I Want to Forget: Fine-Grained Encryption with Full Forward Secrecy in the Distributed Setting

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

[1] DFINITY
david@dfinity.org

[2] AIT Austrian Institute of Technology, Vienna
{sebastian.ramacher|daniel.slamanig|christoph.striecks}@ait.ac.at

Abstract. Managing sensitive data in highly-distributed environments is gaining a lot of attention recently. Often, once data is presented to such environments, this data is persistent there. Being able to "forget" in such environments constitutes a very desired feature due to data security and privacy issues. In particular, applying the European General Data Protection Regulation (GDPR), the "Right to be Forgotten" essentially became a data owner right.

In this work, we seek for cryptographic solutions that offer the possibility to willfully lose access to data in distributed environments (which can be seen equivalent to removing that data). We argue that simple encryption mechanisms do not suffice to cover all desired requirements and provide a solution that offers several strong security and privacy features. In particular, our solution achieves forward secrecy for all participants in the system (i.e., even when user keys leak), ensures strong privacy against public observers of the system (i.e., key anonymity against tracking), and enables fine-grained access control. Having those features in parallel was unknown from the cryptographic literature.

We base our solution on a novel cryptographic primitive we dub *Identity-Based Puncturable Encryption* (IBPE) which significantly enhances previous ideas on Puncturable Encryption (PE) due to Green and Miers (IEEE S&P 2015) and Günther et al. (EUROCRYPT 2017). We argue that black-box constructions from Hierarchical Identity-Based Encryption (HIBE) do not seem to work, albeit we do know how to construct PE from HIBE. We further introduce an important feature being crucial in the setting of always-accessible and public data, namely that of *key-anonymity* for IBPE such that an IBPE ciphertext reveals nothing about the encryption key. We demonstrate the feasibility of our IBPE construction with a practical prototype implementation. Finally, we show that IBPE is a very versatile tool by using it to generically instantiate forward-secret IBE and forward-secret digital signatures, latter also being of importance in a distributed setting.

Keywords: puncturable encryption ⋄ forward secrecy ⋄ distributed setting

1 Introduction

Distributed[3] technologies allow to remove the need for (trusted) central authorities with the benefits of entirely removing single points of failure and undesired concentration of trust.

[‡] Work partly done while the author was with Graz University of Technology.

[‖] Work done while the author was with Graz University of Technology.

[3] We refer to the term distributed as defined by Baran [Bar62], i.e., there is a hierarchy from centralized to decentralized to distributed. Hence, distributed implies the other two.

Consequently, emerging technologies such as distributed storage gained a lot of attention in recent time. In such systems, data is typically accessed or held in a distributed way, e.g, dispersed over or replicated on multiple nodes which bears several threats to data security and privacy.

This distributed nature improves availability of data with several further benefits. However, for reliable erasure of data from such a system, one can quickly conclude that this becomes a non-trivial task. One might think of a node that goes offline immediately after receiving data and only goes online again after a certain erasure operation took place. Here, it is hard to enforce or guarantee the erasure (or, deletion) of data from such systems as data may reside on multiple different locations. Moreover, to allow data owners to control access to their data, data-access control mechanisms have to be deployed in such settings; which by itself is a non-trivial issue and often only realized in software products (that are prone to implementation errors).

Looking at new regulations such as the European General Data Protection Regulation (GDPR)[4], deletion of data becomes a data-owner right from a legal point of view: GDPR's "Right to be Forgotten" (Recitals 65, 66 and Article 17) essentially states that data controllers (e.g., organizations) have to erase personal data under certain conditions if the data owners request erasure of their data. Under the GDPR, organizations most probably would not store personal data in an uncontrolled (e.g., distributed) environment. However, more and more recent technologies (e.g., distributed application and decentralized storage) such as Filecoin[5], Sia [VC14] or Storj [WBB+16] are using such distributed technologies. Moreover, although there are academic and open-source distributed storage frameworks that put a focus on security (e.g., [BCQ+13, LHS15]), most prominent frameworks such as Ceph, Minio, or Swift[6] as well as the increasingly popular storage layer InterPlanetary File System (IPFS) [Ben18] do not.

Motivation. We believe that being able to "forget" data in highly distributed environments is crucial to data security. Furthermore, without appropriate access-control mechanisms, distributed data can even be publicly accessible and permanently analyzable which additionally threatens users' privacy. This is different to more ephemeral settings like the Internet where data is much more versatile and one has to invest significantly more effort to capture Internet traffic. Naturally, encryption mechanisms with build-in privacy features provide a suitable mitigation where only the data owner (relying on encryption keys) is able to access her own data. We see the following core features are essential to be considered in a our (distributed) setting:

Practical data-security mechanism. Protecting data in such settings is an essential goal and should be achieved in a practical way. Encryption techniques provide suitable mechanisms and we will consider different known solutions below before we introduce ours that significantly enhance the state of the art.

Practical key management. As encryption techniques shall be used, key management should be as simple and powerful as possible. Ideally, every participant holds a single compact key that does not need to change frequently, and, in particular, can be used for an unbounded number of encryptions. In addition, users should not be required to keep a (large) local state.

[4] https://gdpr.eu

[5] https://filecoin.io/

[6] https://ceph.com/, https://www.minio.io/, https://wiki.openstack.org/wiki/Swift

Full forward secrecy. Large-scale data breaches (of sensitive passwords protecting the unauthorized use of cryptographic keys) happen quite frequently.[7] In the past, this strongly motivated the use of mitigation methods such as full forward secrecy, which essentially ensures that leakage of a secret key at some point in time only affects encryptions from that time onwards while encryptions created prior to the key leakage are considered not in danger.[8]

Fine-grained access control. Secure access-control management is essential in almost every distributed system. Hence, we seek at providing reliable and strong access control to data and stress that key-leakage mitigations shall apply to all legitimate users that have access to data. Furthermore, we want that users fine-granularly can delegate access to other users (without giving up the forward-secrecy feature for the delegatee).

Encryption-key privacy. In many encryption mechanisms seeing a ciphertext reveals the public-key under which an encryption was generated. Those (public-key) meta-data can lead to severe privacy problems when data is stored in a distributed fashion and available for analysis to many different parties for a potentially very long time. Consequently, encryptions should hide information about the respective (public) keys. This feature is known as *key privacy* or *anonymity* [BBDP01] and we consider it to be an important property in distributed technologies.

We believe that all those requirements above are very important in the distributed setting and we will subsequently argue why prior encryption solutions do not suffice to realize all features in parallel.

The problem. A natural first step for data owners in the distributed regime is the use of cryptographic mechanisms and in particular encryption, e.g., secret-key encryption (SKE), to protect the data from being readable by arbitrary nodes or more generally unauthorized entities. This protects against eavesdroppers and introduces data-access control since only the legitimate data owner can access the plain data. Erasure of the data is then reduced to erasing the (secret) key associated to the encrypted data on the data-owner side regardless of where the encrypted data is distributed to. Now, if persistent encrypted data is not declared as "personal data" (which seems to be a reasonable assumption), then data owners have a sound way of "erasing" their data even in the light of GDPR. Interestingly, key-management for the data owner, i.e., sharing keys with other entities such that they can access the data, is easily realizable as well. It can simply be done by providing the corresponding encryption key for that encrypted data to that entity. Note, however, that this requires additional mechanisms such as public-key encryption (PKE) to efficiently distribute the keys.[9]

Fine-grained key-management allows the data owner to only have a short secret seed (e.g., a master key) from which many secret keys can be derived. Those derived keys can be used to access encrypted data (one key per ciphertext). Particularly, the data owner can now fine-granularly equip delegatees with unique derived keys such that only they can retrieve the intended data. Since derived keys are unique, high security guarantees against eavesdroppers can be claimed. Unfortunately, fine-grained key-management solely is not enough. Once the

[7] See for instance `https://en.wikipedia.org/wiki/List_of_data_breaches`.

[8] Such mitigation strategies are heavily used in interactive settings such as large-scale messaging services, e.g., WhatsApp or Signal, and were made mandatory in version 1.3 of the Transport Layer Security (TLS) protocol [Res18] for all public-key based key exchange mechanisms.

[9] This is inherent in many encryption scenarios as it is in our and, ideally, has only to be done once.

secret seed is leaked, all guarantees for the entire system are lost. *Key-leakage prevention mechanisms* allow to mitigate such scenario in the sense that keys can be updated to a newer period while old-period keys can be deleted. Furthermore, ciphertexts are also associated to periods which implies that once a (secret) key leaks, old-period ciphertexts are not in danger. However, updating keys requires interaction with all of the data holders.

We seek for a non-interactive solution that allows fine-grained key-management *and* key-leakage prevention mechanisms at the data owner side, the latter as well at the delegatee's side. Albeit fine-grained key-management can be very efficiently implemented with SKE techniques, the main concern is key leakage which becomes a much larger threat when dealing with persistent data. (One can think of several-years-old encrypted sensitive data which is still accessible today and within the next several years.) Even the recently established technique of Puncturable Pseudo-Random Functions (PPRFs) [SW14], a symmetric primitive, does not guarantee forward secrecy equipped with fined-grained key-management. The latter ported to the PKE setting yields forward-secure PKE [CHK03], however, with introducing such forward-secret PKE techniques, one still only obtains the same guarantees as with PPRFs. A next step to mitigate the issues would be to rely on the more general (fully) Puncturable Encryption paradigm [GM15, GHJL17a, DKL$^+$18a] in that it allows more fine-grained key-management in comparison with forward-secure PKE, but still cannot guarantee non-interactive forward-secrecy and fine-grained key-management at the same time. We refer to Table 1 for an overview.

