Fine-Grained Forward Secrecy: Allow-List/Deny-List Encryption and Applications[‡]

David Derler[1], Sebastian Ramacher[2], Daniel Slamanig[2], and Christoph Striecks[2]

[1] DFINITY, Zurich, Switzerland
david@dfinity.org
[2] AIT Austrian Institute of Technology, Vienna, Austria
{sebastian.ramacher,daniel.slamanig,christoph.striecks}@ait.ac.at

Abstract. Forward secrecy is an important feature for modern cryptographic systems and is widely used in secure messaging such as Signal and WhatsApp as well as in common Internet protocols such as TLS, IPSec, or SSH. The benefit of forward secrecy is that the damage in case of key-leakage is mitigated. Forward-secret encryption schemes provide security of past ciphertexts even if a secret key leaks, which is interesting in settings where cryptographic keys often reside in memory for quite a long time and could be extracted by an adversary, e.g., in cloud computing. The recent concept of puncturable encryption (PE; Green and Miers, IEEE S&P'15) provides a versatile generalization of forward-secret encryption: it allows to puncture secret keys with respect to ciphertexts to prevent the future decryption of these ciphertexts.
We introduce the abstraction of allow-list/deny-list encryption schemes and classify different types of PE schemes using this abstraction. Based on our classification, we identify and close a gap in existing work by introducing a novel variant of PE which we dub *Dual-Form Puncturable Encryption* (DFPE). DFPE significantly enhances and, in particular, generalizes previous variants of PE by allowing an interleaved application of allow- and deny-list operations.
We present a construction of DFPE in prime-order bilinear groups, discuss a direct application of DPFE for enhancing security guarantees within Cloudflare's Geo Key Manager, and show its generic use to construct forward-secret IBE and forward-secret digital signatures.

Keywords: Puncturable Encryption · Forward Secrecy.

1 Introduction

Leakage of secret keys is a major security risk in modern systems and cryptographic protocols. For example, key-leakage can be a significant problem in secure messaging applications such as Signal or WhatsApp, but also in other well-known Internet protocols such as TLS, IPSec, or SSH. Those applications typically address this risk by providing the property of *forward secrecy*. Forward secrecy mitigates the problems associated to the leakage of a long-term secret key in the sense that the confidentiality of the data encrypted in old ciphertexts is still protected after a key is leaked. However, key-leakage is problematic far beyond the aforementioned applications. One of the prime examples of key-leakage being an important risk is when decryption keys are kept in software or trusted execution environments (TEEs)

[‡] This is the full version of a paper which appears in Financial Cryptography and Data Security – 25th International Conference, FC 2021, Revised Selected Papers, LNCS, Springer.

like ARM's TrustZone or the increasingly popular SGX by Intel. Such scenarios are typically found within heavily virtualized environments such as cloud computing. In such settings, it is well known that the shared resources introduce the danger of information leakage, i.e., extracting decryption keys held in shared memory, e.g., via co-located virtual machines controlled by an attacker [RTSS09, ZJRR12, ZJRR14]. At the same time, microarchitectural attacks such as cache attacks (against TEEs) are getting increasingly sophisticated and more devastating (cf. [SG20]). While rotating keys frequently helps to reduce the risk, frequently deploying new keys becomes impractical if the frequency gets too high.

Concrete example. We will illustrate this problem by a concrete practical example. As the use of TLS for securing communication on the Internet grows, content distribution networks (CDNs) such as Cloudflare face a new issue: all of their endpoints terminating TLS connections deployed in colocations all over the world need access to the secret keys associated to the certificate (i.e., public key) to guarantee low latency. As those secret keys belong to the customers, they need to provide the keys to the CDNs or deploy solutions such as Keyless SSL[3], where customers are required to run their own keyserver answering signing requests from the CDN. The latter comes at the cost of higher latency if users are not close to the location of the key server. The former—while providing better latency for users worldwide—faces a different issue: due to various differences in local laws or other regulations surrounding the use of secret keys, customers might not be interested in having their keys exposed to certain locations and areas. Systems like Cloudflare's Geo Key Manager[4] tackle this issue by giving customers the control on the locations their secret keys are stored when shared with Cloudflare. Effectively, customers are able put whole regions on allow-lists, e.g., Europe or the US. At the same time, they are able to put multiple colocations within those regions, e.g., London in Europe, on deny-lists. Finally, they are also able to directly put colocations on the allow-list that are not inside the regions already on allow-lists, e.g., Singapore.

Since the customer's secret keys are highly sensitive, such a system profits from strong security including forward secrecy. The currently deployed solution *does not* provide forward secrecy, a feature that helps to put the distributed keys at a much lower risk. Looking ahead, with our approach, adding areas to allow-lists, colocations within these areas on deny-lists, as well as allow-listing single colocations is efficiently possible and adds forward secrecy on top.

Fine-grained forward-secrecy. We will follow the approach of restricting (or customizing) the capabilities of secret keys held in memory via cryptographic means to achieve fine-grained forward secrecy. Arguably, this still does not entirely eliminate the problem of key-leakage. Yet, it helps to significantly reduce the damage if key-leakage happens, and, at the same time, removes the requirement to frequently rotate keys. While forward secrecy can be efficiently obtained in interactive protocols, it is more involved for non-interactive primitives. In the past, forward secrecy has been studied for various non-interactive primitives such as digital signatures [BM99], identification schemes [AABN02], public-key encryption [CHK03], symmetric cryptography [BY03], and proxy re-encryption [DKL+18a]. The basic idea is to discretize time into intervals and to have a fixed public key over a potentially long period of time. However, the secret key "evolves" over time such that a leaked secret key in interval i is no longer useful for any interval $j < i$. In particular, for public-key encryption, this

[3] https://www.cloudflare.com/ssl/keyless-ssl/

[4] https://blog.cloudflare.com/introducing-cloudflare-geo-key-manager/

guarantees that all "old" ciphertexts can no longer be decrypted. Obviously, if the switches between intervals happen too frequently, it requires good synchronization, whereas for longer time intervals a looser synchronization (which is desirable) is sufficient. Nevertheless, in any case, the achieved forward-secrecy property is very coarse grained, i.e., switching the interval essentially destroys access to *all* old ciphertexts.

Green and Miers [GM15] introduced the cryptographic concept of puncturable encryption (PE) as a versatile generalization of forward-secret public-key encryption for asynchronous messaging. The idea here is to provide a more fine-grained forward-secrecy property that allows a secret key to be "punctured" on specific ciphertexts (or tags associated to them) in a way that the resulting decryption key will then no longer be useful to decrypt ciphertexts on which the key has been punctured over time. Following these initial works, a number of alternative PE schemes [GHJL17, DJSS18, SSS+20, SDLP20, CRSS20] as well as some variations of PE schemes with different puncturing capabilities [DKL+18b, WCW+19] have been proposed.

Allow-list/deny-list encryption schemes. To provide a comprehensive classification of different cryptographic primitives (and mostly PE schemes), we introduce allow-/deny-list (ALDL) encryption. It represents a very simple abstraction of encryption mechanisms maintaining allow and deny lists. Here, ciphertexts and decryption keys are linked to both allow and deny lists in a certain way. Allow lists incorporate positive tags while deny lists have negative tags. A ciphertext can carry two tags—a positive and negative tag—that are determined during the encryption procedure. Decryption keys can be associated to several positive and negative tags. For example, a ciphertext with a positive tag t_+ and negative tag t_- can be decrypted by a secret key that is associated to t_+ but not to t_-. More generally this means that a decryption key that is linked to a positive tag is able to decrypt the ciphertext if that ciphertext has the positive tag attached. On the other hand, a decryption key that is associated to a negative tag is *not* able to decrypt ciphertexts that have those negative tags attached.

On a high level, this has interesting applications and subsumes several cryptographic primitives as shown in Table 1. For example, an identity-based encryption (IBE) [BF01, Coc01] scheme can be seen as an ALDL encryption scheme where an allow list can contain an identity, i.e., an *id*. Ciphertexts and secret keys are associated to a certain *id* (i.e., a positive tag) from the allow list. If the *id* of the ciphertext matches the secret-key *id*, then decryption works.

Furthermore, a PE scheme can be seen as an ALDL encryption scheme where the deny list contains a set of tags. A ciphertext is associated to a certain negative tag and decryption keys are associated to a deny list of negative tags. Now, when the tag of the ciphertext is on the deny list, then decryption is *not* successful while all other ciphertexts with tags not on the deny list can be successfully decrypted. An interesting application is forward-secret zero round-trip time (0-RTT) key-exchange [GHJL17, DJSS18].

Recently, Derler et al. [DKL+18b] and Wei et al. [WCW+19] proposed the new forward-secret primitives called fully PE (FuPE) and forward-secret puncturable IBE (fs-PIBE), respectively, that can be abstracted by ALDL encryption in the following sense. Within FuPE, ciphertexts are associated to a positive and negative tag while decryption keys can be first associated to several negative tags in a deny list and a final positive tag in an allow list. In fs-PIBE on the other hand, ciphertexts are also associated to a positive and a negative tag while decryption keys can be first associated to one positive tag in the allow list and afterwards to several negative tags inserted to a deny list. FuPE realized the first

forward-secret proxy re-encryption (PRE) scheme while fs-PIBE has been shown to have applications to Cloud e-mails. In these approaches, the order of inserting tags to the allow and deny lists plays a crucial role.

In our work, we want to enhance those capabilities even further. In particular, our work allows to first associate the secret key with several negative tags in a deny list, then with a positive tag in the allow list, and afterwards with several further negative tags again in the deny list, which yields new applications areas not yet covered by existing approaches. Our enhancement gives more flexibility and more fine-grained forward secrecy enhancing techniques from prior works. In Table 1, we compare all discussed approaches.

ALDL Variant	1. Action	2. Action	3. Action	Primitive	Applications
1.	AL (1)	–	–	IBE	E-mail (e.g, [BF01])
2.	DL (∞)	–	–	PE	Key exchange (e.g, [GM15, GHJL17])
3a.	AL (1)	DL (∞)	–	fs-PIBE	Cloud e-mail, weak forward-secret IBE (e.g., [WCW+19])
3b.	DL (∞)	AL (1)	–	FuPE	Forward-secure PRE (e.g., [DKL+18b])
4. (this work)	DL (∞)	AL (1)	DL (∞)	DFPE	Enhanced Geo Key Manager, forward-secret IBE and signatures

Table 1: Overview of allow-list (AL)/deny-list (DL) encryption variants with actions performed on the allow and deny lists and in which order. We use 1 to denote support for a single tag and ∞ to indicate many tags in arbitrary order. We further list cryptographic primitives and applications abstracted by the ALDL-encryption variants.

