Unique-Path Identity Based Encryption With Applications to Strongly Secure Messaging

Paul Rösler[1], Daniel Slamanig[2], and Christoph Striecks[2]

[1] FAU Erlangen-Nürnberg paul.roesler@fau.de
[2] AIT Austrian Institute of Technology
{daniel.slamanig,christoph.striecks}@ait.ac.at

Abstract. *Hierarchical Identity Based Encryption* (HIBE) is a well studied, versatile tool used in many cryptographic protocols. Yet, since the performance of all known HIBE constructions is broadly considered prohibitive, some real-world applications avoid relying on HIBE at the expense of security. A prominent example for this is secure messaging: *Strongly secure* messaging protocols are provably equivalent to *Key-Updatable Key Encapsulation Mechanisms* (KU-KEMs; Balli et al., Asiacrypt 2020); so far, all KU-KEM constructions rely on *adaptive unbounded*-depth HIBE (Poettering and Rösler, Jaeger and Stepanovs, both CRYPTO 2018). By weakening security requirements for better efficiency, many messaging protocols dispense with using HIBE.

In this work, we aim to gain *better efficiency without sacrificing security*. For this, we observe that applications like messaging only need a restricted variant of HIBE for strong security. This variant, that we call *Unique-Path* Identity Based Encryption (UPIBE), restricts HIBE by requiring that each secret key can delegate at most one subordinate secret key. However, in contrast to fixed secret key delegation in Forward-Secure Public Key Encryption, the delegation in UPIBE, as in HIBE, is uniquely determined by variable identity strings from an exponentially large space. We investigate this mild but surprisingly effective restriction and show that it offers substantial complexity and performance advantages.

More concretely, we generically build *bounded-depth* UPIBE from only *bounded-collusion IBE* in the standard model; and we generically build *adaptive unbounded*-depth UPIBE from only *selective bounded*-depth HIBE in the random oracle model. These results significantly extend the range of underlying assumptions and efficient instantiations. We conclude with a rigorous performance evaluation of our UPIBE design. Beyond solving challenging open problems by reducing complexity and improving efficiency of KU-KEM and strongly secure messaging protocols, we offer a new definitional perspective on the bounded-collusion setting.

The full version [38] of this article is available in the IACR eprint archive as article 2023/248, at https://eprint.iacr.org/2023/248.

1 Introduction

Traditionally, Hierarchical Identity Based Encryption (HIBE) [29, 21] is motivated by a real-world scenario in which a sender wants to securely encrypt a message to a receiver without knowing their individual public key. Using a global main *public* key as well as a string that identifies the receiver (e.g., their email address `bob@pc.2023.ec.iacr.org`), the sender can encrypt the message via (H)IBE. To decrypt a ciphertext, the receiver can obtain their individual secret key by requesting delegation from the global main *secret* key. The hierarchy in HIBE provides a fine grained, leveled delegation: the secret key of `bob@pc.2023.ec.iacr.org` is delegated from secret key of `pc.2023.ec.iacr.org` which proceeds up to delegation from secret key of `org`. Thereby, each secret key can only delegate secret keys of subordinate identities. For the specific case of Identity Based Encryption (IBE) [41, 7], only the global main secret key can delegate identity-specific secret keys, which reduces the level depth to 1.

HIBE as a Powerful Building Block. Independent of this real-world use case, HIBE turns out to be a versatile, powerful tool in the design of larger cryptographic protocols. For example, HIBE is used as the main component in designs of Broadcast Encryption (BE) [13], Forward-Secure Public Key Encryption (FS-PKE) [8], Puncturable FS-PKE [25], 0-RTT Key Exchange with Forward Secrecy [27, 12], and Key-Updatable Key Encapsulation Mechanisms (KU-KEM) for Ratcheted Key Exchange (RKE) [37]. In most of these cases, the reason for relying on HIBE is rather the strength of HIBE secret key delegation than the traditional motivation of encrypting messages to an identity whose individual public key is unknown.

Notably, not all of these constructions utilize the full power of standard HIBE. For instance, FS-PKE can be based on relaxed Binary-Tree Encryption (BTE) [8, 33]. Furthermore, KU-KEM constructions [37, 30, 32, 3] only delegate secret keys along a single path of identities.

Introducing Unique-Path IBE. Motivated by such restricted delegations, we introduce the notion of *Unique-Path Identity Based Encryption* (UPIBE). As in HIBE, UPIBE allows a sender to encrypt messages to a receiver whose individual public key is unknown by using only a string that specifies the receiver's identity as well as a global main public key. On the receiver side, UPIBE assumes that a secret key in one level delegates at most one secret key of the subjacent level. In contrast to FS-PKE, unique-path delegation in UPIBE still respects identity (sub-)strings from an exponential size string space on each level. Consequently, a receiver with email address `bob@pc.2023.ec.iacr.org` cannot decrypt ciphertexts encrypted to identity `charlie@pc.2023.ec.iacr.org`. Beyond the cryptographic utility, there are real-world examples for such a unique-path delegation behavior in linear vertical or horizontal hierarchies.[3]

[3] E.g., the chronological succession of presidents in a particular state or a ranking list that results from a competition.

One perspective on UPIBE could be that it lifts the bounded-collusion setting from IBE [15] to HIBE by restricting adversaries in corrupting at most one delegated secret key in the identity hierarchy. Instead, we view the characteristic of UPIBE complementary or even orthogonal to the bounded-collusion setting: While bounded collusion means that the overall number of corrupted secret keys is limited, UPIBE limits the number of delegations—and, hence, corruptions—structurally *per delegated secret key*. In the specific case of *UP*IBE, we permit one delegation per secret key, but this can be extended to two or more delegations per secret key. Indeed, one of our results motivates research on HIBE with *at most two delegations* per secret key (see Section 4), which we leave as a question for future work and concentrate on UPIBE here.

UPIBE as an Abstraction of KU-KEM. In the context of strongly secure messaging, many cryptographic protocols use a building block called Key-Updatable Key Encapsulation Mechanism (KU-KEM) [37, 30, 32, 3]. This extended form of standard KEM provides an update mechanism with which public keys and secret keys can be updated independently with respect to arbitrary bit strings. In addition to the security guarantees of a standard KEM, updates in KU-KEM are required to achieve *forward-secrecy* and *effective divergence*. This means that an updated secret key cannot decrypt ciphertexts directed to prior versions of the secret key; and an incompatibly updated secret key cannot decrypt ciphertexts produced with a corresponding (incompatible) public key.

The only known construction of KU-KEM relies on black-box HIBE with *unbounded hierarchy depth* secure against *adaptive adversaries* [37, 30, 32, 3]. This induces a significant performance penalty and limits the choice of underlying assumptions (e.g., no practical[4] unbounded-depth HIBE from lattices is known). Intuitively, KU-KEM secret key updates are realized via sequential HIBE delegations. Replacing black-box HIBE in this construction by black-box UPIBE is trivial. Thus, using a black-box HIBE scheme to realize UPIBE is henceforth referred to as *trivial UPIBE construction*. By introducing UPIBE as a more general notion for KU-KEM, we are the first to lift this specific tool to a suitable abstraction and reduce the power of (underlying) HIBE to the essential. As we will see, this also allows for a substantial gain in efficiency.

Definitions and Constructions of Secure Messaging. KU-KEM was developed as a building block for constructions of secure messaging protocols. Interestingly, the impractical performance of prior KU-KEM constructions even affected security definitions in the messaging literature. These definitions can be divided into two categories: (1) those that require *full security* with respect to the modeled threats and (2) those that *relax the security requirements* by limiting adversarial power. Generally, relaxed definitions allow for more efficient constructions. Specifically, the majority of fully secure messaging protocols relies on KU-KEM [37, 30, 32, 3], whereas the main motivation for relaxing security

[4] We stress that the construction of selective-secure HIBE with unbounded delegations from CDH [17] or from any fully secure IBE [16] is an impressive, yet rather theoretic result.

definitions was to analyze or develop practical protocols that can dispense with employing KU-KEM for better efficiency [31, 2, 18]. To emphasize and substantiate this partition of the literature, Balli et al. [3] proved that KU-KEM is equivalent to fully secure messaging under weak randomness. We conclude that KU-KEM and, therefore, UPIBE play a central role in (strongly) secure messaging.

Efficiency of UPIBE and KU-KEM. The inefficiency of the trivial KU-KEM construction from black-box HIBE leads to two questions that were posed as open problems in prior work [37, 30, 3] and which we will address via the UPIBE approach:

(1) Can we build (KU-KEM from) UPIBE based on weaker assumptions?
(2) Can we build (KU-KEM from) UPIBE with better efficiency?

We are the first to affirm both questions in three steps. But instead of only giving answers for the specific case of KU-KEM, we generalize it to the UPIBE setting which highlights the reasons for our improvements.

First, we consider bounded-depth UPIBE, which means that the maximal number of secret-key delegation levels is bounded a priori. Our generic construction of bounded-depth UPIBE is based on *bounded-collusion IBE*, for which we have practical instantiations from standard assumptions like DDH or QR in the standard model [15, 23, 42].[5] In a second step, we extend the design of our bounded-depth UPIBE construction to obtain an unbounded-depth UPIBE scheme. This unbounded-depth UPIBE construction with adaptive security can be based on *bounded-depth* HIBE with only *selective* security in the random oracle model. Finally, we prove that KU-KEM can be based on UPIBE, where the number of key updates in KU-KEM is proportionate to the number of key delegations in UPIBE.

Instantiating our unbounded-depth UPIBE construction with the bounded-depth HIBE by Boneh et al. [5] reveals the strengths of our approach. We compare this instantiation to the best known instantiation of *trivial* unbounded-depth UPIBE via the unbounded-depth HIBE by Gong et al. [24]. This comparison shows that our construction is significantly more efficient by most relevant measures. In particular, it outperforms the trivial approach substantially in terms of performance, ciphertext sizes, and encryption key sizes.

A notable feature of our unbounded-depth UPIBE construction is that its efficiency can be dynamically configured via a parameter ε. Roughly, ε trades ciphertext size against secret key size. Depending on the performance priorities in a setting (bandwidth, algorithm runtime, etc.) and depending on the expected user behavior (average length of identity strings, average number of encryptions per identity, etc.), this parameter can optimize our construction for deployment under various conditions. Setting the parameter ε to infinity yields the known

[5] An alternative approach from standard assumptions would be to rely on the fully secure IBE from CDH by Garg and Döttling [17]. Unfortunately, this will not yield a practical instantiation.

trivial UPIBE construction [37, 30]; consequently, there always exists an ε for which our new UPIBE construction is indeed the best known one.

Contributions. Our first contribution is to abstract the tools in KU-KEM constructions to the more general field of Identity Based Encryption by, simultaneously, reducing the power of standard HIBE to the essential: Unique-Path IBE. Our definition from Section 2 shows that this new perspective on structurally limited delegation and collusion is seamlessly embedded in existing (H)IBE notions.

For comprehensibility, we start with building the simpler bounded-depth UPIBE construction in Section 3, which is secure in the standard model (StM):

$$\text{Adaptive Bounded-Collusion IBE} \implies_{\text{StM}} \text{Adaptive Bounded-Depth UPIBE}$$

This construction shows that UPIBE can be based on significantly reduced complexity assumptions with a practically[4] relevant design. We also give a concrete instantiation with small ciphertexts (two group elements) and secret keys (six group elements and one symmetric key) from DDH that takes advantage of construction internals of a bounded-collusion IBE by Dodis et al. [15].

By developing two powerful extensions on top of our first generic UPIBE construction, we are ultimately able to build unbounded-depth UPIBE:

$$\text{Adaptive Bounded-Depth HIBE} \implies_{\text{StM}} \text{Adaptive Unbounded-Depth UPIBE}$$

While conceptually inheriting core ideas of our *bounded*-depth UPIBE, this second *unbounded*-depth UPIBE construction in Section 4 unfolds the full strength of our approach. Its efficiency is dynamically configurable for different deployment settings and, instantiated with the most suitable bounded-depth selective HIBE [5], it reaches the best performance results compared to existing work. Along the way, inspired by techniques that turn *selective* secure bounded-depth HIBEs *adaptive* secure [4, 5], we develop a guessing technique which allows for a significantly broader choice of underlying assumptions and more efficient instantiations in the random oracle model (ROM):

$$\text{Selective Bounded-Depth HIBE} \implies_{\text{ROM}} \text{Adaptive Unbounded-Depth UPIBE}$$

We note that when instantiating our construction with lattice HIBEs [1, 10], we obtain the first KU-KEM secure under conjectured post-quantum assumptions.