Mechanism	Fine-grained encryptions	System-wide full forward secrecy
SKE	✓	✗
PPRF	✗	✓
FuPE	✓	✗
Ours (IBPE)	✓	✓

Table 1. Overview of encryption techniques. SKE, PPRF, and FuPE are acronyms for secret-key encryption, puncturable pseudo-random functions, and fully puncturable encryption, respectively.

We are not aware of any approach in the cryptographic literature that solves this issue. It seems that we are stuck with either having ease of key-management or having mitigation against key-leakage in such settings to convincingly ensure erasure of data.

Our contribution. We propose a strongly secure and versatile encryption solution for multi-user systems that mitigates secret-key leakage in form of non-interactive full forward secrecy for each user, guarantees key anonymity such that ciphertexts do not reveal the respective public key, and minimizes effort of key management, and supporting fine-grained access control for an unbounded number of users. As the foundation, we propose a novel cryptographic primitive dubbed Identity-Based Puncturable Encryption (IBPE) which extends recent works on Puncturable Encryption (PE). PE is a cryptographic paradigm that brought full forward secrecy to important practical applications such as asynchronous messaging and 0-RTT key exchange where the latter is an important building block for TLS, but not sufficient to achieve our goals. We carefully adapt the PE techniques envisioned by Green and Miers [GM15] and Günther, Hale, Jager, and Lauer [GHJL17a] to equip PE with *identity* and *key-anonymity* capabilities. Our overall goal is related to what can be achieved by the recent work by Derler et al. [DKL+18a]. However, their concept of so-called Fully PE (FuPE) does not achieve forward-secrecy for all system participants (in their language, positively punctured keys can no longer be negatively punctured) nor key-anonymity of ciphertexts and also seems much more complicated to handle. In distributed settings, especially the missing two features become highly relevant. Unfortunately – and this is in contrast to Derler et al. who can instantiate their FuPE scheme from any Hierarchical Identity-Based Encryption (HIBE) scheme – we are not able to derive our construction generically from any known cryptographic primitive. It seem that the desired features we are striving for need significantly more technical care in constructing. We summarize our contributions below.

Identity-Based Puncturable Encryption. We introduce Identity-Based Puncturable Encryption (IBPE). Loosely speaking, IBPE allows to puncture secret keys on tags (like within PE), i.e., a key punctured on a tag can no longer decrypt ciphertexts under this tag, but in addition a secret key can be customized to a given identity once (and then further punctured). Keys customized to an identity can only decrypt ciphertexts to this identity and whose tags are distinct from the ones the key was punctured on. We want to stress that although the primitive is called identity-based, we do not view it in the sense of (Hierarchical) Identity-Based Encryption, i.e., that there is a central authority generating keys within a system, but in the sense of public-key encryption where every user generates its own IBPE key-pair. Hence there is no trust in a central authority. Besides introducing the concept of IBPE, we rigorously model its security requirements in the vein of indistinguishability under chosen-plaintext (IBPE-IND-CPA) and chosen-ciphertext attacks (IBPE-IND-CCA). Further, we introduce key anonymity for IBPE (captured in our IND-KEY-ANON notion) which guarantees that ciphertexts do not leak anything about the public key which was used upon encryption. Looking ahead, this will be the essential starting point for key anonymity in our proposed practical solution. We note that key anonymity becomes a privacy issues when many instances of a cryptographic scheme are used.[10] We want to stress that our IBPE scheme also achieves full forward secrecy for secret keys customized to identities and, hence, for all secret keys in a system which clearly distinguishes our work to trivial extensions of encryption techniques to our setting. We then proceed to give an instantiation of a key-anonymous IBPE scheme which takes as a starting point the Hierarchical Identity-Based Encryption (HIBE) scheme due to Boneh-Boyen-Goh (BBG) [BBG05], but requires some major modifications and tweaks to provide all the features required by an IBPE scheme as well as new proof of security. A main benefit of the BBG HIBE is that the size of the ciphertexts is constant. Finally, we give a practical proof-of-concept instantiation of our IBPE scheme with measurements.

Cryptographic applications. We demonstrate that IBPE is a versatile cryptographic tool by showing that it can be used to generically instantiate other cryptographic primitives and immediately yields (new) constructions thereof, potentially under weaker assumptions and more efficient as previously known ones. In particular, we show how to generically construct forward-secure identity-based encryption (IBE) [YFDL04], thereby – to the best of our knowledge – obtaining the first fs-IBE scheme with compact ciphertexts, as well as forward-

[10] In the cryptographic literature, mostly single-instance schemes are considered for security analysis.

secure signatures [BM99, Kra00, IR01, AABN02]. Especially, the latter primitive turned out to be an interesting primitive in the context of distributed ledgers [DGKR18, GW19, DN19]. For instance, they are useful to obtain strong security guarantees for the consensus mechanism within proof-of-stake blockchain protocols as done in [DGKR18].

Application to fine-grained key-management. Integrating PE with identities and key-anonymity allows us to guarantee forward secrecy, key-anonymity, *and* flexibility of fined-grained access control at the same time, but *without* increasing the efforts of key management. This is particularly important in settings where multiple users are present. We believe that those techniques are particularly useful in application scenarios such as distributed ledgers that are thought to store, manage, and share encrypted data without a centralized authority in a fine-grained way, and additionally observers are able to analyze persistent and huge amounts of data. Common symmetric encryption techniques fall short since it is unclear how to combine forward-secrecy for all system users with the concept of identities in the symmetric setting. We discuss how to deploy our solution in real-world environments in Section 5.

1.1 Motivating Example: Receipts and Taxes

We want to sketch a concrete application scenario where our approach is well suited. In particular, we envision efficient handling of processes related to electronic receipts and tax offices via a distributed ledger. We note that this topic is in broad discussion[11,12] and first solutions in China are already in place.[13]

Tax offices often need to inspect the tax-related data of companies. For this purpose, they need to get access to all the relevant data of a certain period of time, e.g., one year. After the inspection, the tax offices need to delete the respective data according to EU privacy law, i.e., the GDPR. This data, for example, includes all receipts which were issued by a particular company for a particular time frame. Those receipts, once issued, must no longer be changed and should also have a time stamp to ensure consistent ordering. Furthermore, companies are legally obliged to keep their receipts for a certain amount of time, and, only after this period has passed, they can delete them. Using our approach, companies could encrypt all their receipts under their own public key and store them in the distributed ledger. This way, it is ensured that it is no longer possible to change the content as well as the order of the stored receipts. In case of an inspection, the tax office can use a customized key received from the data owner. This can be done in a way such that the tax office can only read the data required for the inspection but noting else. Once the inspection has been completed, the tax office can simply puncture its customized key (i.e., to forget the ability to decrypt the data from a certain time frame) with respect to, e.g., a certain time frame, and it will no longer be able to decrypt the respective ciphertexts. Still our approach allows the tax office to update the customized key in a way that it can decrypt data for future time frames in a non-interactive way and this can be done repeatedly, i.e., every year. Similarly, also companies can give up the decryption abilities for all the data they are no longer legally obliged to keep, which in turn greatly helps to reduce the risk associated to key leakage.

[11] https://www.pwc.co.uk/issues/futuretax/how-blockchain-technology-could-improve-tax-system.html

[12] https://www.ey.com/en_gl/trust/how-blockchain-could-transform-the-world-of-indirect-tax

[13] https://www.asiacryptotoday.com/official-announcement-blockchain-tax-receipts-have-arrived-in-china/

1.2 Related Work

In recent years, various advanced encryption primitives that are interesting in context of storing encrypted data (in a distributed setting) have been proposed.

The probably most well-known cryptographic primitive for fine-grained access to encrypted data is attribute-based encryption (ABE) [SW05, GPSW06, BSW07]. In ABE there is a central authority that issues secret keys that can be used to fine-granularly decrypt ciphertexts depending on attributes and policies. The major drawbacks of this flexible encryption primitive are the requirement for a trusted third party for setup and key distribution and the missing forward secrecy feature.

Updatable encryption (UE) [BLMR13, EPRS17, LT18] is a form of encryption that allows one to periodically rotate the encryption key and move already existing ciphertexts from the old to the new key. This mechanism could be used to switch to a unknown random key (which one then forgets) and update ciphertexts to this key. A related approach to implement the right to be forgotten was proposed by Pagnin et al. [PBP18], who suggest to keep the secret key identical but to update the ciphertexts to random plaintexts to essentially "destroy" the ciphertexts.

We note that in specific distributed settings such as distributed ledgers, which typically have immutability as an inherent property, such ideas combined with work due to Ateniese et al. in [AMVA17] and follow up work by Derler et al. [DSSS19] may be used. They propose to integrate rewriting capabilities into blockchains on a block and transaction level respectively, by replacing hash functions to chain blocks or aggregate transactions in a blockchain with variants of chameleon hash functions [KR00]. An alternative non-cryptographic mechanism based on the consensus layer has recently been proposed by Deuber et al. [DMT19]. But the problem in our context with all these mechanisms is that although ciphertexts can be updated, the original ciphertexts may still remain available in the distributed system. Consequently, once the data is distributed one looses control over the "old" ciphertexts and thus such primitives are not useful to us.

Proxy re-encryption (PRE) [BBS98, AFGH05] is an encryption primitive that allows a semi-trusted entity (the proxy) to convert between ciphertext under different keys when provided with a so called re-encryption key. Although forward-secrecy has recently been studied in context of PRE [DKL$^+$18b], the requirement for a always online semi-trusted third party makes this primitive unattractive within our setting. We note that although this can be countered by using threshold PRE [Nuñ18], one is still faced with the same issues as within UE discussed above.