Our contribution. We propose a versatile variant of puncturable encryption dubbed dual-form puncturable encryption (DFPE), which extends recent works on PE that are not expressive enough to achieve our goals. We carefully adapt the PE techniques envisioned by Green and Miers [GM15] and Günther, Hale, Jager, and Lauer [GHJL17] to equip PE with *interleaved negative and positive puncturing*. While the concept of Fully PE (FuPE) due to Derler et al. [DKL+18a, DKL+18b] is related to our solution, it is not sufficient. In their work, positively punctured keys can no longer be negatively punctured. In contrast to Derler et al.—who can instantiate their FuPE scheme from any Hierarchical Identity-Based Encryption (HIBE) [GS02] scheme—we require novel tools and in particular the concept of tagged HIBEs (THIBEs), a generalization of HIBEs. Our ideas on THIBEs are related to the work of Abdalla, Kiltz, and Neven [AKN07], but with different goals.

Dual-Form Puncturable Encryption (DFPE): Loosely speaking, DFPE allows to puncture secret keys on negative tags (like within PE), i.e., a key punctured on a negative tag can no longer decrypt ciphertexts under this tag, but in addition a secret key can be customized to a given positive tag once and then further punctured negatively. Keys customized to a positive tag can only decrypt ciphertexts to this positive tag and whose negative tags are distinct from the ones the key was punctured on. We introduce the concept of DFPE and rigorously model its security requirements. For concrete instantiations of DFPE, we introduce a generalization of HIBEs called tagged HIBEs (THIBEs) along with a suitable security model

and which we instantiate using ideas underlying the Boneh-Boyen-Goh (BBG) [BBG05a] HIBE. Since it requires some modifications and tweaks to provide the features required by a THIBE scheme, we provide a careful proof of security of our THIBE. A main benefit of starting from the BBG HIBE is that the size of the ciphertexts in our THIBE is constant. Finally, we show how DFPE can be generically constructed from any THIBE and provide a proof-of-concept implementation of our concrete DFPE scheme.

Enhancing Cloudflare's Geo Key Manager: We show how the currently used approach based on a combination of both pairing-based identity-based broadcast encryption (IBBE) [Del07] and identity-based revocation (IBR) [ALdP11], which so far does not provide forward-secrecy, can be instantiated using DFPE as a single primitive. Thereby, it supports the required functionality of adding areas to allow-lists, colocations within these areas on deny-lists, allow-listing single colocations is efficiently possible, and at the same time adds forward secrecy on top while achieving comparable parameter sizes.

Cryptographic applications: We demonstrate that DFPE is a versatile cryptographic tool by generically instantiating other primitives. This immediately yields (new) constructions thereof. In particular, we show how to generically construct forward-secure IBE [YFDL04], thereby—to the best of our knowledge—obtaining the first fs-IBE scheme with compact ciphertexts, as well as forward-secure signatures [BM99, Kra00, IR01, AABN02]. Especially, the latter turned out to be an interesting primitive in the context of distributed ledgers [DGKR18, GW19, DN19, DGNW20, DGKR18].

Outline. In Section 2, we provide preliminaries for our work. In Section 3, we define tagged HIBEs (THIBEs) and its CPA- and CCA-based security notions. In Section 3, we present Dual-Form Puncturable Encryption (DFPE) and give a construction from THIBEs. We conclude with applications in Section 5.

2 Preliminaries

Notation. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$, and let $\kappa \in \mathbb{N}$ be the security parameter. For a finite set S, we denote by $s \leftarrow S$ the process of sampling s uniformly from S. For an algorithm A, let $y \leftarrow A(\kappa, x)$ be the process of running A on input (κ, x) with access to uniformly random coins and assigning the result to y. (If not given explicitly, we assume that κ is implicitly given as input.) To make the random coins r explicit, we write $A(\kappa, x; r)$. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in κ. A function f is negligible if its absolute value is smaller than the inverse of any polynomial (i.e., if $\forall c \exists k_0 \forall \kappa \geq k_0 : |f(\kappa)| < 1/\kappa^c$). We may write $q = q(\kappa)$ if we mean that the value q depends polynomially on κ.

Pairings. Let G_1, G_2, G_T be cyclic groups of order p. A *pairing* $e\colon G_1 \times G_2 \to G_T$ is a map that is *bilinear* (i.e., for all $g_1, g_1' \in G_1$ and $g_2, g_2' \in G_2$, we have $e(g_1 \cdot g_1', g_2) = e(g_1, g_2) \cdot e(g_1', g_2)$ and $e(g_1, g_2 \cdot g_2') = e(g_1, g_2) \cdot e(g_1, g_2')$, *non-degenerate* (i.e., for generators $g_1 \in G_1, g_2 \in G_2$, we have that $e(g_1, g_2) \in G_T$ is a generator), and *efficiently computable*. Let BGen be a PPT algorithm that, on input a security parameter κ, outputs $\mathsf{BG} = (p, G_1, G_2, G_T, e, g_1, g_2)$ for generators g_1 and g_2 of G_1 and G_2, respectively, and $\Theta(\kappa)$-bit prime p.

q-wBDHI assumption. We recall the q-wBDHI [BBG05a] assumptions ported to Type-3 groups [CM11]. We define the advantage of an adversary D with respect to q-wBDHI as

$$\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(\kappa) := \left| \Pr[D(pp, e(g_1, g_2^r)^{\alpha^{(q+1)}}) = 1] - \Pr[D(pp, e(g_1, g_2^r)^u) = 1] \right|,$$

where $\mathsf{BG} \leftarrow \mathsf{BGen}(1^\kappa)$, and $pp = (\mathsf{BG}, g_1^\alpha, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_2^\alpha, g_1^r, g_2^r)$, for $\alpha, r, u \leftarrow \mathbb{Z}_p$. We say the q-wBDHI assumption holds if $\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},A}$ is a negligible function in the security parameter κ for all PPT adversaries A.

3 Tagged Hierarchical Identity-Based Encryption

Hierarchical identity-based encryption (HIBE) [HL02, GS02, BBG05a] organizes identities in a tree where identities at some level can delegate secret keys to its descendant entities, but cannot decrypt ciphertexts intended for other higher-level identities. A tagged HIBE (THIBE) is a generalization of HIBEs where secret keys can be tagged and ciphertexts are tagged (a concept related to [AKN07] but adapted to different goals in our work). Correctness now ensures that untagged secret keys are capable of decrypting (tagged) ciphertexts if the identities match while tagged secret keys can only decrypt (tagged) ciphertexts correctly if the identities *and* the tag match. The distinguishing feature between HIBEs and THIBEs is that delegated secret keys on any hierarchy can be tagged and, afterwards, even further delegated. In a certain sense, through tagging, secret keys can be further restricted on the same hierarchy level and beyond in their decryption capabilities.

3.1 Definition, Correctness, and Security Notions of THIBEs

Before constructing THIBEs, we first present our THIBE definition and continue with its correctness property as well as its security notions.

Definition 1 (THIBE). *For some hierarchy parameter $\ell \in \mathbb{N}$, a tagged hierarchical identity-based encryption (THIBE) scheme* THIBE *with message space $\mathcal{M}$, tag space $\mathcal{T}$, and identity space $\mathcal{ID}^{\leq \ell}$, consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{Del}, \mathsf{Tag}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Gen}(1^\kappa, \ell)$: *output a keypair* $(\mathsf{pk}, \mathsf{sk}^\varepsilon_\varepsilon)$. *(We assume that* pk *is given as input to* $\mathsf{Del}, \mathsf{Tag}$, *and* Dec *implicitly; let $\varepsilon \notin \mathcal{ID} \cup \mathcal{T}$ be a distinguished element associated to non-tagged or non-delegated secret keys.)*

$\mathsf{Del}(\mathsf{sk}^t_{id'}, id)$: *output a secret key* sk^t_{id} *if $id' \in \mathcal{ID}^{\ell'-1}$ is a prefix of $id \in \mathcal{ID}^{\ell'}$, for some $\ell' \in [\ell]$ else output* $\mathsf{sk}^t_{id'}$.

$\mathsf{Tag}(\mathsf{sk}^\varepsilon_{id}, t)$: *output a secret key* sk^t_{id} *if $t \in \mathcal{T}$, else output* $\mathsf{sk}^\varepsilon_{id}$.

$\mathsf{Enc}(\mathsf{pk}, M, id, t)$: *for message $M \in \mathcal{M}$, identity $id \in \mathcal{ID}^{\leq \ell}$, and tag $t \in \mathcal{T}$, output a ciphertext C^t_{id}.*

$\mathsf{Dec}(\mathsf{sk}^{t'}_{id'}, C^t_{id})$: *output $M \in \mathcal{M} \cup \{\perp\}$.*

Correctness of THIBE. Essentially, correctness follows the HIBE correctness (i.e., a secret key can decrypt a ciphertext if the identity in such key is a prefix of the identity associated to the ciphertext), but we additionally require that the tag in the ciphertext matches the tag in the secret key as well.

More formally, for all $\kappa, \ell \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk}^\varepsilon_\varepsilon) \leftarrow \mathsf{Gen}(1^\kappa, \ell)$, all $M \in \mathcal{M}$, all $id, id' \in \mathcal{ID}^{\leq \ell} \cup \{\varepsilon\}$ where $id' \in \mathcal{ID}^{\ell'-1}$ is a prefix of $id \in \mathcal{ID}^{\ell'}$, for some $\ell' \in [\ell]$, all $t \in \mathcal{T} \cup \{\varepsilon\}$, all $\mathsf{sk}^t_{id} \leftarrow \mathsf{Tag}(\mathsf{sk}^\varepsilon_{id}, t)$, all $\mathsf{sk}^t_{id} \leftarrow \mathsf{Del}(\mathsf{sk}^t_{id'}, id)$, all $t' \in \mathcal{T}$ all $C^{t'}_{id} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, id, t')$, we have that $\mathsf{Dec}(\mathsf{sk}^t_{id}, C^{t'}_{id}) = M$ if $t = t'$.

