Hierarchical Identity-Based (Lossy) Trapdoor Functions

Alex Escala[1], Javier Herranz[1], Benoît Libert[2], and Carla Ràfols[3]

[1] Universitat Politècnica de Catalunya, Dept. Matemàtica Aplicada IV (Spain)
e-mail: {alex.escala,jherranz}@ma4.upc.edu
[2] Université catholique de Louvain, ICTEAM Institute – Crypto Group (Belgium)
e-mail: benoit.libert@uclouvain.be
[3] Ruhr-Universität Bochum, Horst Görtz Institut für IT-Sicherheit (Germany)
e-mail: Carla.Rafols@rub.de

Abstract. Lossy trapdoor functions, introduced by Peikert and Waters (STOC'08), have received a lot of attention in the last years, because of their wide range of applications in theoretical cryptography. The notion has been recently extended to the identity-based setting by Bellare *et al.* (Eurocrypt'12). We provide one more step in this direction, by considering the notion of hierarchical identity-based (lossy) trapdoor functions (HIB-TDFs). Hierarchical identity-based cryptography has proved very useful both for practical applications and to establish theoretical relations with other cryptographic primitives.
The notion of security for IB-TDFs put forward by Bellare *et al.* easily extends to the hierarchical scenario, but an (H)IB-TDF secure in this sense is not known to generically imply other related primitives with security against adaptive-id adversaries, not even IND-ID-CPA secure encryption. Our first contribution is to define a new security property for (H)IB-TDFs. We show that functions satisfying this property imply secure cryptographic primitives in the adaptive identity-based setting: these include encryption schemes with semantic security under chosen-plaintext attacks, deterministic encryption schemes, and (non-adaptive) hedged encryption schemes that maintain some security when messages are encrypted using randomness of poor quality. We emphasize that some of these primitives were unrealized in the (H)IB setting previous to this work.
As a second contribution, we describe the first pairing-based HIB-TDF realization. This is also the first example of hierarchical trapdoor function based on traditional number theoretic assumptions: so far, constructions were only known under lattice assumptions. Our HIB-TDF construction is based on techniques that differ from those of Bellare *et al.* in that it uses a hierarchical predicate encryption scheme as a key ingredient. The resulting HIB-TDF is proved to satisfy the new security definition, against either selective or, for hierarchies of constant depth, adaptive adversaries. In either case, we only need the underlying predicate encryption system to be selectively secure.

Keywords. Lossy trapdoor functions, hierarchical identity-based encryption, partial lossiness.

1 Introduction

1.1 (Identity-Based) Lossy Trapdoor Functions

Lossy trapdoor functions, as introduced by Peikert and Waters in [31], have been proved very powerful in theoretical cryptography and received a lot of attention in the recent years (see, e.g., [21, 26, 29, 13, 27, 36]). Roughly speaking, a lossy trapdoor function is a family of functions that can be instantiated in two different modes. In the injective mode, the function is injective and can be inverted using the corresponding trapdoor. In lossy mode, the function is (highly) non-injective since its image size is much smaller than the size of the domain. The key point is that lossy instantiations of the function must be indistinguishable from injective instantiations.

In their seminal paper [31], Peikert and Waters showed that lossy trapdoor functions provide black-box constructions of chosen-ciphertext secure (IND-CCA) public-key encryption schemes as well as universal one-way and collision-resistant hash functions. Later on, other applications of lossy trapdoor functions were discovered: they gave rise to deterministic encryption schemes [5]

in the standard model [10], public-key encryption hedged schemes maintaining some security in the absence of reliable encryption coins [6] and even public-key encryption with selective-opening security [8] (*i.e.*, which offer certain security guarantees in case of sender corruption).

Recently, Bellare, Kiltz, Peikert and Waters [9] introduced the notion of identity-based (lossy) trapdoor function (IB-TDF), which is the analogue of lossy trapdoor functions in the setting of identity-based cryptography [34]. In the identity-based scenario, users' public keys are directly derived from their identities, whereas secret keys are delivered by a trusted master entity. In this way, the need for digital certificates, which usually bind public keys to users in traditional public-key cryptography, is drastically reduced. Throughout the last decade, several generalizations of identity-based cryptography were put forth, including hierarchical identity-based cryptography [22], attribute-based cryptography [32, 23] or predicate-based cryptography [11, 28]. In the setting of hierarchical identity-based cryptography, identities are organized in a hierarchical way, so that a user who holds the secret key of an identity id can generate, use and distribute valid secret keys for any identity that is a descendant of id in the hierarchy. Hierarchical identity-based encryption (HIBE) is of great interest due to both practical and theoretical reasons. On the practical side, many organizations and systems that may need (identity-based) cryptographic solutions are organized in a hierarchical way. On the theoretical side, generic constructions [15, 16] are known to transform a weakly secure HIBE scheme (*i.e.*, IND-CPA security against selective adversaries) into (public-key) encryption schemes with strong security properties, like chosen-ciphertext security [16] or forward-security [4, 15], where private keys are updated in such a way that past encryptions remain safe after a private key exposure.

Bellare *et al.* [9] proposed instantiations of identity-based lossy trapdoor functions based on bilinear maps and on lattices (as noted in [9], almost all IBE schemes belong to these families). The former makes clever use of an anonymous IBE system (where the ciphertext hides the receiver's identity) with pseudorandom ciphertexts whereas the latter relies on lossiness properties of learning-with-error-based cryptosystems. Moreover, they show that their definition of partial-lossiness for identity-based trapdoor functions leads to the same cryptographic results as lossy trapdoor functions, but in the selective identity-based setting only, where the attacker must choose the target identity upfront in the attack game. Namely, in the case of selective adversaries, IB-TDFs satisfying their definition imply identity-based encryption with semantic security, identity-based deterministic encryption and identity-based hedged encryption. In [9], it was left as an open problem to prove that the same results hold in the case of adaptive adversaries.

1.2 Our Contributions

This paper extends to the hierarchical setting the notion of identity-based (lossy) trapdoor function.

New Definition of Partial Lossiness. From a theoretical standpoint, we first define a new security property for hierarchical identity-based trapdoor functions (HIB-TDFs). We show that a HIB-TDF which satisfies this property can be used to obtain the same kind of results that are derived from standard lossy trapdoor functions [31]. Namely, they lead to standard encryption schemes, to deterministic encryption schemes, and to non-adaptive hedged encryption schemes, which are secure in the hierarchical identity-based setting, in front of adaptive-id adversaries. Since HIB-TDFs contain IB-TDFs as a particular case, our results for adaptive adversaries solve an open problem in [9]. Interestingly, the pairing-based IB-TDF of Bellare *et al.* [9] can be proved to also satisfy the new security property. As a consequence, it provides adaptively secure deterministic and hedged IBE schemes. Recently, Xie *et al.* [37] designed an adaptively secure D-IBE system using

lattices. The construction of [9] is thus the first adaptively secure pairing-based realization.

Construction of a Pairing-Based Hierarchical Trapdoor Functions. On the constructive side, we focus on pairing-based systems where, as already mentioned in [9], greater challenges are faced. Indeed, in the hierarchical scenario, anonymity – which was used as an important ingredient by Bellare *et al.* [9] – has been significantly harder to obtain in the world of bilinear maps than with lattices [17, 1, 2]: for example, the first pairing-based anonymous HIBE system [12] appeared four years after the first collusion-resistant HIBE [22]. Moreover, very few anonymous IBE systems seem amenable for constructing IB-TDFs, as noted in [9] where a new scheme was specially designed for this purpose.

Using bilinear maps, we thus construct a HIB-TDF and prove that it satisfies our new definition of partial lossiness under mild hardness assumptions in groups of prime order. As an intermediate step, we design a hierarchical predicate encryption (HPE) system [33, 30] with suitable anonymity properties, which may be of independent interest. Perhaps surprisingly, although this scheme is proved secure only against weak *selective* adversaries (who select their target attribute set before seeing the public parameters), we are able to turn it into a HIB-TDF providing security (namely, our new version of partial lossiness) against *adaptive* adversaries for hierarchies of constant depth. To the best of our knowledge, our HIB-TDF gives rise to the first hierarchy of trapdoor functions which does not rely on lattices: realizing such a hierarchy using traditional number theoretic techniques was identified as an open problem by Cash *et al.* [17].

Beyond its hierarchical nature, our construction brings out an alternative design principle for (H)IB-TDFs. The idea is to rely on hierarchical predicate encryption (HPE) to deal with hierarchies. Namely, public parameters consist of a matrix of HPE encryptions and, when the function has to be evaluated, the latter matrix is turned into a matrix of (anonymous) HIBE ciphertexts. The homomorphic properties of the underlying HIBE then make it possible to evaluate the function while guaranteeing a sufficient amount of lossiness in lossy mode.

While the pairing-based construction of Bellare *et al.* [9] builds on an adaptively secure anonymous IBE, our (hierarchical) IB-TDF is obtained from a *selectively* weakly attribute-hiding HPE system. This result is somewhat incomparable with [9]: on one hand, we start from a more powerful primitive – because predicate encryption implies anonymous IBE – but, on the other hand, we need a weaker security level to begin with. Both (H)IB-TDF constructions rely on specific algebraic properties in the underlying IBE/HPE and neither is generic. It would be interesting to see if a more general approach exists for building such functions.

1.3 Discussion on the Implications

Combining our HIB-TDF with the theoretical implications of our new security property, we obtain: (1) a semantically secure HIBE encryption scheme under adaptive adversaries, (2) the first secure deterministic HIBE scheme[4], (3) the first HIBE scheme that (non-adaptively) hedges against bad randomness, as advocated by Bellare *et al.* [6].

In the case of adaptive adversaries, these results only hold for hierarchies of constant depth (said otherwise, we do not provide full security). However, using our definition of partial lossiness or that of Bellare *et al.* [9], this appears very difficult to avoid. The reason is that both definitions seem inherently bound to the partitioning paradigm. Namely, they assume the existence of alternative public parameters, called *lossy parameters*, where the identity space is partitioned into subsets of

[4] See [37] for a recent and independent construction, in the (non-hierarchical) IBE case.

injective and lossy identities. The definition of [9] intuitively captures that a fraction δ of identities are lossy in the case of lossy parameters. In the hierarchical setting, the analogy with HIBE schemes suggests that all ancestors of a lossy identity be lossy themselves. Hence, unless one can make sure that certain lossy identities only have lossy descendants, the fraction δ seems doomed to exponentially decline with the depth of the hierarchy.

Finally, due to the results of Canetti, Halevi and Katz [15], our construction also implies the first forward-secure deterministic and hedged public-key encryption schemes (note that selective security suffices to give forward-secure public-key cryptosystems). Although our scheme is not practical due to very large ciphertexts and key sizes, it provides the first feasibility results in these directions.

2 Hierarchical Identity-Based (Lossy) Trapdoor Functions, and Applications

In this section we extend to the hierarchical scenario the definitions for identity-based (lossy) trapdoor functions given in [9].

Syntax. A hierarchical identity-based trapdoor function (HIB-TDF) is a tuple of efficient algorithms $\mathsf{HF} = (\mathsf{HF.Setup}, \mathsf{HF.MKg}, \mathsf{HF.Kg}, \mathsf{HF.Del}, \mathsf{HF.Eval}, \mathsf{HF.Inv})$. The setup algorithm $\mathsf{HF.Setup}$ takes as input a security parameter $\varrho \in \mathbb{N}$, the (constant) number of levels in the hierarchy $d \in \mathbb{N}$, the length of the identities $\mu \in \mathsf{poly}(\varrho)$ and the length of the function inputs $n \in \mathsf{poly}(\varrho)$, and outputs a set of global public parameters pms, which specifies an input space InpSp, an identity space IdSp and the necessary mathematical objects and hash functions. The master key generation algorithm $\mathsf{HF.MKg}$ takes as input pms and outputs a master public key mpk and a master secret key msk. The key generation algorithm $\mathsf{HF.Kg}$ takes as input pms, msk and a hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in \mathsf{IdSp}$, for some $\ell \geq 1$ and outputs a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$. The delegation algorithm $\mathsf{HF.Del}$ takes as input pms, msk, a hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$ for it, and an additional identity $\mathsf{id}_{\ell+1}$; the output is a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell,\mathsf{id}_{\ell+1})}$ for the hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell, \mathsf{id}_{\ell+1})$ iff $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell, \mathsf{id}_{\ell+1}) \in \mathsf{IdSp}$. The evaluation algorithm $\mathsf{HF.Eval}$ takes as input pms, msk, an identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and a value $X \in \mathsf{InpSp}$; the result of the evaluation is denoted as C. Finally, the inversion algorithm $\mathsf{HF.Inv}$ takes as input pms, msk, a hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, a secret key $\mathbf{SK}_{\mathsf{id}}$ for it and an evaluation C, and outputs a value $\tilde{X} \in \mathsf{InpSp}$.

A HIB-TDF satisfies the property of correctness if

$$\mathsf{HF.Inv}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{id}, \mathbf{SK}_{\mathsf{id}}, \mathsf{HF.Eval}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), X\big)\big) \;=\; X,$$

for any $X \in \mathsf{InpSp}$, any $\mathsf{pms}, (\mathsf{mpk}, \mathsf{msk})$ generated by $\mathsf{HF.Setup}$ and $\mathsf{HF.MKg}$, any hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in \mathsf{IdSp}$ and any secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$ generated either by running $\mathsf{HF.Kg}\big(\mathsf{pms}, \mathsf{msk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)\big)$ or by applying the delegation algorithm $\mathsf{HF.Del}$ to secret keys of shorter hierarchical identities.

Before formalizing the new definition of partial lossiness for a HIB-TDF, let us recall the notion of *lossiness*: if f is a function with domain $\mathrm{Dom}(f)$ and image $\mathrm{Im}(f) = \{f(x) \;:\; x \in \mathrm{Dom}(f)\}$, we say that f is ω-lossy if $\lambda(f) \geq \omega$, where $\lambda(f) = \log \frac{|\mathrm{Dom}(f)|}{|\mathrm{Im}(f)|}$.

To properly define lossiness in the IB setting, it will also be useful to consider extended HIB-TDFs, which differ from standard HIB-TDFs in that, in the latter, the algorithm $\mathsf{HF.Setup}$ specifies in pms an auxiliary input space AuxSp, and $\mathsf{HF.MKg}$ takes as additional auxiliary input $aux \in \mathsf{AuxSp}$. Also, given some HIB-TDF $\mathsf{HF} = (\mathsf{HF.Setup}, \mathsf{HF.MKg}, \mathsf{HF.Kg}, \mathsf{HF.Del}, \mathsf{HF.Eval}, \mathsf{HF.Inv})$, a *sibling* for HF is an extended HIB-TDF $\mathsf{LHF} = (\mathsf{HF.Setup}, \mathsf{LHF.MKg}, \mathsf{LHF.Kg}, \mathsf{HF.Del}, \mathsf{HF.Eval}, \mathsf{HF.Inv})$ whose delegation, evaluation and inversion algorithms are those of HF, and where an auxiliary space

AuxSp is contained in $\mathsf{pms} \leftarrow \mathsf{HF.Setup}(\varrho)$, so that $\mathsf{IdSp} \subset \mathsf{AuxSp}$.

Looking ahead, we will define, as in [9], two different experiments: one corresponding to the standard setup and one corresponding to the lossy setup, in one of them the experiment will interact with a standard HIB-TDF, in the other one with a sibling in which some identities lead to lossy evaluation functions. The notion of extended HIB-TDF will serve to construct both of these functions as an extended HIB-TDF but with different auxiliary inputs $\vec{y}^{(0)}, \vec{y}^{(1)}$.

2.1 A New Security Definition for HIB-TDFs

The basic security property of a trapdoor function is *one-wayness*, which means that the function is hard to invert without the suitable secret key. In the identity-based setting, one-wayness is required to hold even when the adversary has oracle access to the secret keys for some identities. *Partial lossiness* for identity-based trapdoor functions was introduced in [9], where it was proved to imply one-wayness. Roughly speaking, partial lossiness requires that the weighted difference of the probability that any adversary outputs 1 in the lossy or in the real setup is negligible. For the selective case, the weights can simply be set to 1 and it can be proved that an IB-TDF satisfying their notion of partial lossiness in the selective scenario can be used to build: (1) identity-based encryption (IBE) schemes with selective IND-CPA security, (2) selectively secure deterministic IBE schemes, (3) selectively secure hedged IBE schemes. However, these results are not known to be true in the adaptive setting, in particular, the definition is not even known to yield an IND-ID-CPA secure encryption scheme.

To address this question, we propose an alternative definition for the partial lossiness of (hierarchical) identity-based trapdoor functions — in particular, they also result in a new definition when the hierarchy depth is equal to 1, the case considered by Bellare *et al.* [9]. We will show that a HIB-TDF satisfying this new definition gives a secure construction of the same primitives we mentioned for the selective case.

As in [9], we define two different experiments, a lossy experiment and a real experiment. For any adversary $\mathcal{A}$ against a HIB-TDF, and any $\zeta \in (0,1)$, the $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}$ experiment and the $\mathsf{LOSSY}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}$ experiment are parameterized by the security parameter ϱ (which we will usually omit in the notation) and values $\zeta(\varrho), \omega(\varrho)$. The value ζ will be important to show that our construction implies other cryptographic primitives. Intuitively, ζ will be the advantage of an adversary against a cryptographic scheme the security of which is reduced to the security of the HIB-TDF. The experiment also takes as input the specification of some efficient algorithm $\mathcal{P}$ which takes as input $\zeta, \mathsf{pms}, \mathsf{mpk}_1, \mathsf{msk}_1, IS, \mathsf{id}^\star$, and outputs a bit d_2. This procedure $\mathcal{P}$ can be, in general, any probabilistic and polynomial-time algorithm. In the security analysis of the selectively secure version of our HIB-TDF, $\mathcal{P}$ will be the trivial algorithm that always outputs $d_2 = 1$. In other cases, $\mathcal{P}$ could be a more complicated algorithm; for instance, in order to prove the security of the adaptive-id version of our HIB-TDF, we will take as pre-output stage $\mathcal{P}$ Waters' artificial abort step. Actually, procedure $\mathcal{P}$ is a way to relax the security requirements in Denition 1, in order to allow the possibility of building cryptographic schemes from secure HIB-TDFs, in a black-box way.

Finally, the value ω is related to the lossiness of the trapdoor function. To simplify notation, we will simply write REAL instead of $\mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}$ and LOSSY instead of $\mathsf{LOSSY}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}$.

For compactness, we present the two experiments as a single experiment depending of a bit β: the challenger $\mathcal{C}$, who interacts with the adversary $\mathcal{A}$, runs either REAL if $\beta = 0$ or LOSSY if $\beta = 1$. Also, some instructions of both experiments depend on whether *selective* or *adaptive* security is being considered. We say that a hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ is a *prefix* of another one $\mathsf{id}^\star = (\mathsf{id}^\star_1, \ldots, \mathsf{id}^\star_{\ell^\star})$ if $\ell \leq \ell^\star$ and $\mathsf{id}_i = \mathsf{id}^\star_i$ for every $i = 1, \ldots, \ell$. We denote it by $\mathsf{id} \leq \mathsf{id}^\star$.