Access control encryption (ACE) [DHO16, FGKO17, BMM17] is a cryptographic primitive that allows to control the information flow between several parties according to a given policy specifying which parties are, or are not, allowed to communicate. Similarly to PRE, it involves a central party (the sanitizer) that controls all the communication between parties and thus makes it not suitable for our distributed scenario.

Finally, we want to briefly talk about existing approaches to "forgetting" data stored in distributed systems. Vanish [GKLL09] is a system for creating data that automatically "self-destruct" after a period of time. Therefore data is symmetrically encrypted with random keys, the keys are secret shared and shares of the key are stored in a large, public distributed hash table (DHT). The basic idea is that DHTs delete data older than a certain age and after this happens to the key shares, the key is permanently lost, and the encrypted data is permanently unreadable. Unfortunately, it has been shown that Sybil attacks allow to efficiently recover keys for more than 99% of Vanish messages [WHH$^+$10]. Besides the existing attacks, self-destructing data in such a system is not fine-grained with respect to when exactly deleting which data. While there is follow up work [TLLP12] that gets rid of the problem within Vanish and provides more fine-grained policy based deletion, such systems require a set of key managers (holding the encryption keys) which is deployed as a centralized trusted service. While this seems reasonable for enterprise cloud solutions, this is in contrast with the trust assumptions in distributed systems such as distributed ledgers.

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. (We may omit to mention the κ-input explicitly and assume that all algorithms take κas 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 $q = q(k)$ if we mean that the value q depends polynomially on κ. Further, we write $v_{|i} := v_1 \ldots v_i$, for $i \in \mathbb{N}$, and we denote $v_{|0} = \varepsilon$ as the empty set, respectively.

Pairings. Let G_1, G_2, G_T be cyclic groups of order p. A *pairing* $e : G_1 \times G_2 \to G_T$ is a map that is *bilinear* (i.e., for all $g_1, g_2' \in G_1$ 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 $\mathsf{BG} = (p, G_1, G_2, G_T, e, g_1, g_2) \leftarrow \mathsf{BGen}(\kappa)$ for generators g_1 and g_2 of G_1 and G_2, respectively, and $\Theta(\kappa)$-bit prime p.

BDH and q-wBDHI assumptions. We recall the BDH [BF01] and q-wBDHI [BBG05] assumptions ported to Type-III groups [CM11]. We define the advantage of an adversary A as

$$\mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},A}(\kappa) := \left| \Pr\left[\begin{array}{l} r, s, t, u \leftarrow \mathbb{Z}_p, b \leftarrow \{0,1\}, \\ b^* \leftarrow A(\mathsf{BG}, g_1^r, g_1^s, g_2^s, \\ g_1^t, g_2^t, e(g_1, g_2)^{b \cdot rst + (1-b)u}) \end{array} : b^* = b \right] - 1/2 \right|.$$

and

$$\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},A}(\kappa) := \left| \Pr\left[\begin{array}{l} \alpha, r \leftarrow \mathbb{Z}_p, b \leftarrow \{0,1\}, \\ b \leftarrow A(\mathsf{BG}, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, \\ g_2^{\alpha}, g_1^r, g_2^r, e(g_1, g_2^r)^{\alpha^{(q+1)b} + (1-b)u}) \end{array} : b^* = b \right] - 1/2 \right|.$$

We say the BDH and q-SDH assumptions hold if $\mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},A}$ and $\mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},A}$ are negligible functions in the security parameter κ for all PPT adversaries A, respectively.

3 Identity-Based Puncturable Encryption

Puncturable encryption (PE) has been introduced by Green and Miers in [GM15] and subsequently used and refined in several works [CHN+16, CRRV17, GHJL17a, DJSS18, DKL+18a]. 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 a certain sense, PE can be viewed as forward-secure public-key encryption [CHK03] with a much more fined-grained method for restricting decryption capabilities. 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+18b], 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 (and, e.g., given to some delegatee) will loose the forward-secrecy property, a feature that we want to enable for all entities within our approach. This, in particular, allows forward secrecy for keys at a delegatee's end. To the best of our knowledge, no similar approach is known from the cryptographic literature.

To mitigate this problem and to make more powerful concepts of PE more comprehensible, we introduce in this section the new notion of Identity-Based PE (IBPE) which significantly enhances previous work on PE in the sense of the features discussed above. Moreover, we extend the IBPE definition by an anonymity notion in the vein of key anonymity for public-key encryption [BBDP01], which, informally spoken, means that a ciphertext hides the public key under which it was produced. Let us now define IBPE as well as the security notions. We stress that we explicitly model a Setup algorithm to generate public parameters pp, e.g., the description of a bilinear group. This is meaningful if many parties generate their public keys with respect to the same group parameters.

Definition 1 (IBPE). *An Identity-Based Puncturable Encryption (IBPE) scheme* IBPE *with message space* $\mathcal{M}$*, tag space* $\mathcal{T}$*, and identity space* $\mathcal{ID}$*, consists of the PPT algorithms* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Punc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(\kappa, \ell)$: *System setup, on input a security parameter* $\kappa \in \mathbb{N}$ *and maximum number of tags* $\ell \in \mathbb{N}$*, outputs public parameters* pp*. (We assume that* pp *implicitly determines* $\mathcal{M}$*,* $\mathcal{T}$*, and* $\mathcal{ID}$*.*[14]*)*

$\mathsf{Gen}(\mathsf{pp})$: *Key generation, on input public parameters* pp*, outputs public and secret keys* $(\mathsf{pk}, \mathsf{sk}_{\varepsilon}^{\varepsilon})$*. We assume that* $\mathsf{sk}_{\varepsilon}^{\varepsilon}$ *contains* pk *and that* pp *is implicit in* pk*.*

$\mathsf{Punc}(\mathsf{sk}_T^{id}, t)$: *Key puncturing, on input a secret key* sk_T^{id} *with* $T \subset \mathcal{T}$*,* $id \in \mathcal{ID}$*, and a tag* $t \in \mathcal{T}$*, outputs a key* $\mathsf{sk}_{T \cup \{t\}}^{id}$*.*

$\mathsf{Ext}(\mathsf{sk}_T^{\varepsilon}, id)$: *Key extraction, on input a secret key* sk_T *associated to tag set* $T \subset \mathcal{T}$ *and id* id*, outputs a key* sk_T^{id}*.*

$\mathsf{Enc}(\mathsf{pk}, M, t, id)$: *Encryption, on input a public key* pk*, a message* $M \in \mathcal{M}$*, a tag* $t \in \mathcal{T}$*, and an id* $id \in \mathcal{ID}$*, outputs a ciphertext* C_t^{id}*. (We note that* t *and* id *are publicly retrievable from the ciphertext* C_t^{id} *and the public parameters* pp*.)*

[14] We consider the empty identity ε to be part of the identity space $\mathcal{ID}$, i.e. $\varepsilon \in \mathcal{ID}$.

$\mathsf{Dec}(\mathsf{sk}_T^{id'}, C_t^{id})$: *On input a secret key* $\mathsf{sk}_T^{id'}$ *and a ciphertext* C_t^{id}*, outputs* $M \in \mathcal{M} \cup \{\bot\}$.

Correctness. We require that for all $\kappa, \ell \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa, \ell)$, all $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon) \leftarrow \mathsf{Gen}(\mathsf{pp})$, all $T \subset \mathcal{T}$, all $t \in \mathcal{T}$, all $id \in \mathcal{ID}$, all arbitrarily interleaved runs of $\mathsf{sk}_{T \cup \{t\}}^{id} \leftarrow \mathsf{Punc}(\mathsf{sk}_T^{id}, t)$ and $\mathsf{sk}_T^{id} \leftarrow \mathsf{Ext}(\mathsf{sk}_T^\varepsilon, id)$, all $M \in \mathcal{M}$, all $C_t^{id} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, t, id)$, we have that $\mathsf{Dec}(\mathsf{sk}_T^{id}, C_t^{id}) = M$ if and only if $t \notin T$. We want to stress that we allow the Ext algorithm to be called on an empty tag set $T = \emptyset$ and that this definition also allows the Punc-algorithm to be called on keys sk_T^{id} extracted for identity id.

IBPE-IND-CPA and IBPE-IND-CCA security notions. We define security notions in the vein of indistinguishability under chosen plaintext (IND-CPA) and ciphertext attacks (IND-CCA) for IBPE, dubbed IBPE-IND-CPA and IBPE-IND-CCA. In the IBPE-IND-CPA security experiment, public parameter pp as well as public and secret keys $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon)$ are honestly generated. During the experiments, A may adaptively query a $\mathsf{Cor}(\mathsf{sk}_\varepsilon^\varepsilon, \cdot, \cdot)$-oracle, while for the IBPE-IND-CCA notion, A may adaptively query a $\mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, \cdot)$-oracle additionally. The Cor- and Dec'-oracles are defined as follows:

$\mathsf{Cor}(\mathsf{sk}_\varepsilon^\varepsilon, T, id)$, on input secret key $\mathsf{sk}_\varepsilon^\varepsilon$, tag set $T \subset \mathcal{T}$, and identity $id \in \mathcal{ID}$, outputs $\mathsf{sk}_T^{id} \leftarrow \mathsf{Ext}(\mathsf{sk}_{T_\ell}^\varepsilon, id)$, for iteratively punctured secret key $\mathsf{sk}_{T_i}^\varepsilon \leftarrow \mathsf{Punc}(\mathsf{sk}_{T_{i-1}}^\varepsilon, t_{i-1})$, for all pairwise-different tags $(t_0, \ldots, t_{\ell-1}) \in (T)^\ell$ with $\ell := |T|$ and $i \in [\ell]$ in arbitrary order. (Note that the oracle allows the empty identity $id = \varepsilon$ and the empty tag set $T = \varepsilon$ as input.)