We systematically analyze the performance of our approach when being used to instantiate KU-KEM in Section 7. It is notable that all prior KU-KEM constructions are a trivial special case of our new techniques. This means that our new constructions always offer the best (known) performance. For clarity, we first present semantically secure constructions of UPIBE. Using techniques known from KEM combiners [22], we show in Section 5 that our constructions can also be made secure against chosen-ciphertext attacks if the underlying (H)IBE schemes are.

1.1 Technical Overview

To understand the core idea of our UPIBE constructions, we briefly discuss the subtle difference between the security definitions of standard HIBE and UPIBE.

Although these definitions are conceptually identical, the crucial limitation of UPIBE is that at most one delegation per secret key is permitted. This means that the large tree of delegated secret keys in HIBE is reduced to a unique delegation path in UPIBE. Consequently, adversaries will essentially expose at most one UPIBE secret key—all descendant UPIBE keys can be obtained via delegation by the adversary itself. Consider the identity string that corresponds to this exposed UPIBE secret key. In relation to this identity string, our natural security definition requires only two types of challenge ciphertexts to remain secure: (1) those that are encrypted to *true prefix* identity strings and (2) those that are encrypted to identity strings *branching off* the exposed key's identity string. All remaining challenges can be solved trivially with the exposed secret key. Our UPIBE constructions exploit this fact to turn all prefix identity strings (case 1) into branched off identity strings (case 2) by adding a special suffix at the end of every UPIBE identity string.

Combined HIBE Exposure. Having the definitional difference in mind, we will see that multiple colluding exposures in HIBE can be significantly more damaging than the single permitted exposure in UPIBE. More concretely, HIBE constructions have to make sure that challenge ciphertexts remain secure under *any combination* of (non-trivial) secret key exposures in the delegation hierarchy. Since the unique-path delegation in UPIBE permits at most *one exposure*, UPIBE constructions have to protect challenge ciphertexts only against the single most damaging secret key exposure. We illustrate this gap by considering the effect of a specific combination of HIBE secret key exposures.

For this we let two exposed HIBE secret keys have identities $id_{\text{ex},1} = (id'_1)$ and $id_{\text{ex},2} = (id_1, id'_2)$, and a single HIBE challenge have identity $id_{\text{ch}} = (id_1, id_2)$, such that $id_1, id'_1, id_2, id'_2 \in \{0,1\}^\lambda$, where λ is the bit-length per delegated sub-identity string. This means, $id_{\text{ex},2}$ and id_{ch} branch in delegation level 2 with $id'_2 \neq id_2$, and $id_{\text{ex},1}$ branches off the former two identity strings in level 1 with $id'_1 \neq id_1$. Observe that the exposed key with identity $id_{\text{ex},1}$ still contains information for delegating subordinate keys to the second level, e.g., to sub-identity id_2 which results in full identity $id^* = (id'_1, id_2)$. In contrast, the exposed key with identity $id_{\text{ex},2}$ does not (need to) contain this information anymore as it is delegated to level 2 already. However, exposed key with identity $id_{\text{ex},2}$ may contain information about its own delegation path along the first level with sub-identity string id_1, which differs from the information contained in exposed key with identity $id_{\text{ex},1} = (id'_1)$. One major difficulty for building HIBE is to make sure that the information about delegation along id_1 from exposed key $id_{\text{ex},2}$ cannot be combined harmfully with the secrets available for delegation to level 2 from exposed key $id_{\text{ex},1}$. In particular, this combination should not suffice to obtain a secret key for identity $(id_1, id_2) = id_{\text{ch}}$ because this would solve the challenge. Since the single permitted exposure in UPIBE prevents such combined exposures, we can simplify the design of our UPIBE constructions, which makes them more efficient. We stress that this difference between HIBE and UPIBE is an inevitable implication of our natural definition.

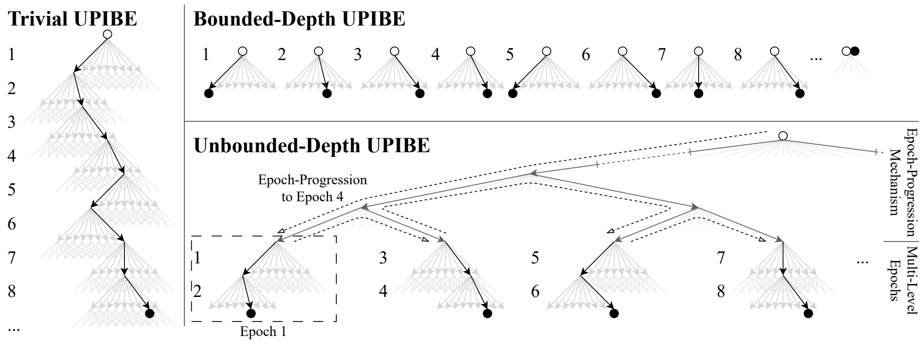

Fig. 1: Conceptual illustration of delegations in the trivial, bounded-depth, and unbounded-depth UPIBE constructions (here with $\varepsilon = 2$). The black (path of) arrows realize delegation of a UPIBE identity string with level depth 8. Light gray arrows indicate alternative and further delegations. White circles represent the (composed) main public key(s) and filled dots represent the (composed) delegated secret key(s).

Bounded-Depth UPIBE. One interpretation of the above observation is that our constructions can assume key material for lower level delegations to be per se harmless. Using this guarantee, our bounded-depth UPIBE construction implements each UPIBE delegation level with an individual IBE instance. Intuitively, this turns the vertical delegation path into a horizontal delegation sequence, as illustrated in Figure 1. Our construction's UPIBE main public and secret key consist of all underlying IBE instances' main public and secret keys, respectively. For encryption, the UPIBE identity string is split into multiple IBE sub-strings. The UPIBE ciphertext is then obtained by executing IBE encryption for each level's sub-string and concatenating the resulting IBE ciphertexts. On UPIBE delegation, the respective level's IBE main secret key is removed after delegating an identity-specific secret key for that level. To prove security of this construction, we use the fact that every challenge identity branches off the exposed key's identity in one of it's passed delegation levels. Our reduction embeds an underlying IBE challenge in this branching level, which turns a successful UPIBE adversary into a successful IBE adversary. The above description of our scheme is highly simplified and neglects subtle enhancements that lead to better performance. Although conceptually simple in the bounded-depth case, this construction does not extend (trivially) to the unbounded-depth setting.

Unbounded-Depth UPIBE. Therefore, we develop two crucial extensions: First, we replace each delegation level by an *ε-level delegation epoch*. In every such epoch, ε many sequential delegations can be processed. (See Figure 1 where $\varepsilon = 2$.) This reduces the number of concatenated ciphertexts by a factor of $1/\varepsilon$. Then, we add an *epoch-progression mechanism* on top of our construction. With this mechanism, delegation from a fully-delegated epoch progresses dynamically to the next fresh epoch. This allows us to dispose of the static list of IBE instances from our bounded-depth construction. One can think of the epoch-progression mechanism as a Forward-Secure PKE scheme that generates at every step a fresh starting point for a multi-level epoch in which the actual UPIBE delegations are conducted. The security proof for our unbounded-depth UPIBE follows the same idea as the one for our bounded-depth construction, only that it reduces to bounded-depth HIBE. To rely on only *selective* bounded-depth HIBE, we develop a special guessing technique that avoids the exponential loss factor induced by known techniques [4, 5] for turning selective HIBE adaptive secure. We believe that the solid design—in addition to its enhanced performance—makes our construction attractive for practical applications (such as secure messengers).

Chosen-Ciphertext Security. We investigate the options to obtain CCA security for UPIBE. Unfortunately, the well known generic BCHK (often also called CHK) compiler for HIBEs [9, 6] is not applicable to UPIBE. While opting for a form of verification-by-re-encryption akin to the Fujisaki-Okamoto (FO) transform [19] is applicable, one introduces significant computational overhead as well as is bound to the ROM. Instead, we leverage chosen-ciphertext security of the underlying building blocks by effectively tying together the concatenated ciphertexts in every UPIBE ciphertext. For simplicity, we referred to UPIBE as a *Message Encryption* primitive so far, but all our results actually consider *Key Encapsulation*. Therefore, in the case of bounded-depth UPIBE we can make use of techniques developed in the context of KEM combiners [22]. These versatile techniques only change the final computation of the encapsulated UPIBE key instead of explicitly authenticating the concatenated ciphertext. A similar idea, though in the ROM, can be applied in the case of unbounded-depth UPIBE where the underlying HIBE instances can be efficiently made CCA secure via the BCHK compiler. As a result, our chosen-ciphertext secure constructions are only minimally less efficient than our semantically secure ones.

2 UPIBE Definition

For clarity, we consider Identity Based *Key Encapsulation* primitives instead of Identity Based *Message Encryption* in this work. In line with this, we call public and secret keys *encapsulation* and *decapsulation* keys, respectively. Since Unique-Path IBE is a special case of Hierarchical IBE, we introduce all relevant IBE notions modularly at once.

Syntax. All of the considered *Identity Based Encapsulation* (IBE) schemes are quadruples $\mathrm{IE} = (\mathrm{IE.gen}, \mathrm{IE.enc}, \mathrm{IE.dec}, \mathrm{IE.del})$ of algorithms with encapsulation and decapsulation key spaces $\mathcal{EK}$ and $\mathcal{DK}$, respectively, symmetric key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$.

We specify the considered types of IBE via parameters L, λ, and D. L fixes the maximal number of *sequential* delegations (i.e., the maximal number of levels aka. the *depth*), λ fixes the bit-length of identity strings for each delegation, and D fixes the maximal number of delegations *per* decapsulation key. That means, for unbounded-depth HIBE we have $(\mathsf{L}, \mathsf{D}) = (\infty, 2^\lambda)$, for bounded-depth HIBE we have $(\mathsf{L}, \mathsf{D}) = (L, 2^\lambda)$ for some fixed value L, for unbounded-depth UPIBE we have $(\mathsf{L},\mathsf{D}) = (\infty, 1)$, and for bounded-depth UPIBE we have $(\mathsf{L},\mathsf{D}) = (L, 1)$ for some fixed value L. We treat bounded-collusion IBE as a bounded-depth HIBE with $\mathsf{L} = 1$ such that the number of colluding users is encoded as the number of maximal delegations for the main decapsulation key $\mathsf{D} = D$ for some constant D.

The four IBE algorithms' syntax is defined as follows:

- IE.gen : $\emptyset \to_\$ \mathcal{EK} \times \mathcal{DK}$
- IE.enc : $\mathcal{EK} \times \{0,1\}^{l\cdot\lambda} \to_\$ \mathcal{C} \times \mathcal{K}$, where $0 < l \le \mathsf{L}$
- IE.dec : $\mathcal{DK} \times \mathcal{C} \to \mathcal{K}$
- IE.del : $\mathcal{DK} \times \{0,1\}^{\lambda} \to_\$ \mathcal{DK}$

For efficiency reasons, we add derivation algorithm IE.der : $\mathcal{EK} \times \{0,1\}^{\lambda} \to_\$ \mathcal{EK}$ that computes (compact) identity-specific encapsulation keys. This allows for reducing the combined size of a main encapsulation key ek and a multi-level identity string $id = (id_1, \ldots, id_l)$, such that IE.enc$(ek, (id_1, \ldots, id_l))$ can be turned into IE.enc(IE.der$(\ldots$ IE.der$(ek, id_1) \ldots, id_l), \epsilon)$.

Correctness. For correctness of all considered types of IBE with parameters L, λ, and D, we require for all $(ek, dk_0) \leftarrow_\$$ IE.gen, all $id = (id_1, \ldots, id_l)$ with $id_i \in \{0,1\}^{\lambda}, 0 < i \le l \le \mathsf{L}$, all $dk_i \leftarrow_\$$ IE.del(dk_{i-1}, id_i), and all $(c, k) \leftarrow_\$$ IE.enc(ek, id), that IE.dec$(dk_l, c) = k$.

Security. We define experiment $\mathrm{IND}^b_{\mathrm{IE}}(\mathcal{A}), b \in \{0,1\}$ that models multi-instance key indistinguishability. For all considered types of IBE schemes IE, this experiment provides the following oracles to adversary $\mathcal{A}$ for which we provide a full pseudo-code specification in the full version [38]:

- Gen: **Generates** a fresh main key pair $(ek, dk) \leftarrow_\$$ IE.gen and returns ek
- Del(i, id, id^*): **Delegates** decapsulation key $dk_{i,(id,id^*)} \leftarrow_\$$ IE.del$(dk_{i,id}, id^*)$ from $dk_{i,id}$ with identity string $id^* \in \{0,1\}^{\lambda}$, unless $dk_{i,id}$ results from L sequential delegations from a main decapsulation key, or D delegations from $dk_{i,id}$ were already queried
- Chall(i, id): Issues a **challenge** encapsulation $(c, k_0) \leftarrow_\$$ IE.enc(ek_i, id) to main encapsulation key ek_i and identity string $id \in \{0,1\}^{l\cdot\lambda}, 0 < l \le \mathsf{L}$ and returns c as well as key k_b, where $k_1 \leftarrow_\$ \mathcal{K}$, unless an exposed decapsulation key was delegated from ek_i's main decapsulation key dk_i with an identity string that equals or is a prefix of id
- Exp(i, id): **Exposes** decapsulation key $dk_{i,id}$, generated or delegated from main decapsulation key dk_i and identity string id, unless a challenge encapsulation to ek_i and identity string id' was queried, such that (ek_i, dk_i) form a main key pair and id equals or is a prefix of id'

Eventually, the adversary terminates by outputting a guess b' and wins iff $b = b'$.