0. $\mathcal{C}$ chooses global parameters pms by running $\mathsf{HF.Setup}$. The parameters pms are given to $\mathcal{A}$, who replies by choosing a hierarchical identity $\mathsf{id}^\dagger = (\mathsf{id}_1^\dagger, \ldots, \mathsf{id}_{\ell^\dagger}^\dagger)$, for some $\ell^\dagger \leq d$.
1. $\mathcal{C}$ runs $(\mathsf{mpk}_0, \mathsf{msk}_0) \leftarrow \mathsf{HF.MKg}(\mathsf{pms})$ and $(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow \mathsf{LHF.MKg}(\mathsf{pms}, aux = \mathsf{id}^\dagger)$. The adversary $\mathcal{A}$ receives mpk_β and lists $IS \leftarrow \emptyset$, $QS \leftarrow \emptyset$ are initialized.
2. $\mathcal{A}$ can make adaptive queries for hierarchical identities $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and identities $\mathsf{id}_{\ell+1}$.
   - **Create-key**: $\mathcal{A}$ provides id and $\mathcal{C}$ creates a private key SK_{id}. If $\beta = 0$, SK_{id} is created by running $\mathsf{HF.Kg}(\mathsf{pms}, \mathsf{msk}_0, \mathsf{id})$. If $\beta = 1$, it is created by running $\mathsf{LHF.Kg}(\mathsf{pms}, \mathsf{msk}_1, \mathsf{id})$. The list QS is updated as $QS = QS \cup \{\mathsf{id}\}$.
   - **Create-delegated-key**: $\mathcal{A}$ provides $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and $\mathsf{id}_{\ell+1}$ such that $\mathsf{id} \in QS$. The challenger $\mathcal{C}$ then computes $SK_{\mathsf{id}'}$ for $\mathsf{id}' = (\mathsf{id}_1, \ldots, \mathsf{id}_{\ell+1})$ by running the delegation algorithm $\mathsf{HF.Del}\big(\mathsf{pms}, \mathsf{mpk}_\beta, SK_{\mathsf{id}}, \mathsf{id}_{\ell+1}\big)$. The list QS is updated as $QS = QS \cup \{\mathsf{id}'\}$.
   - **Reveal-key**: $\mathcal{A}$ provides id with the restriction that if $\mathcal{A}$ is selective, then $\mathsf{id} \not\leq \mathsf{id}^\dagger$. $\mathcal{C}$ returns $\perp$ if $\mathsf{id} \notin QS$. Otherwise, SK_{id} is returned to $\mathcal{A}$ and the list IS is updated as $IS = IS \cup \{\mathsf{id}\}$.
3. The adversary $\mathcal{A}$ outputs a hierarchical identity $\mathsf{id}^\star = (\mathsf{id}_1^\star, \ldots, \mathsf{id}_{\ell^\star}^\star)$ and a bit $d_{\mathcal{A}} \in \{0, 1\}$. If $\mathcal{A}$ is selective, then $\mathsf{id}^\star = \mathsf{id}^\dagger$. In the adaptive case, no element of IS can be a prefix of $\mathsf{id}^\star$. Let d_1 be the bit $d_1 := \big(\forall\ \mathsf{id} \in IS, \lambda\left(\mathsf{HF.Eval}(\mathsf{pms}, \mathsf{mpk}_1, \mathsf{id}, \cdot)\right) = 0\big) \wedge \big(\lambda\left(\mathsf{HF.Eval}(\mathsf{pms}, \mathsf{mpk}_1, \mathsf{id}^\star, \cdot)\right) \geq \omega\big)$.
4. $\mathcal{C}$ sets d_2 to be the output of the pre-output stage $\mathcal{P}$ with input $\zeta, \mathsf{pms}, \mathsf{mpk}_1, \mathsf{msk}_1, IS, \mathsf{id}^\star$.
5. The final output of the experiment consists of $\{d_{\mathcal{A}},\ d_{\neg abort}^{\mathcal{A}}\}$, where $d_{\neg abort}^{\mathcal{A}} = d_1 \wedge d_2 \in \{0, 1\}$.

For notational convenience, from now on, let us define $d_{exp}^{\mathcal{A}} = d_{\mathcal{A}} \wedge d_{\neg abort}^{\mathcal{A}}$.

Note that, in the lossy experiment, some identities may lead to lossy functions, which can be detected by $\mathcal{A}$ if it queries the secret key for such an identity. This causes an asymmetry when comparing the real and the lossy experiments. For this reason, Bellare *et al.* defined the advantage of a distinguisher among the lossy and real experiments as the *weighted* difference of the probability of outputting 1 in the real case minus the same probability in the lossy case. Our solution is different: we force the experiment to adopt the same behavior in the real and lossy settings: if a query would force the LOSSY experiment to set the bits d_1 or d_2 to 0 (and, as a consequence, setting $d_{exp}^{\mathcal{A}} = 0$), then it also forces the REAL experiment to set d_1 or d_2 to 0, respectively. Therefore, the difference of probabilities in our definition (see condition (i) below) is not weighted.

Definition 1. *A HIB-TDF is (ω, δ)-partially lossy if it admits a sibling and an efficient pre-output stage $\mathcal{P}$ such that for all PPT adversaries $\mathcal{A}$ and for all non-negligible ζ, there exist two non-negligible values ϵ_1, ϵ_2 with $\delta = \epsilon_1\epsilon_2$ such that the following three conditions hold:*

(i) the following advantage function is negligible in the security parameter ϱ:

$$\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) = |\Pr[d_{exp}^{\mathcal{A}} = 1|\ \mathsf{REAL}]\ -\ \Pr[d_{exp}^{\mathcal{A}} = 1|\ \mathsf{LOSSY}]| \tag{1}$$

(ii) $\Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid \mathsf{REAL}] \geq \epsilon_1$.

(iii) if $\mathcal{A}$ is such that $\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}]\ -\ \frac{1}{2} > \zeta$, *then*

$$\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL} \wedge d_{\neg abort}^{\mathcal{A}} = 1] - \frac{1}{2} > \epsilon_2 \cdot \zeta, \tag{2}$$

where δ may be a function of q the maximal number of secret key queries of $\mathcal{A}$.

As said above, condition (i) is a simple modification of the definition of partial lossiness of Bellare *et al.* [9]. We add condition (ii) to rule out some cases in which the definition would be trivial to satisfy, like the case where the procedure $\mathcal{P}$ aborts with overwhelming probability or the case where the sibling admits only lossy identities: in any of these scenarios, we would have $d_{\neg abort} = 1$ with negligible probability, which would render the scheme useless with the sole condition (i). Then, we add a third condition (iii) which allows reducing HIB-TDFs to other primitives. Roughly speaking, this condition guarantees that the probability of aborting is somewhat independent of the behavior of any computationally bounded adversary. Interestingly, it is possible to prove (by proceeding exactly as in the proof of our HIB-TDF) that the pairing-based IB-TDF described by Bellare *et al.* [9] satisfies our new partial-lossiness definition.

2.2 Implications of Lossy (H)IB-TDFs: the Example of (H)IBE

Using the same argument as in [9], it is quite easy to prove that a HIB-TDF which enjoys the new version of the partial lossiness property is already one-way, in both the selective and adaptive settings. In this section we prove that a HIB-TDF which satisfies our new security definition can be used to build other primitives in the hierarchical identity-based security, with security against adaptive adversaries. We detail the example of hierarchical identity-based encryption (HIBE) with IND-CPA security[5]. The construction is the direct adaptation of the Peikert-Waters construction [31] in the public-key setting.

Let HF be a HIB-TDF with message space $\{0,1\}^n$ and lossiness ω, and $\mathcal{H}$ a family of pairwise independent hash functions from $\{0,1\}^n$ to $\{0,1\}^l$ where $l \leq \omega - 2\lg(1/\epsilon_{LHL})$ for some negligible ϵ_{LHL}. The HIBE scheme has message space $\{0,1\}^l$. Its setup, key generation and key delegation algorithms are basically the same ones as those for HF, the rest are as follows:

$\mathbf{MKGen}(\mathsf{pms})$	$\mathbf{Enc}(\mathsf{pms}, \mathsf{mpk}, m, \mathsf{id})$	$\mathbf{Dec}(\mathsf{pms}, \mathsf{mpk}, \mathbf{SK}_{\mathsf{id}}, C, \mathsf{id})$
$(\mathsf{mpk}', \mathsf{msk}) \leftarrow \mathsf{HF.MKg}(1^k)$	$x \leftarrow \{0,1\}^n$	$x = \mathsf{HF.Inv}(\mathsf{pms}, \mathsf{mpk}, \mathbf{SK}_{\mathsf{id}}, c_1, \mathsf{id})$
$h \leftarrow \mathcal{H}$	$c_1 = \mathsf{HF.Eval}(\mathsf{pms}, \mathsf{mpk}, \mathsf{id}, x)$	$m = c_2 \oplus h(x)$
$\mathsf{mpk} = (\mathsf{mpk}', h)$	$c_2 = h(x) \oplus m$	Return m
Return mpk	Return $C = (c_1, c_2)$	

We prove the following theorem.

Theorem 1. *If* HF *is* (ω, δ)*-partially lossy for some non-negligible value of* δ*, then the HIBE scheme* Π *described is IND-ID-CPA secure. In particular, for every IND-ID-CPA adversary* $\mathcal{B}$ *against* Π *there exists a PPT adversary* $\mathcal{A}$ *against* HF *such that*

$$\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) \geq \frac{2}{3} \cdot \delta \cdot \mathbf{Adv}^{\text{ind-id-cpa}}(\mathcal{B}) - \nu(\varrho)$$

for some negligible function ν*; both adversaries* $\mathcal{A}$ *and* $\mathcal{B}$ *run in comparable times.*

Proof. Let us assume that an adversary $\mathcal{B}$ has advantage at least ζ in breaking the IND-ID-CPA security of the HIBE scheme Π, for some non-negligible ζ. We build an adversary $\mathcal{A}$ that breaks the condition (i) of Definition 1 assuming that conditions (ii) and (iii) are satisfied. Our adversary $\mathcal{A}$, who interacts with a challenger that runs either the experiment REAL or the experiment LOSSY, proceeds to simulate the challenger in the IND-ID-CPA game with $\mathcal{B}$ as follows.

[5] The cases of deterministic HIBE and hedged HIBE are discussed in Appendices C and D.

$\mathcal{A}$ forwards an identity $\mathsf{id}^\dagger$ to its challenger, which is some random identity in the adaptive case or corresponds to the challenge identity chosen by $\mathcal{B}$ in the selective case. When the challenger runs the setup and gives the output to $\mathcal{A}$, $\mathcal{A}$ forwards this information to $\mathcal{B}$ together with a hash function $h \leftarrow \mathcal{H}$. When $\mathcal{B}$ asks for a secret key for a hierarchical identity id, $\mathcal{A}$ forwards the query to the experiment and forwards the reply to $\mathcal{B}$. At some point, $\mathcal{B}$ outputs $(m_0, m_1, \mathsf{id}^\star)$, with $\mathsf{id}^\dagger = \mathsf{id}^*$ in the selective case. Adversary $\mathcal{A}$ then forwards $\mathsf{id}^\star$ to its challenger, chooses $\gamma \leftarrow \{0,1\}$ at random and encrypts m_γ under the identity $\mathsf{id}^\star$. After some more secret key queries, $\mathcal{B}$ outputs a guess γ' and $\mathcal{A}$ outputs $d_\mathcal{A} = 1$ if $\gamma = \gamma'$ and $d_\mathcal{A} = 0$ otherwise.

In the REAL setting, we will have

$$\Pr[\gamma' = \gamma |\ \mathsf{REAL}] - \frac{1}{2} = \Pr[d_\mathcal{A} = 1 |\ \mathsf{REAL}] - \frac{1}{2} \geq \zeta,$$

since $\mathcal{A}$ perfectly simulated the IND-ID-CPA game with $\mathcal{B}$. This inequality can be combined with conditions (ii) and (iii) of the definition of (ω, δ)-partial lossiness (which we assume to be satisfied by HF), and we obtain

$$\Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{REAL}] \cdot \left(\Pr[d_\mathcal{A} = 1 \mid \mathsf{REAL} \wedge d^\mathcal{A}_{\neg abort} = 1] - \frac{1}{2} \right) > \epsilon_1 \epsilon_2 \zeta. \tag{3}$$

On the other hand, as proved in [9], in the LOSSY setting when $\mathsf{id}^\star$ is lossy, the advantage of $\mathcal{B}$ in guessing γ is negligible. Indeed, since we are using a pairwise independent hash function, the Leftover Hash Lemma [24] (more precisely, its variant proved in [18]) implies that the distribution of c_2 given c_1 is statistically close to the uniform distribution. We thus have $\Pr[d_\mathcal{A} = 1 |\ \mathsf{LOSSY} \wedge d^\mathcal{A}_{\neg abort} = 1] \leq \frac{1}{2} + \epsilon_{LHL}$, for some negligible function ϵ_{LHL}. Since $d^\mathcal{A}_{exp} = d^\mathcal{A}_{\neg abort} \wedge d_\mathcal{A}$, we find

$$\begin{aligned}
\Pr[d^\mathcal{A}_{exp} = 1 \mid \mathsf{LOSSY}] &= \Pr[d_\mathcal{A} = 1 \mid \mathsf{LOSSY} \wedge d^\mathcal{A}_{\neg abort} = 1] \Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{LOSSY}] \\
&\leq \left(\frac{1}{2} + \epsilon_{LHL}\right) \cdot \Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{LOSSY}] \\
&\leq \frac{1}{2} \cdot \left(\Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{REAL}] + \mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) \right) + \nu,
\end{aligned} \tag{4}$$

for some negligible function $\nu \in \mathsf{negl}(\varrho)$. The last equality follows from the fact that we can assume that $\Pr[d^\mathcal{A}_{\neg abort} = 1 |\ \mathsf{LOSSY}] - \Pr[d^\mathcal{A}_{\neg abort} = 1 |\ \mathsf{REAL}] \leq \mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A})$: otherwise, we can easily build a distinguisher[6] against condition (i) of the partial lossiness definition. If we plug (4) into the definition of $\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A})$, we obtain

$$\begin{aligned}
\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) &= \left| \Pr[d^\mathcal{A}_{exp} = 1 \mid \mathsf{REAL}] - \Pr[d^\mathcal{A}_{exp} = 1 \mid \mathsf{LOSSY}] \right| \\
&\geq \left| \Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{REAL}] \cdot \left(\Pr[d_\mathcal{A} = 1 \mid \mathsf{REAL} \wedge d^\mathcal{A}_{\neg abort} = 1] - \frac{1}{2} \right) \right| \\
&\quad - \frac{1}{2} \cdot \mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) - \nu,
\end{aligned} \tag{5}$$

so that there exists $\tilde{\nu} \in \mathsf{negl}(\varrho)$ such that

$$\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) \geq \frac{2}{3} \cdot \left| \Pr[d^\mathcal{A}_{\neg abort} = 1 \mid \mathsf{REAL}] \cdot \left(\Pr[d_\mathcal{A} = 1 \mid \mathsf{REAL} \wedge d^\mathcal{A}_{\neg abort} = 1] - \frac{1}{2} \right) \right| - \tilde{\nu}.$$

[6] This distinguisher $\mathcal{A}_1$ is obtained from $\mathcal{A}$ by ignoring $d_\mathcal{A} \in \{0,1\}$ and replacing it by a 1, so that $d^\mathcal{A}_{\neg abort} = d^\mathcal{A}_{exp}$.

This means that $\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A})$ is non-negligible since $\delta = \epsilon_1\epsilon_2$ and the right-hand-side member of the above expression is at least $(2/3)\cdot\delta\cdot\zeta - \nu$. In other words, any adversary guessing $\gamma' = \gamma$ with non-negligible advantage ζ in REAL necessarily contradicts condition (i). □

3 Interlude: Hierarchical Predicate Encryption

In this section we propose a new hierarchical predicate encryption scheme with the attribute-hiding property, which will be used as an ingredient to build, in the next section, a hierarchical identity-based (lossy) trapdoor function. The syntax and security model for hierarchical predicate encryption schemes are recalled in Appendix A.1.

3.1 Some Complexity Assumptions

We consider groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p for which an asymmetric bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ is efficiently computable. We will assume that the DDH assumption holds in both $\mathbb{G}$ and $\hat{\mathbb{G}}$, which implies that no isomorphism is efficiently computable between $\mathbb{G}$ and $\hat{\mathbb{G}}$. The assumptions that we need are sometimes somewhat stronger than DDH. However, they have *constant* size (*i.e.*, we de not rely on q-type assumptions) and were previously used in [20].

The Bilinear Diffie Hellman Assumption (BDH): in bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p, the distribution $D_1 = \{(g,\ g^a,\ g^c,\ \hat{g},\ \hat{g}^a,\ \hat{g}^b,\ e(g,\hat{g})^{abc}) \ \mid a, b, c \xleftarrow{R} \mathbb{Z}_p\}$, is computationally indistinguishable from $D_2 = \{(g,\ g^a,\ g^c,\ \hat{g},\ \hat{g}^a,\ \hat{g}^b,\ e(g,\hat{g})^z) \ \mid a, b, c, z \xleftarrow{R} \mathbb{Z}_p\}$.

The $\mathcal{P}$-BDH$_1$ Assumption: in asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p, the distribution $D_1 = \{(g, g^b, g^{ab}, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, g^{abc}) \mid a, b, c \xleftarrow{R} \mathbb{Z}_p\}$ is computationally indistinguishable from $D_2 = \{(g, g^b, g^{ab}, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, g^z) \mid a, b, c, z \xleftarrow{R} \mathbb{Z}_p\}$.

The DDH$_2$ Assumption: in asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p, the distribution $D_1 = \{(g,\ \hat{g},\ \hat{g}^a,\ \hat{g}^b,\ \hat{g}^{ab}) \mid a, b \xleftarrow{R} \mathbb{Z}_p\}$ is computationally indistinguishable from the distribution $D_2 = \{(g,\ \hat{g},\ \hat{g}^a,\ \hat{g}^b,\ \hat{g}^z) \mid a, b, z \xleftarrow{R} \mathbb{Z}_p\}$.

3.2 A Selectively Secure Weakly Attribute-Hiding Hierarchical Predicate Encryption Scheme

The construction is inspired by the Shi-Waters delegatable predicate encryption scheme [33] (at a high-level, it also bears similarities with the lattice-based scheme of [3]). However, we have to turn it into a predicate encryption scheme for inner product relations (like the one of Okamoto and Takashima [30]) instead of a hidden vector encryption [11]. Another difficulty to solve is that we cannot use composite order groups as in [33] because, in our HIB-TDF of Section 4, one of the subgroups would eventually leak information on the input in lossy mode (this is actually what happened with our initial attempt). For this reason, we chose to work with prime-order groups and used asymmetric pairing configurations to anonymize ciphertexts. As a benefit, we obtain a better efficiency than by using the techniques of [11] by reducing the number of pairing evaluations.

Setup(ϱ, d, μ): given a security parameter $\varrho \in \mathbb{N}$, the (constant) desired number of levels in the hierarchy $d \in \mathbb{N}$ and the desired length μ of the attribute vectors at each level, choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of order p, where $p > 2^{\varrho}$. Choose $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$. Then, pick $\alpha, \alpha_v, \alpha_w \xleftarrow{R} \mathbb{Z}_p^*$ and set $v = g^{\alpha_v}$, $\hat{v} = \hat{g}^{\alpha_v}$, $w = g^{\alpha_w}$ and $\hat{w} = \hat{g}^{\alpha_w}$. For $i_1 = 1, \ldots, d$ and

$i_2 = 0, \ldots, \mu$, choose $\alpha_{i_1,i_2} \xleftarrow{R} \mathbb{Z}_p^*$ and compute $h_{i_1,i_2} = g^{\alpha_{i_1,i_2}} \in \mathbb{G}$ and $\hat{h}_{i_1,i_2} = \hat{g}^{\alpha_{i_1,i_2}} \in \mathbb{G}$. The master public key is defined to be

$$\mathsf{mpk} := \Big(v,\ w,\ e(g,\hat{v})^{\alpha},\ \{h_{i_1,i_2}\}_{i_1\in\{1,\ldots,d\},\ i_2\in\{0,\ldots,\mu\}}\Big)$$

while the master secret key is $\mathsf{msk} := \big(\hat{g}, \hat{g}^{\alpha}, \hat{v}, \hat{w}, \{\hat{h}_{i_1,i_2}\}_{i_1\in\{1,\ldots,d\},\ i_2\in\{0,\ldots,\mu\}}\big)$.

Keygen$\big(\mathsf{msk}, (\vec{X}_1, \ldots, \vec{X}_\ell)\big)$**:** to generate a private key for vectors $(\vec{X}_1, \ldots, \vec{X}_\ell)$ with $\ell \leq d$, parse msk as $\big(\hat{g}, \hat{v}, \hat{w}, \{\hat{h}_{i_1,i_2}\}_{i_1\in\{1,\ldots,d\},\ i_2\in\{0,\ldots,\mu\}}\big)$. For $i_1 = 1$ to ℓ, parse $\vec{X}_{i_1}$ as $(x_{i_1,1}, \ldots, x_{i_1,\mu}) \in \mathbb{Z}_p^{\mu}$. Choose $r_w \xleftarrow{R} \mathbb{Z}_p^*$ and $r_1, \ldots, r_\ell \xleftarrow{R} \mathbb{Z}_p^*$, for $i_1 \in \{1, \ldots, \ell\}$. Then, compute the decryption component $SK_D = (D, D_w, \{D_{i_1}\}_{i_1=1}^{\ell})$ of the key as

$$D = \hat{g}^{\alpha} \cdot \prod_{i_1=1}^{\ell} \big(\prod_{i_2=1}^{\mu} \hat{h}_{i_1,i_2}^{x_{i_1,i_2}} \big)^{r_{i_1}} \cdot \hat{w}^{r_w}, \qquad D_w = \hat{v}^{r_w}, \qquad D_{i_1} = \hat{v}^{r_{i_1}}.$$

To define the elements of its delegation component SK_{DL}

$$\big(\{K_{j,k}, L_j, L_{j,k,i_1}, L_{w,j,k}\}_{j\in\{\ell+1,\ldots,d\},\ k\in\{1,\ldots,\mu\},\ i_1\in\{1,\ldots,\ell\}}\big)$$

pick $s_j \xleftarrow{R} \mathbb{Z}_p^*$, $s_{j,k,i_1} \xleftarrow{R} \mathbb{Z}_p^*$, $s_{w,j,k} \xleftarrow{R} \mathbb{Z}_p^*$ for $i_1 \in \{1, \ldots, \ell\},\ i_2 \in \{1, \ldots, \mu\}$ and set:

$$K_{j,k} = \prod_{i_1=1}^{\ell} \big(\prod_{i_2=1}^{\mu} \hat{h}_{i_1,i_2}^{x_{i_1,i_2}} \big)^{s_{j,k,i_1}} \cdot \hat{h}_{j,k}^{s_j} \cdot \hat{w}^{s_{w,j,k}}, \quad L_j = \hat{v}^{s_j}, \quad L_{j,k,i_1} = \hat{v}^{s_{j,k,i_1}}, \quad L_{w,j,k} = \hat{v}^{s_{w,j,k}}.$$

Output the private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)} = \big(SK_D, SK_{DL}\big)$.