$\mathsf{Dec}'(\mathsf{sk}_\varepsilon^\varepsilon, C_t^{id})$, on input secret key $\mathsf{sk}_\varepsilon^\varepsilon$ and ciphertext C_t^{id}, computes $\mathsf{sk}_\varepsilon^{id} \leftarrow \mathsf{Ext}(\mathsf{sk}_\varepsilon^\varepsilon, id)$ and outputs $M \leftarrow \mathsf{Dec}(\mathsf{sk}_\varepsilon^{id}, C_t^{id})$. (Note that the oracle allows the ciphertext input associated to the empty identity $id = \varepsilon$ and the empty tag $t = \varepsilon$.)

The public key pk is given to A. A outputs messages (M_0, M_1), a target tag t^*, and a target identity id^*. The target challenge ciphertext $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, t^*, id^*)$, 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 for any valid A, the equation $b = b^*$ holds. More concretely, if any PPT A succeeds the previous experiments only with probability at most negligibly larger than 1/2, then we say an IBPE scheme IBPE is IBPE-IND-CPA and IBPE-IND-CCA secure, respectively. We say that A is valid if and only if A has not queried the Cor-oracle to obtain keys such that the challenge ciphertext can be trivially decrypted and A outputs only equal-length messages; for the IBPE-IND-CCA case, we additionally require that A did not query Dec-oracle with the challenge ciphertext. In Experiment 1, we formally state the IBPE-IND-CPA and IBPE-IND-CCA experiments.

Definition 2. *We define the advantage of an adversary A in the IBPE-IND-*T *experiment* $\mathsf{Exp}_{\mathsf{IBPE},A}^{\mathsf{ibpe\text{-}ind\text{-}T}}(\kappa, \ell)$ *as*

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

An IBPE scheme IBPE *is IBPE-IND-*T*-secure for* $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$*, if* $\mathbf{Adv}_{\mathsf{IBPE},A}^{\mathsf{ibpe\text{-}ind\text{-}T}}(\kappa, \ell)$ *is a negligible function in κ for all valid PPT adversaries A.*

IBPE-ANON-KEY anonymity notion. We present an anonymity notion denoted as IBPE-ANON-KEY which is concerned with key anonymity (also called key privacy). More

```
Experiment $\mathsf{Exp}^{\mathsf{ibpe\text{-}ind\text{-}T}}_{\mathsf{IBPE},A}(\kappa,\ell)$

  $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa,\ell), (\mathsf{pk},\mathsf{sk}^{\varepsilon}_{\varepsilon}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
  $(M_0, M_1, t^*, id^*, \mathsf{st}) \leftarrow A^{\mathsf{Cor}(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot,\cdot),\, \mathsf{Dec}'(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot)}(\mathsf{pk})$
  $b \leftarrow \{0,1\}$
  $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b, t^*, id^*)$
  $b^* \leftarrow A^{\mathsf{Cor}(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot,\cdot),\, \mathsf{Dec}'(\mathsf{sk}^{\varepsilon}_{\varepsilon},\cdot)}(\mathsf{st},\mathsf{pk},C^*)$
  if $A$ is valid and $b = b^*$ return 1, else return 0
```

Experiment 1: IBPE-IND-T-security experiment for an IBPE scheme IBPE: $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

precisely, it is inspired from the notion for public-key encryption due to Bellare et al. [BBDP01] where an adversary is allowed to choose a target tag and an identity, and is then presented with two public keys. It must, however, not be confused with anonymity in context of identity-based encryption where one wants to hide the identity [ABC+05], but is rather related to definitions for IBE where one wants to hide under which (master) public key a ciphertext has been created (cf. [PS08]). Such a notion has never been considered before in context of puncturable encryption[15] and in particular, we require that the adversary when given a ciphertext cannot tell which public key has been used to compute the ciphertext. In the IBPE-ANON-KEY security experiment, public parameter pp as well as public and secret keys $((\mathsf{pk}_0, \mathsf{sk}^{\varepsilon}_{0,\varepsilon}), (\mathsf{pk}_1, \mathsf{sk}^{\varepsilon}_{1,\varepsilon}))$ are honestly generated. The public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$ are given to A. A outputs a message M, a target tag t^*, and a target identity id^*. The target challenge ciphertext $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}_b, M, t^*, id^*)$, 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 for any A, the equation $b = b^*$ holds. More concretely, if any PPT A succeeds the previous experiments only with probability at most negligibly larger than 1/2, then we say an IBPE scheme IBPE provides IBPE-ANON-KEY anonymity. In Experiment 2, we formally state the IBPE-ANON-KEY experiment. Note that for this experiment we need to assume that there is a Setup algorithm which computes common parameters pp (if required by the underlying scheme), e.g., a description of a bilinear group which is very common in practice, and the public keys are set up with respect to these parameters as otherwise this notion might not be not meaningful, i.e,. there might be trivial distinguisher.

Definition 3. *We define the advantage of an adversary A in the IBPE-ANON-KEY experiment* $\mathsf{Exp}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE},A}(\kappa)$ *as*

$$\mathbf{Adv}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE},A}(\kappa,\ell) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE},A}(\kappa,\ell) = 1 \right] - \frac{1}{2} \right| .$$

An IBPE scheme IBPE provides IBPE-ANON-KEY anonymity, if $\mathbf{Adv}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE},A}(\kappa,\ell)$ is a negligible function in κ for all PPT adversaries A.

[15] In a concurrent and independent work Kerber et al. [KKKZ18] introduce such a notion for forward-secure public key encryption, a notion related to puncturable encryption.

```
Experiment $\mathsf{Exp}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE},A}(\kappa,\ell)$

    $\mathsf{pp} \leftarrow \mathsf{Setup}(\kappa,\ell)$
    $(\mathsf{pk}_0, \mathsf{sk}^{\varepsilon}_{0,\varepsilon}) \leftarrow \mathsf{Gen}(\mathsf{pp}), (\mathsf{pk}_1, \mathsf{sk}^{\varepsilon}_{1,\varepsilon}) \leftarrow \mathsf{Gen}(\mathsf{pp})$
    $(M, t^*, id^*, \mathsf{st}) \leftarrow A(\mathsf{pk}_0, \mathsf{pk}_1)$
    $b \leftarrow \{0,1\}$
    $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}_b, M, t^*, id^*)$
    $b^* \leftarrow A(\mathsf{st}, C^*)$
    if $b = b^*$ return 1, else return 0
```

Experiment 2: IBPE-ANON-KEY experiment for an IBPE scheme IBPE.

4 Constructing Identity-Based Puncturable Encryption

Subsequently, we present a construction of an IBPE scheme based on pairings. Unfortunately, we cannot directly instantiate our IBPE scheme directly from Hierarchical Identity-Based Encryption (HIBE) [GS02, BBG05] as done in prior work on Puncturable Encryption (PE) [GHJL17b] and Fully PE (FuPE) [DKL+18b]. The reason is that we want to allow puncturings even after secret keys were extracted for a specific identity such that those keys can be further restricted. FuPE generalizes PE in the sense that one can extract a secret key for a tag (identity), but further puncturings are not possible. IBPE allows for puncturings even after a secret key for some identity has been extracted. This allows for more fine-grained encryption use-cases with forward secrecy and cryptographic applications we have in mind.

Despite that we do not know how to construct IBPE from HIBE in a black-box way, our general construction paradigm borrows ideas from the Boneh-Boyen-Goh (BBG) HIBE construction [BBG05] with a setup algorithm to generate shared public parameters. We significantly change the functionality of the BBG construction and proof our construction under the same assumption used to prove the BBG HIBE. We choose to instantiate our approach using asymmetric Type-III bilinear groups as they represent the state-of-the-art regarding efficiency and similarity of the security levels of the base and target groups.

We construct our IBPE scheme IBPE in a two-way approach which is mainly to ease the understanding. First, we give a simplified version of IBPE we dub IBPE_0 and prove its IPBE-IND-CPA security and IPBE-KEY-ANON anonymity in Subsec. 4.1. This is showcase the main novelties and differences to the BBG approach. In Subsec. 4.2, we lift the simplified construction IBPE_0 to the full version IBPE and argue about IPBE-IND-CCA security while key anonymity is trivially carried over to the full version.

4.1 Simplified Version of our IBPE Scheme

We start with a simplified version of our IBPE scheme IBPE_0 which is presented in Scheme 1. To ease understanding, the scheme only supports a tag space $\mathcal{T} = [\ell]$ and we associate each tag to a unique group element in the public parameters. This is a fairly inefficient scheme; however, with the benefits of showcasing the main ideas more clearly. Via the compiler of Canetti-Halevi-Katz [CHK03]—as discussed in the next subsection—we straightforwardly are able to extend this to a more efficient version with an exponentially large tag space of $O(2^\ell)$ with $O(\ell)$ group elements in the public parameters while achieving the same security guarantees. The public-parameter size can even be reduced to constant-size by applying a hash function which is modeled as a random oracle (RO). Such conversion can also be found in prior work for achieving non-trivial forward secrecy in the standard and RO model, e.g., [CHK03, GHJL17a, DKL+18b, GW19, DN19].

Subsequently, we show that the simple scheme IBPE_0 satisfies both the IBPE-IND-CCA and IBPE-ANON-KEY security notions. Correctness is easy to verify and straightforward. We start with IBPE-IND-CPA in Theorem 1 and use the FO transform [FO99] to extend this to IBPE-IND-CCA security in the next subsection.