If adversary $\mathcal{A}$ specifies the challenge(s) at the beginning of the game without adaptively seeing the return values of other queries, we call $\mathcal{A}$ **selective** and otherwise **adaptive**.

With the above adversarial oracles, we capture **chosen-plaintext** attacks. Selective chosen-plaintext attacks is a rather weak adversary model that helps us focusing on the core of our novel ideas when presenting our constructions. Yet, we also present adaptive **chosen-ciphertext** secure constructions. An adversary, attacking such constructions, can additionally query the following oracle:

– $\mathrm{Dec}(i, id, c)$: **Decapsulates** $k \leftarrow \mathrm{IE.dec}(dk_{i,id}, c)$ of ciphertext c under $dk_{i,id}$ and returns k, unless c was given to $\mathcal{A}$ as a challenge encapsulation to ek_i and id, $dk_{i,id}$ was (sequentially) delegated from dk_i with respect to id, and (ek_i, dk_i) form a main key pair

Definition 1. *The advantage of adversary* $\mathcal{A}$ *in winning* $\mathrm{IND}^b_{\mathrm{IE}}$ *is* $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) := \left|\Pr[\mathrm{IND}^0_{\mathrm{IE}}(\mathcal{A}) = 1] - \Pr[\mathrm{IND}^1_{\mathrm{IE}}(\mathcal{A}) = 1]\right|$.

Compared to standard (bounded-depth) (H)IBE security experiments, the only difference is our restriction to at most D delegation queries per decapsulation key. Yet, challenges can be queried without limiting the choice of identity strings, even for UPIBE.

3 Bounded-Depth UPIBE from Bounded-Collusion IBE

We present our bounded-depth UPIBE construction in Figure 2 by explaining its components one after another, starting with decapsulation keys and ciphertexts.

Structure of Keys and Ciphertexts. The core idea behind our UPIBE constructions is that delegations along the unique 'vertical' path of identity levels are realized 'horizontally'. That means, for each delegation level in our UPIBE construction from Figure 2 with bounded-depth L, we use a separate bounded-collusion IBE instance. Think of these IBE instances being placed horizontally next to one another from left to right as shown in Figure 1.

To understand this idea, we describe the structure of UPIBE decapsulation keys. A *UPIBE* decapsulation key delegated to level l contains three different types of keys, two of which are *IBE* decapsulation keys: (1) One *ordinary delegated* IBE decapsulation key for *each* of the first l levels, (2) an additional *special delegated* IBE decapsulation key for *only* level l, and (3) a symmetric *forwarding* key from which (*un-delegated*) IBE main decapsulation keys for all remaining $\mathsf{L}-l$ levels are computed. See Figure 2 lines 02-06 for UPIBE key generation that consists of generating all IBE main encapsulation keys and sampling the initial symmetric forwarding key.

A UPIBE ciphertext, encapsulated to level l (i.e., to identity $id \in \{0,1\}^{l\cdot\lambda}$), consists of one IBE ciphertext for each of the first $l-1$ levels encoded with suffix $\mathtt{1}$ (lines 23-25) and one additional IBE ciphertext that targets the special delegated IBE decapsulation key at level l encoded with suffix $\mathtt{0}$ (line 26). To decapsulate the former $l-1$ ciphertexts (lines 33-34), the receiver needs to be in possession of the first $l-1$ ordinary delegated IBE decapsulation keys. Hence, successful decapsulation shows that the receiver holds a UPIBE decapsulation key that was correctly delegated along the first $l-1$ levels of the identity path. By also being able to decapsulate the special lth IBE ciphertext (lines 35-36), the

```
Proc IE.gen
00 E[·] ← ⊥; D[·] ← ⊥
01 fk_0 ←$ {0,1}^λ
02 For l = 0 to L − 1:
03    (fk_{l+1}, s) ← G(fk_l)
04    (ek', dk') ← IE.gen'(s)
05    E[l] ← ek'
06 ek ← E; dk ← (0, ⊥, fk_0)
07 Return (ek, dk)

Proc IE.del(dk, id)
08 Require id ∈ {0,1}^λ
09 (l, D, fk) ← dk
10 Require l < L
11 If l > 0:
12    (dk'_0, dk'_1) ← D[l − 1]
13    D[l − 1] ← dk'_1
14 (fk', s) ← G(fk)
15 (ek', dk') ← IE.gen'(s)
16 dk'_0 ←$ IE.del'(dk', id‖0)
17 dk'_1 ←$ IE.del'(dk', id‖1)
18 D[l] ← (dk'_0, dk'_1)
19 dk ← (l + 1, D, fk')
20 Return dk

Proc IE.enc(ek, id)
21 Require id ∈ {0,1}^{l·λ}, 0 < l ≤ L
22 E ← ek
23 id_0‖...‖id_{l−1} ← id with id_j ∈ {0,1}^λ
24 For j = 0 to l − 2:
25    (c'_j, k'_j) ←$ IE.enc'(E[j], id_j‖1)
26 (c'_{l−1}, k'_{l−1}) ←$ IE.enc'(E[l − 1], id_{l−1}‖0)
27 C ← c'_0‖...‖c'_{l−1}
28 K ← W(k'_0, ..., k'_{l−1}, C)
29 Return (C, K)

Proc IE.dec(dk, C)
30 (l, D, fk) ← dk
31 c_0‖...‖c_{l'−1} ← C with c_j ∈ 𝒞
32 Require l = l'
33 For j = 0 to l − 2:
34    k'_j ←$ IE.dec'(D[i], c_j)
35 (dk'_0, dk'_1) ← D[l − 1]
36 k'_{l−1} ←$ IE.dec'(dk'_0, c_{l−1})
37 K ← W(k'_0, ..., k'_{l−1}, C)
38 Return K
```

Fig. 2: Construction of bounded-depth UPIBE IE with parameters $(\mathsf{L}, \lambda, \mathsf{D} = 1)$ from PRG G and bounded-collusion IBE scheme IE′ with parameters $(\mathsf{L}', \lambda', \mathsf{D}') = (1, \lambda + 1, 2)$ and ciphertext space $\mathcal{C}$. Core function W is realized as XOR-sum $\bigoplus_{j=0}^{l-1} k'_j$ and ignores input C. In our chosen-ciphertext secure instantiation, we additionally generate a dummy ciphertext $\hat{c} \leftarrow_\$ \mathcal{C}$ and key $\hat{k} \leftarrow_\$ \mathcal{K}$ in IE.gen, which is included into ek and W to pad all unused indices $i \leq \mathsf{L}$ with $\hat{c}$ and $\hat{k}$ respectively.

receiver additionally shows that it holds the full UPIBE decapsulation key that was delegated along all l levels—and particularly not a UPIBE decapsulation key that was delegated along an *extended* identity path.

While a UPIBE ciphertext is a concatenation of all l IBE ciphertexts, the encapsulated UPIBE key is an XOR-sum of all l encapsulated IBE keys (lines 27-28). We generalize the computation of the encapsulated key via core function W to simplify the description of our chosen-ciphertext secure construction in Section 5.

Delegation of a UPIBE decapsulation key is in line with the above ideas by conducting four steps: (a) Removing the special IBE decapsulation key at current level l, yet keeping all ordinary IBE decapsulation keys until level l (lines 11-13), (b) computing the next forwarding key as well as a seed by evaluating a PRG on the current forwarding key (line 14), (c) generating the main IBE decapsulation key at level $l+1$ from the obtained seed (line 15), and delegating both the special delegated IBE decapsulation key for level $l+1$ (line 16) as well as the ordinary delegated IBE decapsulation key for level $l+1$ (line 17) from this new main IBE decapsulation key, and, lastly, (d) removing the just obtained main IBE decapsulation key at level $l+1$ as well as the old forwarding key.

Intuition for Security. The security argument for this construction uses the fact that adversaries can expose at most one UPIBE decapsulation key per instance during the security experiment.[6] This single exposure reveals precisely one special delegated IBE decapsulation key—the current one—, the chain of ordinary IBE decapsulation keys that were delegated along the exposed UPIBE key's identity path, and the current symmetric forwarding key from which future levels' IBE main decapsulation keys can be obtained. After such an exposure, two types of UPIBE ciphertexts must remain secure: Those that target *true prefixes* of the exposed key's identity string, and those that target identity strings *branching off* the exposed key's identity string.[7] Ciphertexts targeting a true prefix identity string, indeed, remain secure because their decapsulation requires the use of a higher level special delegated IBE decapsulation key. Such prior level special IBE keys were removed before the exposure and are, therefore, not contained in the exposed UPIBE key. Similarly, the decapsulation of ciphertexts that target a branched off identity string require the use of an inaccessible IBE decapsulation key—namely, an ordinary IBE decapsulation key that was delegated along this branch. Consequently, exposed UPIBE decapsulation keys do not affect ciphertexts that are required to remain secure. Finally, we note that at most two delegated decapsulation keys per IBE instance are leaked at an exposure of a UPIBE decapsulation key. Thus, relying on bounded-collusion IBE suffices, where the number of colluding users is at most 2.

Performance. Bounded-depth UPIBE (and bounded-depth KU-KEM) actually often suffice for secure messaging protocols.[8] So far, the only known instantiation of bounded-depth UPIBE is trivially derived from bounded-depth HIBE. With our bounded-depth UPIBE construction we demonstrate a significant reduction in complexity of the underlying hardness assumption: bounded-collusion IBE instead of bounded-depth HIBE. Furthermore, we use this construction to make the reader familiar with the core ideas of our unbounded-depth UPIBE construction in Section 4.

Without any additional assumptions on the underlying bounded-collusion IBE, the size of UPIBE encapsulation keys in our construction is linear in the maximal level depth L, UPIBE decapsulation keys grow with the number of conducted delegations, and UPIBE ciphertexts grow in the bit-length of their corresponding identity string.

When instantiating our construction with the DDH-based bounded-collusion IBE by Dodis et al. [15], we can take advantage of the group structure to aggregate and shrink encapsulation keys, decapsulation keys, and ciphertexts. We give the concrete instantiation in the full version [38] in which a UPIBE decapsulation

[6] With the exposed UPIBE decapsulation key, the adversary can compute all subsequent delegations and decapsulations itself, so further exposures are meaningless.

[7] Branching here means that for two identity strings id, id^* with $\ell^* = \min(|id|, |id^*|)$, strings id and id^* differ in at least one of the first ℓ^* bits.

[8] E.g., the number of conducted key delegations in the bidirectional messaging protocol in [36, see page 22] is upper-bounded by the maximal number of ciphertexts that cross the wire during a round-trip time (i.e., at most a few dozens).

key consists of 6 exponents and 1 symmetric key, a UPIBE ciphertext consists of 2 group elements, and a UPIBE encapsulation key consists of $2 + 3(\mathsf{L} - l)$ group elements, where l is the level for which the current encapsulation key is derived via algorithm IE.der. This is highly efficient for settings in which distribution and storage of large encapsulation keys is cheap.[9] Enhancing this construction to also obtain a compact, constant size encapsulation key remains an interesting open problem.

