1}, T_{\mathbf{u},|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},\lambda}) : \mathbf{u} \in \overline{\Gamma_{\mathbf{t}(e+1)}}\}, m(K_e), pp)$ and $\Delta'_{e+1} = (D_{t,0}, D_{t,1}, D_{t,2})_{t\in\mathcal{S}'\cup\{\forall\}}$, sample $(h_{t,0}, \ldots, h_{t,\lambda+1})_{t\in\mathcal{S}'\cup\{\forall\}}, (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda+1})_{\mathbf{u}\in\Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and compute

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

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

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

Lemma 6 (Game 3.i.0 to Game 3.i.1). *For any PPT adversary A there is a distinguisher D on RS such that*

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

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

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

$\underline{\mathsf{KPunc}'(\mathcal{S}')}$: run $(K'_{e+1}, \Delta'_{e+1}) \leftarrow \mathsf{KPunc}(K_e, \mathcal{S}')$. For

$$K'_{e+1} = (\{(T_{\mathbf{u},j'',0}, T_{\mathbf{u},j'',1}, T_{\mathbf{u},j'',|\mathbf{u}|+1}, \ldots, T_{\mathbf{u},j'',\lambda}) : \mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}, j'' \in [|\Gamma_{\mathbf{t}(e+1)}|]], m(K_e), pp)$$

(we can assume a natural order in K'_{e+1}) and

$$\Delta'_{e+1} = (D_{t,0}, D_{t,1}, D_{t,2})_{t \in \mathcal{S}' \cup \{\forall\}},$$

sample $(h_{t,0}, \ldots, h_{t,\lambda+1})_{t \in \mathcal{S} \cup \{\forall\}}, (h_{\mathbf{u},0}, \ldots, h_{\mathbf{u},\lambda+1})_{\mathbf{u} \in \Gamma_{\mathbf{t}(e+1)}} \leftarrow \mathsf{SampH}(pp)$, and for the (j, j')-th query (j-th token with its j'-th tag-based part, $(j, j') \in [q'] \times [q_t]$ with $q = q' + q_t$) compute token Δ'_{e+1} as follows:

if $j + j' < i + 1 \wedge (e = \hat{e}^* - 1 \vee e = \hat{e}' - 1)$:

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

if $j + j' = i + 1 \wedge (e = \hat{e}^* - 1 \vee e = \hat{e}' - 1)$:

$$(D_{t_{j'},0} \boxed{T_0}, D_{t_{j'},1} \cdot \boxed{T_{\lambda+1}^{t_{j'}}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} \boxed{T_i^{\mathbf{t}(e+1)_i}}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}}$$

else:

$$(D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}}.$$

Moreover, compute key $K'_{\mathbf{u}}$ as follows:

if $q + j'' = i \wedge e = e^*_{\mathsf{exp}}$:

$$(T_{\mathbf{u},j'',0} \boxed{T_0}, T_{\mathbf{u},j'',1} \cdot \prod_{i=1}^{|\mathbf{u}|} \boxed{T_i^{\mathbf{u}_i}}, T_{\mathbf{u},j'',|\mathbf{u}|+1} \boxed{T_{|\mathbf{u}|+1}}, \ldots, T_{\mathbf{u},j'',\lambda} \boxed{T_\lambda})$$

else:

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

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

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

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

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

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

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

Lemma 7 (Game 3.i.1 to Game 3.i.2). *For any PPT adversary A, it holds:*

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

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

Proof. This is reminiscent of Lemma 11 in [GCTC16] and results in a pseudo-normal semi-functional token and keys. For all key elements in the set $\Gamma_{\mathbf{t}(e')}$ (if $K_{e'}$ is queried) or in $\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}$ (if $K_{e'}$ is not queried) and for all tag-based token queries, we information-theoretically embed $(\widehat{h})^\alpha$ in the i-th (key or token) query step-by-step for each i. This can be only be done if the key or token queries have no epoch or tag prefixes with the challenge ciphertext.