$\underline{\mathsf{Setup}(\kappa,\ell)}$: Generate a bilinear group $\mathsf{BG} := (p, e, G_1, G_2, G_T, g_1, g_2) \leftarrow \mathsf{BGen}(\kappa)$, set $\mathcal{T} := [\ell]$, $g, h, h_0, h_1, \ldots, h_\ell \leftarrow G_1$, and return $\mathsf{pp} := (\mathsf{BG}, H, g, h, h_0, h_1, \ldots, h_\ell)$, for hash function $H : \mathcal{ID} \mapsto \mathbb{Z}_p$ where $H(\varepsilon) := 0$.

$\underline{\mathsf{Gen}(\mathsf{pp})}$: Choose $\alpha, r \leftarrow \mathbb{Z}_p$ and return $(\mathsf{pk}, \mathsf{sk}^{\varepsilon}_{\varepsilon}) \leftarrow (g_2^{\alpha}, (h^{\alpha} \cdot h_0^r, g_2^r, h_1^r, \ldots, h_\ell^r, g^r))$.

$\underline{\mathsf{Punc}(\mathsf{sk}^{id}_T, t)}$: If $id = \varepsilon$: parse $\mathsf{sk}^{id}_T =: (a_0, a_1, h'_1, \ldots, h'_{\ell'}, g')$ and set $T' \leftarrow T \cup \{t\}$. Sample $r \leftarrow \mathbb{Z}_p$ and return

$$\left(a_0 \cdot h'_t \cdot \left(\prod_{i=0}^{|T'|} h_i\right)^r, a_1 \cdot g_2^r, (h'_i \cdot h_i^r)_{i\in[\ell'], i\neq t}, g' \cdot g^r\right).$$

If $id \neq \varepsilon$: parse $\mathsf{sk}^{id}_T =: (a_0, a_1, h'_1, \ldots, h'_{\ell'})$ and set $T' \leftarrow T \cup \{t\}$. Sample $r \leftarrow \mathbb{Z}_p$ and return

$$\left(a_0 \cdot h'_t \cdot \left(\prod_{i=0}^{|T'|} h_i\right)^r \cdot g^{H(id)\cdot r}, a_1 \cdot g_2^r, (h'_i \cdot h_i^r)_{i\in[\ell'], i\neq t}\right).$$

$\underline{\mathsf{Ext}(\mathsf{sk}^{\varepsilon}_T, id)}$: Parse $\mathsf{sk}^{\varepsilon}_T =: (a_0, a_1, h'_{t_1}, \ldots, h'_{t_{\ell'}}, g')$. Sample $r \leftarrow \mathbb{Z}_p$ and return $(a_0 \cdot g'^{H(id)} \cdot (\prod_{i=0}^{|T|} h_i)^r \cdot g^{H(id)\cdot r}, a_1 \cdot g_2^r, (h'_{t_i} \cdot h_{t_i}^r)_{i\in[\ell'], t_i\neq t})$.

$\underline{\mathsf{Enc}(\mathsf{pk}, M, t, id)}$: Sample $s \leftarrow \mathbb{Z}_p$, and return

$$(C_1, C_2, C_3) := \left(e(h, \mathsf{pk})^s \cdot M, g_2^s, \left(\prod_{i=0, i\neq t}^{\ell} h_i\right)^s \cdot g^{H(id)\cdot s}\right).$$

$\underline{\mathsf{Dec}(\mathsf{sk}^{id'}_T, C^{id}_t)}$: Run $\mathsf{sk}^{id'}_T \leftarrow \mathsf{Punc}(\mathsf{sk}^{id'}_T, t')$ for all $t' \in T \setminus \{t\}$. If $id' = \varepsilon$, run $\mathsf{sk}^{id'}_T \leftarrow \mathsf{Ext}(\mathsf{sk}^{\varepsilon}_T, id)$. Parse $\mathsf{sk}^{id'}_T$ as $(a_0, a_1, \ldots)$. Return $M' := C_1 \cdot e(C_3, a_1) \cdot e(a_0, C_2)^{-1}$.

Scheme 1: Simplified version of the IBPE-IND-CPA secure IBPE scheme IBPE_0 (which can be extended to IBPE-IND-CCA security for 2^ℓ-size tags with constant-size parameters pp in the RO model).

Theorem 1. *If the q-wBDHI assumption holds, then* IBPE_0 *defined in Scheme 1 is IBPE-IND-CPA-secure in the RO model. Concretely, for any PPT adversary A with at most $q_C = q_C(\kappa)$ corruption queries, there is a distinguisher D on q-wBDHI with $q = \ell + 1$, such that*

$$\mathbf{Adv}^{\mathsf{ibpe\text{-}ind\text{-}cpa}}_{\mathsf{IBPE}_0, A}(\kappa, \ell) \leq q_H \cdot \ell \cdot \mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen}, D}(\kappa),$$

for group generator BGen *defined as above and number of RO-queries $q_H = q_H(\kappa)$.*

Proof. We show the IBPE-IND-CPA security of IBPE_0 for any valid PPT adversary A in two games where:

Game 0. Game 0 is the IBPE-IND-CPA experiment as defined above.
Game 1. Game 1 is defined as Game 0 apart from the fact that the challenge ciphertext is independent of the challenge bit.

In Game 1, A has no advantage (i.e., success probability of $\Pr[S_0] = 1/2$) in the sense of IBPE-IBE-IND-CPA. We denote the event of the adversary winning Game i as S_i.

Description of the reduction. The proof follows the argumentation of the proof for the Boneh-Boyen-Goh (BBG) HIBE scheme under the q-wBDHI assumption [BBG05] transformed to the Type-III pairing setting. Essentially, due to the fact that the IBPE_0-construction is slightly different compared to their HIBE construction, we have to carefully prove our construction again. Nevertheless, many similarities can be found and, hence, we will focus on the differences in more detail (while of course recap the similarities). The main difference is that we have a special element, namely an identity-associated group element, that needs some extra care. However, as it turns out, this element can be handled as all the tag-basis group elements and is retrieved from the assumption as well. We are able to setup the public parameter with the correct distribution as well answer the Cor-queries. We follow the proof line of BBG in the following sense: BBG show that it is sufficient to proof a weaker security variant where the adversary has to commit to the target identity (i.e., target tag t^* and identity id^* in our case) before seeing the public parameters. This is often referred to as selective security and can be easily lifted to adaptive security (which we actually model within the IBPE-IND-CPA experiment where public parameters are retrieved first by the adversary) by relying on the random-oracle (RO) model. (We refer to BBG [BBG05] and later works [GW19, DN19] that all use similar techniques to prove adaptive security of their [signature] schemes.) For our purpose, it suffices to use the hash function H as the identity function for fixed-length identities in $\mathbb{Z}_p$. We want to show that each PPT adversary A with at most $q_C = q_C(\kappa)$ corruption queries on the IBPE-IND-CPA security of IBPE_0 yields a PPT distinguisher D for the q-wBDHI assumption, for $q = \ell + 1$.

The distinguisher D is given as input

$$(\mathsf{BG}, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{\ell+1}}, g_2^{\alpha}, g_1^{r}, g_2^{r}, T),$$

with g_1, g_2 given by the bilinear group parameters BG and T equals either $e(g_1, g_2^r)^{\alpha^{\ell+2}}$ or is a uniform element in G_T. The adversary outputs the target tag t^* and target identity id^*. The public parameters pp are sampled as in the BBG proof, i.e., for $y_i := g_1^{\alpha^i}$ and $\gamma \leftarrow \mathbb{Z}_p$, sets $h := y_{\ell+1} \cdot g_1^{\gamma}$ and $h_i := g_1^{\gamma_i}/y_{\ell-i+1}$ as well as $g := g_1^{\gamma_{\ell+1}}/y_1$, for all $\gamma_i \leftarrow \mathbb{Z}_p$, $i \in [\ell+1]$. The public parameters

$$\mathsf{pp} := (\mathsf{BG}, g, h, h_0, h_1, \ldots, h_\ell)$$

and the $\mathsf{pk} := g_2^{\alpha} \in G_2$ are sent to A. Hence, all parameters and the public key are distributed correctly. Note that D does not know the secret key $\mathsf{sk}_{\varepsilon}^{\varepsilon}$ corresponding to pk; implicitly the secret key is set to $\mathsf{sk}_{\varepsilon}^{\varepsilon} = (h^{\alpha} \cdot h_0^r, g_2^r, h_1^r, \ldots, h_\ell, g^r) = (g_1^{\alpha(\alpha^{\ell+1}+\gamma)} \cdot h_0^r, \ldots)$ for some $r \leftarrow \mathbb{Z}_p$ and $h_0 := g_1^{\delta} \cdot \prod_{i=1, i\neq t^*}^{\ell} g_1^{\alpha^{\ell-i+1}\cdot i} \cdot g_1^{\alpha^{\ell+1}\cdot id^*}$, for some $\delta \leftarrow \mathbb{Z}_p$. (Hence, we embed the term id^* and all tags except for t^* in the h_0 group element, which is essentially used to cancel out when trying to compute key material for the target identity id^* and target tag t^* such that a decryption for a target ciphertext with id^* and t^* would work. See that this slightly different to the BBG proof due to the fact that we allow decryption if and only if the secret key was *not* punctured on t^*.)