Delegate$\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, \vec{X}_{\ell+1}\big)$**:** parse the key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$ as (SK_D, SK_{DL}). Given, $\vec{X}_{\ell+1} = (x_{\ell+1,1}, \ldots, x_{\ell+1,\mu}) \in \mathbb{Z}_p^{\mu}$, do the following.

1. Randomize SK_{DL} by raising all its component to some $z \xleftarrow{R} \mathbb{Z}_p^*$. Call this new key $\widehat{SK}_{DL}$ and write its elements with a hat (e.g., $\widehat{K}_{j,k} = K_{j,k}^z$).
2. Compute a *partial decryption key*

$$K_{\ell+1} = \prod_{k=1}^{\mu} \widehat{K}_{\ell+1,k}^{x_{\ell+1,k}} = \prod_{i_1=1}^{\ell} \big(\prod_{i_2=1}^{\mu} \hat{h}_{i_1,i_2}^{x_{i_1,i_2}} \big)^{s_{\ell+1,i_1}} \cdot \big(\prod_{k=1}^{\mu} \hat{h}_{\ell+1,k}^{x_{\ell+1,k}} \big)^{s_{\ell+1}} \cdot \hat{w}^{s_{w,\ell+1}}, \quad L_{\ell+1,\ell+1} = \widehat{L}_{\ell+1},$$

$$L_{\ell+1,i_1} = \prod_{k=1}^{\mu} \widehat{L}_{\ell+1,k,i_1}^{x_{\ell+1,k}} = \hat{v}^{s_{\ell+1,i_1}} \quad \text{for} \quad i_1 \in \{1, \ldots, \ell\}, \qquad L_{w,\ell+1} = \prod_{k=1}^{\mu} \widehat{L}_{w,\ell+1,k}^{x_{\ell+1,k}} = \hat{v}^{s_{w,\ell+1}}$$

where we define the exponents $s_{\ell+1,i_1} = z \cdot \sum_{k=1}^{\mu} s_{\ell+1,k,i_1} \cdot x_{\ell+1,k}$ for $i_1 \in \{1, \ldots, \ell\}$, and $s_{w,\ell+1} = z \cdot \sum_{k=1}^{\mu} s_{w,\ell+1,k} \cdot x_{\ell+1,k}$.
3. For all $j \in \{\ell+2, \ldots, d\}$, $k \in \{1, \ldots, \mu\}$, compute re-randomized versions of the partial decryption key by raising the partial decryption key to a random power $\tau_{j,k} \xleftarrow{R} \mathbb{Z}_p^*$.

$$K_{\ell+1}^{(j,k)} = K_{\ell+1}^{\tau_{j,k}}, \qquad L_{w,\ell+1}^{(j,k)} = L_{w,\ell+1}^{\tau_{j,k}}, \qquad \{L_{\ell+1,i_1}^{(j,k)} = L_{\ell+1,i_1}^{\tau_{j,k}}\}_{i_1=1}^{\ell+1}.$$

These values will be used to compute the delegation component of the new key at step 5.

4. Compute a decryption component $SK'_D = (D', D'_w, \{D'_{i_1}\}_{i_1=1}^{\ell+1})$ for the delegated key by setting $D' = D \cdot K_{\ell+1}$, $D'_w = D_w \cdot L_{w,\ell+1}$. Then, define $D'_{\ell+1} = L_{\ell+1,\ell+1}$ and, for each $i_1 \in \{1, \ldots, \ell\}$, set $D'_{i_1} = D_{i_1} \cdot L_{\ell+1,i_1}$.
5. Compute a delegation component for the delegated key. For each $j \in \{\ell+2, \ldots, d\}$, set $L'_j = \widehat{L}_j$. Then, for $k = 1$ to μ and $i_1 = 1$ to $\ell + 1$, set

$$K'_{j,k} = \widehat{K}_{j,k} \cdot K_{\ell+1}^{(j,k)}, \quad L'_{w,j,k} = \widehat{L}_{w,j,k} \cdot L_{w,\ell+1}^{(j,k)} \quad L'_{j,k,i_1} = \widehat{L}_{j,k,i_1} \cdot L_{\ell+1,i_1}^{(j,k)},$$

where $\widehat{L}_{j,k,\ell+1} = 1$ for all j, k. The new delegation component SK'_{DL} is

$$\big(\{K'_{j,k}, L'_j, L'_{j,k,i_1}, L'_{w,j,k}\}_{j\in\{\ell+2,\ldots,d\},\ k\in\{1,\ldots,\mu\},\ i_1\in\{1,\ldots,\ell\}}\big)$$

Return the delegated private key $SK_{(\vec{X}_1,\ldots,\vec{X}_{\ell+1})} = (SK'_D, SK'_{DL})$.

Encrypt$\big(\mathsf{mpk}, (\vec{Y}_1, \ldots, \vec{Y}_\kappa), M\big)$**:** given mpk, a plaintext $M \in \mathbb{G}_T$ as well as a hierarchy of vectors $\vec{Y}_1 = (y_{1,1}, \ldots, y_{1,\mu})$, $\ldots$, $\vec{Y}_\kappa = (y_{\kappa,1}, \ldots, y_{\kappa,\mu})$, choose $s \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$C_0 = M \cdot e(g, \hat{v})^{\alpha \cdot s}, \qquad C_v = v^s, \qquad C_w = w^s, \qquad \{C_{i_1,i_2} = \big(h_{i_1,0}^{y_{i_1,i_2}} \cdot h_{i_1,i_2}\big)^s\}_{i_1\in\{1,\ldots,\kappa\},\ i_2\in\{1,\ldots,\mu\}},$$

The ciphertext is $C = \big(C_0, C_v, C_w, \{C_{i_1,i_2}\}_{i_1\in\{1,\ldots,\kappa\},\ i_2\in\{1,\ldots,\mu\}}\big)$.

Decrypt$\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, C\big)$**:** parse the private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$ as $\big(SK_D, SK_{DL}\big)$, where $SK_D = (D, D_w, \{D_{i_1}\}_{i_1=1}^{\ell})$ and the ciphertext C as $\big(C_0, C_v, C_w, \{C_{i_1,i_2}\}_{i_1\in\{1,\ldots,\kappa\},\ i_2\in\{1,\ldots,\mu\}}\big)$.

1. For each $i_1 \in \{1, \ldots, \ell\}$, compute $C_{i_1} = \prod_{i_2=1}^{\mu} C_{i_1,i_2}^{x_{i_1,i_2}} = \big(h_{i_1,0}^{\vec{X}_{i_1}\cdot\vec{Y}_{i_1}} \cdot \prod_{i_2=1}^{\mu} h_{i_1,i_2}^{x_{i_1,i_2}}\big)^s$.
2. Return M if $M = C_0 \cdot e(C_v, D)^{-1} \cdot e(C_w, D_w) \cdot \prod_{i_1=1}^{\ell} e(C_{i_1}, D_{i_1})$ is in the appropriate subspace[7] of $\mathbb{G}_T$. Otherwise, return $\perp$.

We show that our scheme is correct in Appendix A.2. Our HIB-TDF uses the predicate-only variant of the above scheme, obtained by discarding the ciphertext component C_0 (which contains the payload) and the factor $\hat{g}^\alpha$ from the private key component D.

The new HPE scheme is selectively weakly attribute-hiding under the BDH, $\mathcal{P}$-BDH$_1$ and DDH$_2$ assumptions, as established by Theorem 2. The security of its predicate-only variant (which is the one used as a key ingredient in the design of our HIB-TDF) relies only on the latter two assumptions.

Theorem 2. *The HPE scheme is selectively weakly attribute-hiding (in the sense of Definition 2 in Appendix A.1) if the BDH, $\mathcal{P}$-BDH$_1$ and DDH$_2$ assumptions hold in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$. (The proof is given in Appendix A.3).

4 A Hierarchical Identity-Based (Lossy) Trapdoor Function

From the HPE scheme of Section 3.2, our lossy function is obtained by including a $n \times n$ matrix of HPE ciphertexts in the master public parameters. As in the DDH-based function of [31], each row of the matrix is associated with an encryption exponent, which is re-used throughout the entire row. Each column corresponds to a different set of public parameters in the HPE system.

The HIB-TDF that we construct is actually an extended HIB-TDF, and so the master key

[7] As in [11, 28], the plaintext space is restricted to have a size much smaller than $|\mathbb{G}_T|$ to make sure that the decryption algorithm returns $\perp$ if an unauthorized key is used to decrypt.

generation protocol takes an auxiliary input. Depending on the value of this auxiliary input, we obtain the trapdoor (injective) function or a partially lossy function, used in the security proofs. Actually, all HPE ciphertexts in the above-mentioned matrix correspond to different hierarchical vectors $(\mathbf{y}_1, \ldots, \mathbf{y}_d) \in \mathbb{Z}_p^{d \cdot \mu}$, depending on the auxiliary input. The selective weak attribute-hiding property of the HPE scheme guarantees that the two setups are computationally indistinguishable.

In order to evaluate a function for some hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, the first step of the evaluation algorithm computes a transformation on HPE ciphertexts so as to obtain a matrix of HIBE ciphertexts. During this transformation, a set of inner products $\{\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle\}_{i_1=1}^{\ell}$ is calculated in the exponent in the diagonal entries of the matrix. The transformation provides a $n \times n$ matrix (8) of anonymous HIBE ciphertexts that are always well-formed in non-diagonal entries. As for diagonal entries, they contain "perturbed" HIBE ciphertexts: at each level, one ciphertext component contains a perturbation factor of the form $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle$. In this matrix of HIBE ciphertexts, random encryption exponents are again re-used in all positions at each row.

The function evaluation is then carried out as in [31], by computing a matrix-vector product in the exponent and taking advantage of homomorphic properties of the HIBE scheme over the randomness space. The function output can be seen as a set of n anonymous HIBE ciphertexts – one for each input bit – which are well-formed ciphertexts if and only if the corresponding input bit is 0 (*i.e.*, if and only if the perturbation factors $\{\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle\}_{i_1=1}^{\ell}$ are left out when computing the matrix-vector product in the exponent). The function is thus inverted by testing the well-formedness of each HIBE ciphertext using the private key.

4.1 Description

HF.Setup(ϱ, d, n, μ)**:** given a security parameter $\varrho \in \mathbb{N}$, the (constant) desired number of levels in the hierarchy $d \in \mathbb{N}$ and integers $\mu, n \in \mathsf{poly}(\varrho)$ specifying the length of identities and that of function inputs, respectively, choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\varrho$. Define $\mathsf{InpSp} = \{0,1\}^n$, $\Sigma_{\mathsf{ID}} = \{(1, \mathbf{x}) : \mathbf{x} \in \mathbb{Z}_p^{\mu-1}\}$, $\mathsf{IdSp} = \Sigma_{\mathsf{ID}}^{(\leq d)}$ and $\mathsf{AuxSp} = \mathbb{Z}_p^{d \cdot \mu}$. The public parameters are $\mathsf{pms} = \big(p, (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), d, n, \mu, \mathsf{InpSp}, \mathsf{IdSp}, \mathsf{AuxSp}\big)$.

Since HF is an extended HIB-TDF, the master key generation algorithm of our HIB-TDF receives an auxiliary input $\mathbf{y} \in \mathsf{AuxSp}$. Here, it is seen as a concatenation of d row vectors $\mathbf{y}_1, \ldots, \mathbf{y}_d \in \mathbb{Z}_p^{\mu}$.

HF.MKg$(\mathsf{pms}, \mathbf{y})$**:** parse the auxiliary input as $\mathbf{y} = [\mathbf{y}_1 | \ldots | \mathbf{y}_d] \in \mathbb{Z}_p^{d \cdot \mu}$, and proceed as follows.

1. Choose $\alpha_v \xleftarrow{R} \mathbb{Z}_p^*$, $\boldsymbol{\alpha}_w \xleftarrow{R} (\mathbb{Z}_p^*)^n$, and $\boldsymbol{\alpha}_h \xleftarrow{R} (\mathbb{Z}_p^*)^{d \times (\mu+1) \times n}$. Define $v = g^{\alpha_v}$, $\hat{v} = \hat{g}^{\alpha_v}$, $\mathbf{w} = g^{\boldsymbol{\alpha}_w} \in \mathbb{G}^n$ and $\hat{\mathbf{w}} = \hat{g}^{\boldsymbol{\alpha}_w} \in \hat{\mathbb{G}}^n$. Likewise, set up vectors $\mathbf{h} = g^{\boldsymbol{\alpha}_h} \in \mathbb{G}^{d \times (\mu+1) \times n}$ and $\hat{\mathbf{h}} = \hat{g}^{\boldsymbol{\alpha}_h} \in \hat{\mathbb{G}}^{d \times (\mu+1) \times n}$. Define

$$\mathsf{PP}_{core} := \Big(v,\ \{\mathbf{w}[l_1]\}_{l_1=1}^{n},\ \{\mathbf{h}[i_1, i_2, l_1]\}_{i_1 \in \{1,\ldots,d\}, i_2 \in \{0,\ldots,\mu\},\ l_1 \in \{1,\ldots,n\}}\Big)$$

2. For $i_1 = 1$ to d, parse $\mathbf{y}_{i_1}$ as $(\mathbf{y}_{i_1}[1], \ldots, \mathbf{y}_{i_1}[\mu]) \in \mathbb{Z}_p^{\mu}$. For $l_2 = 1$ to n, do the following.

   a. Choose $\mathbf{s}[l_2] \xleftarrow{R} \mathbb{Z}_p^*$ and compute $\mathbf{J}[l_2] = v^{\mathbf{s}[l_2]}$ as well as

$$\mathbf{C}_w[l_2, l_1] = \mathbf{w}[l_1]^{\mathbf{s}[l_2]}, \quad \mathbf{C}[i_1, i_2, l_2, l_1] = \big(\mathbf{h}[i_1, 0, l_1]^{\mathbf{y}_{i_1}[i_2] \cdot \Delta(l_2, l_1)} \cdot \mathbf{h}[i_1, i_2, l_1]\big)^{\mathbf{s}[l_2]}$$

   for each $i_1 \in \{1, \ldots, d\}$, $i_2 \in \{1, \ldots, \mu\}$, $l_1 \in \{1, \ldots, n\}$.

b. Define a $n \times n$ matrix $\{\mathbf{CT}[l_2, l_1]\}_{l_2,l_1 \in \{1,\ldots,n\}}$ of HPE ciphertexts

$$\mathbf{CT}[l_2, l_1] = \big(\mathbf{J}[l_2], \mathbf{C}_w[l_2, l_1], \{\mathbf{C}[i_1, i_2, l_2, l_1]\}_{i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\}}\big). \tag{6}$$

The master public key consists of $\mathsf{mpk} := \big(\mathsf{PP}_{core}, \{\mathbf{CT}[l_2, l_1]\}_{l_2,l_1 \in \{1,\ldots,n\}}\big)$ while the master secret key is $\mathsf{msk} := \big(\hat{v}, \hat{\mathbf{w}}, \hat{\mathbf{h}}\big)$. For each $l_1 \in \{1, \ldots, n\}$, it will be convenient to view $(\mathsf{PP}_{core}, \mathsf{msk})$ as a vector of HPE master key pairs $(\mathsf{mpk}[l_1], \mathsf{msk}[l_1])$, with

$$\mathsf{mpk}[l_1] = \big(v, \mathbf{w}[l_1], \{\mathbf{h}[i_1, i_2, l_1]\}_{i_1 \in \{1,\ldots,d\}, i_2 \in \{0,\ldots,\mu\}}\big)$$
$$\mathsf{msk}[l_1] = \big(\hat{v}, \hat{\mathbf{w}}[l_1], \{\hat{\mathbf{h}}[i_1, i_2, l_1]\}_{i_1 \in \{1,\ldots,d\}, i_2 \in \{0,\ldots,\mu\}}\big).$$

HF.Kg$\big(\mathsf{pms}, \mathsf{msk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)\big)$**:** to generate a key for an identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in \mathsf{IdSp}$, parse msk as $\big(\hat{v}, \hat{\mathbf{w}}, \hat{\mathbf{h}}\big)$ and id_{i_1} as $\mathsf{id}_{i_1}[1] \ldots \mathsf{id}_{i_1}[\mu]$ for $i_1 = 1$ to ℓ. Choose $\mathbf{r}_w, \mathbf{r}_1, \ldots, \mathbf{r}_\ell \xleftarrow{R} (\mathbb{Z}_p^*)^n$. For each $l_1 \in \{1, \ldots, n\}$, compute the decryption component $\mathbf{SK}_D = (\mathbf{D}, \mathbf{D}_w, \{\mathbf{D}_{i_1}\}_{i_1=1}^{\ell})$ of the key as

$$\mathbf{D}[l_1] = \prod_{i_1=1}^{\ell} \Big(\prod_{i_2=1}^{\mu} \hat{\mathbf{h}}[i_1, i_2, l_1]^{\mathsf{id}_{i_1}[i_2]} \Big)^{\mathbf{r}_{i_1}[l_1]} \cdot \hat{\mathbf{w}}[l_1]^{\mathbf{r}_w[l_1]},\ \mathbf{D}_w[l_1] = \hat{v}^{\mathbf{r}_w[l_1]},\ \mathbf{D}_{i_1}[l_1] = \hat{v}^{\mathbf{r}_{i_1}[l_1]} \tag{7}$$

and the delegation component $\mathbf{SK}_{DL} = \big(\{\mathbf{K}[j, k, l_1]\}_{j,k,l_1},\ \{\mathbf{L}[j, l_1]\}_{j,l_1},\ \{\mathbf{L}[j, k, i_1, l_1]\}_{j,k,i_1,l_1}, \{\mathbf{L}_w[j, k, l_1]\}_{j,k,l_1}\big)$, with $j \in \{\ell + 1, \ldots, d\}$, $k \in \{1, \ldots, \mu\}$ and $i_1 \in \{1, \ldots, \ell\}$ as

$$\mathbf{K}[j, k, l_1] = \prod_{i_1=1}^{\ell} \Big(\prod_{i_2=1}^{\mu} \hat{\mathbf{h}}[i_1, i_2, l_1]^{\mathsf{id}_{i_1}[i_2]} \Big)^{\mathbf{s}[j,k,i_1,l_1]} \cdot \hat{\mathbf{h}}[j, k, l_1]^{\mathbf{s}'[j,l_1]} \cdot \hat{\mathbf{w}}[l_1]^{\mathbf{s}_w[j,k,l_1]},$$

$$\mathbf{L}[j, l_1] = \hat{v}^{\mathbf{s}'[j,l_1]}, \qquad \mathbf{L}[j, k, i_1, l_1] = \hat{v}^{\mathbf{s}[j,k,i_1,l_1]} \quad \text{and } \mathbf{L}_w[j, k, l_1] = \hat{v}^{\mathbf{s}_w[j,k,l_1]}.$$

Output $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)} = \big(\mathbf{SK}_D, \mathbf{SK}_{DL}\big)$.

HF.Del$\big(\mathsf{pms}, \mathsf{mpk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}, \mathsf{id}_{\ell+1}\big)$**:** parse $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$ as a HF private key of the form $(\mathbf{SK}_D, \mathbf{SK}_{DL})$, and $\mathsf{id}_{\ell+1}$ as a string $\mathsf{id}_{\ell+1}[1] \ldots \mathsf{id}_{\ell+1}[\mu] \in \Sigma_{\mathsf{ID}}$.

1. For $l_1 = 1$ to n, define ℓ-th level HPE keys $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}[l_1] = (\mathbf{SK}_D[l_1], \mathbf{SK}_{DL}[l_1])$ where

$$\mathbf{SK}_D[l_1] = (\mathbf{D}[l_1], \mathbf{D}_w[l_1], \{\mathbf{D}_{i_1}[l_1]\}_{i_1=1}^{\ell})$$
$$\mathbf{SK}_{DL}[l_1] = \big(\{\mathbf{K}[j, k, l_1]\}_{j,k},\ \{\mathbf{L}[j, l_1]\}_{j},\ \{\mathbf{L}[j, k, i_1, l_1]\}_{j,k,i_1}, \{\mathbf{L}_w[j, k, l_1]\}_{j,k}\big).$$

2. For $l_1 = 1$ to n, run $\mathbf{Delegate}(\mathsf{mpk}[l_1], (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}[l_1], \mathsf{id}_{\ell+1})$ (as specified in Section 3.2) to get $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell,\mathsf{id}_{\ell+1})}[l_1] = (\mathbf{SK}'_D[l_1], \mathbf{SK}'_{DL}[l_1])$.

Finally return $\{\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell,\mathsf{id}_{\ell+1})}[l_1]\}_{l_1=1}^{n}$.