For the tokens $(\Delta_{e^*,t})_{t\in\mathcal{S}\cup\{\forall\}}$ before the challenge epoch, see that such a token has no prefix with the challenge ciphertexts (as the epochs do not match, i.e., such tokens have the epoch e^*-1 encoded and the challenge ciphertext are in epochs $\geq e^*$). We can use this fact to embed a $(\widehat{h})^\alpha$ in $(\Delta_{e^*,t})_{t\in\mathcal{S}\cup\{\forall\}}$ due to [GCTC16]. Moreover, see that by validity condition 2), A is not allowed to query a token $\Delta_{e',t}$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ and $t = t^*$ or $t = \forall$ when it has queried a key $K_{e'}$. Hence, we can use such a fact here to embed an $(\widehat{h})^\alpha$-element in $K_{e'}$ as the tokens $(\Delta_{e',t})_t$ do not share prefixes with the challenge ciphertext. (Otherwise the token can be used to update the challenge ciphertext which would raise the validity to fail.) Moreover, no information on $\widehat{h}$ is given out in any $\Delta_{e',t}$ due to orthogonality (i.e., $m(\widehat{h}) = 1$). For the key $K_{e^*_{\mathsf{exp}}+1}$ see that it does not have any prefixes of the challenge ciphertext and we can safely embed $(\widehat{h})^\alpha$ (if $K_{e'}$ was not queried; otherwise, $(\widehat{h})^\alpha$ has already been embedded in such a key). As shown in [GCTC16], due to non-degeneracy, we have that $(h_s)^{\alpha'}$ can be replaced by some suitable $(\widehat{h})^\alpha$, for suitable $\alpha, \alpha' \in \mathbb{Z}_N$.

Lemma 8 (Game 3.i.2 to Game 3.i.3). *For any PPT adversary A there is a distinguisher D on RS such that*

$$|\Pr[S_{A,3.i.2}] - \Pr[S_{A,3.i.3}]| \leq \mathsf{poly}(\lambda) \cdot \mathsf{Adv}^{\mathrm{rs}}_{\mathsf{DSG},D}(\lambda, \lambda+1),$$

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

Proof. In this step, we undo the game hop from Lemma 6. As a result, we now have a uniform element $(\widehat{h})^\alpha$ embedded in the key and in the token queries for $K_{e'}, K_{e^*_{\mathsf{exp}}+1}$ and $\Delta_{e^*}, \Delta_{e'}$. If $\mathbf{T} = \mathbf{h}\widehat{\mathbf{h}}$, then the tokens $\Delta_{e^*}, \Delta_{e'}$ and the keys $K_{e'}, K_{e^*_{\mathsf{exp}}+1}$, are distributed as in Game 3.i.2. If $\mathbf{T} = \mathbf{h}$, then those are distributed as in Game 3.i.3. This reduction loses a polynomial factor.

Lemma 9 (Game 3.$|\Gamma_{\mathbf{t}(e^*_{\mathsf{exp}})}| + q$.3 to Game 4). *For any PPT adversary A there is a distinguisher D on ND such that*

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

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

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

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

if $e = \hat{e}^* - 1$:

$$(D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \boxed{K \cdot (\widehat{h})^{\alpha} \cdot k_e^{-1} \cdot \delta_{e+1}^{-1}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t_{j'},i}^{\mathbf{t}(e+1)_i}, m(\boxed{K \cdot (\widehat{h})^{\alpha}}))_{t_{j'} \in \mathcal{S}' \cup \{\forall\}}$$

if $e = \hat{e}' - 1$:

$$(D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \boxed{(K \cdot (\widehat{h})^{\alpha} \cdot \prod_{e^*}^{e'} \delta_i^{-1} \cdot k_{e'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_{t_{j'},i}^{\mathbf{t}(e+1)_i},$$
$$m(\boxed{K \cdot (\widehat{h})^{\alpha} \cdot \prod_{e^*}^{e'} \delta_i^{-1}}))_{t_{j'} \in \mathcal{S}' \cup \{\forall\}}$$

else:

$$(D_{t_{j'},0} h_{t_{j'},0}, D_{t_{j'},1} \cdot h_{t_{j'},\lambda+1}^{t_{j'}} \cdot \prod_{i=1}^{|\mathbf{t}(e+1)|} h_i^{\mathbf{t}(e+1)_i}, D_{t_{j'},2})_{t_{j'} \in \mathcal{S}' \cup \{\forall\}}.$$

Moreover, compute key $K'_{\mathbf{u}}$ as follows:

if $e = e^*_{\mathsf{exp}}$:

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

else:

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

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

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

$(\cdot, \hat{e}^* - 1, C_{\hat{e}^*-1,t^*,\hat{e}^*_{\mathsf{exp}}}) \notin \mathcal{L}^*$, then return $\perp$. Set

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

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

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

Lemma 10 (Game 4). *For any PPT adversary A, $\Pr[S_{A,4}] = 1/2$ holds.*

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

Taking Lemmata 3, 4, 5, 6, 7, 8, 9, and 10 together, shows Theorem 2. □

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

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

Acknowledgements. We thank the anonymous reviewers for valuable feedback. This project has received funding from the European Union's Horizon 2020 ECSEL Joint Undertaking project under grant agreement No 783119 (SECREDAS) and No 826610 (COMP4DRONES), from the Austrian Science Fund (FWF) and netidee SCIENCE grant P31621-N38 (PROFET), and from the European Union's Horizon 2020 Research and Innovation Programme project under grant agreement No 101019808 (TEAMAWARE).