THIBE-IND-CPA and THIBE-IND-CCA security notions. A THIBE scheme is THIBE-IND-CPA-secure or THIBE-IND-CCA-secure if and only if any PPT adversary A succeeds in the following experiments only with probability at most negligibly larger than $1/2$.

First, A receives an honestly generated pk. Let $\mathsf{Ext}(\cdot,\cdot,\cdot)$ be a key-extraction oracle that, given $\mathsf{sk}_\varepsilon^\varepsilon$, an identity $id \in \mathcal{ID}^{\leq \ell}$, and a tag $t \in \mathcal{T} \cup \{\varepsilon\}$, outputs a secret key sk_{id}^t via iteratively running Del to compute $\mathsf{sk}_{id}^\varepsilon$ and, afterwards, returning $\mathsf{Tag}(\mathsf{sk}_{id}^\varepsilon, t)$. Furthermore, let Dec' be a decryption oracle that, given $\mathsf{sk}_\varepsilon^\varepsilon$ and a ciphertext C_{id}^t, outputs $\mathsf{Dec}(\mathsf{sk}_{id}^t, C_{id}^t)$, where $\mathsf{sk}_{id}^t \leftarrow \mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, id, t)$. During the experiment, A may adaptively query the $\mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, \cdot, \cdot)$-oracle for corresponding secret key $\mathsf{sk}_\varepsilon^\varepsilon$ to pk. Only for THIBE-IND-CCA security, A has access to the decryption oracle Dec'. At some point, A outputs two equal-length messages M_0, M_1 and receives a target ciphertext $C_{id^*}^{t^*} \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, id^*, t^*)$ in return, for uniform $b \leftarrow \{0,1\}$. Eventually, A outputs a guess b^*. We say that A is valid if and only if A never queried the Ext-oracle on a prefix of id^* for tag $t \in \{t^*, \varepsilon\}$, and only outputs equal-length messages. For THIBE-IND-CCA security, A is only valid if it additionally did not query Dec' on the challenge ciphertext. We say that any valid A succeeds if $b = b^*$. More formally, the experiments are given in Experiment 1.

Experiment $\mathsf{Exp}^{\mathsf{thibe\text{-}ind\text{-}T}}_{\mathsf{THIBE},A}(1^\kappa, \ell)$
$(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon) \leftarrow \mathsf{Gen}(1^\kappa, \ell)$
$(M_0, M_1, id^*, t^*, \mathsf{state}) \leftarrow A^{\mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, \cdot, \cdot), \mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, \cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, id^*, t^*)$
$b^* \leftarrow A^{\mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, \cdot, \cdot), \mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, \cdot)}(\mathsf{state}, C^*)$
if $b = b^*$ return then 1, else return 0

Experiment 1: THIBE-IND-T-security for THIBE: $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

Definition 2. *For any PPT adversary A, we define the advantage function as*

$$\mathbf{Adv}^{\mathsf{thibe\text{-}ind\text{-}T}}_{\mathsf{THIBE},A}(1^\kappa, \ell) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{thibe\text{-}ind\text{-}T}}_{\mathsf{THIBE},A}(1^\kappa, \ell) = 1\right] - \frac{1}{2} \right|,$$

for integer $\ell \in \mathbb{N}$, for $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

3.2 Constructing Tagged Hierarchical Identity-Based Encryption

We present our construction of a THIBE. The scheme construction closely follows the construction of the Boneh-Boyen-Goh (BBG) HIBE [BBG05a], but has one additional distinguished element in the secret keys (used for positive puncturings later in our DFPE construction). This element is not related to any hierarchy level and can be embedded into the secret key at any stage. In Scheme 1, we formally construct our THIBE.

Correctness of THIBE. Correctness essentially follows from the correctness of the Boneh-Boyen-Goh HIBE [BBG05a]; in particular, see that decryption succeeds for matching secret keys $\mathsf{sk}_{id}^t =: (a_0, a_1, \dots) = (h^\alpha \cdot (h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i} \cdot g^{H(t)})^r, g_2^r)$ and ciphertexts $C_{id}^t =:$

$\underline{\mathsf{Gen}(1^\kappa, \ell)}$: Generate a bilinear group $\mathsf{BG} := (p, e, G_1, G_2, G_T, g_1, g_2) \leftarrow \mathsf{BGen}(1^\kappa)$, set $\mathcal{M} := G_T$, set $\mathcal{T} := \{0,1\}^\kappa$, and set $\mathcal{ID} := \mathbb{Z}_p$, sample $g, h, h_0, h_1, \ldots, h_\ell \leftarrow G_1$, choose $\alpha, r \leftarrow \mathbb{Z}_p$, set $\mathsf{pk} := (\mathsf{BG}, H, g, h, h_0, h_1, \ldots, h_\ell, g_2^\alpha)$, for hash function $H\colon \mathcal{T} \mapsto \mathbb{Z}_p$ (modelled as RO in the security proof) where $H(\varepsilon) := 0$, and $\mathsf{sk}_\varepsilon^\varepsilon := (h^\alpha \cdot h_0^r, g_2^r, h_1^r, \ldots, h_\ell^r, g^r)$.

$\underline{\mathsf{Del}(\mathsf{sk}_{id'}^t, id)}$: For $id =: (id_1, \ldots, id_{\ell'+1})$ and $\ell' := |id'|$, if $id \neq (id', id_{\ell'+1})$, then return $\mathsf{sk}_{id'}^t$. Otherwise, if $t = \varepsilon$, parse $\mathsf{sk}_{id'}^t =: (a_0, a_1, K_{\ell'+1}, \ldots, K_\ell, g')$. Sample $r' \leftarrow \mathbb{Z}_p$ and return

$$\left(a_0 \cdot K_{\ell'+1}^{id_{\ell'+1}} \cdot \left(h_0 \cdot \prod_{i=1}^{|\ell'+1|} h_i^{id_i}\right)^{r'}, a_1 \cdot g_2^{r'}, (K_i \cdot h_i^{r'})_{\ell'+1<i\leq\ell}, g' \cdot g^{r'}\right).$$

Otherwise, if $t \neq \varepsilon$, parse $\mathsf{sk}_{id'}^t =: (a_0, a_1, K_{\ell'+1}, \ldots, K_\ell)$. Sample $r' \leftarrow \mathbb{Z}_p$ and return

$$\left(a_0 \cdot K_{\ell'+1}^{id_{\ell'+1}} \cdot \left(h_0 \cdot \prod_{i=1}^{\ell'+1} h_i^{id_i}\right)^{r'} \cdot g^{H(t)\cdot r'}, a_1 \cdot g_2^{r'}, (K_i \cdot h_i^{r'})_{\ell'+1<i\leq\ell}\right).$$

$\underline{\mathsf{Tag}(\mathsf{sk}_{id}^\varepsilon, t)}$: If $t = \varepsilon$, return $\mathsf{sk}_{id}^\varepsilon$. Otherwise, set $\ell' := |id|$ and $id =: (id_1, \ldots, id_{\ell'})$. Parse $\mathsf{sk}_T^\varepsilon =: (a_0, a_1, K_{\ell'+1}, \ldots, K_\ell, g')$. Sample $r' \leftarrow \mathbb{Z}_p$ and return

$$\left(a_0 \cdot g'^{H(t)} \cdot \left(h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i}\right)^{r'} \cdot g^{H(t)\cdot r'}, a_1 \cdot g_2^{r'}, (K_i \cdot h_i^{r'})_{\ell'<i\leq\ell}\right).$$

$\underline{\mathsf{Enc}(\mathsf{pk}, M, id, t)}$: Set $\ell' := |id|$ and $id =: (id_1, \ldots, id_{\ell'})$. Sample $s \leftarrow \mathbb{Z}_p$, and return

$$(C_1, C_2, C_3) := \left(e(h, g_2^\alpha)^s \cdot M, g_2^s, \left(h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i}\right)^s \cdot g^{H(t)\cdot s}\right).$$

$\underline{\mathsf{Dec}(\mathsf{sk}_{id'}^{t'}, C_{id}^t)}$: If $id' \neq id$ or $t' \neq t$, return $\perp$. Otherwise, parse $\mathsf{sk}_{id'}^t$ as $(a_0, a_1, \ldots)$ and $(C_1, C_2, C_3) := C_{id}^t$. Return $M' := C_1 \cdot e(C_3, a_1) \cdot e(a_0, C_2)^{-1}$.

Scheme 1: Construction of THIBE.

$(C_1, C_2, C_3) = (e(h, g_2^\alpha)^s \cdot M, g_2^s, (h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i} \cdot g^{H(t)})^s)$, for $id =: (id_1, \ldots, id_{\ell'})$:

$$C_1 \cdot \frac{e(C_3, a_1)}{e(a_0, C_2)} = e(h, g_2^\alpha)^s \cdot M \cdot \frac{e((h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i} \cdot g^{H(t)})^s, g_2^r)}{e(h^\alpha \cdot (h_0 \cdot \prod_{i=1}^{\ell'} h_i^{id_i} \cdot g^{H(t)})^r, g_2^s)} = M.$$

Theorem 1. *If the q-wBDHI assumption holds, then* THIBE *defined in Scheme 1 is THIBE-IND-CPA-secure in the random-oracle (RO) model. Concretely, for any valid PPT adversary A with at most $q_k = q_k(\kappa)$ key queries, there is a distinguisher D on q-wBDHI with $q = \ell+1$, such that*

$$\mathbf{Adv}^{\mathsf{thibe\text{-}sind\text{-}cpa}}_{\mathsf{THIBE},A}(1^\kappa, \ell) \leq q_k \cdot \mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(1^\kappa),$$

for group generator BGen *and number of RO-queries $q_k = q_k(\kappa)$.*

Proof. We show the THIBE-IND-CPA security of THIBE for any valid PPT adversary A in two games where:

Game 0. Game 0 is the THIBE-IND-CPA experiment.

Game 1. Game 1 is defined as Game 0 except that the encrypted message is a uniformly random group element independent of the (challenge) bit b.