Corruption-oracle queries (up to q_C) in the reduction are answered as follows which is the same as in the BBG proof for "identity vector" $((t_i)_{i\in[n],\exists i:t_i=t^*}, id)$. As long as t^* is part of the vector, we can even allow $id = id^*$. If t^* is not part of the vector and $id = id^*$ is queried, there exist at least one element which involves $g_2^{\alpha^{\ell+2}}$ that cannot be computed by D. For all other queries, such term cancels out due to the partitioning argument (the same way as in the BBG proof).

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

$$C^* := \left(M_b \cdot T \cdot e(g_1^\alpha, g_2^r), g_2^r, g_1^{r\cdot\delta + \sum_{i=1,i\neq t^*}^{\ell+1} i\cdot\gamma_i\cdot id^*} \right)$$

to A.

Analysis of the reduction. The public parameters pp and the public key pk as well as all secret keys (that are returned after querying the Cor-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. The distinghuisher further has to guess the tag (from a polynomially large tag space) which induced a multiplicative factor of $|T| = \ell$.

If relying on the RO model (to achieve adaptive security), a multiplicative factor of q_H (which is the number of RO queries) is introduced; see also the discussions in [BBG05, GW19, DN19]. Consequently, we have that $|\Pr[S_0] - \Pr[S_1]| \leq q_H \cdot \ell \cdot \mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(\kappa)$. By taking $\mathbf{Adv}^{\mathsf{ibpe\text{-}ind\text{-}cpa}}_{\mathsf{IBPE}_0,A}(\kappa,\ell) = |\Pr[S_1] - \frac{1}{2}|$, we obtain that

$$\mathbf{Adv}^{\mathsf{ibpe\text{-}ind\text{-}cpa}}_{\mathsf{IBPE}_0,A}(\kappa,\ell) \leq q_H \cdot \ell \cdot \mathbf{Adv}^{\mathsf{q\text{-}wBDHI}}_{\mathsf{BGen},D}(\kappa),$$

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

Theorem 2. *If the BDH assumption holds, then* IBPE_0 *defined in Scheme 1 is IBPE-ANON-KEY-secure. Concretely, for any PPT adversary A, there is a distinguisher D on BDH such that*

$$\mathbf{Adv}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE}_0,A}(\kappa,\ell) \leq \mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},B}(\kappa),$$

for group generator BGen *defined as above.*

Proof. We prove the theorem using a sequence of games. We denote the event of the adversary winning Game i as S_i.

Game 0. The original IBPE-KEY-ANON game.

Game 1. As in Game 0, but in the Setup instead of sampling the elements $g, h, h_0, h_1, \ldots, h_\ell$ uniformly at random from G_1, we choose $\gamma_0, \ldots, \gamma_{\ell+1} \leftarrow \mathbb{Z}_p$ and set $h_i \leftarrow g_1^{\gamma_i}$ for $0 \leq i \leq \ell$ and $g \leftarrow g_1^{\gamma_{\ell+1}}$. Note that the parameters are distributed exactly as in the original game and thus $\Pr[S_1] = \Pr[S_0]$.

Game 2. As in the original game, but in the computation of the challenge ciphertext (C_1, C_2, C_3), we sample $\mathbf{R} \leftarrow G_T$ and compute the component C_1 as $C_1 \leftarrow \mathbf{R} \cdot M$. Observe that in Game 2 the challenge ciphertext information-theoretically hides the used public key pk_b and so we have $\Pr[S_2] = \frac{1}{2}$.

Claim 1. We claim that $|\Pr[S_2] - \Pr[S_1]| \le \mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},B}(\kappa)$. We prove this claim below.

Description of the reduction. We construct a BDH adversary B. It runs the adversary on the security parameter κ. It obtains an instance $(\mathsf{BG}, g_1^r, g_1^s, g_2^s, g_1^t, g_2^t, e(g_1,g_2)^z)$ of the BDH problem, and uses the random self-reducibility of the DDH to obtain a second instance of the problem with shared values g_1^t and g_2^t. Therefore, set the first instance to

$$(g_1^{r_0}, g_1^{s_0}, g_2^{s_0}, g_1^{t_0}, g_2^{t_0}, e(g_1,g_2)^{z_0}) \leftarrow (g_1^r, g_1^s, g_2^s, g_1^t, g_2^t, e(g_1,g_2)^z).$$

Then, choose $u, v, w \leftarrow \mathbb{Z}_p$ and set the second instance to

$$\begin{aligned}(g_1^{r_1}, g_1^{s_1}, g_2^{s_1}, g_1^{t_1}, g_2^{t_1}, e(g_1,g_2)^{z_1}) \leftarrow (&g_1^r \cdot g_1^u, (g_1^s)^w \cdot g_1^v, (g_2^s)^w \cdot g_2^v,\\ &g_1^t, g_2^t, (e(g_1,g_2)^z)^w \cdot e(g_1^r, g_2^t)^v \cdot e(g_1^s, g_2^t)^{uw} \cdot e(g_1, g_2^t)^{uv}).\end{aligned}$$

Note that we now have two independent instances (sharing the same value t). Sample $b \leftarrow \{0,1\}$ and run the Setup, but set $h \leftarrow g_1^t$. Then B runs the adversary on pk_0 and pk_1. After receiving the challenge message M, the challenge tag t^*, and the challenge id^*, B sets the challenge ciphertext as

$$C^* \leftarrow \left(e(g_1,g_2)^{z_b} \cdot M, g_2^{s_b}, (g_1^{s_b})^{\sum_{i=0, i\neq t^*}^{\ell} \gamma_i + \gamma_{\ell+1} H(id^*)}\right),$$

starts the adversary on the challenge ciphertext and receives and outputs a bit b^*.

Analysis of the reduction. We observe that independent of the validity of the BDH instance the public keys are distributed as required. In case the BDH instance and the ciphertext has the form $(e(g_1,g_2)^{z_b} \cdot M, g_2^{s_b}, (\prod_{i=1, i\neq t^*}^{\ell} h_i \cdot g^{H(id^*)})^{s_b})$ and since $z_b \in \mathbb{Z}_p$ is uniformly random, $e(g_1,g_2)^{z_b}$ is a uniformly random element in G_T (we are simulating Game 2). If the BDH instance is valid, then the ciphertext has the form $(e(g_1,g_2)^{r_b s_b t_b} \cdot M, g_2^{s_b}, (\prod_{i=1, i\neq t^*}^{\ell} h_i \cdot g^{H(id^*)})^{s_b})$ and is distributed as a original ciphertext (we are simulating Game 1). Consequently, we have that $|\Pr[S_1] - \Pr[S_2]| \le \mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},B}(\kappa)$.

By taking $\mathbf{Adv}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE}_0,A}(\kappa,\ell) = |\Pr[S_0] - \frac{1}{2}|$ and that $\Pr[S_0] = \Pr[S_1]$, we obtain that $\mathbf{Adv}^{\mathsf{ibpe\text{-}anon\text{-}key}}_{\mathsf{IBPE}_0,A}(\kappa,\ell) \le \mathbf{Adv}^{\mathsf{BDH}}_{\mathsf{BGen},B}(\kappa)$, which concludes the proof.

4.2 Full Version of our IBPE Scheme

We lift our simplified version of our IBPE scheme IBPE_0 to a full version IBPE by applying the compiler due to Canetti, Halevi, and Katz (CHK) [CHK03]. Albeit this is a well-established method in the forward-secrecy domain, e.g., [BBG05, GHJL17a], we sketch the construction here for completeness. Consider the tags of our IBPE scheme associated to leafs of a binary tree. In particular, level i in the tree corresponds to the i-th bit of the tag. For each level, we associate two special group elements in the public parameters, one associated to the bit 1 and one for bit 0. Furthermore, we have one specific group element in the public parameter which corresponds to the identities in the IBPE scheme. All in all, this yields $O(\ell)$ group elements where 2^ℓ is the maximum number of tags and ℓ the bit-length of the tags. Secret keys now contain $O((\log \ell)^2)$ group elements such that each (inner) node at level i is now associated to a substring $t_{|j}, j < \ell$ of the tags and the potential to puncture subsequent tags t' that $t_{|j}$ is prefix of t'. Puncturing and key extraction are only changed in the way, that they treat all parts of the secret key associated to a specific node in the tree. (Note here that each node in the binary tree has a unique basis representation which we also exploit in the security proof of IBPE_0.) Encryption is straight-forward as well as decryption (while the secret is used that is associated to the smallest tag such that decryption is possible). We want to stress that the authors of [GHJL17a] in the full version [GHJL17b, Sec. 2.3] have a concise illustration how the tags are associated to the tree while explaining the details in the context of HIBEs and forward secrecy. The security arguments do not change and can be analogously shown compared to the simplified version. Again, this is due to the fact that each node can be represented by a unique basis. Hence, we derive at an IBPE-IND-CPA and IBPE-ANON-KEY secure IBPE scheme IBPE via the CHK compiler.

CCA Security. We now discuss how to obtain IBPE-IND-CCA security for our full version construction IBPE 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 bitstring (this requires to injectively encode (m, r) into the message space of the IBPE scheme). The IBPE-encryption is derandomized and uses as random coins $H(r)$ where H is a hash function modeled as a random oracle to obtain the ciphertext C. The decryption algorithm applies the original decryption algorithm from the IBPE-IND-CPA-secure IBPE scheme to receive (m', r'). Then, it re-encrypts (m', r') using random coins $H(r, m')$ to obtain the ciphertext C'. If it holds that $C = C'$, it outputs m' and otherwise it outputs $\perp$.