HF.Eval$\big(\mathsf{pms}, \mathsf{mpk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), X\big)$**:** Given a n-bit input $X = x_1 \ldots x_n \in \{0,1\}^n$, for $i_1 = 1$ to ℓ, parse id_{i_1} as $\mathsf{id}_{i_1}[1] \ldots \mathsf{id}_{i_1}[\mu]$. For $l_1 = 1$ to n, do the following.

1. For each $l_2 \in \{1, \ldots, n\}$, compute modified HPE ciphertexts by defining

$$\mathbf{C}_{\mathsf{id}}[i_1, l_2, l_1] = \prod_{i_2=1}^{\mu} \mathbf{C}[i_1, i_2, l_2, l_1]^{\mathsf{id}_{i_1}[i_2]} = \Big(\mathbf{h}[i_1, 0, l_1]^{\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle \cdot \Delta(l_2, l_1)} \cdot \prod_{i_2=1}^{\mu} \mathbf{h}[i_1, i_2, l_1]^{\mathsf{id}_{i_1}[i_2]} \Big)^{\mathbf{s}[l_2]}$$

for each $i_1 \in \{1,\ldots,\ell\}$, $l_1, l_2 \in \{1,\ldots,n\}$. The modified ciphertexts are

$$\mathbf{CT}_{\mathsf{id}}[l_2, l_1] = \big(\mathbf{J}[l_2], \{\mathbf{CT}_{\mathsf{id}}[i_1, l_2, l_1]\}_{i_1=1}^{\ell}\big) \in \mathbb{G}^{\ell+1}. \quad (8)$$

The resulting $\{\mathbf{CT}_{id}[l_2, l_1]\}_{l_2,l_1 \in \{1,\ldots,n\}}$ thus form a $n \times n$ matrix of anonymous HIBE ciphertexts for the identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$.

2. Compute $C_{\mathsf{id},v} = \prod_{l_2=1}^{n} \mathbf{J}[l_2]^{x_{l_2}} = v^{\langle \mathbf{s}, X\rangle}$, $\mathbf{CT}_{\mathsf{id},w}[l_1] = \prod_{l_2=1}^{n} \mathbf{C}_w[l_2, l_1]^{x_{l_2}} = \mathbf{w}[l_1]^{\langle s, X\rangle}$ and

$$\mathbf{CT}_{\mathsf{id}}[i_1, l_1] = \prod_{l_2=1}^{n} \mathbf{C}_{\mathsf{id}}[i_1, l_2, l_1]^{x_{l_2}} = \mathbf{h}[i_1, 0, l_1]^{\mathbf{s}[l_1]\cdot x_{l_1}\cdot\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1}\rangle} \cdot \Big(\prod_{i_2=1}^{\mu} \mathbf{h}[i_1, i_2, l_1]^{\mathsf{id}_{i_1}[i_2]}\Big)^{\langle \mathbf{s}, X\rangle} \quad (9)$$

Then, output $C = \big(C_{\mathsf{id},v}, \{\mathbf{CT}_{\mathsf{id},w}[l_1]\}_{l_1=1}^{n}, \{\mathbf{CT}_{\mathsf{id}}[i_1, l_1]\}_{i_1 \in \{1,\ldots,\ell\}, l_1 \in \{1,\ldots,n\}}\big) \in \mathbb{G}^{n+1+n\times\ell}$. (10)

$\mathsf{HF.Inv}\big(\mathsf{pms}, \mathsf{mpk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}, C\big)$: parse the decryption component $\mathbf{SK}_D$ of the private key as a tuple of the form $(\mathbf{D}, \mathbf{D}_w, \mathbf{D}_{\bar{w}}, \{\mathbf{D}_{i_1}\}_{i_1=1}^{\ell})$ and the output C as per (10). Then, for $l_1 = 1$ to n, set $x_{l_1} = 0$ if

$$e(C_{\mathsf{id},v}, \mathbf{D}[l_1]) \cdot e(\mathbf{CT}_{\mathsf{id},w}[l_1], \mathbf{D}_w[l_1])^{-1} \cdot \prod_{i_1=1}^{\ell} e(\mathbf{CT}_{\mathsf{id}}[i_1, l_1], \mathbf{D}_{i_1}[l_1])^{-1} = 1_{\mathbb{G}_T}. \quad (11)$$

Otherwise, set $x_{l_1} = 1$. Eventually, return $X = x_1 \ldots x_n \in \{0,1\}^n$.

From (9), we notice that, with overwhelming probability, if there exists some $i_1 \in \{1,\ldots,d\}$ such that $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1}\rangle \neq 0$, relation (11) is satisfied if and only if $x_{l_1} = 0$. Indeed, in this case, the output (10) is distributed as a vector of n Boneh-Boyen HIBE ciphertexts (in their anonymous variant considered in [20]). These ciphertexts correspond to the same encryption exponent $\langle \mathbf{s}, X\rangle$ and are generated under n distinct master public keys sharing the same component $v \in \mathbb{G}$.

When the function is implemented in injective mode, the auxiliary input consists of a vector $\mathbf{y}^{(0)} = [(1,0,\ldots,0)|\ldots|(1,0,\ldots,0)] \in \mathbb{Z}_p^{d\cdot\mu}$. Since $\mathsf{id}_{i_1}[1] = 1$ for each i_1, this guarantees injectivity since $\langle \mathbf{y}_{i_1}^{(0)}, \mathsf{id}_{i_1}\rangle \neq 0$ for each i_1. In the partially lossy mode, we have $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1}\rangle = 0$ for each $i_1 \in \{1,\ldots,\ell\}$ with non-negligible probability, which leads to high non-injectivity.

4.2 Security Analysis

To analyze the security of the scheme (w.r.t. the new definition, in Section 2.1), in both the adaptive and the selective cases, we define two experiments, RL_0 and RL_n. In both of them, $\mathsf{HF.Setup}$ is run and the public parameters are given to the adversary $\mathcal{A}$. Algorithm $\mathsf{HF.MKg}$ is run with auxiliary input $\mathbf{y}^{(0)} = [(1,0,\ldots,0)|\ldots|(1,0,\ldots,0)]$ in RL_0, and auxiliary input $\mathbf{y}^{(1)} = [\mathbf{y}_1^{(1)}|\ldots|\mathbf{y}_d^{(1)}]$ in RL_n, where $\mathbf{y}^{(1)}$ is produced by an auxiliary input generator $\mathsf{Aux}(\mathsf{id})$ taking as input a special hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$. The master public key mpk is given to the $\mathcal{A}$. $\mathcal{A}$ can request secret keys for identities id, which will be answered using $\mathsf{HF.Kg}$ and $\mathsf{HF.Del}$ and will be added to IS, initialized to $IS = \{\emptyset\}$. Also, $\mathcal{A}$ will output a hierarchical identity $\mathsf{id}^\star$. Finally, $\mathcal{A}$ will output a guess $d_{\mathcal{A}}$. Both in RL_0 and RL_n, the experiment will halt and output $d' = 0$ if: a) for any $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in IS$ we have that $\langle y_{i_1}^{(1)}, \vec{\mathsf{id}}_{i_1}\rangle = 0$ for each $i_1 \in \{1,\ldots,\ell\}$, or b) if $\langle y_{i_1}^{(1)}, \vec{\mathsf{id}}_{i_1}^{\star}\rangle \neq 0$ for some $i_1 \in \{1,\ldots,\ell^\star\}$. If the experiment has not aborted, it will output the bit $d' = d_{\mathcal{A}}$. The result in the following lemma will be used to prove that HF enjoys partial lossiness.

Lemma 1. *Under the $\mathcal{P}$-BDH$_1$ and DDH$_2$ assumptions, the experiments RL_0 and RL_n return 1 with nearly identical probabilities. Namely, there exist PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$|\Pr[RL_0 \Rightarrow 1] - \Pr[RL_n \Rightarrow 1]| \leq n \cdot \big((d \cdot \mu + 1) \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_1}(\mathcal{B}_1) + q \cdot \mathbf{Adv}^{\mathcal{P}\text{-DDH}_2}(\mathcal{B}_2)\big),$$

where q is the number of "Reveal-key" queries made by $\mathcal{A}$.(The proof is in Appendix B.1).

Adaptive-id Security. The details on the security analysis of HF with respect to selective adversaries are given in Appendix B.2. For adaptive security we consider our construction of HF with a restricted identity space, namely taking $\Sigma_{ID} = \{(1, \mathbf{x}) : \mathbf{x} \in \{0,1\}^{\mu-1}\}$. Define a sibling LHF with $\mathsf{AuxSp} = \mathbb{Z}_p^{\mu \cdot d}$, where the auxiliary input $\mathbf{y}^{(1)} = [\mathbf{y}_1^{(1)} | \ldots | \mathbf{y}_d^{(1)}]$, for any i_1 from 1 to d, is defined as

$$y'_{i_1} \xleftarrow{R} \{0, \ldots, 2q-1\},\ \xi_{i_1} \xleftarrow{R} \{0, \ldots, \mu+1\},\ \mathbf{y}_{i_1}^{(1)}[1] = y'_{i_1} - 2\xi_{i_1} q,\ \{\mathbf{y}_{i_1}^{(1)}[i] \xleftarrow{R} \{0, \ldots, 2q-1\}\}_{i=2}^{\mu}.$$

The pre-output stage $\mathcal{P}$ associated to the scheme HF in the adaptive case will be the artificial abort used in [35]: if IS is the set of queried identities (only taking into account "Reveal key" queries) and $\mathsf{id}^\star$ is the challenge identity, let $E(IS, \mathsf{id}^\star)$ be the event that the REAL or the LOSSY experiments set d_1 to be 1, and let $\eta(IS, \mathsf{id}^\star) = \Pr[E(IS, \mathsf{id}^\star)]$. We prove in Appendix B.3:

Lemma 2. $\eta_{low} = 1/\left(2 \cdot (2q\mu)^d\right) \leq \eta(IS, \mathsf{id}^\star)$

$\mathcal{P}$ computes an approximation $\eta'(IS, \mathsf{id}^\star)$ of $\eta(IS, id^\star)$, by using $O(\zeta^{-2} \ln(\zeta^{-1}) \eta_{low}^{-1} \ln(\eta_{low}^{-1}))$ samples, for a non-negligible ζ, which is part of the input to $\mathcal{P}$. If $\eta'(IS, \mathsf{id}^\star) \leq \eta_{low}$, then d_2 is always set to 1. If $\eta'(IS, \mathsf{id}^\star) > \eta_{low}$, then $d_2 = 0$ with probability $1 - \eta_{low}/\eta'(IS, \mathsf{id}^\star)$ and $d_2 = 1$ with probability $\eta_{low}/\eta'(IS, \mathsf{id}^\star)$. Adaptive security of HF is established in the following theorem.

Theorem 3. *Let $n > \log p$ and let $\omega = n - \log p$. Let HF be the HIB-TDF with parameters $n, d, \mu, \Sigma_{ID} = \{(1, \mathbf{x}) : \mathbf{x} \in \{0,1\}^{\mu-1}\}, \mathsf{IdSp} = \Sigma_{ID}^{(\leq d)}$. Let LHF be the sibling with auxiliary input as above, let the pre-output stage associated to HF be the one specified above. In front of adaptive-id adversaries that make a maximal number of $q \leq p/(2\mu)$ queries, HF is (ω, δ)-partially lossy, for $\delta = 13/\left(32 \cdot (2q\mu)^d\right)$.*

Specifically regarding condition (i), for any such adaptive-id adversary $\mathcal{A}$ there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that $\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\omega,\zeta}(\mathcal{A}) \leq 2n \cdot \big((d \cdot \mu + 1) \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_1}(\mathcal{B}_1) + q \cdot \mathbf{Adv}^{\mathcal{P}\text{-DDH}_2}(\mathcal{B}_2)\big)$, where the running time of $\mathcal{B}_1$ and $\mathcal{B}_2$ is that of $\mathcal{A}$ plus a $O(\zeta^{-2} \ln(\zeta^{-1}) \eta_{low}^{-1} \ln(\eta_{low}^{-1}))$ overhead, with $\eta_{low} = 1/\left(2 \cdot (2q\mu)^d\right)$. (The proof is given in Appendix B.4).

References

1. S. Agrawal, D. Boneh, X. Boyen. Efficient Lattice (H)IBE in the Standard Model. *Eurocrypt'10*, *LNCS* 6110, pp. 553–572, 2010.
2. S. Agrawal, D. Boneh, X. Boyen. Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. *Crypto'10*, *LNCS* 6223, pp. 98–115, 2010.
3. S. Agrawal, D. Freeman, V. Vaikuntanathan. Functional Encryption for Inner Product Predicates from Learning with Errors. *Asiacrypt'11*, *LNCS* 7073, pp. 21–40, 2011.
4. R. Anderson. Two Remarks on Public Key Cryptology. Invited lecture, *ACM Conference on Computer and Communications Security*, 1997.
5. M. Bellare, A. Boldyreva, A. O'Neill. Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In *Crypto'07*, *LNCS* 4622, pp. 535–552, 2007.

6. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, S. Yilek. Hedged Public-Key Encryption: How to Protect against Bad Randomness. In *Asiacrypt'09*, *LNCS* 5912, pp. 232-249, 2009. Full version available at `http://eprint.iacr.org/2012/220` .
7. M. Bellare, M. Fischlin, A. O'Neill and T. Ristenpart. Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In *Crypto'08*, LNCS 5157, 2008.
8. M. Bellare, D. Hofheinz, S. Yilek. Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening. In *Eurocrypt'09*, *LNCS* 5479, pp. 1–35, 2009.
9. M. Bellare, E. Kiltz, C. Peikert, B. Waters. Identity-Based (Lossy) Trapdoor Functions and Applications. In *Eurocrypt'12*, *LNCS* 7237, pp. 228-245, 2012
10. A. Boldyreva, S. Fehr, A. O'Neill. On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. In *Crypto'08*, LNCS 5157, 2008.
11. D. Boneh, B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In *4th Theory of Cryptography Conference* (TCC 2007), *LNCS* 4392, pp. 535–554, 2007.
12. X. Boyen, B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Crypto'06*, *LNCS* 4117, pp. 290–307, 2006.
13. X. Boyen, B. Waters. Shrinking the Keys of Discrete-Log-Type Lossy Trapdoor Functions. In *ACNS'10*, *LNCS* 6123, pp. 35–52, 2010.
14. Z. Brakerski, G. Segev. Better Security for Deterministic Public-Key Encryption: The Auxiliary-Input Setting. In *Crypto'11*, *LNCS* 6841, pp. 543–560, 2011.
15. R. Canetti, S. Halevi, J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *Eurocrypt'03*, *LNCS* 2656, pp. 254–271, 2003.
16. R. Canetti, S. Halevi, J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *Eurocrypt'04*, *LNCS* 3027, pp. 207–222, 2004.
17. D. Cash, D. Hofheinz, E. Kiltz, C. Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. In *Eurocrypt'10*, *LNCS* 6110, pp. 523–552, 2004.
18. Y. Dodis, L. Reyzin, A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Eurocrypt'04*, *LNCS* 3027, pp. 523–540, 2004.
19. Y. Dodis, A. Smith. Entropic Security and the Encryption of High Entropy Messages. In *TCC'05*, LNCS 3378, 2005.
20. L. Ducas. Anonymity from Asymmetry: New Constructions for Anonymous HIBE. In *CT-RSA'10*, *LNCS* 5985, pp. 148–164, 2010.
21. D. Freeman, O. Goldreich, E. Kiltz, A. Rosen, G. Segev. More Constructions of Lossy and Correlation-Secure Trapdoor Functions. In *PKC'10*, *LNCS* 6056, pp. 279–295, 2010.
22. C. Gentry, A. Silverberg. Hierarchical ID-Based Cryptography. In *Asiacrypt'02*, *LNCS* 2501, pp. 548–566, 2002.
23. V. Goyal, O. Pandey, A. Sahai, B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS'06*, pp. 89–98, 2006.
24. J. Håstad, R. Impagliazzo, L. Levin, M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, vol. 28(4), pp. 1364–1396, 1999.
25. B. Hemenway, B. Libert, R. Ostrovsky, D. Vergnaud. Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security. In *Asiacrypt'11, LNCS* 7073, pp. 70-88, 2011.
26. B. Hemenway, R. Ostrovsky. Lossy Trapdoor Functions from Smooth Homomorphic Hash Proof Systems. In *Electronic Colloquium on Computational Complexity (ECCC)* 16: 127, 2009.
27. D. Hofheinz. All-But-Many Lossy Trapdoor Functions. In *Eurocrypt'12*, *LNCS* 7237, pp. 209–227, 2012.
28. J. Katz, A. Sahai, B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In *Eurocrypt'08*, *LNCS* 4965, pp. 146-162, 2008.
29. P. Mol, S. Yilek. Chosen-Ciphertext Security from Slightly Lossy Trapdoor Functions. In *PKC'10*, *LNCS* 6056, pp. 296–311, 2010.
30. T. Okamoto, K. Takashima. Hierarchical Predicate Encryption for Inner-Products.In *Asiacrypt'09*, *LNCS* 5912, pp. 214–231, 2009.
31. C. Peikert, B. Waters. Lossy trapdoor functions and their applications. In *STOC'08*, ACM Press, pp. 187–196, 2008.
32. A. Sahai, B. Waters. Fuzzy Identity-Based Encryption In *Eurocrypt'05*, *LNCS* 3494, pp. 457–473, 2005.
33. E. Shi, B. Waters. Delegating Capabilities in Predicate Encryption Systems. In *ICALP'08*, *LNCS* 5126, pp. 560–578, 2008.
34. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Crypto'84*, *LNCS* 196, pp. 47–53, 1984.
35. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, pp. 114–127, 2005.

36. H. Wee. Dual Projective Hashing and its Applications - Lossy Trapdoor Functions and More. In *Eurocrypt'12*, *LNCS* 7237, pp. 246–262, 2012.
37. X. Xie, R. Xue, R. Zhang. Deterministic Public Key Encryption and Identity-Based Encryption from Lattices in the Auxiliary-Input Setting. In *SCN'12*, *LNCS* series, to appear, 2012. Cryptology ePrint Archive: Report 2012/463

A Deferred Definitions and Proofs for Hierarchical Predicate Encryption

A.1 Definitions for Hierarchical Predicate Encryption

A tuple of integers $\vec{\nu} = (\mu_1, d; \nu_1, \ldots, \nu_d)$ such that $\nu_0 = 0 \leq \nu_1 < \nu_2 < \cdots, < \nu_d = \mu_1$ is called a *format hierarchy* of depth d. Such a hierarchy is associated with attribute spaces $\{\Sigma_i\}_{i=0}^{d}$. In our setting, we set $\Sigma_i = \mathbb{Z}_p^{\nu_i - \nu_{i-1}} \backslash \{\vec{0}\}$ for $i = 1$ to d, for some $p \in \mathbb{N}$, and also define the universe of hierarchical attributes as $\Sigma := \cup_{i=1}^{d}(\Sigma_1 \times \cdots \times \Sigma_i)$. For vectors $\{\vec{X}_i \in \Sigma_i\}_{i \in \{1,\ldots,d\}}$, we will consider the (inner-product) hierarchical predicate $f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1, \ldots, \vec{Y}_\kappa) = 1$ iff $\ell \leq \kappa$ and $\vec{X}_i \cdot \vec{Y}_i = 0$ for all $i \in \{1, \ldots, \ell\}$. The space of hierarchical predicates is defined to be

$$\mathcal{F} = \{f_{(\vec{X}_1,\ldots,\vec{X}_\ell)} | \{\vec{X}_i \in \Sigma_i\}_{i \in \{1,\ldots,\ell\}}\}$$

and the integer κ (resp. ℓ) is called the depth (resp. the level) of $(\vec{Y}_1, \ldots, \vec{Y}_\kappa)$ (resp. $(\vec{X}_1, \ldots, \vec{X}_\ell)$). Most of the time we will assume that $\nu_i / \nu_{i-1} = \mu$ for any i, and to specify the format hierarchy we will only have to give μ and d.

Let $\vec{\nu} = (\mu_1, d; \nu_1, \ldots, \nu_d)$ be a format hierarchy. A hierarchical predicate encryption (HPE) scheme for a predicate family $\mathcal{F}$ consists of these algorithms.

Setup$(\varrho, \vec{\nu})$**:** takes as input a security parameter $\varrho \in \mathbb{N}$ and a format hierarchy $\vec{\nu} = (n_1, d; \nu_1, \ldots, \nu_d)$. It outputs a master secret key msk and a master public key mpk that includes the description of a hierarchical attribute space Σ.

Keygen$\big(\mathsf{msk}, (\vec{X}_1, \ldots, \vec{X}_\ell)\big)$**:** takes as input predicate vectors $(\vec{X}_1, \ldots, \vec{X}_\ell) \in \Sigma_1 \times \cdots \times \Sigma_\ell$ and the master secret key msk. It outputs a private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$.

Encrypt$\big(\mathsf{mpk}, (\vec{Y}_1, \ldots, \vec{Y}_\kappa), M\big)$: takes as input attribute vectors $(\vec{Y}_1, \ldots, \vec{Y}_\kappa) \in \Sigma_1 \times \cdots \times \Sigma_k$, the master public key mpk and a message M. It outputs a ciphertext C.