We prove that Game 0 and Game 1 are indistinguishable, by presenting a reduction that uses a q-wBDHI challenger to interpolate between the two games. The proof follows the argumentation of the proof for the BBG HIBE scheme under the q-wBDHI assumption [BBG05a] transformed to the Type-3 pairing setting.

Essentially, due to the fact that the THIBE-construction is slightly different compared to the BBG-HIBE construction, we have to carefully argue that our construction is THIBE-IND-CPA secure under the q-wBDHI assumption. Nevertheless, many similarities to the proof of the BBG-HIBE can be found and, hence, we will focus on the differences in more detail (while of course point out the similarities).

The main difference is that we have a special element, namely a tag-associated group element, that needs some extra care. However, as it turns out, this element can be handled as all the identity-basis group elements and is retrieved from the assumption as well. We are able to setup the public key with the correct distribution as well as answer the Ext-queries consistently. Essentially, this distinguishing tag-based element is handled as the "leaf" target identity in the BBG HIBE proof.

We follow the proof line of BBG [BBG05b, Proof of Theorem 3.1] in the following sense: BBG show a weaker variant where the adversary has to commit to the target (hierarchical) identity before seeing the public key (i.e., identity id^* and tag t^* in our case). This is often referred to as selective security and can be easily lifted to adaptive security (which we actually model within the THIBE-IND-CPA experiment where the public key is retrieved first by the adversary before it commits to the target identity and tag) by relying on the random-oracle (RO) model. (We refer to BBG [BBG05a] and later works [GW19, DN19, DGNW20] that all use similar techniques to prove adaptive security of their schemes in the RO model.)

For our purpose, it suffices to use the hash function H as the tag function for fixed-length tags mapped to $\mathbb{Z}_p$ (with restriction for $H(\varepsilon) := 0$). We want to show that each PPT adversary A with at most $q_k = q_k(\kappa)$ extraction queries on the THIBE-IND-CPA security of THIBE yields a PPT distinguisher D for the q-wBDHI assumption, for $q = \ell + 1$.

The distinguisher D is given as input

$$(pars := (\mathsf{BG}, y_1 := g_1^{\alpha}, y_2 := g_1^{\alpha^2}, \ldots, y_{\ell+1} := g_1^{\alpha^{\ell+1}}, g_2^{\alpha}, g_1^{r}, g_2^{r}), T),$$

with g_1, g_2 is given by the bilinear group parameters $\mathsf{BG} = (p, G_1, G_2, G_T, g_1, g_2)$ and T equals either $e(g_1, g_2^r)^{\alpha^{\ell+2}}$ or is a uniform element in G_T.

A outputs the target (hierarchical) identity $id^* = (id_1^*, \ldots, id_{\ell'}^*) \in (\mathbb{Z}_p)^{\ell'}$, for $\ell' \in [\ell]$, and target tag t^*. If $\ell' < \ell$, pad id^* with $\ell - \ell'$ *zeroed* part-identities such that $|id^*| = \ell$ and we can assume $id^* = (id_1^*, \ldots, id_\ell^*)$ as done in the BBG proof. The hash evaluation $H(t^*) \in \mathbb{Z}_p$ will be set as the $(\ell+1)$-th "identity." The public key pk is sampled similarly to the BBG proof, i.e., set $h := y_{\ell+1} \cdot g_1^{\gamma}$, for $\gamma \leftarrow \mathbb{Z}_p$, and $h_i' := g_1^{\gamma_i}/y_{\ell-i+2}$, for all $\gamma_i \leftarrow \mathbb{Z}_p$, $i \in [\ell+1]$. Furthermore, set $g := h_1'$ and $h_0 := g_1^{\gamma_0} \cdot \prod_{i=1}^{\ell} y_{\ell-i+2}^{id_i^*} \cdot y_1^{H(t^*)}$, for some $\gamma_0 \leftarrow \mathbb{Z}_p$. The public key

$$\mathsf{pk} := (\mathsf{BG}, H, g, h, h_0, h_1 := h_{\ell+1}', \ldots, h_\ell := h_2', g_2^{\alpha})$$

is sent to A.

Note that D does not know the secret key $\mathsf{sk}_{\varepsilon}^{\varepsilon}$ corresponding to pk; implicitly the secret key is set to $\mathsf{sk}_{\varepsilon}^{\varepsilon} = (h^{\alpha} \cdot h_0^{r'}, g_2^{r'}, h_1^{r'}, \ldots, h_\ell^{r'}, g^{r'}) = (g_1^{\alpha(\alpha^{\ell+1}+\gamma)} \cdot h_0^{r'}, \ldots)$ for some $r' \in \mathbb{Z}_p$ as

in the BBG proof. Hence, we embed the tag as $H(t^*)$ and the target identity id^* in the h_0 group element, where essentially key material cannot be computed for the target tag t^* and identity id^* by the distinguisher D.

Extraction-oracle queries in the reduction are answered as in the BBG proof for "identity vector" $((id_i)_{i\in[\ell'']}, t)$, for some $\ell'' \in [\ell]$ (we note that there exists a smallest $k \in [\ell'']$ such that $id_k \neq id_k^*$ if $t = t^* \neq \varepsilon$ due to the validity of A in querying Ext we assume here). As long as id_k^* is part of the vector, we can even allow $t = t^* \neq \varepsilon$. If no such k exist, then $t \neq t^*$ is queried (again, due to the validity of A we assume here) and there still exists at least one element which involves $g_1^{\alpha^{\ell+2}}$ that cannot be computed by D. For all other queries, $g_1^{\alpha^{\ell+2}}$ cancels out due to the partitioning argument (the same way as in the BBG proof) and extraction queries can be computed.

At some point, A outputs two equal-length messages M_0, M_1, D samples $b \leftarrow \{0,1\}$, and sends the challenge ciphertext

$$C^* := \left(M_b \cdot T \cdot e(g_1^{\alpha}, (g_2^r)^{\gamma}), g_2^r, g_1^{r(\gamma_0 + \sum_{i=1}^{\ell} id_i^* \cdot \gamma_i + H(t^*)\cdot\gamma_{\ell+1})}\right) \text{ to } A.$$

Eventually, A outputs a guess b^* on b. If $b^* = b$, then D outputs 1 (i.e., guesses $T = e(g_1, g_2^r)^{\alpha^{\ell+2}}$), else outputs 0 (i.e., guesses that T is uniformly random in G_T).

Analysis of the reduction. The public key pk as well as all secret keys (that are returned after querying the Ext-oracle) are distributed correctly. If $T = e(g_1, g_2^r)^{\alpha^{\ell+2}}$, the ciphertext C^* is a valid encryption of M_b; otherwise, i.e., if T is uniformly random in G_T, then C^* is independent of b. If relying on the RO model (to achieve adaptive security), a multiplicative factor of q_k (which is the number of RO queries) is introduced (cf. [BBG05a, GW19, DN19, DGNW20]). Let S_0 and S_1 be the events that D outputs 1 to its $\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}$-challenger in Game 0 and Game 1, respectively. Consequently, we have that $\Pr[S_0] = \mathbf{Adv}^{\mathsf{thibe\text{-}ind\text{-}cpa}}_{\mathsf{THIBE},A}(1^\kappa, \ell)/q_k + 1/2$ and $\Pr[S_1] = 1/2$. Hence, for $\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(1^\kappa) = |\Pr[D(pars, e(g_1, g_2^r)^{\alpha^{\ell+2}}) = 1] - \Pr[D(pars, e(g_1, g_2^r)^u) = 1]|$, this yields

$$\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(1^\kappa) \geq |\Pr[S_0] - \Pr[S_1]| = \mathbf{Adv}^{\mathsf{thibe\text{-}ind\text{-}cpa}}_{\mathsf{THIBE},A}(1^\kappa, \ell)/q_k,$$

with $q = \ell + 1$, which concludes the proof.

THIBE-IND-CCA security. We now discuss how to obtain THIBE-IND-CCA security for our construction THIBE by applying the well-known Fujisaki-Okamoto transform [FO99]. Basically, the encryption algorithm will encrypt as its message (M, r) with M the original message and r a sufficiently large randomly sampled bit string (this requires an injective encoding (M, r) into the message space of the THIBE scheme). The THIBE-encryption is de-randomized and uses as random coins $H(r)$ where H is a hash function modeled as a random oracle (RO) to obtain the ciphertext C^t_{id}. The decryption algorithm applies the original decryption algorithm from the THIBE-IND-CPA-secure THIBE scheme to receive (M', r'). Then, it re-encrypts (M', r') using random coins $H(r, M')$ to obtain the ciphertext $\overline{C}^t_{id}$. If it holds that $C^t_{id} = \overline{C}^t_{id}$, it outputs M' and otherwise it outputs $\perp$.

Corollary 1. *If the q-wBDHI assumption holds, then* THIBE *defined in Scheme 1 is THIBE-IND-CCA-secure in the RO model. Concretely, for any valid PPT adversary A with at most $q_k = q_k(\kappa)$ key queries, there is a distinguisher D on q-wBDHI with $q = \ell + 1$, such that*

$$\mathbf{Adv}^{\mathsf{thibe\text{-}sind\text{-}cca}}_{\mathsf{THIBE},A}(1^\kappa, \ell) \leq q_k \cdot q_c \cdot \mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(1^\kappa),$$

for group generator BGen *and number of RO-queries $q_k = q_k(\kappa)$ and $q_c = q_c(\kappa)$.*

4 Dual-Form Puncturable Encryption

Puncturable encryption (PE) has been introduced by Green and Miers in [GM15] and subsequently used and refined in several works, e.g., in [CHN+16, CRRV17, GHJL17, DJSS18, DKL+18b]. We recall, that a PE scheme is a public-key encryption scheme where each ciphertext can be encrypted with respect to one (or more tags). PE features an additional puncturing algorithm that takes a secret key and a tag t as input and produces an updated secret key. This updated secret key is able to decrypt all ciphertexts *except* those tagged with t and (updated) secret keys can be iteratively "punctured" on distinct tags. (In our generalized allow-/deny-list encryption concept, this will correspond to our secret-key manipulation with respect to a deny list.)