Security. For clarity, we first consider chosen-plaintext security $\mathrm{IND}^b_{\mathrm{IE}}$ of our UPIBE construction:

Theorem 1. *Bounded-depth UPIBE protocol* IE *from Figure 2 offers adaptive key indistinguishability in the standard model. More precisely, for every adaptive chosen-plaintext adversary $\mathcal{A}$ attacking protocol* IE *in games $\mathrm{IND}^b_{\mathrm{IE}}$ according to Definition 1 with parameters $(\mathsf{L}, \lambda, \mathsf{D} = 1)$, there exists an adversary $\mathcal{B}_{\mathrm{G}}$ attacking PRG* G *and an adaptive chosen-plaintext adversary $\mathcal{B}_{\mathrm{IE}'}$ attacking bounded-collusion IBE* IE′ *in games $\mathrm{IND}^b_{\mathrm{IE}'}$ according to Definition 1 with parameters $(\mathsf{L}', \lambda', \mathsf{D}') = (1, \lambda+1, 2)$ such that $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) \leq q_{\mathrm{Gen}} \cdot \mathsf{L}^2 \cdot \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{G}}(\mathcal{B}_{\mathrm{G}}) + q_{\mathrm{Gen}} l \cdot q_{\mathrm{Chall}} \cdot \mathsf{L} \cdot \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}'}(\mathcal{B}_{\mathrm{IE}'})$, where q_{Gen} and q_{Chall} are the number of queries to oracles* Gen *and* Chall *by adversary $\mathcal{A}$, respectively, and the running time of $\mathcal{B}_{\mathrm{G}}$ and $\mathcal{B}_{\mathrm{IE}'}$ is about that of $\mathcal{A}$.*

Security Proof Overview. For clarity in notation, we refer to oracles in game IND^b_X by adding the scheme's identifier X as a subscript to the oracle names (i.e., Gen_X, Chall_X, etc). Also, we first sketch our proof by focusing on a reduction from single-instance security of UPIBE to multi-instance security of IBE.

Using the PRG, we begin with a hybrid argument that replaces all unexposed symmetric forwarding keys and IBE main key pairs with independently sampled ones. Our reduction $\mathcal{B}_{\mathrm{IE}'}$ then almost directly passes oracle queries from adversary $\mathcal{A}$ against our UPIBE construction IE in game $\mathrm{IND}_{\mathrm{IE}}$ to oracles of game $\mathrm{IND}_{\mathrm{IE}'}$ against the underlying bounded-collusion IBE scheme IE′. The responses of oracles in game $\mathrm{IND}_{\mathrm{IE}'}$ can then be used almost directly to answer adversary $\mathcal{A}$'s oracle queries in game $\mathrm{IND}_{\mathrm{IE}}$. That means, $\mathcal{A}$'s queries to oracle $\mathrm{Gen}_{\mathrm{IE}}$ can be answered by using responses of simple queries to oracle $\mathrm{Gen}_{\mathrm{IE}'}$; the same holds for queries to oracle $\mathrm{Del}_{\mathrm{IE}}$.

However, embedding challenges from game $\mathrm{IND}_{\mathrm{IE}'}$ in challenges of game $\mathrm{IND}_{\mathrm{IE}}$ is non-trivial. To understand this, we observe that the hardness of a challenge in game $\mathrm{IND}_{\mathrm{IE}}$ depends on the delegation path of the first (and w.l.o.g. only) exposed UPIBE decapsulation key in game $\mathrm{IND}_{\mathrm{IE}}$. More precisely, let id^* be the identity string that corresponds to the delegation path of the first exposure via oracle $\mathrm{Exp}_{\mathrm{IE}}$. A challenge directed to identity string id is only considered hard if id is a true prefix of id^*, or if id and id^* differ in at least one of their

[9] Consider asymmetric communication for which ciphertexts should be small and encapsulation keys can be large: E.g., sending large encapsulation keys on hardware memory from time to time via resupply flights to the International Space Station, and sending ciphertexts over the air back to earth.

first ℓ^* bits, where $\ell^* = \min(|id|, |id^*|)$. On a query to oracle $\mathrm{Chall}_{\mathrm{IE}}$ with identity string id, our reduction $\mathcal{B}_{\mathrm{IE}'}$ splits id into its λ-long sub-strings and then identifies in which of these sub-strings the first difference between id and id^* occurs. For this branching sub-string, reduction $\mathcal{B}_{\mathrm{IE}'}$ queries an IBE challenge via oracle $\mathrm{Chall}_{\mathrm{IE}'}$. The resulting IBE challenge-ciphertext and IBE challenge-key are then embedded in the corresponding UPIBE challenge-ciphertext and UPIBE challenge-key output of oracle $\mathrm{Chall}_{\mathrm{IE}}$. However, reduction $\mathcal{B}_{\mathrm{IE}'}$ learns string id^* only as soon as adversary $\mathcal{A}$ calls oracle $\mathrm{Exp}_{\mathrm{IE}}$. Hence, for each challenge issued before this first exposure query, reduction $\mathcal{B}_{\mathrm{IE}'}$ has to guess in which sub-string the identities branch. Embedding this guessing step in a hybrid argument introduces a loss factor of at most $q_{\mathrm{Gen}} \cdot q_{\mathrm{Chall}} \cdot \mathsf{L}$, where q_{Gen} and q_{Chall} are the numbers of queries to oracles $\mathrm{Gen}_{\mathrm{IE}}$ and $\mathrm{Chall}_{\mathrm{IE}}$ by adversary $\mathcal{A}$, resp., and L is the maximal number of delegation levels for our UPIBE construction. We provide our formal proof for multi-instance security in the full version [39].

4 Unbounded-Depth UPIBE from Bounded-Depth HIBE

Our unbounded-depth UPIBE construction extends our bounded-depth construction from Section 3 twofold: Horizontally, it replaces each level—realized by an IBE instance in our bounded-depth construction—by a *multi-level epoch*. Each epoch can internally handle up to ε sub-identity levels/delegations. The second extension replaces the static list of IBE main keys at the top of our bounded-depth UPIBE construction by a dynamic *epoch-progression* mechanism. This mechanism realizes a dynamic progression from one epoch to another and, thereby, eliminates the a-priori bounded number of sub-identity levels/delegations; see Figure 1 for a schematic illustration.

The only component used to build our unbounded-depth UPIBE construction is a single bounded-depth HIBE scheme. To understand how the (unbounded number of) UPIBE delegations are processed by this bounded-depth HIBE, we invite the reader to look at the tree of identities/delegations in this HIBE that is indicated by gray (dotted) lines and arrows in Figure 1.

Epoch-Progression via Forward-Secure PKE Technique. In the top α levels of the HIBE tree, we implement the epoch-progression mechanism, where $\alpha = \lceil \log(2^\kappa/\varepsilon) \rceil$ and κ is the security parameter. Of these α top HIBE delegation levels, we only make use of a binary delegation (sub-)tree. Each path in this binary tree part of the HIBE tree is the binary-encoding of an epoch number, where first epoch 0 is encoded as the left-most path and last epoch $2^\kappa/\varepsilon - 1$ is encoded as the right-most path. The lowest nodes in this top binary tree part (i.e., nodes in level α) represent *epoch starting nodes*. The first epoch starts at the left-most node which corresponds to the identity string that binary-encodes 0 (i.e., $\mathsf{0}^{\alpha \cdot \lambda'}$, where λ' is the bit-length of HIBE identity sub-strings per level/delegation). We defer the explanation of how UPIBE delegations are realized *within* epochs to the next paragraph. As soon as an epoch is completed, the next epoch starts at the adjacent binary-tree node to the right in level α. (That is, starting nodes of epochs 2 and 3 correspond to identity strings $0^{\alpha\cdot\lambda'-1}\|1$ and $0^{(\alpha-1)\cdot\lambda'-1}\|1\|0^{\lambda'}$, respectively, where each level's identity sub-string contains a $(\lambda'-1)$-long 0-bit padding prefix.)

Progression from one epoch starting node to the next one follows the well known idea of Forward-Secure PKE from Binary Tree Encryption [8].[10] Roughly, the epoch-progression mechanism iteratively delegates HIBE decapsulation keys along the α-long path from the root to the current epoch starting node. During this path delegation, also decapsulation keys of (binary-tree) siblings along this path are delegated. After each delegation on this path, the respective parent node's key from which the two sibling keys were delegated is deleted. Only the first epoch progression starts at the root of the HIBE tree. All following epoch progressions start from the lowest level for which a delegated sibling key exists. This mechanism ensures that only starting nodes of future epochs but not of previous epochs are accessible.

Multi-Level Epochs. Our UPIBE construction splits identity strings of length $l\cdot\lambda$ into $\varepsilon\cdot\lambda$-long *epoch sub-strings*. Each individual epoch sub-string is delegated in ε steps vertically in the HIBE tree under its epoch starting node (i.e., each epoch contains ε delegation levels). Hence, every epoch sub-string in the HIBE tree looks exactly the same as its UPIBE identity sub-string counterpart (see Figure 1). However, instead of being concatenated vertically in the HIBE tree, one can think of the vertical epoch sub-strings hanging next to one another from left to right under their epoch starting nodes in level α.

Structure of Keys and Ciphertexts. Despite these two crucial extensions, the overall idea of our unbounded-depth UPIBE construction is very close to its bounded-depth counterpart from Section 3. This becomes evident when looking at the structure of UPIBE decapsulation keys and ciphertexts.

A *UPIBE* decapsulation key at delegation level l contains three types of delegated *HIBE* decapsulation keys: (1) up to α *epoch-progression* decapsulation keys, (2) one *ordinary* decapsulation key for each of the previous $\lceil l/\varepsilon\rceil-1$ epochs and, potentially, one *ordinary* decapsulation key for the current epoch, and (3) a *special* decapsulation key for the current epoch. The epoch-progression decapsulation keys replace the single symmetric forwarding key from our bounded-depth construction. This allows for efficient delegation of *future* epochs' initial decapsulation keys, yet preventing access to *previous* epochs' initial decapsulation keys. Ordinary and special decapsulation keys are used for the actual decapsulation of UPIBE ciphertexts (almost) as in our bounded-depth construction.

The concrete components of a UPIBE decapsulation key are as follows. One *ordinary* HIBE decapsulation key, delegated to the lowest HIBE tree level $\alpha+\varepsilon$, is stored for each finished epoch. All remaining HIBE decapsulation keys, ever delegated in these prior epochs, are removed from the (delegated) UPIBE

[10] For clarity in our explanation, we slightly deviate from the original BTE-to-FS-PKE idea by Canetti et al. [8]: We do not use all nodes in the BTE tree as epoch starting points but only nodes in the lowest level of this BTE component.

decapsulation key. For the current epoch, a *special* decapsulation key delegated to HIBE level $\alpha + (l \mod \varepsilon)$ in that epoch is stored in the UPIBE decapsulation key, where l is the overall number of UPIBE delegations so far. When delegating the UPIBE decapsulation key, this *special* HIBE decapsulation key is replaced by a new one for the next level. Only in the last level $\alpha + \varepsilon$ of the current epoch where $(l = -1 \mod \varepsilon)$, the UPIBE decapsulation key contains two HIBE keys: a special *and* an ordinary HIBE decapsulation key.

A UPIBE ciphertext for level l consists of one HIBE ciphertext per existing epoch, where $\lceil l/\varepsilon \rceil$ is the number of existing epochs. Each of the first $\lceil l/\varepsilon \rceil - 1$ ciphertexts is directed to its epoch's ordinary decapsulation key, and the last ciphertext is directed to the current epoch's special decapsulation key.

All UPIBE delegations within an epoch delegate a new special HIBE decapsulation key from the previous level's special HIBE decapsulation key. After each delegation, this previous special HIBE decapsulation key is removed. In the lowest level of an epoch—in HIBE tree level $\alpha + \varepsilon$—an additional ordinary HIBE decapsulation key is delegated the from previous level's special HIBE decapsulation key. This ordinary HIBE decapsulation key is never removed from the UPIBE decapsulation key.

Intuition for Security. The intuitive security argument for this construction resembles the one from Section 3. Recall that, on exposure of a UPIBE decapsulation key, only those UPIBE encapsulations must remain secure whose targeted identity string either is a *true prefix* of the exposed key's identity string or *branches off* the exposed key's identity string.[7] Encapsulations to true prefix identity strings have their last HIBE encapsulation directed to an *earlier* special HIBE decapsulation key. This special key is not stored in the exposed UPIBE decapsulation key anymore, since the latter only contains the *current* level's special HIBE decapsulation key. Encapsulations to branched off identity strings have the HIBE encapsulation of the branching epoch directed to an ordinary HIBE decapsulation key that was never stored in the exposed UPIBE decapsulation key. Finally, all exposed decapsulation keys of the epoch-progression mechanism only reveal parts of the HIBE tree from which future epochs can be delegated. Thus, UPIBE encapsulations of our unbounded-depth construction remain secure under non-trivial exposures of UPIBE decapsulation keys.

Construction. We specify our unbounded-depth UPIBE construction formally in Figure 3. This construction uses a bounded-depth HIBE with maximal level depth $\mathsf{L} = \alpha + \varepsilon = \lceil \log(2^\kappa/\varepsilon) \rceil + \varepsilon$, where κ is the security parameter.