Decrypt$\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, C\big)$**:** takes in a private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$ for the vectors $(\vec{X}_1, \ldots, \vec{X}_\ell)$, the master public key mpk and a ciphertext C. It outputs a plaintext M or $\perp$.

Delegate$\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, \vec{X}_{\ell+1}\big)$**:** takes in a ℓ-th level private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$, the corresponding vectors $(\vec{X}_1, \ldots, \vec{X}_\ell)$ and a vector $\vec{X}_{\ell+1}$. It outputs a $(\ell+1)$-th level private key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell,\vec{X}_{\ell+1})}$.

Correctness mandates that, for any message M and hierarchical vectors $(\vec{X}_1, \ldots, \vec{X}_\ell)$, $(\vec{Y}_1, \ldots, \vec{Y}_\kappa)$, we have $\textbf{Decrypt}\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, \textbf{Encrypt}(\mathsf{mpk}, (\vec{Y}_1, \ldots, \vec{Y}_\kappa, M)\big) = M$ whenever the condition $f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1, \ldots, \vec{Y}_\kappa) = 1$ is satisfied.

Following [30], we write $f' \leq f$ to express that the predicate vector of f is a prefix of that for f', meaning that f' is at least as constraining as f.

Definition 2. *A Hierarchical Predicate Encryption scheme is* **selectively weakly attribute-hiding** *if no PPT adversary has non-negligible advantage in the following game:*

1. *The adversary $\mathcal{A}$ chooses a format hierarchy $\vec{\nu} = (n_1, d; \nu_1, \ldots, \nu_d)$ and vectors $(\vec{Y}_1^0, \ldots, \vec{Y}_{d^\star}^0)$, $(\vec{Y}_1^1, \ldots, \vec{Y}_{d^\star}^1)$, for some $d^\star \leq d$. The challenger generates a master key pair $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathbf{Setup}(\varrho, \vec{\nu})$ and mpk is given to $\mathcal{A}$.*
2. *$\mathcal{A}$ is allowed to make a number of adaptive queries.*
   - **Create-key**: *$\mathcal{A}$ provides a predicate $f \in \mathcal{F}$ and the challenger creates a private key SK_f for f without revealing it to $\mathcal{A}$.*
   - **Create-delegated-key**: *$\mathcal{A}$ chooses a private key that was previously created for some predicate f and also specifies another predicate $f' \leq f$ that f is a prefix of. The challenger then computes a delegated key $SK_{f'}$ for f' without revealing it to $\mathcal{A}$.*
   - **Reveal-key**: *$\mathcal{A}$ asks the challenger to give out a previously created key.*

   For each Reveal-key query $(\vec{X}_1, \ldots, \vec{X}_\ell)$, it is required that

$$f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1^0, \ldots, \vec{Y}_{d^\star}^0) = f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1^1, \ldots, \vec{Y}_{d^\star}^1) = 0.$$

3. *$\mathcal{A}$ outputs messages M_0, M_1. Then, the challenger chooses $\beta \xleftarrow{R} \{0,1\}$ and computes a challenge ciphertext $C^\star = \mathbf{Encrypt}\big(\mathsf{mpk}, (\vec{Y}_1^\beta, \ldots, \vec{Y}_{d^\star}^\beta), M_\beta\big)$, which is sent to $\mathcal{A}$.*
4. *$\mathcal{A}$ makes further private key queries for hierarchical vectors $(\vec{X}_1, \ldots, \vec{X}_\ell)$ under the same restriction as above.*
5. *$\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$ and wins if $\beta' = \beta$.*

$\mathcal{A}$'s advantage is quantified as the distance $\mathbf{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - 1/2|$.

A.2 Correctness of the HIPE scheme

Lemma 3. *The HIPE scheme of Section 3.2 is correct. Namely, for any message M and any vectors $\vec{X}_1, \ldots, \vec{X}_\ell$, $\vec{Y}_1, \ldots, \vec{Y}_\kappa$, we have*

$$\mathbf{Decrypt}\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, \mathbf{Encrypt}(\mathsf{mpk}, (\vec{Y}_1, \ldots, \vec{Y}_\kappa), M\big) = M$$

whenever $f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1, \ldots, \vec{Y}_\kappa) = 1$. This fact does not depend on whether the key $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}$ was created using **Delegate** *or* **Keygen**.

Proof. Let $SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)} = \big(SK_D, SK_{DL}\big)$ be obtained by running $\mathbf{Keygen}\big(\mathsf{msk}, (\vec{X}_1, \ldots, \vec{X}_\ell)\big)$, where each attribute vector is $\vec{X}_{i_1} = (x_{i_1,1}, \ldots, x_{i_1,\mu}) \in \mathbb{Z}_p^\mu$, for each $i_1 \in \{1, \ldots, \ell\}$. Let us write the decryption component of the key as $SK_D = (D, D_w, \{D_{i_1}\}_{i_1=1}^\ell)$.

Let $C = \big(C_0, C_v, C_w, \{C_{i_1,i_2}\}_{i_1 \in \{1,\ldots,\kappa\},\ i_2 \in \{1,\ldots,\mu\}}\big)$ be the output of the encryption algorithm $\mathbf{Encrypt}\big(\mathsf{mpk}, (\vec{Y}_1, \ldots, \vec{Y}_\kappa), M\big)$, for vectors $\vec{Y}_1 = (y_{1,1}, \ldots, y_{1,\mu})$, …, $\vec{Y}_\kappa = (y_{\kappa,1}, \ldots, y_{\kappa,\mu})$.

Since $f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1, \ldots, \vec{Y}_\kappa) = 1$, and by the definition of hierarchical inner-product predicates, we know that $\ell \leq \kappa$ and $\vec{X}_i \cdot \vec{Y}_i = 0$ for all $i \in \{1, \ldots, \ell\}$. Therefore, when the decryption protocol $\mathbf{Decrypt}\big(\mathsf{mpk}, (\vec{X}_1, \ldots, \vec{X}_\ell), SK_{(\vec{X}_1,\ldots,\vec{X}_\ell)}, C\big)$ computes C_{i_1}, for each $i_1 \in \{1, \ldots, \ell\}$, the obtained value equals $C_{i_1} = \big(\prod_{i_2=1}^\mu h_{i_1,i_2}^{x_{i_1,i_2}}\big)^s$. The decryption protocol computes then the pairing of this element C_{i_1} with $D_{i_1} = \hat{v}^{r_{i_1}}$. Multiplying all these pairings, for indices $i_1 \in \{1, \ldots, \ell\}$, one obtains $e\big(v^s\ ,\ \prod_{i_1=1}^\ell \big(\prod_{i_2=1}^\mu \hat{h}_{i_1,i_2}^{x_{i_1,i_2}}\big)^{r_{i_1}}\big)$. This value is canceled out with one of the factors of $e(C_v, D)$, when computing the final decryption operation $C_0 \cdot e(C_v, D)^{-1} \cdot e(C_w, D_w) \cdot \prod_{i_1=1}^\ell e(C_{i_1}, D_{i_1})$. The other two factors of $e(C_v, D)$ are $e(v^s, \hat{g}^\alpha)$ and $e(v^s, \hat{w}^{r_w})$, which cancel out the factor $e(g, \hat{v})^{\alpha s}$,

contained in $C_0 = M \cdot e(g, \hat{v})^{\alpha s}$, and the factor $e(C_w, D_w) = e(w^s, \hat{v}^{r_w})$, respectively. Therefore, the final computation of the decryption protocol results in the plaintext M contained in C_0.

In this way, we have proved that the encryption and decryption protocols work correctly when the original secret keys (resulting from Keygen) are used. The fact that the decryption protocol works fine also with delegated secret keys (resulting from Delegate) is a consequence of Lemma 4, inside the proof of Theorem 2. If a delegated secret key could lead to an incorrect decryption, then this fact could be used to distinguish original secret keys from delegated ones, which would contradict the statement of Lemma 4.

A.3 Proof of Theorem 2

The proof considers a sequence of games starting with the real game and ending with a game where the adversary has no advantage and wins with probability exactly 1/2.

For each i, we denote by S_i the event that the adversary wins in Game_i. In the whole sequence of games, we call $d^\star$ the depth of the challenge hierarchical vectors $(\vec{Y}_1^0, \ldots, \vec{Y}_{d^\star}^0)$ and $(\vec{Y}_1^1, \ldots, \vec{Y}_{d^\star}^1)$ and

$$C^\star = \left(C_0^\star, C_v^\star, C_w^\star, \{C_{i_1,i_2}^\star\}_{i_1 \in \{1,\ldots,d^\star\}, i_2 \in \{1,\ldots,\mu\}}\right)$$

denotes the challenge ciphertext.

Game_0: is the real attack game at the end of which the challenger outputs 1 in the event, called S_0, that the adversary $\mathcal{A}$ manages to output $\beta' \in \{0,1\}$ such that $\beta' = \beta$, where $\beta \in \{0,1\}$ is the challenger's hidden bit in the challenge phase. If $\beta' \neq \beta$, the challenger outputs 0.

Game_1: is identical to Game_0 with the difference that the challenger always answers private key queries by returning fresh private keys (*i.e.*, keys produced by **Keygen**) instead of deriving those keys using the delegation algorithm.

Game_2: is like Game_1 but the challenge ciphertext $C^\star$ is now an encryption under $(\vec{Y}_1^\beta, \ldots, \vec{Y}_{d^\star}^\beta)$ of a random plaintext $M \xleftarrow{R} \mathbb{G}_T$, which is chosen independently of M_0 and M_1.

Game_3: is identical to Game_2 with the difference that, in the challenge ciphertext, $C_w^\star$ is replaced by a random group element chosen uniformly and independently in $\mathbb{G}$.

$\text{Game}_{4,i,j}$ $(1 \leq i \leq d^\star,\ 1 \leq j \leq \mu)$: is identical to Game 3 with the difference that, in the challenge ciphertext, $C_{i_1,i_2}^\star$ are replaced by random elements of $\mathbb{G}$ if $i_1 < i$ or $(i = i_1) \wedge (i_2 \leq j)$. Other group elements (*i.e.*, for which $i > i_1$ or $(i = i_1) \wedge (i_2 > j)$) are still computed as in a normal challenge ciphertext.

In $\text{Game}_{4,d^\star,\mu}$, it is easy to see that the adversary $\mathcal{A}$ cannot guess $\beta \in \{0,1\}$ with higher probability than $\Pr[S_{4,d^\star,\mu}] = 1/2$ since the challenge ciphertext $C^\star$ is completely independent of $\beta \in \{0,1\}$. □

Lemma 4. *Game_0 and Game_1 are computationally indistinguishable if the DDH_2 assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$.*

Proof. The lemma will be proved by a hybrid argument. We define $\text{Game}_{0,i}$ for all $0 \leq i \leq q$. $\text{Game}_{0,i}$ differs from Game_0 in the fact that, when the adversary issues the first i delegation queries, instead of generating the delegated keys faithfully using the Delegate algorithm, the challenger calls the Keygen algorithm to generate these delegated keys. For all the remaining queries, the challenger computes keys and responds faithfully as in Game_0. Under the above definition, $\text{Game}_{0,0}$ is the same as Game_0 and $\text{Game}_{0,q}$ is the same as Game_1. We will prove that $\text{Game}_{0,\kappa}$ is indistinguishable from $\text{Game}_{0,\kappa+1}$ for all $0 \leq \kappa \leq q-1$. To this end, we will proceed similarly to [33] and rely on a generalized version (called GDDH hereafter) of the DDH problem in $\hat{\mathbb{G}}$.

Given a group generator GG, define the following distribution $P(\varrho)$:

$$
\begin{array}{l}
\big(p, (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), e\big) \xleftarrow{R} \mathsf{GG}(\varrho, 1), \\
g \xleftarrow{R} \mathbb{G}, \quad \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, \\
\hat{h}_1, \hat{h_2}, \ldots, \hat{h_\ell} \xleftarrow{R} \hat{\mathbb{G}} \\
\tau \xleftarrow{R} \mathbb{Z}_p \\
X \leftarrow \big(\big(p, (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), e\big),\ g, \hat{g}, \hat{h_1}, \hat{h_2}, \ldots, \hat{h_\ell}\big) \\
Q \leftarrow \big(\hat{h_1^\tau}, \hat{h_2^\tau}, \ldots, \hat{h_\ell^\tau}\big), \\
\text{Output } (X, Q)
\end{array}
$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the above problem:

$$\ell\text{-GDDH } \mathsf{Adv}_{\mathsf{GG},\mathcal{A}}(\varrho) := |\Pr[\mathcal{A}(X, Q) = 1] - \Pr[\mathcal{A}(X, R)]|$$

where $(X, Q) \leftarrow P(\varrho)$ and $R \leftarrow \hat{\mathbb{G}}^\ell$. It is immediate[8] that GDDH is not easier than DDH_2 and that the latter advantage function is negligible if the DDH_2 assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$.

To prove that $\mathrm{Game}_{0,\kappa}$ is indistinguishable from $\mathrm{Game}_{0,\kappa+1}$ we will use another hybrid argument. We define $\mathrm{Game}'_{0,\kappa}$, which differs from $\mathrm{Game}_{0,\kappa}$ in that, for the $(\kappa+1)$-th delegation query, $\widehat{SK}_{DL}$ is the delegation component of a fresh key, instead of a delegation component obtained by raising every element in SK_{DL} to the same random power $z \in_R \mathbb{Z}_p$. We show that a PPT adversary cannot distinguish between the two games.

We also define $\mathrm{Game}''_{0,\kappa}$, which differs from $\mathrm{Game}'_{0,\kappa}$ in that, for the $(\kappa+1)$-th delegation query, instead of re-randomizing the components of the partial decryption key with the same exponent $\tau_{j.k}$, $\{L^{(j,k)}_{\ell+1,i_1}\}_{i_1=1}^{\ell+1}, L^{(j,k)}_{w,\ell+1}$ are randomized with different independently chosen exponents, while $K'_{j,k}$ is chosen in such a way that the resulting key is still valid. We also prove that no PPT adversary can notice the difference.

We will argue that $\mathrm{Game}''_{0,\kappa} = \mathrm{Game}_{0,\kappa+1}$. Indeed, in the first step, we change $\widehat{SK}_{DL}$ so that, in step 2 of the Delegate algorithm, we obtain a randomized decryption key (except for the g^α term). When multiplied by SK_D, it gives a randomized decryption key for $(\vec{X}_1, \ldots, \vec{X}_{\ell+1})$. On the other hand, in step 2 of the hybrid proof we change the partial decryption keys so that they also are randomized keys except the g^α term.

Claim. $\mathrm{Game}_{0,\kappa}$ is computationally indistinguishable from $\mathrm{Game}'_{0,\kappa}$.

Proof. Let q_0 denote the maximum number of secret key queries (taking into account both the "Create-key" and "Create-delegated-key" queries) made by the adversary. We build a simulator $\mathcal{B}$

[8] The straightforward reduction computes a GDDH instance from a DDH_2 instance $\big(g, \hat{g}, \hat{g}^a, \hat{g}^b, \hat{\eta} \stackrel{?}{=} \hat{g}^{ab}\big)$ by setting $\hat{h_i} = \hat{g}^{\alpha_i} \cdot (\hat{g}^b)^{\beta_i}$ and $Q_i = (\hat{g}^a)^{\alpha_i} \cdot \hat{\eta}^{\beta_i}$ for $i = 1$ to ℓ with $\alpha_1, \ldots, \alpha_\ell \xleftarrow{R} \mathbb{Z}_p$, $\beta_1, \ldots, \beta_\ell \xleftarrow{R} \mathbb{Z}_p$. If $\hat{\eta} = \hat{g}^{ab}$, we have $Q = (\hat{h}_1^a, \ldots, \hat{h}_\ell^a)$ whereas, if $\eta \in_R \hat{\mathbb{G}}$, Q is a random vector of $\hat{\mathbb{G}}^\ell$.

that uses $\mathcal{A}$ to break the following $(q_0 d\mu(d+1))$-GDDH assumption.

$$
\begin{aligned}
&(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e) \xleftarrow{R} \mathsf{GG}(\varrho, 1),\\
&g \xleftarrow{R} \mathbb{G}, \quad \hat{g} \xleftarrow{R} \hat{\mathbb{G}}\\
&\{\hat{v}_{i,i_1,i_2,k} \leftarrow \hat{\mathbb{G}}\}_{i\in\{1,\ldots,q_0\},i_1\in\{1,\ldots,d\},i_2\in\{0,\ldots,\mu\},k\in\{1,\ldots,d+1\}}\\
&\tau \xleftarrow{R} \mathbb{Z}_p\\
&X \leftarrow \big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e),\ g, \hat{g},\ \ \{\hat{v}_{i,i_1,i_2,k}\}_{i\in\{1,\ldots,q_0\},i_1\in\{1,\ldots,d\},i_2\in\{1,\ldots,\mu\},k\in\{1,\ldots,d+1\}}\big)\\
&Q \leftarrow (\{\hat{v}^{\tau}_{i,i_1,i_2,k}\}_{i\in\{1,\ldots,q_0\},i_1\in\{1,\ldots,d\},i_2\in\{1,\ldots,\mu\},k\in\{1,\ldots,d+1\}})
\end{aligned}
$$

Then, the challenger randomly decides to give $(X, Q' = Q)$ or $(X, Q' = R)$, where R is a random vector of elements in $\hat{\mathbb{G}}$ of the size of Q. The simulator will use $\mathcal{A}$ as a subroutine to break the above problem.

Init and Setup. At the beginning of the security game, the adversary commits to two hierarchical vectors $(\vec{Y_1}^0, \ldots, \vec{Y}^0_{d^\star})$ and $(\vec{Y_1}^1, \ldots, \vec{Y}^1_{d^\star})$, where $d^\star \leq d$. The simulator chooses the public and the secret key as usual according to the Setup algorithm. Let $c = \log_v(w)$ and $a_{i_1,i_2} = \log_v(h_{i_1,i_2})$. Note that these values are known to the simulator since they are easily computable from msk.

Secret key queries. We distinguish three cases:

- When a "create-key" query or one of the first κ delegated secret key queries is made, the simulator computes and saves a private key, which is given to $\mathcal{A}$ when a "reveal-key" query is made. To compute this secret key, the simulator uses the elements from the GDDH instance, in such a way that the exponents are distributed at random. In particular, if it is the i-th query, the simulator defines the components of the decryption component of the key as:

$$
\begin{aligned}
D &= g^{\alpha} \prod_{i_1=1}^{\ell} D_{i_1}^{(\sum_{i_2=1}^{\mu} a_{i_1,i_2} x_{i_1,i_2})} \cdot D_w^c, \qquad D_w = \hat{v}_{i,0,1,d+1},\\
D_{i_1} &= \hat{v}_{i,0,1,i_1} \quad i_1 \in \{1,\ldots,\ell\}.
\end{aligned}
$$

  For the delegation component of the key, for all $j \in \{\ell+1 \ldots d\}$, all $k \in \{1,\ldots,\mu\}$, the simulator lets:

$$
L_j = \hat{v}_{i,j,1,\ell+1}, \qquad L_{j,k,i_1} = \hat{v}_{i,j,k,i_1} \quad i_1 \in \{1,\ldots,\ell\}, \qquad L_{w,j,k} = \hat{v}_{i,j,k,d+1}.
$$

  As the simulator knows the discrete logarithms $c = \log_{\hat{v}}(\hat{w})$ and $a_{i_1,i_2} = \log_{\hat{v}}(\hat{h}_{i_1,i_2})$, for each $j \in \{\ell+1, \ldots d\}$ and all $k \in \{1,\ldots,\mu\}$, it can compute the remaining components of the key as follows:

$$
K_{j,k} = \prod_{i_1=1}^{\ell} L_{j,k,i_1}^{(\sum_{i_2=1}^{k} a_{i_1,i_2} \cdot x_{i_1,i_2})} \cdot L_j^{a_{j,k}} \cdot L_{w,j,k}^c. \tag{12}
$$

- When the adversary makes the $(\kappa+1)$-th delegation query, it specifies a parent key and asks to fix the level of the hierarchy to some vector $\vec{X}_{\ell+1}$. In particular, assume that the parent

key was created in the i-th query. When performing Step 1 of the Delegate algorithm, for all $j \in \{\ell+2, \ldots d\},\ k \in \{1, \ldots, \mu\}$, the simulator sets

$$\begin{aligned} \widehat{L}_j &= Q'_{i,j,1,\ell+1}, \\ \widehat{L}_{j,k,i_1} &= Q'_{i,j,k,i_1} \qquad\qquad i_1 \in \{1, \ldots, \ell\}, \\ \widehat{L}_{w,j,k} &= Q'_{i,j,k,d+1} \end{aligned}$$

and computes $K_{j,k}$, for each $j \in \{\ell+2, \ldots d\},\ k \in \{1, \ldots, \mu\}$ exactly in the same way as in expression (12).

- For all the remaining queries, the simulator responds faithfully as in the real game.