Despite being slightly different in their concrete formulation (e.g., some schemes allow single tags, others multiple tags), existing PE schemes all provide the same basic idea in their functionality, i.e., that they allow to puncture secret keys in a way that they can no longer decrypt certain ciphertexts. A notable difference is in the formulation of Fully PE (FuPE) from Derler et al. [DKL+18a] where secret keys can be punctured with respect to so-called negative tags (resembling the functionality of PE) and in addition to so-called *positive* tags. If a secret key is punctured with respect to a positive tag, then it can *only* decrypt ciphertexts that are tagged with respect to the corresponding positive tag. Although this approach adds more flexibility, it still lacks an important feature, namely, once keys are positively punctured, they can no longer be negatively punctured. Mapped to the application that we have in mind, this means that derived FuPE keys will loose the key-manipulation property (a versatile feature that we want to enable). To mitigate this problem and to make the concepts of PE more comprehensible, we introduce the new notion of Dual-Form PE (DFPE) which enables the negative-puncturing features of keys after those keys have already been positively punctured.

4.1 Definition, Correctness, and Security Notions of DFPE

Before constructing DFPE, we first present our DFPE definition and continue with its correctness property as well as its security notions.

Definition 3 (DFPE). *A Dual-Form Puncturable Encryption (DFPE) scheme* DFPE *with message space* $\mathcal{M}$*, positive and negative tag spaces* $\mathcal{T}_+$ *and* $\mathcal{T}_-$*, respectively, consists of the PPT algorithms* $(\mathsf{Gen}, \mathsf{NPunc}, \mathsf{PPunc}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Gen}(1^\kappa, \ell_-)$: *key generation, on input a unary security parameter* $1^\kappa \in \mathbb{N}$ *and maximum number of negative tags* $\ell_- \in \mathbb{N}$*, outputs public and secret keys* $(\mathsf{pk}, \mathsf{sk}^\varepsilon_\varepsilon)$*. (We assume that* pp *implicitly determines* $\mathcal{M}$*,* $\mathcal{T}_+$*, and* $\mathcal{T}_-$*; we consider* ε *to be not part of the positive and negative tag spaces.)*

$\mathsf{NPunc}(\mathsf{sk}^{t_+}_T, t_-)$: *negative puncturing, on input a secret key* $\mathsf{sk}^{t_+}_T$ *with* $T \subset \mathcal{T}_- \cup \{\varepsilon\}$ *and* $t_+ \in \mathcal{T}_+ \cup \{\varepsilon\}$*, and a tag* $t_- \in \mathcal{T}_-$*, outputs* $\mathsf{sk}^{t_+}_{T \cup \{t_-\}}$*.*

$\mathsf{PPunc}(\mathsf{sk}^\varepsilon_T, t_+)$: *positive puncturing, on input a secret key* $\mathsf{sk}^\varepsilon_T$ *and positive tag* $t_+ \in \mathcal{T}_+$*, outputs a key* $\mathsf{sk}^{t_+}_T$*.*

$\mathsf{Enc}(\mathsf{pk}, M, t_-, t_+)$: *encryption, on input a public key* pk*, a message* $M \in \mathcal{M}$*, a negative tag* $t_- \in \mathcal{T}_-$*, and a positive tag* $t_+ \in \mathcal{T}_+$*, outputs a ciphertext* $C^{t_+}_{t_-}$*. (We note that* t_+ *and* t_- *are publicly retrievable given the ciphertext and the public key* pk*.)*

$\mathsf{Dec}(\mathsf{sk}_T^{t'_+}, C_{t_-}^{t_+})$: *on input a secret key* $\mathsf{sk}_T^{t'_+}$ *and a ciphertext* $C_{t_-}^{t_+}$*, outputs* $M \in \mathcal{M}$ *if* $t_- \notin T$ *and* $t'_+ = t_+$*; else output* $\perp$*.*

Correctness of DFPE. Essentially, correctness ensures that even if a secret key is negatively punctured and afterwards positively punctured, or vice versa, decryption succeeds if the resulting secret key matches the positive tag of the ciphertext and the negative tag of the ciphertext was not already punctured.

More formally, for all $\kappa, \ell_- \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon) \leftarrow \mathsf{Gen}(1^\kappa, \ell_-)$, all $T \subset \mathcal{T}_- \cup \{\varepsilon\}$, all $t_- \in \mathcal{T}_-$, all $t_+ \in \mathcal{T}_+ \cup \{\varepsilon\}$, all arbitrarily interleaved runs of $\mathsf{sk}_{T\cup\{t_-\}}^{t_+} \leftarrow \mathsf{NPunc}(\mathsf{sk}_T^{t_+}, t_-)$, all $t'_+ \in t_+$ and $\mathsf{sk}_T^{t_+} \leftarrow \mathsf{PPunc}(\mathsf{sk}_T^\varepsilon, t'_+)$, all $M \in \mathcal{M}$, all $C_{t_-}^{t'_+} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, t_-, t'_+)$, we have that $\mathsf{Dec}(\mathsf{sk}_T^{t'_+}, C_{t_-}^{t'_+}) = M$ if $t_- \notin T$.

DFPE-IND-CPA and DFPE-IND-CCA security notions. We define security notions for DFPE, dubbed DFPE-IND-CPA and DFPE-IND-CCA. A DFPE scheme is DFPE-IND-CPA-secure or DFPE-IND-CCA-secure if any PPT adversary A succeeds in the following experiment only with probability at most negligibly larger than $1/2$. First, public and secret keys $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon)$ are honestly generated. During the experiments, A may adaptively query a $\mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, \cdot, \cdot)$-oracle, while for the DFPE-IND-CCA experiment, A may adaptively query a $\mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, \cdot)$-oracle additionally:

- $\mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, T, t_+)$, on input secret key $\mathsf{sk}_\varepsilon^\varepsilon$, negative-tag set $T \subset \mathcal{T}_-$, and positive tag $t_+ \in \mathcal{T}_+ \cup \{\varepsilon\}$, outputs $\mathsf{sk}_T^{t_+} \leftarrow \mathsf{PPunc}(\mathsf{sk}_{T_\ell}^\varepsilon, t_+)$, for iteratively punctured secret key $\mathsf{sk}_{T_i}^\varepsilon \leftarrow \mathsf{NPunc}(\mathsf{sk}_{T_{i-1}}^\varepsilon, t_{i-1})$, for all pairwise-different tags $(t_0, \ldots, t_{\ell-1}) \in (T)^\ell$ with $\ell := |T|$ and $i \in [\ell]$ in arbitrary order. (It allows the positive tag $t_+ = \varepsilon$ but not the negative-tag set $T = \{\varepsilon\}$ nor the empty set $T = \emptyset$ as input.)
- $\mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, C_{t_-}^{t_+})$, on input secret key $\mathsf{sk}_\varepsilon^\varepsilon$ and ciphertext $C_{t_-}^{t_+}$, derives $\mathsf{sk}_\varepsilon^{t_+} \leftarrow \mathsf{PPunc}(\mathsf{sk}_\varepsilon^\varepsilon, t_+)$ and outputs $M \leftarrow \mathsf{Dec}(\mathsf{sk}_\varepsilon^{t_+}, C_{t_-}^{t_+})$. (The oracle does not allow a ciphertext input associated to the tags $t_- = \varepsilon$ and $t_+ = \varepsilon$.)

The public key pk is given to A. A outputs equal-length messages (M_0, M_1), a target negative tag $t_-^* \in \mathcal{T}_-$, and a target positive tag $t_+^* \in \mathcal{T}_+$. The target challenge ciphertext $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, t_-^*, t_+^*)$, for uniform $b \leftarrow \{0,1\}$, is given to A. Eventually, A outputs a guess b^*, and succeeds, i.e., the experiment outputs 1, if the equation $b = b^*$ holds.

We say that A is *valid* if and only if A has not queried the Ext-oracle to obtain keys such that the challenge ciphertext can be trivially decrypted; for the DFPE-IND-CCA case, we additionally require that A did not query Dec'-oracle with the challenge ciphertext. More concretely, if any valid PPT A succeeds only with probability at most negligibly larger than $1/2$, then we say an DFPE scheme is DFPE-IND-CPA and DFPE-IND-CCA secure, respectively. In Experiment 2, we formally state the security experiments.

Definition 4. *We define the advantage of an adversary* A *in the DFPE-IND-*T *experiment* $\mathsf{Exp}_{\mathsf{DFPE},A}^{\mathsf{dfpe\text{-}ind\text{-}T}}(1^\kappa, \ell_-)$ *as*

$$\mathbf{Adv}_{\mathsf{DFPE},A}^{\mathsf{dfpe\text{-}ind\text{-}T}}(1^\kappa, \ell_-) := \left|\Pr\left[\mathsf{Exp}_{\mathsf{DFPE},A}^{\mathsf{dfpe\text{-}ind\text{-}T}}(1^\kappa, \ell_-) = 1\right] - \tfrac{1}{2}\right|.$$

We say a DFPE scheme DFPE *is DFPE-IND-*T*-secure for* $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$*, if* $\mathbf{Adv}_{\mathsf{DFPE},A}^{\mathsf{dfpe\text{-}ind\text{-}T}}(1^\kappa, \ell_-)$ *is a negligible function in* κ *for all valid PPT* A*.*

Experiment $\mathsf{Exp}^{\mathsf{dfpe\text{-}ind\text{-}T}}_{\mathsf{DFPE},A}(1^\kappa, \ell_-)$

$(\mathsf{pk}, \mathsf{sk}^\varepsilon_\varepsilon) \leftarrow \mathsf{Gen}(1^\kappa, \ell_-)$
$(M_0, M_1, t^*_-, t^*_+, \mathsf{st}) \leftarrow A^{\mathsf{Ext}(\mathsf{sk}^\varepsilon_\varepsilon, \cdot, \cdot), \mathsf{Dec}'(\mathsf{sk}^\varepsilon_\varepsilon, \cdot)}(\mathsf{pk})$
$b \leftarrow \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, t^*_-, t^*_+)$
$b^* \leftarrow A^{\mathsf{Ext}(\mathsf{sk}^\varepsilon_\varepsilon, \cdot, \cdot), \mathsf{Dec}'(\mathsf{sk}^\varepsilon_\varepsilon, \cdot)}(\mathsf{st}, \mathsf{pk}, C^*)$
if $b = b^*$ return 1, else return 0

Experiment 2: DFPE-IND-T-security for DFPE: $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

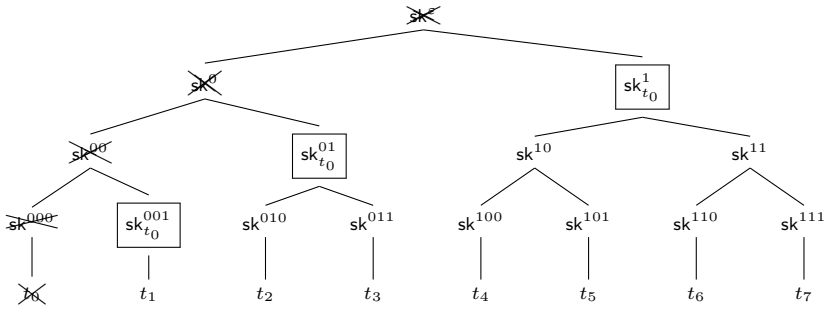

Figure 1. Example of a DFPE secret key that has been punctured on t_0. The secret key $\mathsf{sk}^{t_+}_{t_0}$ has the boxed elements $(\mathsf{sk}^{001}_{t_0}, \mathsf{sk}^{01}_{t_0}, \mathsf{sk}^{1}_{t_0})$.