The UPIBE encapsulation key consists solely of the main HIBE encapsulation key. The initial UPIBE decapsulation key is generated by executing the epoch-progression mechanism with the main HIBE decapsulation key to derive the first epoch's starting decapsulation key (Figure 3, lines 02-06). More concretely, this mechanism delegates one ephemeral and one stored decapsulation key in each of the first α HIBE levels (lines 03-04). Ephemeral key dk'_0 is replaced after delegating the two decapsulation keys of the next level. Stored key dk'_1 will be used for future epoch progressions. In level α, ephemeral key dk'_0 is set as the

Proc IE.gen

00 $E[\cdot] \leftarrow \bot;\ D_{ep}[\cdot] \leftarrow \bot;\ D_{fs}[\cdot] \leftarrow \bot$
01 $(ek', dk'_0) \leftarrow_\$ \mathrm{IE.gen}'$
02 For $j = 0$ to $\alpha - 1$:
03 $dk''_0 \leftarrow_\$ \mathrm{IE.del}'(dk'_0, \mathtt{0}^{\lambda+1})$
04 $dk''_1 \leftarrow_\$ \mathrm{IE.del}'(dk'_0, \mathtt{0}^{\lambda} \| \mathtt{1})$
05 $dk'_0 \leftarrow dk''_0;\ D_{fs}[j] \leftarrow dk''_1$
06 $D_{ep}[0] \leftarrow dk'_0$
07 $ek \leftarrow ek';\ dk \leftarrow (0, D_{fs}, D_{ep})$
08 Return (ek, dk)

Proc IE.enc(ek, id)

09 Require $id \in \{0,1\}^{l\cdot\lambda}, l \in \mathbb{N}^+$
10 $id_0\|\dots\|id_{l-1} \leftarrow id$ with $id_j \in \{0,1\}^\lambda$
11 $d \leftarrow l \mod \varepsilon;\ e \leftarrow \lceil l/\varepsilon \rceil$
12 For $e' = 0$ to $e-2$:
13 $id' \leftarrow \epsilon$
14 $(e'_0,\dots,e'_{\alpha-1}) \leftarrow e'$ with $e'_j \in \{0,1\}$
15 For $j = 0$ to $\alpha - 1$:
16 $id' \xleftarrow{\|} \mathtt{0}^\lambda \| e'_j$
17 For $d' = 0$ to $\varepsilon - 2$:
18 $id' \xleftarrow{\|} id_{e'\cdot\varepsilon+d'}\|\mathtt{1}$
19 $id' \xleftarrow{\|} id_{e'\cdot\varepsilon+\varepsilon-1}\|\mathtt{0}$
20 $(c'_{e'}, k'_{e'}) \leftarrow \mathrm{IE.enc}'(ek, id')$
21 $id' \leftarrow \epsilon$
22 $(e'_0,\dots,e'_{\alpha-1}) \leftarrow e-1$ with $e'_j \in \{0,1\}$
23 For $j = 0$ to $\alpha - 1$:
24 $id' \xleftarrow{\|} \mathtt{0}^\lambda \| e'_i$
25 For $d' = 0$ to $d-1$:
26 $id' \xleftarrow{\|} id_{(e-1)\cdot\varepsilon+d'}\|\mathtt{1}$
27 $(c'_{e-1}, k'_{e-1}) \leftarrow \mathrm{IE.enc}'(ek, id')$
28 $C \leftarrow c'_0\|\dots\|c'_{e-1}$
29 $K \leftarrow \mathrm{W}(k'_0,\dots,k'_{e-1},C)$
30 Return (C, K)

Proc IE.dec(dk, C)

31 $(l, D_{fs}, D_{ep}) \leftarrow dk$
32 $d \leftarrow l \mod \varepsilon;\ e \leftarrow \lceil l/\varepsilon \rceil$
33 $c_0\|\dots\|c_{e'-1} \leftarrow C$ with $c_j \in \mathcal{C}$
34 Require $e = e'$
35 For $j = 0$ to $e-2$
36 $k'_j \leftarrow_\$ \mathrm{IE.dec}'(D_{ep}[j], c_j)$
37 If $d \neq \varepsilon - 1$: $dk'_1 \leftarrow D_{ep}[e-1]$
38 Else: $(dk'_0, dk'_1) \leftarrow D_{ep}[e-1]$
39 $k'_{e-1} \leftarrow_\$ \mathrm{IE.dec}'(dk'_1, c_{e-1})$
40 $K \leftarrow \mathrm{W}(k'_0,\dots,k'_{e-1},C)$
41 Return K

Proc IE.del(dk, id)

42 Require $id \in \{0,1\}^\lambda$
43 $(l, D_{fs}, D_{ep}) \leftarrow dk$
44 $d \leftarrow l \mod \varepsilon;\ e \leftarrow \lfloor l/\varepsilon \rfloor$
45 If $d = 0 \wedge e > 0$:
46 $(dk'_0, dk'_1) \leftarrow D_{ep}[e-1]$
47 $D_{ep}[e-1] \leftarrow dk'_0$
48 $j \leftarrow \mathrm{msdb}(e-1, e)$
49 $dk'_0 \leftarrow D_{fs}[j];\ D_{fs}[j] \leftarrow \bot$
50 For j to $\alpha - 1$:
51 $dk''_0 \leftarrow_\$ \mathrm{IE.del}'(dk'_0, \mathtt{0}^{\lambda+1})$
52 $dk''_1 \leftarrow_\$ \mathrm{IE.del}'(dk'_0, \mathtt{0}^{\lambda} \| \mathtt{1})$
53 $dk'_0 \leftarrow dk''_0;\ D_{fs}[j] \leftarrow dk''_1$
54 $D_{ep}[e] \leftarrow dk'_0$
55 If $d \neq \varepsilon - 1$:
56 $D_{ep}[e] \leftarrow_\$ \mathrm{IE.del}'(D_{ep}[e], id\|\mathtt{1})$
57 Else:
58 $dk'_0 \leftarrow_\$ \mathrm{IE.del}'(D_{ep}[e], id\|\mathtt{0})$
59 $dk'_1 \leftarrow_\$ \mathrm{IE.del}'(D_{ep}[e], id\|\mathtt{1})$
60 $D_{ep}[e] \leftarrow (dk'_0, dk'_1)$
61 $dk \leftarrow (l+1, D_{fs}, D_{ep})$
62 Return dk

Fig. 3: Generic construction of unbounded-depth UPIBE IE from bounded-depth HIBE scheme IE′ with ciphertext space $\mathcal{C}$. Function $\mathrm{msdb}(x, y)$ computes the most significant bit in which the bit-representations of x and y differ and core function W is realized as XOR-sum $\bigoplus_{j=0}^{e-1} k'_j$ and ignores input C. In our chosen-ciphertext secure instantiation we instantiate W with random oracle $\mathrm{H}^\star$.

first epoch's starting decapsulation key. We explain the specific encoding- and padding-scheme for identity strings at the end of this paragraph.

UPIBE encapsulation splits the targeted identity string id into $\varepsilon\cdot\lambda$-long *epoch sub-strings*. Our pseudo-code separates the processing of the first $e-1$ epoch sub-strings (lines 12-20) from the last epoch's sub-string (lines 21-27). Roughly, each epoch sub-string (composed in lines 17-19 resp. 25-26) is prepended with a binary encoding of the corresponding epoch number (lines 14-16 resp. 22-24). The binary encoding prefix represents the epoch-progression path to the epoch's starting node. For every epoch, an HIBE encapsulation directed to the concatenated string of binary-encoded *epoch number* and *epoch identity sub-string* is executed (line 20 resp. 27). The final UPIBE ciphertext is a simple concatenation of all epoch HIBE ciphertexts; the output UPIBE key is an XOR-sum of all encapsulated epoch HIBE keys.

On UPIBE decapsulation, the input ciphertext is decomposed, and each of the resulting HIBE ciphertexts is decapsulated. For all previous epochs, the stored lowest level ordinary decapsulation key is used for decapsulation (lines 35-36). In the current epoch, the special decapsulation key is used for this (line 39). Depending on whether the current epoch reached its lowest level or not, the special decapsulation key is stored solitarily (line 37) or together with the ordinary decapsulation key (line 38).

In most cases, UPIBE delegation simply uses the current epoch's special HIBE decapsulation key together with input identity string *id* to delegate a new special HIBE decapsulation key that replaces the prior one (lines 56). Only if the lowest level of the current epoch is reached, an additional ordinary HIBE decapsulation key is delegated and stored (line 58-60). A subsequent delegation starts a new epoch and, therefore, uses the epoch-progression mechanism (lines 45-54). This mechanism starts by deleting the previous epoch's special decapsulation key (lines 46-47). Then, it identifies the lowest existing decapsulation key in the underlying binary-tree structure (line 48) with which the next epoch starting node is delegated (lines 50-54). This subsequent starting node—basically the immediate neighbor node in the binary tree—is used as the new epoch's initial decapsulation key.

We elaborate on some implementation details. To realize a binary tree in the epoch-progression mechanism, the binary encoding of epoch numbers is padded with $(\lambda' - 1 = \lambda)$ leading 0-bits in every level (lines 03-04, 16, 24, 51-52). For the composition of epoch sub-strings, each *level's* identity *sub-string* is appended with a 1-bit (lines 18, 26) except for the last level in any previous epoch; previous epochs' last level sub-strings have an appended 0-bit (lines 19). This corresponds to the use and delegation of special and ordinary decapsulation keys (lines 56, 58-59).

Depth of Multi-Level Epochs. Our unbounded-depth UPIBE construction is parameterized by variable ε that determines the number of delegations per epoch. We note that for $\varepsilon = \infty$, our UPIBE construction reduces to the known trivial delegation design via *unbounded*-depth HIBE [37, 30, 3]. Thus, there is *always* an ε for which our construction is *at least as efficient* as the previous approach. Beyond that, using the flexibility of parameter ε, our construction's performance can be adapted to different use cases. For example, depending on whether ciphertexts or decapsulation keys should be small, and depending on the expected number of delegations in a setting, an optimal value ε can be configured. Our full evaluation is in Section 7.

2-Delegation HIBE. We want to note that each HIBE decapsulation key in our construction from Figure 3 delegates at most two child decapsulation keys. Thus, while reducing the level depth parameter L substantially from infinity in UPIBE to a bounded value in the underlying HIBE, parameter D only grows from 1 delegation per secret key in UPIBE to 2 in the underlying HIBE. With our definition framework from Section 2 and our new perspective on delegation-restricted HIBE, we lay the foundation for future work that may investigate whether bounded-depth HIBE with limited delegation of $\mathsf{D} = 2$ can be built more efficiently than general bounded-depth HIBE.

Security. To support comprehensibility and avoid idealized assumptions, we first reduce adaptive chosen-plaintext security $\mathrm{IND}^b_{\mathrm{IE}}$ of our UPIBE construction to adaptive security of the underlying HIBE in the standard model. In Section 4.1, we augment our reduction with a new guessing technique that allows us to trade the strength of the underlying HIBE (only selective security instead of adaptive security) against idealized assumptions (random oracle model instead of standard model). Relying only on selective secure HIBEs for adaptive security of our UPIBE significantly extends the class of available HIBE constructions from the literature. For full security against chosen-ciphertext attacks, we consider different generic and direct techniques in Section 5.

Theorem 2. *Unbounded-depth UPIBE protocol* IE *from Figure 3 offers adaptive key indistinguishability in the standard model. More precisely, for every adaptive chosen-plaintext adversary $\mathcal{A}$ attacking protocol* IE *in games* $\mathrm{IND}^b_{\mathrm{IE}}$ *according to Definition 1 with parameters* $(\mathsf{L} = \infty, \lambda, \mathsf{D} = 1)$*, there exists an adaptive chosen-plaintext adversary $\mathcal{B}$ attacking bounded-depth HIBE* IE' *in games* $\mathrm{IND}^b_{\mathrm{IE}'}$ *according to Definition 1 with parameters* $(\mathsf{L}', \lambda', \mathsf{D}') = (\lceil \log(2^\kappa/\varepsilon) \rceil + \varepsilon, \lambda + 1, 2)$ *such that* $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) \leq q_{\mathrm{Gen}} \cdot q_{\mathrm{Chall}} \cdot \lceil l_{\mathrm{long}}/\varepsilon \rceil \cdot \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}'}(\mathcal{B})$*, where κ is the security parameter, ε is the construction's epoch-depth parameter, q_{Chall} and q_{Chall} are the numbers of queries to oracles* Gen *and* Chall *by adversary $\mathcal{A}$, respectively, l_{long} is the level-depth of the longest identity string queried to oracle* Chall *by adversary $\mathcal{A}$, and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.*

Security Proof Overview. Our security proof for Theorem 2 is very similar to the one for Theorem 1. The major technical difference is that here the security of each UPIBE instance is reduced to only one bounded-depth HIBE instance's security. Reduction $\mathcal{B}$, again, simulates all oracle queries of adversary $\mathcal{A}$ in game $\mathrm{IND}_{\mathrm{IE}}$ via queries to oracles in game $\mathrm{IND}_{\mathrm{IE}'}$. As in our proof from Section 3, for certain UPIBE challenge queries to oracle $\mathrm{Chall}_{\mathrm{IE}}$, the reduction has to guess where to embed underlying HIBE challenges of game $\mathrm{IND}_{\mathrm{IE}'}$. A hybrid argument that implements theses guesses cause the loss factor in our advantage bound. The general strategy for embedding challenges is to determine where the identity string input of oracle $\mathrm{Chall}_{\mathrm{IE}}$ branches off the delegation path of (potentially) exposed UPIBE decapsulation keys. In contrast to our proof of Theorem 1, reduction $\mathcal{B}$ here only needs to guess the *epoch of the sub-string* in which the identity strings of challenge and exposure branch lie. We provide our formal proof in the full version [38].