References

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

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

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

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

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

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

[BGP22] Matilda Backendal, Felix Günther, and Kenneth G. Paterson. Puncturable key wrapping and its applications. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 651–681. Springer, Heidelberg, December 2022.

[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.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999.

[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.

[BRS23] Sonja Bruckner, Sebastian Ramacher, and Christoph Striecks. Muckle+: End-to-end hybrid authenticated key exchanges. In Thomas Johansson and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023, College Park, MD, USA, August 16-18, 2023, Proceedings*, volume 14154 of *Lecture Notes in Computer Science*, pages 601–633. Springer, 2023.

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

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

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

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

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

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

[DCM20] Emma Dauterman, Henry Corrigan-Gibbs, and David Mazières. Safetypin: Encrypted backups with human-memorable secrets. In *14th USENIX Symposium on Operating Systems Design and Implementation*, 2020.

[DDG+20] Fynn Dallmeier, Jan P. Drees, Kai Gellert, Tobias Handirk, Tibor Jager, Jonas Klauke, Simon Nachtigall, Timo Renzelmann, and Rudi Wolf. Forward-secure 0-RTT goes live: Implementation and performance analysis in QUIC. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 211–231. Springer, Heidelberg, December 2020.

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

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

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

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

[DKW23] Pratish Datta, Ilan Komargodski, and Brent Waters. Fully adaptive decentralized multi-authority ABE. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 447–478. Springer, Heidelberg, April 2023.

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

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.*, 2(2):107–125, 1992.

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

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

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

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

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

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

[GP23] Yao Jiang Galteland and Jiaxin Pan. Backward-leak uni-directional updatable encryption from (homomorphic) public key encryption. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 399–428. Springer, Heidelberg, May 2023.

[Gro21] Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021. https://eprint.iacr.org/2021/339.

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

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

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

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

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

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

[KPMS23] Stefan Kölbl, Anvita Pandit, Rafael Misoczki, and Sophie Schmieg. Crypto agility and post-quantum cryptography at google. *Real-World Crypto Symposium*, 2023.

[Lew11] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. Cryptology ePrint Archive, Report 2011/490, 2011. https://eprint.iacr.org/2011/490.

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

[LGM+20] Sebastian Lauer, Kai Gellert, Robert Merget, Tobias Handirk, and Jörg Schwenk. T0RTT: Non-interactive immediate forward-secret single-pass circuit construction. *PoPETs*, 2020(2):336–357, April 2020.

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

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

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

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

[MPW23] Peihan Miao, Sikhar Patranabis, and Gaven J. Watson. Unidirectional updatable encryption and proxy re-encryption from DDH. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 368–398. Springer, Heidelberg, May 2023.

[Nis22] Ryo Nishimaki. The direction of updatable encryption does matter. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 194–224. Springer, Heidelberg, March 2022.

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

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

[RSS23] Paul Rösler, Daniel Slamanig, and Christoph Striecks. Unique-path identity based encryption with applications to strongly secure messaging. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 3–34. Springer, Heidelberg, April 2023.

[SS23] Daniel Slamanig and Christoph Striecks. Revisiting updatable encryption: Controlled forward security, constructions and a puncturable perspective. In *TCC 2023 (to appear)*, 2023.

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

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

A Concrete Instantiation Under the *d*-Lin Assumption

For completeness, we provide the concrete DSG instantiation of Gong et al. [GCTC16]. The pairing operation is defined as $\widehat{e}((\mathbf{a}_1,\mathbf{a}_2),(\mathbf{b}_1,\mathbf{b}_2)) = e(\mathbf{a}_1,\mathbf{b}_1)/e(\mathbf{a}_2,\mathbf{b}_2)$. Let π_L,π_M,π_R be function that map the leftmost d columns, the $d+1$-th column, and rightmost column of a matrix. We require the d-LIN assumption:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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