4.2 Constructing Dual-Form Puncturable Encryption

Subsequently, we present a construction of a DFPE scheme based on pairings. Unfortunately, we cannot instantiate our DFPE scheme directly from HIBEs as done in prior work on PE [GHJL17] and Fully PE [DKL+18a, DKL+18b]. The reason is that we want to allow puncturings even after secret keys were extracted for a specific positive tag such that those keys can be further restricted with respect to negative tags. Realizing this generically from HIBEs stays unknown, but we use tagged HIBEs (THIBEs) to construct DFPE-IND-CCA-secure DFPE. This allows to fulfill the needs for our applications we have in mind not being achieved before by FuPE. By applying THIBEs, we are able to instantiate our DFPE scheme using Type-3 bilinear groups as they represent the state-of-the-art regarding efficiency and similarity of the security levels of the base and target groups.

To construct a DFPE scheme from THIBEs, we implicitly arrange negative tags of the DFPE scheme associated to secret keys in a complete binary tree, i.e., the nodes represent a prefix bit representation of the negative tag and, hence, the root of the tree is associated with keys $\mathsf{sk}^{t_+}_\varepsilon$ of the DFPE. In Figure 1, we give an example with a secret key punctured on a negative tag t_0.

We define an additional PPT helper algorithm Trunc to prune the tree to output a punctured secret key that corresponds to a given set of tags. This is reminiscent of prior works, e.g., [CHK03, GHJL17, DJSS18, DKL+18a].

Intuition of Trunc**.** Essentially, Trunc takes the current tree configuration as provided in the secret key (i.e., which tags are already punctured and, hence, how the tree is pruned for such tags). It further receives a negative tag t_- that will be punctured. Trunc first finds all

elements from the root to the associated leaf of tag t_-. (Since those elements can be used to derive a secret key for tag t_-.) It delegates the key elements on that path such that no ancestor elements for t_- are available anymore and keeps the other key elements. The result is a pruned tree that excludes secret-key material for t_- for the new set of punctured tags $T \cup \{t_-\}$. The concrete PPT algorithms works as follows (see that the positive tag t_+ is not touched in Trunc).

$\mathsf{Trunc}(\mathsf{sk}_T^{t_+}, t_-)$: on input keys $(\mathsf{sk}_{T,1}, \ldots, \mathsf{sk}_{T,m}) := \mathsf{sk}_T^{t_+}$, for some integer m, output a punctured secret key according to $t_- = (t_1, \ldots, t_\ell)$ as follows:

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

1b. if there exist a leaf associated to a t_--secret key $\mathsf{sk}_{T,i}$, for $i \in [m]$, then set $\mathsf{sk}_{T\cup\{t_-\}}^{t_+} := \mathsf{sk}'_{T,\leq m}$, where $\mathsf{sk}'_{T,\leq m}$ is the same as $\mathsf{sk}_T^{t_+}$, but without the leaf-associated secret key $\mathsf{sk}_{T,i}$.

2. Output $\mathsf{sk}_{T\cup\{t_-\}}^{t_+}$.

Concrete construction of DFPE. The intuition of the concrete DFPE construction is as follows. Key generation returns the public-secret key pair of the THIBE key generation as its initial public and secret keys. The negative and positive tag spaces are set to $\mathcal{T}_- = |\mathcal{ID}^\ell|$ (i.e., corresponding to a leaf in the tree) and $\mathcal{T}_+ = \mathcal{T}$ (i.e., corresponding to THIBE's tags), respectively. Negative puncturing takes a secret key, runs Trunc to truncate the tree, and returns the punctured secret key (according to the pruned-tree configuration). Positive puncturing takes a secret key and a positive tag t_+, and punctures all part secret keys with t_+ using Tag. Encryption takes the public key, negative and positive tags, and the message to return the output of THIBE's encryption algorithm. Decryption finds the associated secret key part such that the negative tag t_- of the ciphertext is matched (i.e., if t_- was not yet punctured in the secret key, then such key material is available). Furthermore, if the positive tag t_+ of the secret key matches the positive tag of the ciphertext, then decryption returns the output of THIBE's decryption algorithm.

More formally, let $\mathsf{THIBE} = (\mathsf{THIBE.Gen}, \mathsf{THIBE.Del}, \mathsf{THIBE.Tag}, \mathsf{THIBE.Enc}, \mathsf{THIBE.Dec})$ with message space $\mathcal{M}_{\mathsf{THIBE}}$, identity space $\mathcal{ID}^{\leq\ell}$, and tag space $\mathcal{T}$ be a THIBE scheme. We present our DFPE scheme $\mathsf{DFPE} = (\mathsf{Gen}, \mathsf{NPunc}, \mathsf{PPunc}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M} := \mathcal{M}_{\mathsf{THIBE}}$, negative tag space $\mathcal{T}_- := \mathcal{ID}^\ell$, and positive tag space $\mathcal{T}_+ := \mathcal{T}$ in Scheme 2 and further show that it satisfies correctness and the DFPE-IND-CCA security notion.

Correctness of DFPE. Correctness essentially follows from the correctness of THIBE; in particular, see that if $t_- \notin T$, then there exist a part-secret key that is capable of decrypting the ciphertext. A proof of the following theorem is deferred to Appendix B.

Theorem 2. *If* THIBE *is a THIBE-IND-CCA-secure THIBE, then* DFPE *defined in Scheme 2 is DFPE-IND-CCA-secure. Concretely, for any valid PPT adversary A, there is an adversary D on the THIBE-IND-CCA-security, such that*

$$\mathbf{Adv}^{\mathsf{dfpe\text{-}ind\text{-}cca}}_{\mathsf{DFPE},A}(1^\kappa, \ell_-) \leq \mathbf{Adv}^{\mathsf{thibe\text{-}sind\text{-}cca}}_{\mathsf{THIBE},D}(1^\kappa, \ell_-),$$

$\underline{\mathsf{Gen}(1^\kappa, \ell)}$: Return $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon) \leftarrow \mathsf{THIBE.Gen}(1^\kappa, \ell)$. (We assume that the negative tag space $\mathcal{T}_-$ is of size $|\mathcal{ID}^\ell|$ for simplicity.)

$\underline{\mathsf{NPunc}(\mathsf{sk}_T^{t_+}, t_-)}$: Return $\mathsf{sk}_{T \cup \{t_-\}}^{t_+} \leftarrow \mathsf{Trunc}(\mathsf{sk}_T^{t_+}, t_-)$.

$\underline{\mathsf{PPunc}(\mathsf{sk}_T^\varepsilon, t_+)}$: Compute $\mathsf{sk}_{t_i}^{t_+} \leftarrow \mathsf{THIBE.Tag}(\mathsf{sk}_{t_i}^\varepsilon, t_+)$, for all $\mathsf{sk}_T^\varepsilon =: (\mathsf{sk}_{t_1}^\varepsilon, \ldots, \mathsf{sk}_{t_m}^\varepsilon)$, for some integer m, and output $\mathsf{sk}_T^{t_+} := (\mathsf{sk}_{t_i}^{t_+})_{i \in [m]}$.

$\underline{\mathsf{Enc}(\mathsf{pk}, M, t_-, t_+)}$: Return $C_{t_-}^{t_+} \leftarrow \mathsf{THIBE.Enc}(\mathsf{pk}, M, t_-, t_+)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_T^{t'_+}, C_{t_-}^{t_+})}$: Parse $\mathsf{sk}_T^{t'_+} =: (\ldots, \mathsf{sk}_{t'_-}^{t'_+}, \ldots)$ such that t'_- is a prefix of t_-, run $\mathsf{sk}_{t_-}^{t'_+} \leftarrow \mathsf{THIBE.Del}(\mathsf{sk}_{t'_-}^{t'_+}, t)$, and return $M := \mathsf{THIBE.Dec}(\mathsf{sk}_{t_-}^{t'_+}, C_{t_-}^{t_+})$. (Note that if $t_- \notin T$, such prefix always exists.)

Scheme 2: DFPE-IND-CCA-secure DFPE scheme DFPE.

for some integer $\ell_- \in \mathbb{N}$.

4.3 Implementation and Evaluation

We have implemented our DFPE-IND-CPA-secure DFPE scheme as presented in Section 4.2 in Python 3.8 based on pyrelic[5] using the `BN254` curve that yields a security level of around 100 bit [MSS16, BD19] with the relic pairing library version 0.5.0 [AGM+]. The measurements were performed on a laptop with an Intel Core i7-8650U @ 1.9 GHz running Ubuntu 20.04. In Table 2, we present the average runtime over 100 runs each and sizes of public keys, secret keys and ciphertexts using negative tag spaces of size 2^{48}, 2^{64}, and 2^{80} for a random message, respectively. From Table 2, one can see that the algorithms Gen, Enc, Dec, and

Tags	Gen	Enc	Dec	PPunc	NPunc	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}_\varepsilon^\varepsilon\rvert$	$\max \lvert\mathsf{sk}_T^{id}\rvert$	$\lvert C\rvert$
2^ℓ	G	$\mathcal{O}(\ell)G$	$2P$	$\mathcal{O}(\ell)G$	$\mathcal{O}(\ell^2)G$	G_2	$(\ell+2)G_1, G_2$	$\mathcal{O}(\ell^2)G_1, \mathcal{O}(\ell)G_2$	G_1, G_2
2^{48}	2.0	2.7	0.4	2.3	110.2	64	1664	78336	96
2^{64}	2.6	3.4	0.4	3.0	192.0	64	2176	137216	96
2^{80}	3.1	4.1	0.4	3.7	292.6	64	2688	212480	96

Table 2: Performance estimation and evaluation: exponentiations in G_1 and G_2 are denoted as G, pairings as P. Runtime in ms and sizes in bytes.