4.1 Relaxing Assumptions: Adaptive UPIBE from Selective HIBE

The above outlined standard model proof for our unbounded-depth UPIBE construction from Figure 3 relies on *adaptive* secure bounded-depth HIBE. Yet, the most suitable bounded-depth HIBEs (e.g., [5]) are only *selective* secure. Generic techniques for turning selective secure schemes adaptive secure, as done in [4, 5, 1, 10], rely on the random oracle model and induce an exponential loss factor in the HIBE's maximal level depth L. The simple idea of these techniques is to replace each identity sub-string in the construction by the output of a random oracle evaluated on this identity sub-string (i.e., $id_0 \| \dots \| id_l$ is replaced by $\mathrm{H}(id_0) \| \dots \| \mathrm{H}(id_l)$). The reduction then embeds sub-strings of the selective challenge identity in randomly chosen random-oracle-outputs. A reduction succeeds if it embeds the *selective challenge* sub-strings in those random-oracle-outputs whose input identity sub-strings form the *adaptive challenge*. This induces an exponential loss in the maximal number of identity sub-strings per adaptive challenge. This is problematic because our UPIBE construction relies on an adaptive secure bounded-depth HIBE with parameter $\mathsf{L} = \alpha + \varepsilon = \lceil \log(2^\kappa / \varepsilon) \rceil + \varepsilon$, which is linear in the security parameter κ. Thus, the loss factor would be exponential in κ when following the generic approach of turning the underlying HIBE adaptive secure [4, 5, 1, 10] before using this HIBE to instantiate our unbounded-depth UPIBE construction.

Solution: Guessing Essentials Only. Due to the way our construction makes use of the underlying bounded-depth HIBE, we can carefully change the generic approach from [4, 5] in order to relax the assumption on the HIBE from adaptive to selective security. Our main observation is that the two (virtual) components in our UPIBE construction—*epoch-progression mechanism* and *multi-level epochs*—encode information of different density. For this, consider an HIBE identity string to which our UPIBE encapsulation internally issues an HIBE encapsulation. The first part of such an HIBE identity string encodes an integer that represents the epoch number in the upper epoch-progression mechanism. The second part encodes a sub-string of the actual UPIBE identity string (i.e., the identity sub-string for one epoch).

In order to embed a selective HIBE challenge in the adaptive UPIBE challenge, our reduction has to predict the branching epoch's full HIBE identity string in advance. In this epoch, the UPIBE challenge identity branches off the delegated identity of the corresponding (exposed) UPIBE decapsulation key. To predict this epoch's full HIBE identity string, we treat the two parts—epoch number and sub-string of UPIBE identity—differently. The branching *epoch number* can simply be guessed with high probability. The reason is that polynomially bounded users (and adversaries) only issue UPIBE identity strings of polynomial length. Thus, also the number of epochs used to represent a UPIBE identity string is polynomially bounded. To predict the second part of the HIBE identity string—the branching epoch's sub-string of the actual UPIBE identity string—we employ the generic technique [4, 5] based on the random oracle model. Since the depth of each multi-level epoch is bounded by constant parameter ε, the loss induced by this technique is only polynomial (not exponential) in κ.

Concrete Adjustments. We interpose a random oracle H in the following lines of our construction in Figure 3: 18: $id' \xleftarrow{\shortparallel} \mathrm{H}(id_{e' \cdot \varepsilon + d'} \| \mathtt{1})$; 19: $id' \xleftarrow{\shortparallel} \mathrm{H}(id_{e' \cdot \varepsilon + \varepsilon - 1} \| \mathtt{0})$;

26: $id' \xleftarrow{\shortparallel} \mathrm{H}(id_{(e-1)\cdot\varepsilon+d'}\|\mathtt{1})$; 56: $D_{ep}[e] \leftarrow_{\$} \mathrm{IE.del}'(D_{ep}[e], \mathrm{H}(id\|\mathtt{1}))$; 58: $dk'_0 \leftarrow_{\$} \mathrm{IE.del}'(D_{ep}[e], \mathrm{H}(id\|\mathtt{0}))$; 59: $dk'_1 \leftarrow_{\$} \mathrm{IE.del}'(D_{ep}[e], \mathrm{H}(id\|\mathtt{1}))$. However, we leave the identity sub-strings of the upper epoch-progression mechanism untouched. Thus, lines 03-04, 16, 24, and 51-52 remain the same. A full proof of following Theorem 3 is given in the full version [39].

Theorem 3. *Adjusting unbounded-depth UPIBE protocol* IE *from Figure 3 offers adaptive key indistinguishability in the random oracle model. More precisely, let* H *be a random oracle, then for every adaptive chosen-plaintext adversary* $\mathcal{A}$ *attacking protocol* IE *in games* $\mathrm{IND}^b_{\mathrm{IE}}$ *according to Definition 1 with parameters* $(\mathsf{L} = \infty, \lambda, \mathsf{D} = 1)$*, there exists a* selective *chosen-plaintext adversary* $\mathcal{B}$ *attacking bounded-depth HIBE* IE' *in games* $\mathrm{IND}^b_{\mathrm{IE}'}$ *according to Definition 1 with parameters* $(\mathsf{L}', \lambda', \mathsf{D}') = (\lceil \log(2^\kappa/\varepsilon) \rceil + \varepsilon, \lambda + 1, 2)$ *such that* $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) \leq q_{\mathrm{Gen}} \cdot q_{\mathrm{Chall}} \cdot ((l_{\mathrm{long}})^2 \cdot (q_{\mathrm{H}})^{\varepsilon}) \cdot \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}'}(\mathcal{B})$*, where* κ *is the security parameter,* ε *is the construction's epoch-depth parameter,* q_{Gen}*,* q_{Chall}*, and* q_{H} *are the number of queries to oracles* Gen*,* Chall *and the random oracle by adversary* $\mathcal{A}$*, respectively,* l_{long} *is the level-depth of the longest identity string queried to oracle* Chall *by adversary* $\mathcal{A}$*, and the running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

5 CCA Secure UPIBE

Now we turn our focus on the task of achieving chosen-ciphertext security for bounded- and unbounded-depth UPIBE. While it might be tempting to think that similar to HIBEs one could generically convert CPA-secure UPIBE into CCA-secure ones using the BCHK (often also called CHK) compiler [9, 6], this unfortunately does not work: BCHK needs one delegation *per* decapsulation from *the same* decapsulation key, but UPIBE only offers one delegation for each decapsulation key *in total*. Thus, we need to adopt different strategies for constructing CCA-secure UPIBE.

5.1 Bounded-depth UPIBE

FO-Transform. Having in mind that we construct bounded-depth UPIBE from (bounded-collusion) IBE, a natural choice is to apply the Fujisaki-Okamoto (FO) transform [19] and in particular one of its modular variants [28]. FO typically considers single instances, but in our construction of UPIBE one has to deal with multiple parallel IBE ciphertexts and this requires some care. Recently, Cini et al. in [11] considered this issue of parallel ciphertexts in FO for reducing decryption errors as well as constructing Bloom-Filter KEMs (BFKEMs) from IBE. Though [11] relies on a single IBE instance, it is quite straightforward to adapt their approach to UPIBE.[11] Unfortunately, using FO in this way, besides being bound to the random oracle model (ROM), requires an overhead of l encryptions of the underlying IBE during decapsulation, which can be significant.

[11] We would sample a random key k and derive $(r_0, \ldots, r_{l-1}, k') = G(k)$ from a random oracle G and encapsulate k_i with randomness r_i for the i'th instance such that $K = k_0 \oplus \ldots \oplus k_{l-1}$ and then use k' as the overall encapsulation key.

Split-Key PRF. An alternative, more efficient, and more flexible approach is made possible when we view our UPIBE construction in Section 3 as parallel bounded-collusion IBE and take inspiration from Giacon et al. [22]. In particular, recall that our overall ciphertext $C = c'_0 \| \ldots \| c'_{l-1}$ is the concatenation of l ciphertexts of independent IBEs and the encapsulation key is computed as $K \leftarrow \mathrm{W}(k'_0, \ldots, k'_{l-1}, C)$, where W represents what is called a core function by Giacon et al. [22]. We note that [22] focuses on parallel KEM combiners, and show that if W is a split-key pseudorandom function (skPRF), it yields a CCA-secure KEM if at least one of the l KEMs is CCA secure. Various instantiations of skPRFs in the ROM and standard model with different types of trade-offs are discussed in [22]. For instance the PRF-then-XOR composition $\mathrm{W}(k'_0, \ldots, k'_{l-1}, C) := \bigoplus_{i=0}^{l-1} F_i(k'_i, C)$, where F_i's are PRFs, is a skPRF in the standard model. Our focus now is not on combiners and as the use of our instances is dynamic (i.e., the depth can vary), this does not work for UPIBE. Here we need to require that *all* instances are CCA secure. Nevertheless, as we discuss below, the use of an skPRF still gives advantages when it comes to standard model constructions.

Achieving CCA-secure IBE. While CCA security can be easily achieved in the ROM by starting from a CPA-secure (bounded-collusion) IBE and applying the FO transform, the overall overhead due to the FO is identical when directly applying FO (as discussed above). However, we can obtain CCA-secure bounded-depth UPIBE in the standard model when relying on an IBE scheme that directly provides CCA security in the standard model (e.g,. [20] or the CCA-secure version of the bounded-collusion IBE in [15]). Alternatively, if one accepts that the IBEs are replaced by CPA-secure depth 2 HIBEs, one can simply use the BCHK compiler [9, 6].

Now, we will show that the bounded-depth UPIBE protocol from Figure 2 is CCA-secure when the underlying bounded-collusion IBE IE' is CCA-secure (e.g., [15]) and the core function W is based on a split-key pseudorandom function F with $n = \mathsf{L}$ (cf. the full version [38] for the definition). For reasons that we will discuss below, we include a special KEM key $\hat{k}$ and a special ciphertext $\hat{c}$ into ek of the UPIBE protocol IE in order to "pad" calls to W to always take L inputs (for all cases where depth $l < \mathsf{L}$).

Theorem 4. *Bounded-depth UPIBE protocol* IE *from Figure 2 offers adaptive key indistinguishability under chosen-ciphertext attacks in the standard model. More precisely, for every adaptive chosen-ciphertext adversary $\mathcal{A}$ attacking protocol* IE *in games* $\mathrm{IND}^b_{\mathrm{IE}}$ *according to Definition 1 with parameters* $(\mathsf{L}, \lambda, \mathsf{D} = 1)$, *there exists an adversary $\mathcal{B}_{\mathrm{G}}$ attacking PRG* G, *an adversary $\mathcal{B}_{\mathrm{W}}$ against the split-key pseudorandomness of* W, *and an adaptive chosen-ciphertext adversary $\mathcal{B}_{\mathrm{IE}'}$ attacking bounded-collusion IBE* IE' *in games* $\mathrm{IND}^b_{\mathrm{IE}'}$ *according to Definition 1 with parameters* $(\mathsf{L}', \lambda', \mathsf{D}') = (1, \lambda{+}1, 2)$ *such that* $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) \leq q_{\mathrm{Gen}} \mathsf{L}^2 \cdot \Big(q_{\mathrm{Gen}} q_{\mathrm{Chall}} \mathsf{L}\, \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{G}}(\mathcal{B}_{\mathrm{G}}) + 1\Big) + 2 q_{\mathrm{Gen}} q_{\mathrm{Chall}} \mathsf{L} \cdot \Big(q_{\mathrm{Chall}}\, \mathrm{Adv}^{\mathsf{pr}}_{F_i}(\mathcal{B}_{\mathrm{W}}) + \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}'}(\mathcal{B}_{\mathrm{IE}'})\Big)$, *where q_{Chall} and q_{Gen} are the number of queries to oracle* Chall *and* Gen *by adversary $\mathcal{A}$, and the running times of $\mathcal{B}_{\mathrm{G}}$, $\mathcal{B}_{\mathrm{W}}$, and $\mathcal{B}_{\mathrm{IE}'}$ is about that of $\mathcal{A}$.*

Security Proof Overview. The strategy for the proof is analogous to that of Theorem 1, but we will proceed in a sequence of Games moving from the game $\mathrm{IND}^0_{\mathrm{IE}}$ to $\mathrm{IND}^1_{\mathrm{IE}}$, which allows us to follow the strategy by Giacon et al. [22]. In contrast to their proof, in our case all instances are required to be CCA secure. This is since we require CCA security of the underlying IBE IE' at the branching positions of identities that are asked to the challenge oracle, which can be placed at any of the L positions adaptively. We need to take some care when using the pseudorandomness of the split-key pseudorandom function for W, as we use $n = \mathsf{L}$ but the number of required inputs vary with the actual depth of the identities l. Therefore, we always use L inputs for calls to W where for the $\mathsf{L} - l$ rightmost inputs we simply use a fixed key $\hat{k}$ and ciphertext $\hat{c}$ (we will not make this fact explicit in the proof). We provide a formal proof in the full version [38].