Clearly, if $Q' = Q$ in the GDDH instance, then the above simulation is identical to $\text{Game}_{0,\kappa}$. Otherwise, it is identical to $\text{Game}'_{0,\kappa}$ since, in the the $(\kappa+1)$-th delegation query, $Q = R$ implicitly defines a set of fresh random values for $s_j, s_{j,k,i_1}, s_{w,j,k}$, for the appropriate values of j, k, i_1.

Challenge The simulator generates the challenge ciphertext as normal.

Guess If the adversary has a difference of ϵ in its advantage in $\text{Game}_{0,\kappa}$ and $\text{Game}'_{0,\kappa}$, the simulator has a comparable advantage in solving the GDDH instance. □

Claim. $\text{Game}'_{0,\kappa}$ is computationally indistinguishable from $\text{Game}''_{0,\kappa}$.

Proof. To prove this claim, we will appeal to a nested hybrid argument. Let $\text{Game}'_{0,\kappa,0,0} = \text{Game}'_{0,\kappa}$, and for $1 \le \eta \le (d-\ell-1)$, $1 \le \nu \le \mu$, define $\text{Game}'_{0,\kappa,\eta,\nu}$ as the game that differs from $\text{Game}'_{0,\kappa}$ in the following: in the step of the delegation algorithm where the components $\{L^{(j,k)}_{\ell+1,i_1}\}_{i_1 \in \{1,\ldots,\ell+1\}}, L^{(j,k)}_{w,\ell+1}$ are created by re-randomizing the partial decryption key with some exponent $\tau_{j,k}$, we re-randomize instead each of these components with a different exponent chosen uniformly and independently at random whenever $(j,k) \le (\eta+\ell+1, \nu)$ (in lexicographic order). Observe that, by definition, $\text{Game}'_{0,\kappa,d-\ell-1,\mu} = \text{Game}''_{0,\kappa}$. We will show that an adversary cannot distinguish between one game and the next. That is, if we define and $\text{Game}'_{0,\kappa,\eta,\mu+1} = \text{Game}'_{0,\kappa,\eta+1,0}$ to simplify the notation, what we we will show is that no polynomial time adversary can distinguish between $\text{Game}'_{0,\kappa,\eta,\nu}$ and $\text{Game}'_{0,\kappa,\eta,\nu+1}$.
The simulator tries to solve the following $(\mu(d+1))$-GDDH instance.

$$\begin{aligned} &(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e) \xleftarrow{R} \mathsf{GG}(\varrho, 1), \\ &g \xleftarrow{R} \mathbb{G},\ \hat{g} \xleftarrow{R} \hat{\mathbb{G}} \\ &\{\hat{v}_{i,k} \leftarrow \hat{\mathbb{G}}\}_{i \in \{1,\ldots,\mu\}, k \in \{1,\ldots,d+1\}} \\ &\tau \xleftarrow{R} \mathbb{Z}_p \\ &X \leftarrow ((n, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e),\ g, \hat{g},\ \ \{\hat{v}_{i,k}\}_{i \in \{1,\ldots,\mu\}, k \in \{1,\ldots,d+1\}}) \\ &Q \leftarrow (\{\hat{v}^{\tau}_{i,k}\}_{i \in \{1,\ldots,\mu\}, k \in \{1,\ldots,d+1\}}) \end{aligned}$$

The simulator tries to distinguish between $(X, Q' = Q)$ and $(X, Q' = R)$ where R is a random vector from $\hat{\mathbb{G}}$. The simulator uses as a subroutine an adversary $\mathcal{A}$ who can distinguish between $\mathsf{Game}'_{0,\kappa,\eta,\nu}$ and $\mathsf{Game}'_{0,\kappa,\eta,\nu+1}$.

The simulator runs the Setup algorithm as usual in such a way that it knows the discrete logarithms $c = \log_v(w) = \log_{\hat{v}}(\hat{w})$ and $a_{i_1 i_2} = \log_v(h_{i_1,i_2}) = \log_{\hat{v}}(\hat{h}_{i_1,i_2})$ for any $1 \leq i_1 \leq d$, $0 \leq i_2 \leq \mu$.
To answer secret key queries, the simulator proceeds as follows:

- For the first κ-th delegation queries and all of the "create-key" queries, the simulator computes the keys freshly at random.
- At the $(\kappa+1)$-th delegation query, the adversary specifies a parent key and requests to fix the $(\ell+1)$-th level of the hierarchy to $\vec{X}_{\ell+1}$. To answer this query, the simulator first generates some components of $\widehat{SK}_{DL}$ simply by choosing at random the values $\widehat{L}_j, \widehat{L}_{w,j,k}$ for all $j \in \{\ell+2, \ldots, d\}, k \in \{1, \ldots, \mu\}, i_1 \in \{1, \ldots, \ell\}$. To compute the remaining components and the decryption key component, the simulator sets

$$\begin{aligned} L_{\ell+1,i_1} &= \prod_{k=1}^{\mu} \widehat{L}_{\ell+1,k,i_1}^{x_{\ell+1,k}} = \prod_{k=1}^{\mu} (\hat{v}_{k,i_1})^{x_{\ell+1,k}} && i_1 \in \{1, \ldots, \ell\} \\ L_{\ell+1,\ell+1} &= \widehat{L}_{\ell+1} = \hat{v}_{1,\ell+1} \\ L_{w,\ell+1} &= \prod_{k=1}^{\mu} \widehat{L}_{w,\ell+1,k}^{x_{\ell+1,k}} = \prod_{k=1}^{\mu} (\hat{v}_{k,d+1})^{x_{\ell+1,k}}. \end{aligned}$$

  Since the simulator knows the discrete logarithms $c = \log_{\hat{v}}(\hat{w})$ and $a_{i_1,i_2} = \log_{\hat{v}}(\hat{h}_{i_1,i_2})$, the remaining components of $\widehat{SK}_{DL}$ and those of the partial decryption key can be generated efficiently in the same way as in the proof of indistinguishability of $\mathrm{Game}_{0,\kappa}$ and $\mathrm{Game}'_{0,\kappa}$. In particular, the simulator can compute

$$K_{\ell+1} = \prod_{i_1=1}^{\ell} \prod_{i_2=1}^{\mu} L_{\ell+1,i_1}^{a_{i_1,i_2}} \cdot L_{\ell+1,\ell+1}^{\sum_{k=1}^{\mu} a_{\ell+1,k}} \cdot L_{w,\ell+1}^{c}.$$

  To create $K_{\ell+1}^{(j,k)}, \{L_{\ell+1,i_1}^{(j,k)}\}_{i_1 \in \{1,\ldots,\ell+1\}}, L_{w,\ell+1}^{(j,k)}$, if $(j,k) \leq (\eta+\ell+1, \nu)$ (in lexicographic order), the values are chosen as fresh random delegation keys. For the $(\eta+\ell+1, \nu+1)$ partial decryption key, the simulator lets

$$\begin{aligned} L_{\ell+1,i_1}^{(\eta+\ell+1,\nu+1)} &= \prod_{k=1}^{\mu} (Q'_{k,i_1})^{x_{\ell+1,k}} && i_1 \in \{1, \ldots, \ell\} \\ L_{\ell+1,\ell+1}^{(\eta+\ell+1,\nu+1)} &= Q'_{1,\ell+1} \\ L_{w,\ell+1}^{(\eta+\ell+1,\nu+1)} &= \prod_{k=1}^{\mu} (Q'_{k,d+1})^{x_{\ell+1,k}}. \end{aligned}$$

  Again, as the simulator knows the discrete logarithm of $\hat{w}, \hat{h}_{i_1,i_2}$ w.r.t. the base $\hat{v}$, the remaining terms – including $K_{\ell+1}^{(\eta+\ell+1,\nu+1)}$ – can be generated efficiently. For the remaining partial decryption keys, the simulator generates them faithfully using the Delegate algorithm.
- The remaining delegated key queries are generated faithfully.

Clearly, if $Q' = Q$ in the GDDH instance, then the above simulation is identically distributed as $\mathrm{Game}'_{0,\kappa,\eta,\nu}$, otherwise it is identically distributed as $\mathrm{Game}'_{0,\kappa,\eta,\nu+1}$.

The simulator generates the challenge ciphertext as normal and sends it to the adversary. If the adversary has ϵ difference in its advantage in $\text{Game}'_{0,\kappa,\eta,\nu}$ and $\text{Game}'_{0,\kappa,\eta,\nu+1}$, it is not hard to see that the simulator has a comparable advantage in solving the GDDH instance. □

Lemma 5. *Game_1 and Game_2 are computationally indistinguishable if the BDH assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.*

Proof. Assume that there's an adversary $\mathcal{A}$ that can distinguish between Game_1 and Game_2. We build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to break the BDH assumption. The simulator $\mathcal{B}$ receives as input

$$(g, g^a, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, Q'),$$

where Q' is either $e(g,\hat{g})^{abc}$ or an element chosen uniformly at random in $\mathbb{G}_T$.

The adversary $\mathcal{A}$ commits to two vectors $(\vec{Y}_1^0, \ldots, \vec{Y}_{d^\star}^0)$ and $(\vec{Y}_1^1, \ldots, \vec{Y}_{d^\star}^1)$. The challenger $\mathcal{B}$ picks $\beta \xleftarrow{R} \{0,1\}$. If $d^\star < d$, the simulator picks at random $d - d^\star$ vectors $Y_{d^\star+1}^\beta, \ldots, Y_d^\beta \in \mathbb{Z}_p^\mu$. The simulator $\mathcal{B}$ runs the Setup algorithm as usual except that it implicitly sets α to be ab by defining $e(g,\hat{v})^\alpha = e(g^a, \hat{g}^b)^{\alpha_v}$, and then, for $i_1 = 1, \ldots, d$ and $i_2 = 1, \ldots, \mu$, it defines $h_{i_1,i_2} = (g^a)^{z_{i_1} y_{i_1,i_2}} g^{t_{i_1,i_2}}$ for $i_1 \in \{1 \ldots, d\}$, $i_2 \in \{1, \ldots, \mu\}$ and $h_{i_1,0} = (g^a)^{-z_{i_1}}$, for some random exponents z_{i_1}, t_{i_1,i_2}. The values of $\hat{h}_{i_1,i_2}$ for $i_1 \in \{1 \ldots, d\}$, $i_2 \in \{0, \ldots, \mu\}$ are defined similarly from the value $\hat{g}^a$ of the BDH instance. Observe that β remains hidden from $\mathcal{A}$ and that the parameters are correctly distributed.

For the secret key queries, in the last lemma we have just proven that delegated keys are indistinguishable from freshly generated ones. Therefore, $\mathcal{B}$ will generate the secret keys using algorithm Keygen when a reveal query is made. Note that α is defined as ab, which is not known to $\mathcal{B}$. However, $\mathcal{B}$ needs to simulate secret keys for $(\vec{X}_1, \ldots, \vec{X}_\ell)$ as long as $f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\vec{Y}_1^\beta, \ldots, \vec{Y}_{d^\star}^\beta) = 0$. As α does not appear in the delegating component of the key, the delegation component of the secret keys SK_{DL} can be created using the parameters as usual. Therefore, from now on we focus on how to create the decryption component of the secret key. Denote by ℓ' the index of the smallest element of the vector $(\vec{X}_1, \ldots, \vec{X}_\ell)$ for which $\vec{X}_{\ell'} \cdot \vec{Y}_{\ell'}^\beta \neq 0$. This value ℓ' always exists if $\ell \leq d^\star$ by hypothesis and exists with overwhelming probability if $\ell > d^\star$, since the vectors $Y_{d^\star+1}^\beta, \ldots, Y_d^\beta \in \mathbb{Z}_p^\mu$ are completely hidden from $\mathcal{A}$'s view.

The simulator creates the decryption component of the secret key as follows:

$$D = \hat{g}^{ab} \prod_{i_1=1}^{\ell} \left(\prod_{i_2=1}^{\mu} \hat{h}_{i_1,i_2}^{x_{i_1,i_2}} \right)^{r_{i_1}} \hat{w}^{r_w}$$

where the term $\hat{g}^{ab} \left(\prod_{i_2=1}^{\mu} \hat{h}_{\ell',i_2}^{x_{\ell',i_2}} \right)^{r_{\ell'}}$ is computed as

$$\left(\prod_{i_2=1}^{\mu} \hat{h}_{\ell',i_2}^{x_{\ell',i_2}} \right)^{\hat{r}_{\ell'}} (\hat{g}^b)^{-(\sum_{i_2=1}^{\mu} x_{\ell',i_2} t_{\ell',i_2})/(z_{\ell'} \vec{X}_{\ell'} \cdot \vec{Y}_{\ell'}^\beta)},$$

while the other terms in the product are just computed as usual. It is not hard to see that, if we define $r_{\ell'} = \hat{r}_{\ell'} - b/(z_{\ell'} \vec{X}_{\ell'} \cdot \vec{Y}_{\ell'}^\beta)$, then the computation is correct. All the other terms in the decryption component can be computed efficiently, since the simulator knows all the parameters needed and it also knows $\hat{g}^b$.

At the challenge step, the adversary $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0 and M_1. $\mathcal{B}$ then computes

$$C_0 = M_\beta \cdot (Q')^{\alpha_v}, \qquad C_v = (g^c)^{\alpha_v}, \qquad C_w = (g^c)^{\alpha_w},$$
$$\{C_{i_1,i_2} = (g^c)^{t_{i_1,i_2}}\}_{i_1 \in \{1,\ldots,d^\star\},\ i_2 \in \{1,\ldots,\mu\}}$$

It is not hard to check that the challenge ciphertext is correctly distributed.

Finally, when the adversary $\mathcal{A}$ outputs a guess β', if $\beta = \beta'$, then $\mathcal{B}$ guesses that $Q' = e(g,\hat{g})^{abc}$ and if $\beta \neq \beta'$ guesses that $Q' = R$. If $\mathcal{A}$ has ϵ advantage in distinguishing between the two cases, then $\mathcal{B}$ also has ϵ advantage in solving the assumption G3DH instance, except in the case that $\mathcal{A}$ managed to output some secret key query for vector $(\vec{X}_1,\ldots,\vec{X}_\ell)$ for which $\vec{X}_i \cdot \vec{Y}_i^\beta = 0$ for all $i = d^\star+1,\ldots,d$ was zero for all $i \in \{1,\ldots,d\}$, which occurs only with negligible probability.

Lemma 6. *Game$_2$ and Game$_3$ are computationally indistinguishable if the $\mathcal{P}$-BDH$_1$ assumption holds in $(\mathbb{G},\hat{\mathbb{G}})$.*

Proof. Assuming that the adversary $\mathcal{A}$ outputs $\beta' = \beta$ with noticeably different probabilities in Game$_2$ and Game$_3$, we build an distinguisher $\mathcal{B}$ for the $\mathcal{P}$-BDH$_1$ assumption .

Namely, algorithm $\mathcal{B}$ receives as input a tuple

$$\left(g,\ \ g^b,\ g^{ab},\ g^c,\ \hat{g},\ \hat{g}^a,\ \ \hat{g}^b,\ g^z\right),$$

where $a,b,c \xleftarrow{R} \mathbb{Z}_p$. Its goal is to decide if $z = abc$ or $z \in_R \mathbb{Z}_p$.

To this end, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows. It first receives the vectors $(\vec{Y}_0^0,\ldots,\vec{Y}_{d^\star}^0)$, $(\vec{Y}_0^1,\ldots,\vec{Y}_{d^\star}^1)$, at some depth $d^\star \leq d$, that $\mathcal{A}$ wishes to be challenged upon. We may assume w.l.o.g. that $\mathcal{B}$ chooses its challenge bit $\beta \xleftarrow{R} \{0,1\}$ at the beginning of the game. Also, if $d^\star < d$, for each $i \in \{d^\star+1,\ldots,d\}$, $\mathcal{B}$ defines the vector $\vec{Y}_i^\beta$ as a random vector of $\mathbb{Z}_p^\mu$.

It defines the master public key by setting $e(g,\hat{v})^\alpha = e(g,\hat{g})^{\alpha\cdot\gamma_v}$ and

$$\begin{aligned}
v &= g^{\gamma_v}, \\
h_{i_1,0} &= (g^b)^{\gamma_{i_1,0}} && i_1 \in \{1,\ldots,d\} \\
h_{i_1,i_2} &= (g^b)^{-\gamma_{i_1,0}\cdot y_{i_1,i_2}^\beta} \cdot g^{\gamma_{i_1,i_2}} && i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\} \\
w &= g^y \cdot (g^{ab})^x,
\end{aligned} \tag{13}$$

where $\alpha,\gamma_v,x,y \xleftarrow{R} \mathbb{Z}_p$ and $\gamma_{i_1,i_2} \xleftarrow{R} \mathbb{Z}_p$, for each $i_1 \in \{1,\ldots,d\}$, $i_2 \in \{0,\ldots,\mu\}$. For each i_1, we also define the vector $\vec{\gamma}_{i_1} = (\gamma_{i_1,1},\ldots,\gamma_{i_1,\mu}) \in \mathbb{Z}_p^\mu$. We observe that $\mathcal{B}$ does not know $\hat{w}$ (which depends on the unavailable term $\hat{g}^{ab}$) but *can* compute $\{\hat{h}_{i_1,i_2}\}_{i_1\in\{1,\ldots,d\},i_2\in\{1,\ldots,\mu\}}$.

When the adversary $\mathcal{A}$ requests a key for a hierarchical vector $\mathbf{X} = (\vec{X}_1,\ldots,\vec{X}_\ell)$, $\mathcal{B}$ parses $\vec{X}_{i_1}$ as $(x_{i_1,1},\ldots,x_{i_1,\mu}) \in \mathbb{Z}_p^\mu$ for each $i_1 \in \{1,\ldots,\ell\}$. Then, $\mathcal{B}$ responds as follows.

• If $\ell \leq d^\star$, let $i \in \{1,\ldots,\ell\}$ be the smallest index such that $\vec{X}_i \cdot \vec{Y}_i^\beta \neq 0$ (by hypothesis, this index must exist). By choosing $r_w \xleftarrow{R} \mathbb{Z}_p$ and $r_i \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{B}$ implicitly defines the exponent

$$\tilde{r_i} = r_i + \frac{a \cdot r_w \cdot x}{\gamma_{i,0} \cdot \vec{X}_i \cdot \vec{Y}_i^\beta}$$

and can compute

$$\begin{aligned}(\prod_{i_2=1}^{\mu} \hat{h}_{i,i_2}^{x_{i,i_2}})^{\tilde{r}_i} \cdot \hat{w}^{r_w} &= \big(\prod_{i_2=1}^{\mu} \hat{h}_{i,i_2}^{x_{i,i_2}}\big)^{r_i} \cdot \big((\hat{g}^b)^{-\gamma_{i,0}\cdot\vec{X}_i\cdot\vec{Y}_i^{\beta}} \cdot \hat{g}^{\vec{\gamma}_i\cdot\vec{Y}_i^{\beta}}\big)^{a\cdot\frac{r_w\cdot x}{\gamma_{i,0}\cdot\vec{X}\cdot\vec{Y}_i^{\beta}}} \\ &\quad \cdot(\hat{g}^{ab})^{r_w\cdot x} \cdot \hat{g}^{r_w\cdot y} \\ &= \big(\prod_{i_2=1}^{\mu} \hat{h}_{i,i_2}^{x_{i,i_2}}\big)^{r_i} \cdot (\hat{g}^a)^{\vec{\gamma}_i\cdot\vec{Y}_i^{\beta}\cdot\frac{r_w\cdot x}{\gamma_{i,0}\cdot\vec{X}\cdot\vec{Y}_i^{\beta}}} \cdot \hat{g}^{r_w\cdot y}\end{aligned} \tag{14}$$

without knowing $\hat{g}^{ab}$. Similarly, it can compute

$$D_i = \hat{v}^{\tilde{r}_i} = \hat{g}^{\gamma_v\cdot r_i} \cdot (\hat{g}^a)^{\gamma_v\cdot r_w\cdot x/(\gamma_{i,0}\cdot\vec{X}\cdot\vec{Y}_i^{\beta})}$$

as well as $D_w = \hat{v}^{r_w}$.

We now turn to indices $i_1 \in \{1,\ldots,\ell\}\backslash\{i\}$, for which $\mathcal{B}$ can trivially compute $(\prod_{i_2=1}^{\mu} \hat{h}_{i,i_2}^{x_{i,i_2}})^{r_{i_1}}$ and $D_{i_1} = \hat{v}^{r_{i_1}} \cdot S_{v,i_1}$ since it knows $\{\hat{h}_{i_1,i_2}\}_{i_2=0}^{\mu}$ and v. This suffices for computing the whole decryption component SK_D of the private key.

As for the delegation component SK_{DL}, $\mathcal{B}$ can compute $\{K_{j,k}\}_{j\in\{\ell+1,\ldots,d\},k\in\{1,\ldots,\mu\}}$, $\{L_j\}_{j=\ell+1}^{d}$ and $\{L_{w,j,k}\}_{j,k}$ by applying exactly the same procedure as for D, D_i and D_w (and taking advantage of the fact that $\vec{X}_i \cdot \vec{Y}_i^{\beta} \neq 0$ for at least one $i \in \{1,\ldots,\ell\}$). Remaining pieces of SK_{DL} are then trivially computable since $\mathcal{B}$ has $\{\hat{h}_{i_1,i_2}\}_{i_2=0}^{\mu}$ and $\hat{v}$ at disposal.