PPunc are very efficient. The benchmarks of the Dec algorithm assumes that no additional key extraction is necessary. Thus, the runtime of the decryption is independent of the size of the parameter space. The NPunc algorithm needs less than a second for all levels, but is still the slowest algorithm overall.[6] However, (negative) puncturing is often an offline operation and thus our performance results are perfectly acceptable.

[5] `https://github.com/sebastinas/pyrelic`, commit `264e6396`

[6] It is a central open issue in the context of PE to make these algorithms more efficient.

Furthermore, we can observe that the size of ciphertexts do not depend on the size of the negative tag space. In the random oracle model, the basis elements stored in the public key can be derived via the random oracle, so its size can be made independent of the tag space. The size of the secret key depends both on the tag space as well as the performed puncturings. Depending on the exact choice of tags, punctured secret keys can grow up to at most 78 MB, 137 MB and 212 MB, respectively.

5 Applications

5.1 Cloudflare's Geo Key Manager

Let us now continue the discussion on Cloudflare's Geo Key Manager[7] in more detail. The currently used system combines pairing-based constructions of identity-based broadcast encryption (IBBE) [Del07] as well as identity-based revocation (IBR) [ALdP11] schemes with compact ciphertexts in the following way: first, the private key[8] sk is secret shared in two shares sk_1 and sk_2. Using the IBBE scheme, the first share, sk_1 is encrypted with respect to the allowed regions. The second share, sk_2 is encrypted with the IBR scheme revoking access for all denied colocations. Now, if a colocation receives the two ciphertexts, it can recover the encrypted shares only if it is within the allowed region and not one of the denied colocations. Otherwise, it can only recover at most one of the shares. For adding allowed colocations not already contained within one of those allowed areas, sk is additionally encrypted with the IBBE scheme for all allowed colocations. So overall, ciphertexts contain two constant-size IBBE ciphertexts and a constant-size IBR ciphertext. We can obtain the same functionality also from *only* using DFPE. In addition, we also obtain the important property of forward secrecy.

Using DFPE. The idea is to allow the regions using the positive tags and deny colocations by puncturing on unique negative tags assigned to each colocation. Indeed, assume that $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon)$ is Cloudflare's DFPE key-pair. Next, Cloudflare would derive keys for each region by using the name of the region as positive tag in the DFPE scheme, e.g., obtaining $\mathsf{sk}_\varepsilon^{\mathrm{EU}}$ for Europe. Each colocation is assigned a unique negative tag and they receive the secret key for the region additionally punctured on that tag, e.g., $\mathsf{sk}_{\mathrm{London}}^{\mathrm{EU}}$ for the European data center in London. If customers now want to store their secret key, they encrypt the key for each allowed region using the region as positive tag and the denied colocations of the corresponding region as negative tags. If a colocation needs to access the key, it can only decrypt if one of the ciphertexts was encrypted for the region and that particular ciphertext was not tagged with one of the positive tags of the colocation. They are unable to decrypt the other ciphertexts, since they do not have access to the positively tagged keys. For denying colocations in allowed regions, we follow the same approach, but encrypt the ciphertext including all negative tags of the region's colocations without the ones being allowed.

Achieving forward secrecy. Since DFPE allows to puncture on multiple negative tags, we can additionally obtain forward secrecy as an important new feature: we can partition the tag space into one part containing the colocation tags, and another part identifying time periods by viewing this part as ordered sequence. Thereby, the customers can specify a time epoch as additional negative tag, say $t = 2021\text{-}02$ for ciphertexts decryptable in February 2021. Once

[7] https://blog.cloudflare.com/geo-key-manager-how-it-works

[8] Technically, an encryption key for sk, but that does not make a difference.

the month passed, all colocations puncture the secret keys on the month's tag, and are then no longer able to decrypt those ciphertexts. The time periods can be designed in such a way, that they match the renewal periods of certificates. For example, Let's Encrypt [ABC+19] certificates are renewed every three months, and their `certbot` reference client generates new secret keys during certificate renewal.

We note that when switching to DFPE, each colocation only has to manage one DFPE secret key instead of an IBBE and an IBR secret key and DFPE easily achieves both allowing and denying of areas and colocations with one single primitive, respectively, while at the same time it additionally provides forward-secrecy. The approach of using DFPE, however, comes at the cost of having the size of the ciphertexts depend linearly on the number of allowed areas instead of 3 ciphertexts when using IBBE and IBR. When considering continents, the number of regions is very small (< 10), thus our ciphertext size can be considered quasi-constant. Also note that the combined IBBE and IBR ciphertext requires 576 bytes[9] using the same curve, which corresponds to six regions in the DFPE-based approach. Therefore, when considering the continents as regions, we obtain ciphertexts that are at most the same size and, thus, forward secrecy is achieved without additional cost. Additionally, Cloudflare only needs to send the ciphertexts for specific regions to their colocations. Hence, colocations only have to store a single ciphertext. We note that the number of negative tags (denied colocations together with the epoch) does not influence the ciphertext size.

5.2 Cryptographic Primitives

Forward-Secret Identity-Based Encryption. Interestingly, although there are some works on forward-secret IBE [CRF+11, SPB13, LL17], they all consider a very weak model in which the master secret key stays constant and, hence, the private key generator (PKG) is able to generate user keys for arbitrary time periods and, thus, inherently invalidating an important aspect of forward secrecy. We are only aware of a dedicated construction of a forward-secret hierarchical IBE (HIBE) by Yao et al. [YFDL04], which also yields a forward-secret IBE as a special case. This works also considers forward-secrecy for the master secret key. As we will show, DFPE generically yields forward-secret IBE and, thus, offering new instantiations thereof. In particular, to the best of our knowledge, using the concrete DFPE construction, this leads to the first fs-IBE scheme with compact ciphertexts. We recall the definition of an fs-IBE scheme and its security in Appendix A.

fs-IBE construction. Having a DFPE scheme allows to construct an fs-IBE scheme by mapping time intervals to negative tags. The only syntactical difference is that the NPunc and PPunc algorithms of DFPE are mapped to the Update and Ext algorithms of fs-IBE. In particular, when we are at a time interval i in the fs-IBE scheme, this corresponds to secret keys that are punctured with respect to tag set $T = \{1, \ldots, i-1\}$ in the DFPE scheme and moving from time interval i to interval $i+1$ corresponds to puncturing the secret key at tag i, i.e., $T := T \cup \{i\}$. It is straightforward to show the following:

Corollary 2. *If the* DFPE *scheme provides DFPE-IND-*T*-security, then the resulting* fs-IBE *scheme provides fs-IBE-IND-*T*-security, for* $\mathsf{T} \in \{CPA, CCA\}$.

Forward-Secret Signatures. Forward-secret signatures [BM99, Kra00, IR01, AABN02] are a primitive that has recently found interest in distributed ledgers [DGKR18, GW19, DN19, DGNW20].

[9] https://rwc.iacr.org/2018/Slides/Sullivan.pdf

Having a DFPE scheme and, in particular, a fs-IBE scheme, we can generically construct a forward-secret signature scheme. The idea is simply to adopt the Naor-transform [BF01], which converts any IBE-IND-CPA secure IBE scheme into an EUF-CMA secure signature scheme. We first briefly recall this transform: We consider an IBE scheme and the master secret key sk acts as the signing key. Let $id = m$, the message to be signed, then sk^m extracted with sk for identity m acts as the signature for m. The signature verification is done by checking if sk^m functions properly as a correct IBE decryption key for identity m by encrypting a random plaintext and checking if decryption yields to the original plaintext.

The basic idea of this transform applied to the forward-secret setting is as follows. We start with the master secret key $\mathsf{sk}_\varepsilon^\varepsilon$ as initial signing key and to develop the signing key over time, we update the secret key to the next time period, i.e., to update the signing key from interval i to interval $i+1$ we run $\mathsf{sk}_{i+1}^\varepsilon \leftarrow \mathsf{Update}(\mathsf{sk}_i^\varepsilon, \varepsilon, i)$. Now, within every time interval i one uses the current signing key with the above Naor-transform. One straightforwardly obtains:

Corollary 3. *If the* fs-IBE *scheme provides fs-IBE-IND-CPA-security, then the signature scheme obtained via the Naor-transform provides EUF-CMA-security.*

Using our DFPE from Section 4.2 in the above compiler, this yields forward-secret signatures with the same efficiency as in recent work [DN19, GW19, DGNW20].

Acknowledgements. This work was supported by the European commission through ECSEL Joint Undertaking (JU) under grant agreement n°783119 (SECREDAS), grant agreement n°826610 (COMP4DRONES), through the Horizon 2020 research and innovation programme under grant agreement n°871743 (KRAKEN), project IoT4CPS funded by the Austrian "ICT of the future" program of the Austrian Research Promotion Agency (FFG) and the Federal Ministry of Austria for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (PROFET). The work of the first two authors was partly done while at Graz University of Technology.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT*, volume 2332 of *LNCS*, pages 418–433. Springer, 2002.

[ABC+19] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth D. Schoen, and Brad Warren. Let's encrypt: An automated certificate authority to encrypt the entire web. In *ACM CCS*, pages 2473–2487. ACM, 2019.

[AGM+] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[AKN07] Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. In *ESORICS 2007*, 2007.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *PKC*, volume 6571 of *LNCS*, pages 90–108. Springer, 2011.

[BBG05a] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BBG05b] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/15, 2005.

[BD19] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *J. Cryptology*, 32(4):1298–1336, 2019.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *CRYPTO*, volume 1666 of *LNCS*, pages 431–448. Springer, 1999.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In *CT-RSA*, volume 2612 of *LNCS*, pages 1–18. Springer, 2003.