5.2 Unbounded-depth UPIBE

For the same reasons as discussed in Section 5.1 we prefer to avoid a generic use of the FO transform for proving CCA security of our unbounded-depth UPIBE. Unfortunately, the generic skPRF approach pursued in Section 5.1 requires an a priori bound on the depth, which is not the case for unbounded-depth UPIBE.

Consequently, although we follow the same overall idea, as already mentioned in Figure 3, we instantiate the core function W directly by a random oracle $\mathrm{H}^\star$, i.e., derive the overall key as $K \leftarrow \mathrm{H}^\star(k_1, \ldots, k_l, c_1, \ldots, c_l)$ where (k_i, c_i) are the encapsulation outputs of the chosen-ciphertext secure bounded HIBE. Since our focus is on efficiency, and the strategy to prove Theorem 3 already requires the ROM, this seems to be a meaningful choice. For CCA security of the single ciphertexts of the underlying bounded-depth HIBE, the most efficient approach is a use of the BCHK compiler [9, 6]. This yields a very flexible approach as due to the choice of the required strongly secure signature scheme there are many performance and bandwidth trade-offs available (see also Section 7). Using this strategy we can show the following for our unbounded-depth UPIBE. The proof of Theorem 5 is provided in the full version [38].

Theorem 5. *Adjusting unbounded-depth UPIBE protocol* IE *from Figure 3 as described in Section 4.1 offers adaptive key indistinguishability under chosen-ciphertext attacks in the random oracle model. More precisely, let* H *and* $\mathrm{H}^\star$ *be random oracles, then for every adaptive chosen-ciphertext adversary $\mathcal{A}$ attacking protocol* IE *in games* $\mathrm{IND}^b_{\mathrm{IE}}$ *according to Definition 1 with parameters* $(\mathsf{L} = \infty, \lambda, \mathsf{D} = 1)$*, there exists a selective chosen-ciphertext adversary $\mathcal{B}$ attacking bounded-depth HIBE* IE' *in games* $\mathrm{IND}^b_{\mathrm{IE}'}$ *according to Definition 1 with parameters* $(\mathsf{L}', \lambda', \mathsf{D}') = (\lceil \log(2^\kappa/\varepsilon) \rceil + \varepsilon, \lambda + 1, 2)$ *such that* $\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{A}) \leq q_{\mathrm{Gen}} \cdot q_{\mathrm{Chall}} \cdot ((l_{\mathrm{long}})^2 \cdot (q_{\mathrm{H}})^\varepsilon) \cdot \left(\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}'}(\mathcal{B}) + \frac{q_{\mathrm{Chall}} \cdot q_{\mathrm{H}^\star}}{|\mathcal{K}|}\right)$ *where κ is the security parameter, ε is the construction's epoch-depth parameter,* q_{Chall}*,* q_{Gen}*,* q_{H} *and* $q_{\mathrm{H}^\star}$ *are queries to oracles* Chall*,* Gen *and random oracles* H *and* $\mathrm{H}^\star$ *by adversary $\mathcal{A}$, respectively,* l_{long} *is the level-depth of the longest identity string queried to oracle* Chall *by adversary $\mathcal{A}$, and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.*

6 Key-Updatable KEM from UPIBE

A Key-Updatable Key Encapsulation Mechanism (KU-KEM) [30, 37] is a KEM $\text{K} = (\text{K.gen}, \text{K.enc}, \text{K.dec}, \text{K.up})$ with additional update algorithms K.up for encapsulation keys and decapsulation keys. The computation of each update $ek' \leftarrow_\$ \text{K.up}(ek, ad)$ resp. $dk' \leftarrow_\$ \text{K.up}(dk, ad)$ is determined by a bit string ad that is arbitrarily chosen by the user. One can think of these update bit strings as new information (aka. *associated data*) that is added to the context of the ongoing session. Updates of encapsulation keys and decapsulation keys can be conducted independently without information being transmitted between holders of encapsulation and decapsulation key. The feature of *independent* updates *with respect to bit strings* constitutes the crucial difference to significantly weaker notions like Updatable PKE [31, 14] that offer more efficient instantiations. We refer the interested reader to a discussion by Balli et al. [3] who elaborate on the shortcomings of Updatable PKE in the context of *strongly* secure messaging.

As long as both components of a KU-KEM key pair are updated with respect to the same bit strings—meaning, their context is updated compatibly—, the key pair remains compatible. More precisely, a generated pair consisting of encapsulation key and decapsulation key remains compatible if the list of bit strings for updates applied on the encapsulation key equals the list of bit strings for updates applied on the decapsulation key. We follow the slightly stronger variant of KU-KEM by Balli et al. [3] that furthermore requires for compatibility of a key pair that the list of bit strings for updates together with the list of sent and received encapsulation ciphertexts equals on both sides.

For security of KU-KEM, two goals beyond pure key-indistinguishability are required: (1) *Forward-secrecy*, meaning that an updated future version of the current decapsulation key can be exposed to an adversary without harming confidentiality of ciphertexts produced with a current or previous (compatible) version of the corresponding encapsulation key—in short, old ciphertexts remain secure if future decapsulation keys are exposed; (2) *Effective divergence*, meaning that an incompatible decapsulation key can be exposed to an adversary without harming confidentiality of ciphertexts produced with the corresponding (incompatible) encapsulation key—in short, any difference in update bit strings makes encapsulation key and decapsulation key fully independent.

KU-KEM is a special form of UPIBE where KU-KEM update bit strings are implemented via UPIBE identity sub-strings, KU-KEM decapsulation key updates are realized via UPIBE delegations, and KU-KEM encapsulation key updates are realized via UPIBE derivations. The construction of KU-KEM from UPIBE is, therefore, straight forward: $\text{K.gen} \coloneqq \text{IE.gen}$; $\text{K.up}(ek, ad) \coloneqq \text{IE.der}(ek, ad = id)$ resp. $\text{K.up}(dk, ad) \coloneqq \text{IE.del}(dk, ad = id)$; $\text{K.enc}(ek)$ executes $\text{IE.enc}(ek, \epsilon)$ and updates ek via $\text{IE.der}(ek, ad = c)$; $\text{K.dec}(dk, c)$ executes $\text{IE.dec}(dk, c)$ and updates dk via $\text{IE.del}(dk, ad = c)$. (Pseudo-code is given in the full version [38].) This construction was first proposed by Poettering and Rösler [37] and slightly adapted in other works [30, 3]. Yet, we are the first to reduce the underlying assumption from general unbounded-depth HIBE to unbounded-depth UPIBE. For space reasons, we defer the formal definition of KU-KEM by Balli et al. [3]

as well as our proof of Theorem 6 to the full version [38]. This proof tightly reduces the security of the KU-KEM construction to adaptive chosen-ciphertext security of the underlying unbounded-depth UPIBE scheme.

Theorem 6. *KU-KEM protocol* K *offers one-wayness of encapsulated keys. More precisely, for every adaptive chosen-ciphertext adversary* $\mathcal{A}$ *attacking protocol* K*, there exists an adaptive chosen-ciphertext adversary* $\mathcal{B}$ *attacking unbounded-depth HIBE* IE *in games* $\mathrm{IND}^b_{\mathrm{IE}}$ *according to Definition 1 with parameters* $(\mathsf{L}', \lambda', \mathsf{D}') = (\infty, \lambda, 1)$ *such that* $\mathrm{Adv}^{\mathrm{kuow}}_{\mathrm{K}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{ind}}_{\mathrm{IE}}(\mathcal{B})$*, where the running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

7 Evaluation

Our evaluation considers (asymptotic and concrete) parameter sizes of one-way CCA (formally, KUOW) secure KU-KEMs built *trivially* from unbounded-depth HIBEs on the one side and KU-KEMs based on our UPIBE construction that relies on bounded-depth HIBEs from Section 5.2 on the other side. Before starting the concrete analysis, we note that CCA security of (un)bounded-depth HIBEs can be generically achieved efficiently via the BCHK transform [9, 6] using a strongly secure one-time-signature scheme.[12]

Since we have applicability and performance in mind for our application towards optimally secure messaging protocols, we include bounded-depth HIBE schemes that are secure in the random-oracle model (ROM). Moreover, we looked at all applicable unbounded-depth HIBEs and selected three constructions [35, 34, 24] that suit the application we have in mind best. Depending on the concrete bounded-depth HIBE scheme, it is a common technique to reduce public parameter sizes in the ROM [5]. This, however, does not work generically. Particularly, in the HIBE scheme by Gong et al. (GCTC) [24], the underlying encapsulation key structure seemingly prevents this form of parameter compression. The same seems to be the case for Langrehr-Pan (LP) [34], while Lewko (L) [35] already has compact encapsulation keys (however, with a large constant).

For our KU-KEM construction via the UPIBE paradigm (where we only require a selectively secure HIBE with polynomially bounded depth), the strongest candidate is the Boneh-Boyen-Goh (BBG) HIBE [5]. Here, encapsulation key size is only two group elements using the ROM. However, we cannot utilize the ROM to reduce the size of BBG decapsulation keys since these keys require a certain structure. Hence, the BBG HIBE has linear-size decapsulation keys, but enjoys constant-size encapsulation keys *and* ciphertexts (all in the maximal depth).

By considering the most efficient (un)bounded-depth HIBE schemes, we conduct a fair comparison between KU-KEMs from trivial UPIBE via unbounded-depth HIBE and KU-KEMs from our novel UPIBE construction. In Table 1, we list CCA secure KU-KEMs from CCA secure (un)bounded-depth HIBEs with relevant size and performance parameters.

[12] In our concrete setting, for standard-model HIBEs, we use Groth's pairing-free signature scheme [26] while for HIBEs in the ROM, we use Schnorr signatures [40].

UPIBE	Via HIBE	Encapsulation key size	Ciphertext size	Decapsulation key size	Model
Triv.	L [35]	$60\|G_1\| + 2\|G_T\| + l\lambda$	$(10l + 12)\|G_1\|$	$(10l + 60)\|G_2\|$	StM
Triv.	LP [34]	$(2\gamma + 4)\|G_1\| + (2\gamma + 6)\|G_2\| + l\lambda$	$(7l + 11)\|G_1\|$	$(7l + 2)\|G_2\|$	StM
Triv.	GCTC [24]	$(3n + 9 + \lceil l/n \rceil)\|G_1\| + 3\|G_T\|$	$(9\lceil (l + 1)/n \rceil + 2)\|G_1\|$	$((9 + 3n)\lceil l/n \rceil + 3n + 9 - 3l)\|G_2\|$	StM
Ours	BBG [5]	$(1 + \lceil l/\varepsilon \rceil)\|G_1\| + 1\|G_2\|$	$(3\lceil l/\varepsilon \rceil)\|G_1\|$	$(\mathcal{O}(\alpha \cdot (\alpha + \varepsilon)) + \lceil l/\varepsilon \rceil + \alpha)\|G_2\|$	ROM

UPIBE	Via HIBE	Key generation (# exp.)	Encapsulation (# exp.)	Decapsulation (# exp., # pairings)	Ass.
Triv.	L [35]	$60\ (G_1)$, $80\ (G_2)$, $2\ (G_T)$	$60l + 62\ (G_1)$, $2\ (G_T)$	$(61l\ (G_2), 10l + 1)$	DLIN
Triv.	LP [34]	$(2\gamma + 4)\ (G_1)$, $(2\gamma + 6)\ (G_2)$	$(7l + 11)\ (G_2)$, $2\ (G_T)$	$((7(l + 1) + 2)\ (G_2), (7l + 2) + 1)$	SXDH
Triv.	GCTC [24]	$6(n + 3)\ (G_2)$, $1\ (G_T)$	$(15\lceil l/n \rceil + 3l)\ (G_1)$, $3\ (G_T)$	$(15\lceil l/n \rceil + 3l\ (G_2), 9\lceil l/n \rceil + 1)$	SXDH
Ours	BBG [5]	$1\ (G_1)$, $1\ (G_2)$	$((\lceil l/\varepsilon \rceil + 5)\ (G_1)$, $1\ (G_T)$	$(\varepsilon + \alpha/\varepsilon + 2)\ (G_2)$, $2\lceil l/\varepsilon \rceil)$	BDHE

Table 1: Comparison of CCA secure KU-KEMs with parameter sizes and performance instantiated from the standard-model unbounded-depth HIBEs L [35], LP [34], and GCTC [24] (trivially) and the bounded-depth HIBE BBG [5] (via our KU-KEM-from-UPIBE approach from Section 6). Here, $\alpha + \varepsilon$ is the maximum level (and α can be considered linear in the security parameter), l is the current number of key updates, γ is the output bit length of a collision-resistant hash function, and ε is the epoch-depth in our UPIBE. $n \geq 1$ is the performance parameter of GCTC [24]. We use the type-3 pairing setting with $e : G_1 \times G_2 \rightarrow G_T$ for prime-order groups G_1, G_2, and G_T. Here, we do not consider the tightness of the reductions to the underlying assumptions.