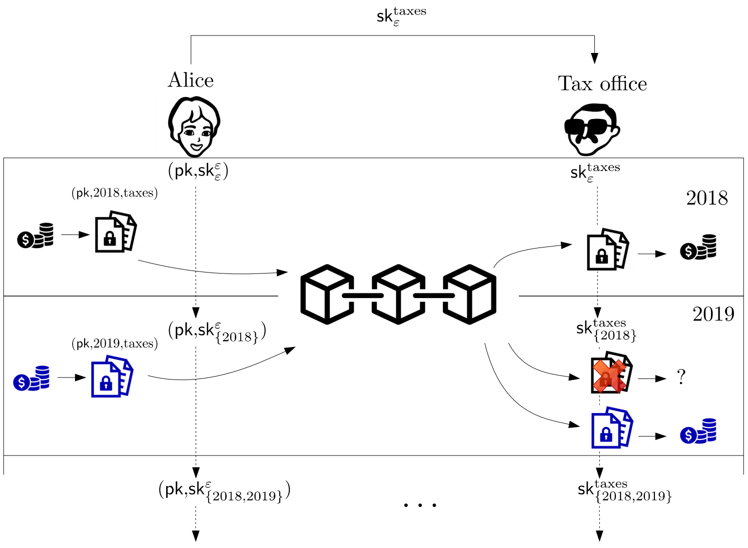
5 Application to Fine-Grained Key-Management

For the sake of illustration (see Figure 1), we come back to the example from the introduction (Section 1.1), and therefore let us assume that we have an individual, say Alice, who maintains her data in a distributed environment (e.g., a distributed ledger) and has an IBPE public-secret key pair. For illustration purposes, we assume that the application started in year 2018 and we currently are in the year 2019. We stress that the described scenario is still a simplification and intended mainly to transport the idea. Consequently, Alice starts with a key-pair $(\mathsf{pk}, \mathsf{sk}_\varepsilon^\varepsilon)$. Recall that Alice can create updated versions of her secret key which are no longer useful to decrypt ciphertexts associated to tags T[16] , and furthermore can be customized to identities id such that only specific ciphertexts can be decrypted.

To realize access to encrypted data, Alice now computes a derived secret key for the tax office and, therefore, Alice hands over this secret key which has been customized with respect to identity id = taxes to the tax office *once*. Thus, this key can at most be used to decrypt ciphertexts associated to id = taxes, but no other ciphertext (regardless of the tags). Consequently, using the secret key $\mathsf{sk}_\varepsilon^{\text{taxes}}$, the tax office can decrypt *all* ciphertext computed with respect to id = taxes.

On the other hand, recall that a secret key which has been punctured with respect to tag t is no longer useful to decrypt ciphertexts associated to t but can decrypt all other ciphertexts. In our example, Alice and the tax office puncture their secret key in the year 2019 (after everything is handled) with respect to tag $t = 2018$ and, consequently, both parties can no longer decrypt ciphertext from the year 2018 anymore. Note that this puncturing is possible for the tax office this since secret keys for id (held on the delegatee's side) can be further

[16] We note that well-known hybrid-encryption approach [CS03] can be obviously used in a straight-forward way to achieve file encryption, which we do not discuss here.



$\mathsf{sk}_\varepsilon^{\text{taxes}}$
Alice
Tax office
$(\mathsf{pk},\mathsf{sk}_\varepsilon^\varepsilon)$
$\mathsf{sk}_\varepsilon^{\text{taxes}}$
(pk,2018,taxes)
2018
$(\mathsf{pk},\mathsf{sk}^\varepsilon_{\{2018\}})$
$\mathsf{sk}^{\text{taxes}}_{\{2018\}}$
2019
(pk,2019,taxes)
?
$(\mathsf{pk},\mathsf{sk}^\varepsilon_{\{2018,2019\}})$
. . .
$\mathsf{sk}^{\text{taxes}}_{\{2018,2019\}}$

Fig. 1. Showcase: Alice makes her tax records available to the tax office.[†]
[†]Icons by Adrien Coquet and Antonius, from thenounproject.com.

punctured. Consequently, when it is required to give up the ability of being able to decrypt some specific ciphertext, i.e., all ciphertexts associated to the year 2018, the tax office is able to loose access to the data; while Alice may or may not puncture her secret key depending on the application.

We note that the identity and even the tags could be derived by evaluating a Pseudo-Random Function (PRF), e.g, an HMAC instantiated with SHA-3, with s_u as secret PRF key and some identifier string for the data (e.g., "taxes") only known to the data controller. This way the resulting identity *id* looks random to any passive observer of the environment and, looking ahead, allows the data controller to delegate decryption rights to other participants without touching ciphertexts stored in the distributed environment (which can be of benefit when potentially large environments are used, e.g., distributed ledgers).

Note that this easily scales to multiple instances. In such case, our system has the benefit of public-key anonymity such that no ciphertext leaks the associated public key of the data controller.

On the design of the tag space. While the use of the identities is clear, we have two ways of interpreting the tags. For the most generic view, we can consider the tag space exponentially large (for example we could simply use the hash of arbitrary large strings as tags) and use a *single* tag for every message to encrypt to obtain a very fine-grained control over puncturing, i.e., the keys can be punctured on every ciphertext. Alternatively, we can also consider tags from a polynomially bounded space and view them as an ordered sequence (e.g., as time periods as done in the previous section). In this approach one can, for instance, reuse a tag over many encryptions and thus *group* ciphertexts (e.g., within one time period). Secret keys can then be punctured in a single operation for all the ciphertexts of the respective group. While latter approach may require to rely on loosely synchronized clocks for the users, we believe this to be very reasonable assumption especially in the context of distributed applications.

6 Applications to Cryptographic Primitives

In this section, we demonstrate that IBPE represents an interesting cryptographic building block which allows to generically instantiating forward-secure identity-based encryption and forward-secure signatures.

6.1 Forward-Secure Identity-Based Encryption

An identity-based encryption (IBE) scheme [BF01] is a public-key encryption scheme in which any string (i.e., identity) can serve as a public key and anyone can encrypt a message for a party with some identity id only using the system wide master public key and the string id. Forward-security clearly is an interesting feature also in context of IBE. Interestingly, although there are some works on forward-secure IBE [CRF⁺11, SPB13, LL17], they all consider a very weak model in which the master secret key stays constant and thus the private key generator (PKG) is able to generate user keys for arbitrary time periods and thus essentially and inherently invalidating an important aspect of forward-security. We are only aware of a dedicated construction of a forward secure hierarchical IBE (HIBE) by Yao et al. [YFDL04], which also yields a forward-secure IBE as a special case. This works also considers forward-secrecy for the master secret key. As we will show in this section, IBPE generically yields forward-secure IBE and thus offering new instantiations thereof. And in particular, to the best of our knowledge this leads to the first fs-IBE scheme with compact ciphertexts.

Definition 4 (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* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Ext}, \mathsf{Update}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(1^\kappa, \ell)$: *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 $\mathsf{Dec}(\mathsf{sk}^{id}_i, C^{id}_i)$.

```
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 $A$ is valid and $b = b^*$ return 1, else return 0
```

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

fs-IBE Construction Having an IBPE scheme allows to construct an fs-IBE scheme by mapping time intervals to tags. The only syntactical difference is that the Punc-algorithm of IBPE is mapped to the Update-algorithm 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 IBPE 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 and thus we omit the proof.

Corollary 1. *If the* IBPE *scheme provides IBPE-IND-*T*-security, then the resulting* fs-IBE *scheme provides fs-IBE-IND-*T*-security.*

6.2 Forward-Secure Signatures

Forward-secure signatures [BM99, Kra00, IR01, AABN02] are a fascinating cryptographic primitive that has recently found interest in the context of distributed ledgers [DGKR18, GW19, DN19].

Having an IBPE scheme and, in particular, a fs-IBE scheme, we can generically construct a forward-secure signature scheme. The idea is simply to adopt the Naor-transform [BF01], which converts any 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 the ciphertext decrypts to the original plaintext.

The basic idea of this transform applied to the forward-secure 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 $\mathsf{sk}^{\varepsilon}_{i+1} \leftarrow \mathsf{Update}(\mathsf{sk}^{\varepsilon}_{i}, id, i)$. Now, within every time interval i one uses the current signing key with the above Naor-transform. It is straightforward to show the following:

Corollary 2. *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 instantiation of an IBPE from Section 4.1 in the above compiler, this yields forward-secure signatures with the same efficiency as in recent work [DN19, GW19].

7 Implementation and Evaluation

In this section, we report on a practical implementation of our IBPE scheme as presented in Section 4. This implementation of the IBPE-IND-CCA- and IBPE-ANON-KEY-secure IBPE directly yields an efficient implementation of the outlined confidentiality solution with very strong security guarantees. But more importantly, the evaluations results below show that the essential goals discussed in Section 5 can be achieved efficiently with our proposal.

Regarding a concrete choice of bilinear groups, we choose the popular BN curve family [BN05] and in particular the `BN254` curve. Respecting recent assessments by Menezes et al. [MSS16] as well as Barbulescu and Duquesne [BD18], this choice yields a security level of around 100 bit.