• If $\ell > d^\star$, it can be the case that $\vec{X}_i \cdot \vec{Y}_i^{\beta} = 0$ for $i = 1$ to ℓ. However, with overwhelming probability, there must exist $i \in \{d^\star+1,\ldots,\ell\}$ such that $\vec{X}_i \cdot \vec{Y}_i^{\beta} \neq 0$ since the vectors $\vec{Y}_{d^\star+1},\ldots,\vec{Y}_d$ have been chosen at random and, due to the generation of the public key as per (13), they are completely independent of $\mathcal{A}$'s view. It comes that $\mathcal{B}$ can generate a private key in the same way as in the case $\ell < d^\star$ (see equation (14)).

When $\mathcal{B}$ has to construct the challenge ciphertext, $\mathcal{B}$ sets $C_0 = M \cdot e(g^c, \hat{g}^{\gamma_v})^{\alpha}$, where $M \xleftarrow{R} \mathbb{G}_T$, and

$$C_v = (g^c)^{\gamma_v}, \qquad C_w = (g^c)^y \cdot (g^z)^x,$$

$$C_{i_1,i_2} = (g^c)^{\gamma_{i_1,i_2}} \qquad i_1 \in \{1,\ldots,d^\star\},\ i_2 \in \{1,\ldots,\mu\}$$

We observe that, if $z = abc$, $(C_0, C_v, C_w, \{C_{i_1,i_2}\}_{i_1\in\{1,\ldots,d^\star\},i_2\in\{1,\ldots,\mu\}})$ corresponds to a valid ciphertext with the encryption exponent $s = c$. In this situation, $\mathcal{B}$ is playing Game_2 with $\mathcal{A}$.

In contrast, if $z \in_R \mathbb{Z}_p$, we have $z \neq c$ with overwhelming probability. In this case, we have $g^z = g^{abc+\theta}$, for some $\theta \neq 0$, and we can thus write $C_w = w^c \cdot g^{\theta\cdot x}$. This means that C_w looks uniformly random and independent from $\mathcal{A}$'s view. Indeed, until the challenge phase, $\mathcal{A}$ has no information about $\theta \in \mathbb{Z}_p$ (recall that public parameters do not depend on c) and the value $x \in \mathbb{Z}_{p_1}$ is also independent of $\mathcal{A}$'s view. We conclude that, if g^z is such that $z \in_R \mathbb{Z}_p$, we are in Game_3. □

Lemma 7. *For each $\delta_1 \in \{1,\ldots,d\}$ and each $\delta_2 \in \{2,\ldots,\mu\}$, $\text{Game}_{4,\delta_1,\delta_2-1}$ and $\text{Game}_{4,\delta_1,\delta_2}$ are computationally indistinguishable if the $\mathcal{P}$-BDH$_1$ assumption holds in $(\mathbb{G},\hat{\mathbb{G}})$.*

Proof. Towards a contradiction, we assume there exists δ_1, δ_2 such that the adversary $\mathcal{A}$ outputs $\beta' = \beta$ with significantly different probabilities in $\text{Game}_{4,\delta_1,\delta_2-1}$ and $\text{Game}_{4,\delta_1,\delta_2}$. We show that $\mathcal{A}$

implies a distinguisher $\mathcal{B}$ against $\mathcal{P}$-BDH$_1$.

Our distinguisher $\mathcal{B}$ receives as input $\big(g, g^b, g^{ab}, g^c, \hat{g}, \hat{g}^a, \hat{g}^b, g^z\big)$, with $a, b, c \xleftarrow{R} \mathbb{Z}_p$. It aims to decide if $z = abc$ or $z \in_R \mathbb{Z}_p$.

To do this, $\mathcal{B}$ runs the adversary $\mathcal{A}$ as follows. It first receives the challenge vectors $(\vec{Y}_0^0, \ldots, \vec{Y}_{d^\star}^0)$, $(\vec{Y}_0^1, \ldots, \vec{Y}_{d^\star}^1)$, at some depth $d^\star \leq d$, that are chosen by $\mathcal{A}$. We assume that $\mathcal{B}$ chooses its challenge bit $\beta \xleftarrow{R} \{0,1\}$ at the outset of the game. Also, if $d^\star < d$, for each $i \in \{d^\star + 1, \ldots, d\}$, $\mathcal{B}$ defines the vector $\vec{Y}_i^\beta$ as a random vector of $\mathbb{Z}_p^\mu$.

It defines the master public key by setting $e(g, \hat{v})^\alpha = e(g, \hat{g})^{\alpha \cdot \gamma_v}$ and

$$\begin{aligned}
v &= g^{\gamma_v}, \\
h_{i_1,0} &= g_{p_1}^{\gamma_{i_1,0}} && \text{for } i_1 \in \{1, \ldots, d\} \\
h_{i_1,i_2} &= g^{-\gamma_{i_1,0} \cdot y^\beta_{i_1,i_2}} \cdot g^{\gamma_{i_1,i_2}} && \text{if } i_1 \in \{1, \ldots, d\} \backslash \{\delta_1\} \text{ or } \; i_2 \in \{1, \ldots, \mu\} \backslash \{\delta_2\} \\
h_{\delta_1,\delta_2} &= g^{-\gamma_{\delta_1,0} \cdot y^\beta_{\delta_1,\delta_2}} \cdot (g^{ab})^{\gamma_{\delta_1,\delta_2}} && \\
w &= g^y \cdot (g^b)^x, &&
\end{aligned} \tag{15}$$

where $\alpha, \gamma_v, y \xleftarrow{R} \mathbb{Z}_p$, $x \xleftarrow{R} \mathbb{Z}_p^*$ and $\gamma_{i_1,i_2} \xleftarrow{R} \mathbb{Z}_p$, for each $i_1 \in \{1, \ldots, d\}$, $i_2 \in \{0, \ldots, \mu\}$. For each i_1, we also define the vector $\vec{\gamma}_{i_1} = (\gamma_{i_1,1}, \ldots, \gamma_{i_1,\mu}) \in \mathbb{Z}_p^\mu$. Note that, in the implicitly defined master secret key msk, the distinguisher $\mathcal{B}$ knows all the components but $\hat{h}_{\delta_1,\delta_2}$, which depends on the unknown term $\hat{g}^{ab}$.

When the adversary $\mathcal{A}$ requests a key for a hierarchical vector $\mathbf{X} = (\vec{X}_1, \ldots, \vec{X}_\ell)$, $\mathcal{B}$ parses $\vec{X}_{i_1}$ as $(x_{i_1,1}, \ldots, x_{i_1,\mu}) \in \mathbb{Z}_p^\mu$ for each $i_1 \in \{1, \ldots, \ell\}$. Then, $\mathcal{B}$ responds as follows.

• If $\ell \geq \delta_1$, $\mathcal{B}$ chooses $r'_w \xleftarrow{R} \mathbb{Z}_p$ and $r_{\delta_1} \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{B}$ implicitly defines the exponent

$$r_w = r'_w - \frac{a \cdot r_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}}{x} \tag{16}$$

and can compute the product

$$\begin{aligned}
\Big(\hat{h}_{\delta_1,0}^{y^\beta_{\delta_1,\delta_2}} \cdot \hat{h}_{\delta_1,\delta_2}\Big)^{r_{\delta_1}} \cdot \hat{w}^{r_w} &= (\hat{g}^{ab})^{\gamma_{\delta_1,\delta_2} \cdot r_{\delta_1}} \cdot \hat{w}^{r'_w} \cdot \big(\hat{g}^y (\hat{g}^b)^x\big)^{-a \cdot r_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x} \\
&= \hat{w}^{r'_w} \cdot (\hat{g}^a)^{-y \cdot r_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x},
\end{aligned} \tag{17}$$

which is the only factor of D that it cannot trivially compute without knowing $\hat{g}^{ab}$. Similarly, it can compute $D_w = v^{r_w} = \hat{g}^{\gamma_v \cdot r'_w} \cdot (\hat{g}^a)^{-\gamma_v \cdot r_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x}$.

To generate the delegation component SK_{DL} of the key, the reduction $\mathcal{B}$ is able to compute $\{K_{j,k}, L_{w,j,k}\}_{j \in \{\ell+1,\ldots,d\}, k \in \{1,\ldots,\mu\}}$ by repeating $(d-\ell) \cdot \mu$ times the same procedure as for computing D and D_w.

• If $\ell < \delta_1$, $\mathcal{B}$ can directly compute $(D, D_w, \{D_{i_1}\}_{i_1=1}^\ell)$ since it knows $\{\hat{h}_{i_1,i_2}\}_{i_1 \in \{1,\ldots,\ell\}, i_2 \in \{1,\ldots,\mu\}}$, $\hat{v}$ and $\hat{w}$. The difficulty is to compute the delegation components $\{K_{\delta_1,k}\}_{k=1}^\mu$ without knowing $\hat{g}^{ab}$. In fact, among these components, the only factor of K_{δ_1,δ_2} that $\mathcal{B}$ cannot trivially compute is $\hat{h}_{\delta_1,\delta_2}^{s_{\delta_1}}$. However, similarly to (16)-(17), it can choose $s_{\delta_1}, s'_{w,\delta_1,\delta_2} \xleftarrow{R} \mathbb{Z}_p$, define $s_{w,\delta_1,\delta_2} = s'_{w,\delta_1,\delta_2} - \frac{a \cdot s_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}}{x}$ and compute the product

$$\begin{aligned}
h_{\delta_1,\delta_2}^{s_{\delta_1}} \cdot \hat{w}^{s_{w,\delta_1,\delta_2}} &= (\hat{g}^{-\gamma_{\delta_1,0} \cdot y^\beta_{\delta_1,\delta_2} \cdot s_{\delta_1}}) \cdot (\hat{g}^{ab})^{\gamma_{\delta_1,\delta_2} \cdot s_{\delta_1}} \cdot \big(\hat{g}^y (\hat{g}^b)^x\big)^{-a \cdot s_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x} \cdot \hat{w}^{s'_{w,\delta_1,\delta_2}} \\
&= (\hat{g}^{-\gamma_{\delta_1,0} \cdot y^\beta_{\delta_1,\delta_2} \cdot s_{\delta_1}}) \cdot \hat{w}^{s'_{w,\delta_1,\delta_2}} \cdot (\hat{g}^a)^{-y \cdot s_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x}
\end{aligned} \tag{18}$$

In the same way, $\mathcal{B}$ computes

$$L_{w,\delta_1,\delta_2} = \hat{v}^{s_{w,\delta_1,\delta_2}} = \hat{g}^{\gamma_v \cdot s'_{w,\delta_1,\delta_2}} \cdot (\hat{g}^a)^{-\gamma_v \cdot s_{\delta_1} \cdot \gamma_{\delta_1,\delta_2}/x}.$$

Note that, for each $k \in \{1,\ldots,\mu\}\backslash\{\delta_2\}$, $\mathcal{B}$ has to generate $K_{\delta_1,k}$ by computing $\hat{h}_{\delta_1,k}^{s_{\delta_1}}$ using the same random exponent s_{δ_1} as in (18). This is always possible since $\mathcal{B}$ knows that exponent.

When it comes to construct the challenge ciphertext, algorithm $\mathcal{B}$ first sets $C_0 = M \cdot e(g^c, \hat{g}^{\gamma_v})^\alpha$, where $M \stackrel{R}{\leftarrow} \mathbb{G}_T$. It also chooses $\Gamma_w \stackrel{R}{\leftarrow} \mathbb{G}$ and computes

$$C_v = (g^c)^{\gamma_v}, \qquad\qquad C_w = \Gamma_w,$$

as well as

$$\begin{aligned} C_{i_1,i_2} &= (g^c)^{\gamma_{i_1,i_2}} && \text{if} \quad (i_1 > \delta_1) \vee \big((i_1 = \delta_1) \wedge (i_2 > \delta_2)\big) \\ C_{\delta_1,\delta_2} &= (g^z)^{\gamma_{\delta_1,\delta_2}} \end{aligned}$$

If $i_1 < \delta_1$ or $i_1 = \delta_1$ and $i_2 < \delta_2$, then C_{i_1,i_2} is chosen uniformly in $\mathbb{G}$.

We observe that, in the situation where $z = abc$, $(C_0, C_v, C_w, \{C_{i_1,i_2}\}_{i_1 \in \{1,\ldots,d^\star\}, i_2 \in \{1,\ldots,\mu\}})$ is distributed in the same way as in $\text{Game}_{4,\delta_1,\delta_2-1}$.

In contrast, if $z \in_R \mathbb{Z}_p$, we have $z \neq c$ with overwhelming probability. In this case, C_{δ_1,δ_2} looks random to the adversary and $\mathcal{B}$ is thus playing $\text{Game}_{4,\delta_1,\delta_2}$. □

B Deferred Proofs for the Security of HF

B.1 Proof of Lemma 1

We consider a sequence of $n+1$ hybrid experiments $RL_0, \ldots, RL_n$. For each $k \in \{0,\ldots,n\}$, RL_k is defined to be an experiment where public parameters are generated as follows. First, the simulator $\mathcal{B}$ chooses $v \stackrel{R}{\leftarrow} \mathbb{G}$, $\mathbf{w} \stackrel{R}{\leftarrow} \mathbb{G}^n, \hat{\mathbf{w}} \stackrel{R}{\leftarrow} \hat{\mathbb{G}}^n$, $\mathbf{h} \stackrel{R}{\leftarrow} \mathbb{G}^{d\times(\mu+1)\times n}$, $\hat{\mathbf{h}} \stackrel{R}{\leftarrow} \hat{\mathbb{G}}^{d\times(\mu+1)\times n}$ and computes PP_{core} in the same way as in the real scheme.

In the second step of the setup procedure, the simulator $\mathcal{B}$ chooses a vector $\mathbf{s} \stackrel{R}{\leftarrow} (\mathbb{Z}_p^*)^n$. For $l_1, l_2 \in \{1,\ldots,n\}$, it first computes

$$\begin{aligned} \mathbf{J}[l_2] &= v^{\mathbf{s}[l_2]}, \\ \mathbf{C}_w[l_2, l_1] &= \mathbf{w}[l_1]^{\mathbf{s}[l_2]}. \end{aligned}$$

Then, for each pair (l_1, l_2) such that $l_1 \neq l_2$, $\mathcal{B}$ sets

$$\mathbf{C}[i_1, i_2, l_2, l_1] = \mathbf{h}[i_1, i_2, l_1]^{\mathbf{s}[l_2]}.$$

Finally, for each $l \in \{1,\ldots,n\}$, $i_1 \in \{1,\ldots,d\}$ and $i_2 \in \{1,\ldots,\mu\}$, $\mathcal{B}$ defines

$$\begin{aligned} \mathbf{C}[i_1, i_2, l, l] &= \Big(\mathbf{h}[i_1, 0, l]^{\mathbf{y}_{i_1}^{(0)}}[i_2] \cdot \mathbf{h}[i_1, i_2, l]\Big)^{\mathbf{s}[l]} && \text{if } l \leq k, \\ \mathbf{C}[i_1, i_2, l, l] &= \Big(\mathbf{h}[i_1, 0, l]^{\mathbf{y}_{i_1}^{(1)}}[i_2] \cdot \mathbf{h}[i_1, i_2, l]\Big)^{\mathbf{s}[l]} && \text{if } l > k. \end{aligned}$$

Lemma 8 below demonstrates that, for each $k \in \{1,\ldots,n\}$, experiment RL_k is computationally indistinguishable from experiment RL_{k-1}.

If we assume that the statement of Lemma 1 is false, there must exist $k \in \{1,\ldots,n\}$ such that the adversary can distinguish RL_k from RL_{k-1} and we obtain a contradiction. □

Lemma 8. *If the HPE scheme described in Section 3.2 is selectively weakly attribute-hiding, then Game RL_k is indistinguishable from Game RL_{k-1} for each $k \in \{1, \ldots, n\}$. Namely, for each k, there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$|\Pr[RL_k \Rightarrow 1] - \Pr[RL_{k-1} \Rightarrow 1]| \leq (d \cdot \mu + 1) \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_1}(\mathcal{B}_1) + q \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_2}(\mathcal{B}_2),$$

where q is the number of "Reveal-key" queries made by $\mathcal{A}$.

Proof. For the sake of contradiction, let us assume that there exist two auxiliary hierarchical vector $\mathbf{y}^{(0)} = [\mathbf{y}_1^{(0)}| \ldots |\mathbf{y}_d^{(0)}]$, $\mathbf{y}^{(1)} = [\mathbf{y}_1^{(1)}| \ldots |\mathbf{y}_d^{(1)}]$ and an index $k \in \{1, \ldots, n\}$ such that the adversary $\mathcal{A}$ has noticeably different behaviors in experiments RL_k and RL_{k-1}. Using $\mathcal{A}$, we construct a selective weakly attribute-hiding adversary $\mathcal{B}$ against the HPE scheme described in Section 3.2 (in its predicate-only variant).

Our adversary $\mathcal{B}$ first declares $\mathbf{y}^{(0)}, \mathbf{y}^{(1)} \in \mathbb{Z}_p^{d \cdot \mu}$ as the vectors that it wishes to be challenged upon. Then, the HPE challenger provides $\mathcal{B}$ with public parameters

$$\mathsf{mpk}_{\text{HPE}} = \Big(v,\ w,\ \{h_{i_1,i_2}\}_{i_1 \in \{1,\ldots,d\}, i_2 \in \{0,\ldots,\mu\}}\Big).$$

Then, $\mathcal{B}$ chooses a vector $\boldsymbol{\zeta} \xleftarrow{R} \mathbb{Z}_p^n$ and a matrix $\boldsymbol{\gamma} \in \mathbb{Z}_p^{d \times (\mu+1) \times n}$, which it uses to compute

$$\begin{aligned} \mathbf{w}[l_1] &= v^{\boldsymbol{\zeta}[l_1]} && \text{for } l_1 \in \{1, \ldots, n\} \backslash \{k\} \\ \mathbf{h}[i_1, i_2, l_1] &= v^{\boldsymbol{\gamma}[i_1,i_2,l_1]} && \text{for } i_1 \in \{1, \ldots, d\},\ i_2 \in \{0, \ldots, \mu\},\ l_1 \in \{1, \ldots, n\} \backslash \{k\}. \end{aligned}$$

It also sets $\mathbf{w}[k] = w$ as well as

$$\mathbf{h}[i_1, i_2, k] = h_{i_1,i_2} \qquad \text{for } i_1 \in \{1, \ldots, d\},\ i_2 \in \{0, \ldots, \mu\}.$$

Then, $\mathcal{B}$ defines core public parameters

$$\mathsf{PP}_{core} = \Big(v,\ \{\mathbf{w}[l_1],\ \ \{\mathbf{h}[i_1, i_2, l_1]\}_{i_1 \in \{1,\ldots,d\}, i_2 \in \{0,\ldots,\mu\},\ l_1 \in \{1,\ldots,n\}}\Big),$$

that correspond to the master secret key $\mathsf{msk} = (\hat{v}, \hat{\mathbf{w}}, \hat{\mathbf{h}})$, which is not completely known to $\mathcal{B}$ (specifically, $\hat{v}$, $\hat{\mathbf{w}}[k]$ and $\hat{\mathbf{h}}[., ., k]$ are not available). Then, $\mathcal{B}$ notifies its HPE challenger that it wishes to directly enter the challenge phase without making any pre-challenge query. The challenger replies with the challenge ciphertext

$$C^\star = \big(C_v, C_w, \{C_{i_1,i_2}\}_{i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\}}\big),$$

where

$$\begin{aligned} C_v &= v^s, && C_w = w^s, \\ \{\ C_{i_1,i_2} &= \big(h_{i_1,0}^{y_{i_1,i_2}^{(\beta)}} \cdot h_{i_1,i_2}\big)^s\ \}_{i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\}}, \end{aligned}$$

for a random element $s \xleftarrow{R} \mathbb{Z}_p^*$ and a random bit $\beta \in \{0, 1\}$. Here we are using, for $\beta \in \{0, 1\}$, the notation $\mathbf{y}(\beta) = [\mathbf{y}_1^{(\beta)}| \ldots |\mathbf{y}_d^{(\beta)}] \in \mathbb{Z}_p^{d \cdot \mu}$, where each $\mathbf{y}_{i_1}^{(\beta)} = (y_{i_1,1}^{(\beta)}, \ldots, y_{i_1,\mu}^{(\beta)}) \in \mathbb{Z}_p^\mu$, for $i_1 \in \{1, \ldots, d\}$.