We see that all but one known trivial KU-KEM instantiations via [35, 24, 34] have ciphertext and decapsulation-key sizes that scale linearly in the number of delegations (which corresponds to KU-KEM key updates). Only GCTC [24] has a trade-off for ciphertext and key sizes via their performance parameter n. With our non-trivial UPIBE approach from bounded-depth HIBEs, taking the BBG scheme [5] as instantiation, we obtain ciphertext sizes that only scale linearly in the number of *epochs*, which can be adjusted by the depth-parameter ε as described in Section 4. Moreover, our KU-KEM approach via BBG enjoys very short encapsulation keys. This yields a significant reduction in encapsulation key and ciphertext sizes for KU-KEMs compared to other approaches (see Table 1).

Detailed Analysis. For our following analysis concerning parameter sizes and performance, from the three trivial standard-model KU-KEMs based on unbounded-depth HIBEs [35, 24, 34], we chose GCTC [24] which outperforms the other two—particularly because of their scalability parameter n that allows to trade-off ciphertext and encapsulation/decapsulation key sizes.[13] Hence, the GCTC scheme is the best suitable reference instantiation of KU-KEM via the *trivial* UPIBE construction for a concrete comparison regarding the applications we have in mind.

Application Requirements. Our focus is on short ciphertexts and encapsulations keys (for bandwidth reasons) while on the sender and the receiver sides, we

[13] Essentially, GCTC [24] improves Lewko [35] towards shorter ciphertext sizes and LP [34] deals with tightness of the Lewko scheme [35], at the expense of rather large encapsulation keys (see γ-factor).

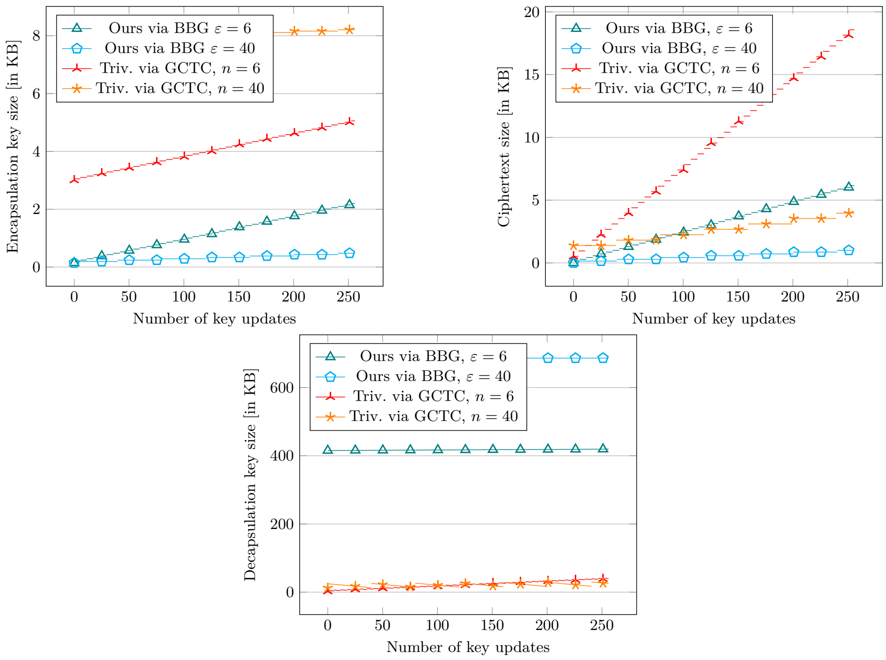

Fig. 4: Comparison of CCA secure KU-KEM encapsulation and decapsulation key as well as ciphertext sizes in kilobytes (KB) from (un)bounded-depth HIBEs. For the pairing group, we chose BLS12-381 (which gives around 128 bit security); this means per element in G_1, G_2, and G_T, we have 382, 764, 4572 bits.

want fast encapsulation and fast decapsulation, respectively. As we argue now, our non-trivial UPIBE approach with BBG outperforms the trivial KU-KEM construction with GCTC in all of the metrics mentioned above. We recall that our KU-KEM decapsulation is based on the actual ciphertext decapsulation and an additional key delegation of the underlying HIBE. Moreover, we can compress the identity string via algorithm IE.der to compute an identity-specific encapsulation key for BBG *and* GCTC. We currently do not see how to perform this compression for [35, 34].

Bandwidth Comparison. We observe that the performance parameter n in GCTC plays a similar role as our depth parameter ε in UPIBE; hence, we compare it at the same level. As illustrative examples, we choose $\varepsilon = n = 6$ and $\varepsilon = n = 40$. From the graphs in Figures 4 and 5, we see that the encapsulation key for the BBG-based KU-KEM is very short. The ciphertext size of all KU-KEMs scales with ε and n. Our BBG-based approach has the shortest ciphertext sizes of all. For decapsulation key sizes, the GCTC approach is more efficient; however, as we argued with the application of secure messaging in mind, this is tolerable. Hence, concerning parameter sizes, we conclude that the BBG approach has shorter ciphertexts and smaller encapsulation key at the expense of slightly larger decapsulation keys compared to the trivial GCTC-based KU-KEM approach.

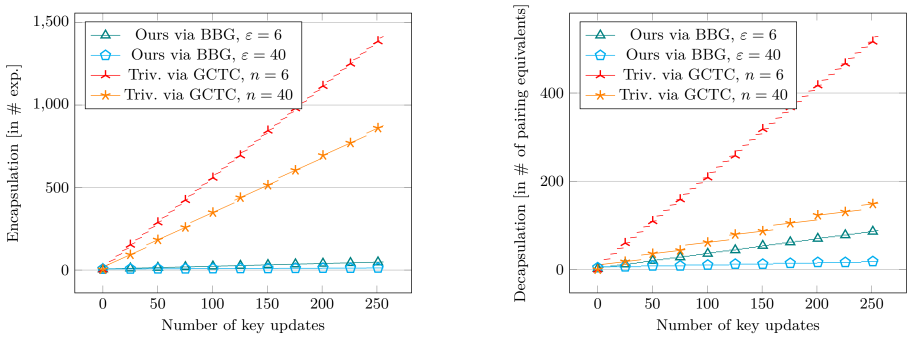

Fig. 5: Comparison of CCA secure KU-KEM key generation, encapsulation, and decapsulation performance from un-/bounded-depth HIBEs. We estimate that a G_1 exponentiation is 10 times more efficient than a pairing.

Computation Comparison. In terms of computation complexity (Figure 5), we see that the BBG approach significantly outperforms the GCTC-based approach for encapsulation and decapsulation. The (initial) key generation for the BBG-based and for GCTC-based approaches are comparable efficiency-wise and constant in the number of key updates; our approach needs α many exponentiations while GCTC's number of exponentiations scales linearly in their performance parameter n. For encapsulation and decapsulation (where latter uses key delegation *and* decryption of the underlying HIBE), the BBG-based KU-KEM is more efficient; particularly, in situations when a large number of key updates is needed. See that the larger ε, the more efficient is the decapsulation of the BBG-based KU-KEM approach. The reason is that the BBG HIBE ciphertexts are of constant size and need only a constant number of pairings per ciphertext for decryption.

Summary. In conclusion, a KU-KEM via our unbounded-depth UPIBE construction, instantiated with the BBG HIBE, has shorter ciphertext and encapsulation-key sizes compared to the GCTC-based solution with analogous parameter choices (being the most efficient unbounded-depth HIBE known for trivial UPIBE) at the expense of a slightly larger decapsulation key. Additionally, the decapsulation and, particularly, the encapsulation of the BBG-based KU-KEM are significantly more efficient compared to the GCTC-based trivial KU-KEM. Hence, for our envisioned application of strongly secure messaging, we can tolerate slightly larger decapsulation keys while achieving more efficient decapsulation and encapsulation as those operations happen rather often in KU-KEMs.

Acknowledgements. This work was supported by the ECSEL Joint Undertaking (JU) under grant agreement No 826610 (Comp4Drones) and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (Profet).

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_28
2. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 129–158. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_5
3. Balli, F., Rösler, P., Vaudenay, S.: Determining the core primitive for optimally secure ratcheting. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 621–650. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_21
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_14
5. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_26
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing **36**(5), 1301–1328 (2007)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_16
9. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_13
10. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_27
11. Cini, V., Ramacher, S., Slamanig, D., Striecks, C.: CCA-secure (puncturable) KEMs from encryption with non-negligible decryption errors. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 159–190. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_6
12. Derler, D., Jager, T., Slamanig, D., Striecks, C.: Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 425–455. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_14
13. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) ACM CCS-9 DRM Workshop 2002 (2002)
14. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III (2021)

15. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_5
16. Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 372–408. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_13
17. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_18
18. Durak, F.B., Vaudenay, S.: Bidirectional asynchronous ratcheted key agreement with linear complexity. In: Attrapadung, N., Yagi, T. (eds.) IWSEC 19. LNCS, vol. 11689, pp. 343–362. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26834-3_20
19. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
20. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_27
21. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (Dec 2002). https://doi.org/10.1007/3-540-36178-2_34
22. Giacon, F., Heuer, F., Poettering, B.: KEM combiners. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 190–218. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76578-5_7
23. Goldwasser, S., Lewko, A.B., Wilson, D.A.: Bounded-collusion IBE from key homomorphism. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 564–581. Springer, Heidelberg (Mar 2012). https://doi.org/10.1007/978-3-642-28914-9_32
24. Gong, J., Cao, Z., Tang, S., Chen, J.: Extended dual system group and shorter unbounded hierarchical identity based encryption. Des. Codes Cryptogr. 2016 **80**(3), 525–559 (2016)
25. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: 2015 IEEE Symposium on Security and Privacy. pp. 305–320. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.26
26. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
27. Günther, F., Hale, B., Jager, T., Lauer, S.: 0-RTT key exchange with full forward secrecy. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 519–548. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56617-7_18
28. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
29. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_31

30. Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: The safety of messaging. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_2
31. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 159–188. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_6
32. Jost, D., Maurer, U., Mularczyk, M.: A unified and composable take on ratcheting. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 180–210. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_7
33. Katz, J.: Binary tree encryption: Constructions and applications. In: Lim, J.I., Lee, D.H. (eds.) ICISC 03. LNCS, vol. 2971, pp. 1–11. Springer, Heidelberg (Nov 2004)
34. Langrehr, R., Pan, J.: Unbounded HIBE with tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 129–159. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_5
35. Lewko, A.B.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_20
36. Poettering, B., Rösler, P.: Asynchronous ratcheted key exchange. Cryptology ePrint Archive, Report 2018/296 (2018), https://eprint.iacr.org/2018/296
37. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 3–32. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_1
38. Rösler, P., Slamanig, D., Striecks, C.: Unique-path identity based encryption with applications to strongly secure messaging. Cryptology ePrint Archive, Paper 2023/248 (2023), https://eprint.iacr.org/2023/248, https://eprint.iacr.org/2023/248
39. Rösler, P., Slamanig, D., Striecks, C.: Unique-path identity based encryption with applications to strongly secure messaging. In: Advances in Cryptology - EUROCRYPT 2023 - 42th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2023 Proceedings. Lecture Notes in Computer Science, Springer (2023)
40. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
41. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (Aug 1984)
42. Tessaro, S., Wilson, D.A.: Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 257–274. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_15