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

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

[CM11] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. *Discret. Appl. Math.*, 159(13):1311–1322, 2011.

[Coc01] Clifford C. Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, 2001.

[CRF+11] Dario Catalano, Mario Di Raimondo, Dario Fiore, Rosario Gennaro, and Orazio Puglisi. Fully non-interactive onion routing with forward-secrecy. In *ACNS*, volume 6715 of *LNCS*, pages 255–273, 2011.

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-ciphertext secure fully homomorphic encryption. In *PKC (2)*, volume 10175 of *LNCS*, pages 213–240. Springer, 2017.

[CRSS20] Valerio Cini, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Cca-secure (puncturable) kems from encryption with non-negligible decryption errors. In *ASIACRYPT*, 2020.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *ASIACRYPT*, volume 4833 of *LNCS*, pages 200–215. Springer, 2007.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)*, volume 10821 of *LNCS*, pages 66–98. Springer, 2018.

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In *USENIX*, pages 2093–2110. USENIX Association, 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 *EUROCRYPT (3)*, volume 10822 of *LNCS*, pages 425–455. Springer, 2018.

[DKL+18a] David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In *PKC (1)*, volume 10769 of *LNCS*, pages 219–250. Springer, 2018.

[DKL+18b] David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. *IACR ePrint*, 2018:321, 2018.

[DN19] Manu Drijvers and Gregory Neven. Forward-secure multi-signatures. *IACR Cryptol. ePrint Arch.*, 2019:261, 2019.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.

[GHJL17] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. In *EUROCRYPT (3)*, volume 10212 of *LNCS*, pages 519–548, 2017.

[GM15] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *IEEE S&P*, pages 305–320. IEEE, 2015.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[GW19] Sergey Gorbunov and Hoeteck Wee. Digital signatures for consensus. Cryptology ePrint Archive, Report 2019/269, 2019.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, volume 2332 of *LNCS*, pages 466–481. Springer, 2002.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO*, volume 2139 of *LNCS*, pages 332–354. Springer, 2001.

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *ACM CCS*, pages 108–115. ACM, 2000.

[LL17] Yang Lu and Jiguo Li. Forward-secure identity-based encryption with direct chosen-ciphertext security in the standard model. *Adv. Math. Commun.*, 11(1):161–177, 2017.

[MSS16] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In *Mycrypt*, volume 10311 of *LNCS*, pages 83–108. Springer, 2016.

[RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM CCS*, pages 199–212. ACM, 2009.

[SDLP20] Willy Susilo, Dung Hoang Duong, Huy Quoc Le, and Josef Pieprzyk. Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption. In *ESORICS (2)*, volume 12309 of *LNCS*, pages 107–127. Springer, 2020.

[SG20] M. Schwarz and D. Gruss. How trusted execution environments fuel research on microarchitectural attacks. *IEEE Security Privacy*, 18(5):18–27, 2020.

[SPB13] Kunwar Singh, C. Pandurangan, and A. K. Banerjee. Lattice based forward-secure identity based encryption scheme with shorter ciphertext. *J. Internet Serv. Inf. Secur.*, 3(1/2):5–19, 2013.

[SSS+20] Shifeng Sun, Amin Sakzad, Ron Steinfeld, Joseph K. Liu, and Dawu Gu. Public-key puncturable encryption: Modular and compact constructions. In *PKC (1)*, volume 12110 of *LNCS*, pages 309–338. Springer, 2020.

[WCW+19] Jianghong Wei, Xiaofeng Chen, Jianfeng Wang, Xuexian Hu, and Jianfeng Ma. Forward-secure puncturable identity-based encryption for securing cloud emails. In *ESORICS (2)*, volume 11736 of *LNCS*, pages 134–150. Springer, 2019.

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM CCS*, pages 354–363. ACM, 2004.

[ZJRR12] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-vm side channels and their use to extract private keys. In *ACM CCS*, pages 305–316. ACM, 2012.

[ZJRR14] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-tenant side-channel attacks in paas clouds. In *ACM CCS*, pages 990–1003. ACM, 2014.

Experiment $\mathsf{Exp}^{\mathsf{fs\text{-}ibe\text{-}ind\text{-}T}}_{\mathsf{fs\text{-}IBE},A}(\kappa,\ell_-)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa,\ell_-), (\mathsf{pk},\mathsf{sk}^{\varepsilon}_{\varepsilon}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
$(M_0, M_1, i^*, id^*, \mathsf{st}) \leftarrow A^{\mathsf{Ext}(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot,\cdot),\, \mathsf{Dec}'(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot)}(\mathsf{pk})$
$b \leftarrow \{0,1\}$
$C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, i^*, id^*)$
$b^* \leftarrow A^{\mathsf{Ext}(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot,\cdot),\, \mathsf{Dec}'(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot)}(\mathsf{st},\mathsf{pk},C^*)$
if $b = b^*$ return 1, else return 0

Experiment 3: fs-IBE-IND-T-security for fs-IBE: $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

A Forward-Secure Identity-Based Encryption

Definition 5 (fs-IBE). *A forward-secure identity-based encryption (*fs-IBE*) scheme with message space $\mathcal{M}$ and identity space $\mathcal{ID}$ consists of the PPT algorithms* (Setup, Gen, Ext, Update, Enc, Dec)*:*

$\mathsf{Setup}(\kappa,\ell_-)$: *On input security parameter κ and maximum number of time periods ℓ_- and outputs public parameters* pp*.*
$\mathsf{Gen}(\mathsf{pp}, id, N)$: *On input public parameters parameter* pp *(which is in implicit input to all algorithms) and the total number of time periods, outputs a master keypair* $(\mathsf{pk},\mathsf{sk}^{\varepsilon}_{\varepsilon})$*.*
$\mathsf{Ext}(\mathsf{sk}^{\varepsilon}_{j}, id, i)$: *On input a master secret key $\mathsf{sk}^{\varepsilon}_{j}$ and identity id and a time period i with $i \geq j$ output a secret key sk^{id}_{i}.*
$\mathsf{Update}(\mathsf{sk}^{id}_{i}, id, i)$: *On input a (master) secret key sk^{id}_{i}, identity id and time period i output a secret key sk^{id}_{i+1}.*
$\mathsf{Enc}(\mathsf{pk}, M, id, i)$: *On input a master public key* pk*, message $M \in \mathcal{M}$ identity $id \in \mathcal{ID}$ and time period i, outputs a ciphertext C^{id}_{i} for identity id and time period i.*
$\mathsf{Dec}(\mathsf{sk}^{id'}_{i}, C^{id}_{i}, i)$: *On input a secret key $\mathsf{sk}^{id'}_{i}$ and a ciphertext C^{id}_{i}, outputs $M \in \mathcal{M} \cup \{\perp\}$.*

We require that for all $\kappa,\ell_- \in \mathbb{N}$, all $N \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa,\ell_-)$, all $(\mathsf{pk},\mathsf{sk}^{\varepsilon}_{\varepsilon}) \leftarrow \mathsf{Gen}(\mathsf{pp},N)$, all $I \subset [N]$, all $i,j \in I$, $i \geq j$, all $id \in \mathcal{ID}$, all $\mathsf{sk}^{id}_{i} \leftarrow \mathsf{Ext}(\mathsf{sk}^{\varepsilon}_{j}, id, i)$, all $\mathsf{sk}^{id}_{i+1} \leftarrow \mathsf{Update}(\mathsf{sk}^{id}_{i}, id, i)$, all $M \in \mathcal{M}$, all $C^{id}_{i} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, id, i)$, we have that $\mathsf{Dec}(\mathsf{sk}^{id}_{i}, C^{id}_{i}) = M$. Now, we define the security which we require for an fs-IBE scheme in Experiment 3. We call an adversary A valid if for the challenge messages $M_0, M_1 \in \mathcal{M}$ and $|M_0| = |M_1|$, it does not query Ext with id^* for time period i^* or with $id = \varepsilon$ for any time period $j \leq i^*$, nor does it query C^* to the Dec' oracle in case of CCA security. Note that the decryption oracle Dec' determines id and i from the given ciphertext C^{id}, then runs $\mathsf{sk}^{id}_{i} \leftarrow \mathsf{Ext}(\mathsf{sk}^{\varepsilon}_{\varepsilon}, id, i)$ and returns $\mathsf{Dec}(\mathsf{sk}^{id}_{i}, C^{id}_{i})$.

B Proof of Theorem 2

Proof. We prove this theorem by an efficient reduction, i.e., an adversary A against the DFPE-IND-CCA-security of the DFPE scheme can be transformed into a successful THIBE-IND-CCA-adversary D on THIBE as follows:

– D is started on pk, ℓ, and itself starts the DFPE-IND-CCA-security experiment with $\ell_- := |\mathcal{ID}|^{\ell}$, sets $\mathcal{T}_- := \mathcal{ID}^{\ell}, \mathcal{T}_+ := \mathcal{ID}, \mathcal{M} := \mathcal{M}_{\mathsf{THIBE}}$, and obtains $(M_0, M_1, t^*_+, t^*_-) \leftarrow A(\mathsf{pk})$.

– Ext-queries by A are answered as follows. On input $T \subset \mathcal{T}_-$ and $t_+ \in \mathcal{T}_+ \cup \{\varepsilon\}$, D uses the Trunc-mechanism to determine those (hierarchical) "identities" such that those are not a prefix of t^*_-. For those "identities", D queries its Ext-oracle to receive the part-secret keys. After all part-secret keys are queried, D return them to A. (Note that validity requires that if $t_- \notin T$, then $t_+ \notin \{t^*_+, \varepsilon\}$. If $t_- \in T$, then we can allow $t_+ \in \{t^*_+, \varepsilon\}$.)
– Dec′-queries are forwarded by D to its own Dec′-oracle where the answered are returned to A.
– D forwards (M_0, M_1, t^*_+, t^*_-) to its own challenger and receives a challenge ciphertext C^* in return which is forwarded to A.
– D forwards the A-guess b^* to its own challenger.

All values are consistently distributed. It follows that if A has a non-negligible advantage in the DFPE-IND-CCA-security experiment, then D has a non-negligible advantage in winning the THIBE-IND-CCA-security experiment. □