7.1 Evaluation Results

The IBPE scheme is implemented in Python 3.7 based on the current development version of the Charm[17] framework [AGM+13] using the `BN254` curve with the PBC pairing library version 0.5.10.[18] The measurements were performed on a laptop with an Intel Core i5-4300U @ 1.9 GHz with 8 GB 1600Hz DDR3 running Ubuntu 19.04. In Table 2, we present the average runtimes over 100 runs each. We measured three different sizes of the tag space (e.g., time intervals), i.e., 2^{48}, 2^{64}, and 2^{80} for a random message, respectively. When using point compression, the base-group elements (G_1) are of size at most 255 bits, the elements of the group G_2 are of size at most 510 bits, and the target-group elements have 3048 bits. Since we are in the random oracle model, the base group elements stored in the public parameters of our IBPE schemes can be derived using the random oracle. Thus the size of the public parameters are independent of the chosen tag space. Each public key pk consists of one G_2 element (resp. 64 bytes), and each master secret key $\mathsf{sk}_{\varepsilon}^{\varepsilon}$ has 50, 66, or 82 base-group element and one G_2 element (1664, 2176, or 2688 bytes, respectively). Each punctured key or extracted key sk_{T}^{id}, respectively, has one G_2 element (128 bytes) and at most 49, 65, or 81 base group elements (1504, 2016, or 2528 bytes, respectively) per node, leading to keys of at most 78,336, 137,216, or 212,480 bytes respectively. The ciphertext C consists of one element in each group, which yields 96 bytes in total for the ciphertext size.

From Table 2, one can see that the algorithms Setup, Gen, Enc, Dec, and Ext are very efficient for each tag space, i.e. always close to 100 ms. The benchmarks of the Dec algorithm assumes that no additional key extraction, i.e., an additional call to Ext which we benchmark separately, is necessary. Thus, the runtime of the decryption is independent of the size of the parameter space and the observed variation can be attributed to noise. The Punc algorithm need several seconds for all levels.[19] However, in our case we argue that puncturing is an offline operation, which, for example, only needs to happen upon puncturing a group of ciphertexts, and thus our performance results are perfectly acceptable. Indeed, the algorithms Enc, Dec, and Ext are used much more frequently in our setting.

[17] `https://github.com/JHUISI/charm`, commit `3eb33d69`

[18] `https://crypto.stanford.edu/pbc/`

[19] It is a central open issue in the context of puncturable-encryption techniques to make these algorithms more efficient, which already has received a considerable amount of attention recently [GM15, GHJL17a, DJSS18].

IBPE algorithm	2^{48}	2^{64}	2^{80}
Setup and Gen	68	89	108
Enc	98	120	137
Dec	57	59	57
Ext	63	84	103
Punc	3,027	5,402	8,243

Table 2. Performance evaluation: runtimes in integers and in [ms] for respective numbers of tags, e.g., intervals, 2^{48}, 2^{64}, and 2^{80}

8 Conclusions

We seek for a non-interactive solution that allow fine-grained key-management and key-leakage prevention mechanisms at the data owner as well as the delegatee side in a persistent-data setting. We show that simple encryption techniques do not suffice to meet all desired requirements. On the foundation, we propose Identity-Based Puncturable Encryption (IBPE) as a novel cryptographic primitive to underpin the important security features of system-wide full forward secrecy and fine-grained key-management with key-anonymity in the distributed settings. Thereby, we equip the well-known puncturable-encryption paradigm to guarantee *identity-based* and *key-anonymity* capabilities which results in our proposed solution with fine-grained key-management and full forward secrecy for all system participants. In particular, the latter was not achieved before from the cryptographic literature. We show how to construct IBPE from pairings and present concrete implementation results in the Type-III setting. Our current definition of the key-anonymity notion (IBPE-KEY-ANON anonymity) is in the CPA sense (i.e., without adversarially chosen ciphertexts). Even though we currently do not see concrete attack vectors imposed by not defining and achieving key-anonymity in the CCA setting, we believe that constructing schemes that provide key-private in a CCA sense poses an interesting open question for future work. Another interesting open question are instantiations under other assumptions and in particular lattice-based schemes.

Acknowledgements

This work was supported by the EUs Horizon 2020 ECSEL Joint Undertaking project SECREDAS under grant agreement n°783119 and by the Austrian Science Fund (FWF) and netidee SCIENCE project PROFET (grant agreement P31621-N38).

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 2002*, 2002. 1, 6.2

[ABC$^+$05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *CRYPTO 2005*, 2005. 3

[AFGH05] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, 2005. 1.2

[AGM$^+$13] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *J. Cryptographic Engineering*, 3(2):111–128, 2013. 7.1

[AMVA17] Giuseppe Ateniese, Bernardo Magri, Daniele Venturi, and Ewerton R. Andrade. Redactable blockchain - or - rewriting history in bitcoin and friends. In *EuroS&P*, 2017. 1.2

[Bar62] Paul Baran. On distributed communication networks. 1962. 3

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, 2001. 1, 3, 3

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, 2005. 1, 2, 4, 4.1, 4.2

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, 1998. 1.2

[BCQ$^+$13] Alysson Neves Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. *TOS*, 9(4):12:1–12:33, 2013. 1

[BD18] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *J. Cryptology*, 2018. 7

[Ben18] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. Whitepaper, `https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf`, 2018. 1

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, 2001. 2, 6.1, 6.2

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO 2013*, 2013. 1.2

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *CRYPTO*, 1999. 1, 6.2

[BMM17] Christian Badertscher, Christian Matt, and Ueli Maurer. Strengthening access control encryption. In *ASIACRYPT 2017*, 2017. 1.2

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography 2005*, 2005. 7

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S&P*, 2007. 1.2

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT 2003*, 2003. 1, 3, 4.1, 4.2

[CHN$^+$16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In *STOC 2016*, 2016. 3

[CM11] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011. 2

[CRF$^+$11] Dario Catalano, Mario Di Raimondo, Dario Fiore, Rosario Gennaro, and Orazio Puglisi. Fully non-interactive onion routing with forward-secrecy. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, pages 255–273, 2011. 6.1

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-ciphertext secure fully homomorphic encryption. In *Public Key Cryptography 2017*, 2017. 3

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003. 16

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT 2018*, 2018. 1, 6.2

[DHO16] Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In *TCC B 2016*, 2016. 1.2

[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 2018*, 2018. 3, 19

[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. *IACR Cryptology ePrint Archive*, 2018. 1, 1, 3

[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. In *Public Key Cryptography 2018*, 2018. 1.2, 3, 4, 4.1

[DMT19] Dominic Deuber, Bernardo Magri, and Sri Aravinda Krishnan Thyagarajan. Redactable blockchain in the permissionless setting. *CoRR*, abs/1901.03206, 2019. 1.2

[DN19] Manu Drijvers and Gregory Neven. Forward-secure multi-signatures. *IACR Cryptology ePrint Archive*, 2019. 1, 4.1, 4.1, 6.2, 6.2

[DSSS19] David Derler, Kai Samelin, Daniel Slamanig, and Christoph Striecks. Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based. In *NDSS*, 2019. 1.2

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In *CRYPTO 2017*, 2017. 1.2

[FGKO17] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access control encryption for equality, comparison, and more. In *Public Key Cryptography 2017*, 2017. 1.2

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*, 1999. 4.1, 4.2

[GHJL17a] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. In *EUROCRYPT 2017*, 2017. 1, 1, 3, 4.1, 4.2, 19

[GHJL17b] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. Cryptology ePrint Archive, Report 2017/223, 2017. 4, 4.2

[GKLL09] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security Symposium*, 2009. 1.2

[GM15] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *IEEE S&P*, 2015. 1, 1, 3, 19

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006. 1.2

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT 2002*, 2002. 4

[GW19] Sergey Gorbunov and Hoeteck Wee. Digital signatures for consensus. Cryptology ePrint Archive, Report 2019/269, 2019. 1, 4.1, 4.1, 6.2, 6.2

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO 2001*, 2001. 1, 6.2

[KKKZ18] Thomas Kerber, Markulf Kohlweiss, Aggelos Kiayias, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. *IACR Cryptology ePrint Archive*, 2018. 15

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*, 2000. 1.2

[Kra00] Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *ACM CCS*. ACM, 2000. 1, 6.2

[LHS15] Thomas Lorünser, Andreas Happe, and Daniel Slamanig. ARCHISTAR: towards secure and robust cloud based data sharing. In *CloudCom*, 2015. 1

[LL17] Yang Lu and Jiguo Li. Forward-secure identity-based encryption with direct chosen-ciphertext security in the standard model. *Adv. in Math. of Comm.*, 11(1):161–177, 2017. 6.1

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In *EUROCRYPT 2018*, 2018. 1.2

[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 2016*, 2016. 7

[Nuñ18] David Nuñez. UMBRAL: A threshold proxy re-encryption scheme. NuCypher White Paper, 2018. `https://github.com/nucypher/umbral-doc/blob/master/umbral-doc.pdf`. 1.2

[PBP18] Elena Pagnin, Carlo Brunetta, and Pablo Picazo-Sanchez. HIKE: Walking the privacy trail. In *CANS 2018*, 2018. 1.2

[PS08] Kenneth G. Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In *Pairing 2008*, 2008. 3

[Res18] Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018. 8

[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. 6.1

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, 2005. 1.2

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC 2014*, pages 475–484, 2014. 1

[TLLP12] Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia J. Perlman. Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Dependable Sec. Comput.*, 9(6):903–916, 2012. 1.2

[VC14] David Vorick and Luke Champine. Sia: Simple decentralized storage. White paper, 2014. `https://sia.tech/sia.pdf`. 1

[WBB$^+$16] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, James Prestwich, Gordon Hall, Patrick Gerbes, Philip Hutchins, and Chris Pollard. Storj – a peer-to-peer cloud storage network. White paper, 2016. `https://storj.io/storj.pdf`. 1

[WHH$^+$10] Scott Wolchok, Owen S. Hofmann, Nadia Heninger, Edward W. Felten, J. Alex Halderman, Christopher J. Rossbach, Brent Waters, and Emmett Witchel. Defeating vanish with low-cost sybil attacks against large dhts. In *NDSS*, 2010. 1.2

[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*, 2004. 1, 6.1