At this point, $\mathcal{B}$ constructs the matrix $\{\mathbf{CT}[i_1, i_2]\}_{l_1,l_2 \in \{1,\ldots,n\}}$ of HPE ciphertexts by setting

$$\begin{aligned} \mathbf{J}[k] &= C_v \\ \mathbf{C}_w[k, k] &= C_w \\ \mathbf{C}[i_1, i_2, k, k] &= C_{i_1,i_2} && \text{for } i_1 \in \{1, \ldots, d\},\ i_2 \in \{1, \ldots, \mu\}. \end{aligned}$$

and, for each $l_1 \in \{1,\ldots,n\}\backslash\{k\}$,

$$\begin{aligned}\mathbf{C}_w[k,l_1] &= C_v^{\boldsymbol{\zeta}[l_1]}\\ \mathbf{C}[i_1,i_2,k,l_1] &= C_v^{\boldsymbol{\gamma}[i_1,i_2,k,l_1]} \qquad \text{for } i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\}.\end{aligned}$$

Note that this implicity sets $\mathbf{s}[k] = s$, where s is the encryption exponent chosen by the HPE challenger to compute $C^\star$. Then, for each $l_2 \in \{1,\ldots,n\}\backslash\{k\}$, $\mathcal{B}$ chooses a random exponent $\mathbf{s}[l_2] \xleftarrow{R} \mathbb{Z}_p^*$ and computes

$$\mathbf{J}[l_2] = v^{\mathbf{s}[l_2]} \tag{19}$$

$$\begin{aligned}\mathbf{C}_w[l_2,l_1] &= \mathbf{w}[l_1]^{\mathbf{s}[l_2]} \qquad \text{for } l_1 \in \{1,\ldots,n\}\backslash\{l_2\}\\ \mathbf{C}[i_1,i_2,l_2,l_1] &= \mathbf{h}[i_1,i_2,l_1]^{\mathbf{s}[l_2]} \qquad \text{for } i_1 \in \{1,\ldots,d\},\ i_2 \in \{1,\ldots,\mu\},\\ &\qquad\qquad l_1 \in \{1,\ldots,n\}\backslash\{l_2\}\end{aligned} \tag{20}$$

As for entries of the form $\{\mathbf{C}[i_1,i_2,l,l]\}_{i_1,i_2,l\neq k}$, $\mathcal{B}$ computes them as

$$\begin{aligned}\mathbf{C}[i_1,i_2,l] &= \big(\mathbf{h}[i_1,0,l]^{\mathbf{y}_{i_1}^{(0)}}[i_2]\cdot\mathbf{h}[i_1,i_2,l]\big)^{\mathbf{s}[l]} \qquad \text{if } l<k\\ \mathbf{C}[i_1,i_2,l] &= \big(\mathbf{h}[i_1,0,l]^{\mathbf{y}_{i_1}^{(1)}}[i_2]\cdot\mathbf{h}[i_1,i_2,l]\big)^{\mathbf{s}[l]} \qquad \text{if } l>k.\end{aligned}$$

using the exponents $\mathbf{s}[l] \in \mathbb{Z}_p^*$ that were chosen in (19). Finally, our adversary $\mathcal{B}$ defines the $n \times n$ matrix $\{\mathbf{CT}[l_2,l_1]\}_{l_2,l_1\in\{1,\ldots,n\}}$ of HPE ciphertexts

$$\mathbf{CT}[l_2,l_1] = \big(\mathbf{J}[l_2], \mathbf{C}_w[l_2,l_1], \{\mathbf{C}[i_1,i_2,l_2,l_1]\}_{i_1\in\{1,\ldots,d\},\ i_2\in\{1,\ldots,\mu\}}\big).$$

Finally, $\mathcal{B}$ defines $\mathsf{mpk} := \big(\mathsf{PP}_{core}, \{\mathbf{CT}[l_2,l_1]\}_{l_2,l_1\in\{1,\ldots,n\}}\big)$ and sends it to the adversary $\mathcal{A}$.

When it comes to answer $\mathcal{A}$'s private key queries for hierarchical identities $(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)$, $\mathcal{B}$ first encodes each level's identity $\mathsf{id}_{i_1} \in \{0,1\}^\mu$ as a μ-vector $\vec{X}_{i_1} = (\mathsf{id}_{i_1}[1],\ldots,\mathsf{id}_{i_1}[\mu])$ for each $i_1 \in \{1,\ldots,\ell\}$. Although $\mathcal{B}$ does not entirely know msk, the decryption components $(\mathbf{D}[l_1], \mathbf{D}_w[l_1], \mathbf{D}_{i_1}[l_1])$ of the private key are always directly computable when $l_1 \neq k$: namely, $\mathcal{B}$ chooses $\mathbf{D}_w[l_1] \xleftarrow{R} \hat{\mathbb{G}}$ and $\mathbf{D}_{i_1}[l_1] \xleftarrow{R} \hat{\mathbb{G}}$, for $i_1 = 1$ to ℓ, and computes

$$\mathbf{D}[l_1] = \prod_{i_1=1}^{\ell} \mathbf{D}_{i_1}[l_1]^{\sum_{i_2=1}^{\mu}\boldsymbol{\gamma}[i_1,i_2,l_1]\cdot\mathsf{id}_{i_1}[i_2]} \cdot \mathbf{D}_w[l_1]^{\boldsymbol{\zeta}[l_1]}.$$

It is easy to see that $(\mathbf{D}[l_1], \mathbf{D}_w, \{\mathbf{D}_{i_1}[l_1]\}_{i_1=1}^{\ell})$ forms a decryption component of the form (7). Moreover, the delegation components can be obtained exactly in the same way.

As for the remaining coordinate $l_1 = k$, the simulator $\mathcal{B}$ aborts if, for any $\tilde{\gamma} \in \{0,1\}$, the obtained hierarchical vector $(\vec{X}_1,\ldots,\vec{X}_\ell)$ is one for which $\langle y_{i_1}^{(\tilde{\gamma})}, X_{i_1}\rangle = 0$ for each $i_1 \in \{1,\ldots,\ell\}$ (which translates into

$$f_{(\vec{X}_1,\ldots,\vec{X}_\ell)}(\mathbf{y}_1^{(\tilde{\gamma})},\ldots,\mathbf{y}_d^{(\tilde{\gamma})}) = 1$$

in the predicate encryption language). Otherwise, $\mathcal{B}$ can obtain the missing private key components by invoking its HPE challenger to obtain a complete private key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)} = (\mathbf{SK}_D, \mathbf{SK}_{DL})$.

It is easy to check that, if the HPE challenger's bit is $\beta = 0$, mpk is distributed as in Game RL_k. In contrast, if $\beta = 1$, mpk has the same distribution as in Game RL_{k-1}. □

B.2 Selective Security of HF

We consider our HF with $\mu = 2$, $\Sigma_{\mathsf{ID}} = \{(1, x) : x \in \mathbb{Z}_p^*\}$ and $\mathsf{IdSp} = (\Sigma_{\mathsf{ID}})^{(\leq d)}$. Let $(\mathsf{id}_1^\star, \ldots, \mathsf{id}_{\ell^\star}^\star) \in \mathsf{IdSp}$ be the identity chosen by the adversary. Define the sibling LHF with $\mathsf{AuxSp} = \mathbb{Z}_p^{2d}$ and auxiliary input $\mathbf{y}^{(1)} = [\mathbf{y}_1^{(1)} | \ldots | \mathbf{y}_d^{(1)}] \in \mathbb{Z}_p^{2d}$ where $\mathbf{y}_{i_1}^{(1)} = (-id_{i_1}^\star, 1)$ for all $i_1 \in \{1, \ldots, \ell^\star\}$ and $\mathbf{y}_{i_1}^{(1)} = (1, 0)$ for $i_1 \in \{\ell^\star + 1, \ldots, d\}$. Selective security is established in the following theorem. In this selective setting, we will omit ζ in the notation, because the trivial pre-output stage that always outputs $d_2 = 1$ will be enough in this case and, as a consequence, the experiments do not depend on ζ.

Theorem 4. *Let $n > \log p$ and let $\omega = n - \log p$. Let HF be the HIB-TDF with parameters $n, d, \mu = 2$, $\Sigma_{\mathsf{ID}} = \{(1, x) : x \in \mathbb{Z}_p^*\}$ and $\mathsf{IdSp} = (\Sigma_{\mathsf{ID}})^{(\leq d)}$. Let LHF be the sibling associated with it as above. Then HF is $(\omega, 1)$-partially lossy against selective-id adversaries.*

Specifically regarding condition (i), for any selective-id adversary $\mathcal{A}$ there exist algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\omega}(\mathcal{A}) \leq n \cdot \big((2d+1) \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_1}(\mathcal{B}_1) + q \cdot \mathbf{Adv}^{\mathcal{P}\text{-DDH}_2}(\mathcal{B}_2)\big)$$

The running time of $\mathcal{B}_1$ and $\mathcal{B}_2$ are comparable to the running time of $\mathcal{A}$.

Proof. Let RL_0 and RL_n be the games specified above. We claim that both

$$\Pr[d_{exp}^{\mathcal{A}} = 1 \mid \mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\omega}] = \Pr[RL_0 \Rightarrow 1]$$

and

$$|\Pr[d_{exp}^{\mathcal{A}} = 1 \mid \mathsf{LOSSY}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\omega}] - \Pr[RL_n \Rightarrow 1]| \in \mathsf{negl}(\varrho),$$

which implies condition (i) of partial lossiness, when combined with Lemma 1.

To prove that, we just need to justify that, when using $\mathbf{y}^{(0)}$ as the auxiliary input, the function HF.Eval is injective for all identities id. On the other hand, when using $\mathbf{y}^{(1)}$, the function HF.Eval will be lossy for identities $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ such that $\langle \mathbf{y}_{i_1}^{(1)}, \mathsf{id}_{i_1} \rangle = 0$ for all $i_1 \in \{1, \ldots, \ell\}$ and injective for all other identities.

To see that, note that all the terms of the output of HF.Eval are determined by $\langle \mathbf{s}, X \rangle$ except the term $\mathbf{CT}_{\mathsf{id}}[i_1, l_1]$, which is also determined by $\mathbf{s}[l_1] \cdot x_{l_1} \cdot \langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle$. Therefore, when $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle = 0$ for all $i_1 \in \{1, \ldots, \ell\}$, the function will be determined only by $\langle \mathbf{s}, X \rangle$, which can take only p values, so HF.Eval will be lossy with lossiness $\lambda\big(\mathsf{HF.Eval}\big(\mathsf{pms}, \mathsf{mpk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), .\big)\big) \geq n - \log p = \omega$. On the other hand, if there is at least one index i_1 for which $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle \neq 0$, then HF.Eval will be injective with overwhelming probability and HF.Inv will be its inverse, also with overwhelming probability. Finally, observe that when the auxiliary input is $\mathbf{y}^{(0)}$, then for all identities $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and for all $i_1 \in \{1, \ldots, \ell\}$, $\langle \mathbf{y}_{i_1}^{(0)}, \mathsf{id}_{i_1} \rangle \equiv 1$ by construction.

With this, we conclude that $\mathbf{y}^{(0)}$ makes HF.Eval injective for all identities and $\mathbf{y}^{(1)}$ makes HF.Eval lossy only for identities such that $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle = 0$ for all $i_1 \in \{1, \ldots, \ell\}$, and in this case the lossiness is $n - \log p$ (where when we claim that HF.Eval is injective, it holds with overwhelming probability). This concludes the proof of condition (i).

Regarding conditions (ii) and (iii), when the auxiliary input is $\mathbf{y}^{(1)}$, then HF.Eval is injective for all valid queried id in "Reveal key" queries and HF.Eval is lossy for $\mathsf{id}^\star = (\mathsf{id}_1^\star, \ldots, \mathsf{id}_{\ell^\star}^\star)$. Indeed, we are setting $\mathbf{y}_{i_1}^{(1)} = (-id_{i_1}^\star, 1)$ for all $i_1 \in \{1, \ldots, \ell^\star\}$ and $\mathbf{y}_{i_1}^{(1)} = (1, 0)$ for $i_1 \in \{\ell^\star + 1, \ldots, d\}$. This implies that if $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ is an identity, then $\langle \mathbf{y}_{i_1}, \mathsf{id}_{i_1} \rangle = 0$ for all $i_1 \in \{1, \ldots, \ell\}$ if and only if id is a prefix of $\mathsf{id}^\star$. As the adversary is not allowed to make "Reveal key" queries for

prefixes of $\mathsf{id}^\star$, then the condition is satisfied. This implies that $d_1 = 1$ with overwhelming probability. Also, the pre-output stage $\mathcal{P}$ in this case simply outputs $d_2 = 1$, always. This implies that $d^{\mathcal{A}}_{\neg abort}$ is 1 with overwhelming probability and, as a consequence, $\Pr[d^{\mathcal{A}}_{\neg abort} = 1 \mid \mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\omega,\zeta}]$ is negligibly close to 1 and $\left(\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\omega,\zeta} \wedge d^{\mathcal{A}}_{\neg abort} = 1] - \frac{1}{2}\right)$ is negligibly close to $\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}^{\mathcal{A}}_{\mathsf{HF},\mathsf{LHF},\omega}] - \frac{1}{2}$. These are precisely the requirements for HF to fulfill conditions (ii) and (iii), with $\delta = 1$. □

B.3 Adaptive Security: Proof of Lemma 2

Fix the view of the adversary $\mathcal{A}$, which implies fixing the queried identities $\mathsf{id}^{(1)}, \ldots, \mathsf{id}^{(q)}, \mathsf{id}^\star$. Although we are assuming that the adversary $\mathcal{A}$ makes the maximum number of queries, with a smaller number of queries we would have the same bounds. We abbreviate $\eta = \eta(IS, \mathsf{id}^\star)$, $\mathbf{y} = \mathbf{y}^{(1)}$ and also call $\ell^\star$ the depth of the challenge identity $\mathsf{id}^\star$. For an integer t, define the event

$$E_t: \quad \bigwedge_{i=1}^{q} \left(\bigvee_{i_1=1}^{\ell^{(i)}} \left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1} \rangle \neq 0 \mod t \right) \right) \wedge \bigwedge_{i_1=1}^{\ell^\star} \left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1} \rangle = 0 \mod t \right)$$

We denote $\mathbf{Y} = \{y'_1, \ldots, y'_d, \mathbf{y}_1, \ldots, \mathbf{y}_d, \xi_1, \ldots, \xi_d\}$. For all $i_1 \in \{1, \ldots, \ell^\star\}$, we have

$$\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1} \rangle = y'_{i_1} + \sum_{k=2}^{\mu} \mathbf{y}_{i_1}[k] \mathsf{id}_{i_1}[k] - 2\xi_{i_1} q$$

for some $0 \leq \xi_{i_1} \leq \mu - 1$. In particular, observe that

$$0 \leq y'_{i_1} + \sum_{k=2}^{\mu} \mathbf{y}_{i_1}[k] \mathsf{id}_{i_1}[k] < 2q\mu < p.$$

Let us define the value $\xi^\star_{i_1} := \lfloor (y'_{i_1} + \sum_{k=2}^{\mu} \mathbf{y}_{i_1}[k] \mathsf{id}^\star_{i_1}[k])/2q \rfloor$. If we have the two conditions

$$\begin{aligned} &\xi_{i_1} = \xi^\star_{i_1} && \text{for each } i_1 \in \{1, \ldots, \ell^\star\} \\ &\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q, \end{aligned}$$

then clearly $\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1} \rangle = 0 \mod p$. Also, if $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1} \rangle \neq 0 \mod 2q$, then we also have $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1} \rangle \neq 0 \mod p$. Using these observations, we have

$$\begin{aligned} \eta &\geq \Pr[\xi_{i_1} = \xi^\star_{i_1} \quad \forall i_1 \in \{1, \ldots, \ell^\star\}] \Pr_{\mathbf{Y}}[E_p \mid \xi_{i_1} = \xi^\star_{i_1} \quad \forall i_1 \in \{1, \ldots, \ell^\star\}] \\ &= \frac{1}{\mu^{\ell^\star}} \Pr_{\mathbf{Y}}[E_p \mid \xi_{i_1} = \xi^\star_{i_1} \quad \forall i_1 \in \{1, \ldots, \ell^\star\}] \\ &\geq \frac{1}{\mu^{\ell^\star}} \Pr_{\mathbf{Y}}[E_{2q} \mid \xi_{i_1} = \xi^\star_{i_1} \quad \forall i_1 \in \{1, \ldots, \ell^\star\}] \\ &= \frac{1}{\mu^{\ell^\star}} \Pr_{\mathbf{Y}'}[E_{2q} \mid \xi_{i_1} = \xi^\star_{i_1} \quad \forall i_1 \in \{1, \ldots, \ell^\star\}] \end{aligned}$$

where $\mathbf{Y}'$ contains $\{\mathbf{y}'_1, \ldots, \mathbf{y}'_d\}$ and $\mathbf{y}'_{i_1} = (y'_{i_1}, \mathbf{y}_{i_1}[2], \ldots, \mathbf{y}_{i_1}[\mu])$ for all $i \in \{1, \ldots, d\}$. Note that the second inequality above holds because of the condition $\xi_{i_1} = \xi^\star_{i_1}$. If we now define

$\eta_{2q} = \Pr_{\mathbf{Y}'}[E_{2q} \mid \xi_{i_1} = \xi^\star_{i_1}\ \forall i_1 \in \{1,\ldots,\ell^\star\}]$, we just showed that $\eta \geq \frac{1}{\mu^{\ell^\star}} \cdot \eta_{2q}$.

Now, we observe some facts about $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle$. First, observe that $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle$ and $\langle \mathsf{id}_{i'_1}, \mathbf{y}_{i'_1}\rangle$ are independent for $i_1 \neq i'_1$. This is because of the way $\mathbf{Y}$ is chosen.

Also, note that for any id_{i_1}, $a \in \mathbb{Z}$, $\Pr_{\mathbf{Y}'}[\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle = a \mod 2q] = 1/2q$. This is because for any choice of $\mathbf{y}_{i_1}[2], \ldots, \mathbf{y}_{i_1}[\mu]$, there is only one value of y'_{i_1} for which the equality holds.

Consider $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \neq \mathsf{id}' = (\mathsf{id}'_1, \ldots, \mathsf{id}'_{\ell'})$ and id not being a prefix of id' and $a, b \in \mathbb{Z}$. First, if $\ell > \ell'$, for each $i_1 \in \{\ell'+1, \ldots, \ell\}$ such that $\mathsf{id}_{i_1} \neq \mathsf{id}'_{i_1}$, we have

$$\Pr_{\mathbf{Y}'}[\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle = a \mod 2q | \bigwedge_{k=1}^{\ell'} \langle \mathsf{id}'_k, \mathbf{y}_k\rangle = b \mod 2q] = \Pr_{\mathbf{Y}'}[\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle = a \mod 2q] = 1/2q.$$

This happens because $\bigwedge_{k=1}^{\ell'} \langle \mathsf{id}'_k, \mathbf{y}_k\rangle = b \mod 2q$ does not impose any condition on $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle$ and we can apply the same arguments as previously.

On the other hand, for all $i_1 \leq \ell'$, $\Pr_{\mathbf{Y}'}[\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle = a \mod 2q | \bigwedge_{k=1}^{\ell'} \langle \mathsf{id}'_k, \mathbf{y}_k\rangle = b \mod 2q]$ is either 0, if $\mathsf{id}_{i_1} = \mathsf{id}'_{i_1}$ or $1/2q$, if $\mathsf{id}_{i_1} \neq \mathsf{id}'_{i_1}$. The second fact is because, if $\mathsf{id}_{i_1} \neq \mathsf{id}'_{i_1}$, there exists an index j for which $\mathsf{id}_{i_1}[j] = 1$ and $\mathsf{id}'_{i_1}[j] = 0$ or the other way around. We see that, if we fix all $\mathbf{y}_{i_1}[i]$ for $i \neq j$ so that $\langle \mathsf{id}'_{i_1}, \mathbf{y}_{i_1}\rangle = b \mod 2q$, then there is only one value for $\mathbf{y}_{i_1}[j]$ so that $\langle \mathsf{id}_{i_1}, \mathbf{y}_{i_1}\rangle = a \mod 2q$.

With all these observations, we calculate the following bound on η_{2q}:

$$\begin{aligned}
\eta_{2q} &= \Pr_{\mathbf{Y}'}\left[\bigwedge_{i=1}^{q}\left(\bigvee_{i_1=1}^{\ell^{(i)}}\left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1}\rangle \neq 0 \mod 2q\right)\right) \mid \bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right] \cdot \\
&\qquad \cdot \Pr_{\mathbf{Y}'}\left[\bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right] \\
&= 1/(2q)^{\ell^\star} \Pr_{\mathbf{Y}'}\left[\bigwedge_{i=1}^{q}\left(\bigvee_{i_1=1}^{\ell^{(i)}}\left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1}\rangle \neq 0 \mod 2q\right)\right) \mid \bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right] \\
&= 1/(2q)^{\ell^\star}\left(1 - \Pr_{\mathbf{Y}'}\left[\bigvee_{i=1}^{q}\left(\bigwedge_{i_1=1}^{\ell^{(i)}}\left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right) \mid \bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right]\right) \qquad (21) \\
&\geq 1/(2q)^{\ell^\star}\left(1 - \sum_{i=1}^{q}\Pr_{\mathbf{Y}'}\left[\bigwedge_{i_1=1}^{\ell^{(i)}}\left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right) \mid \bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right]\right)
\end{aligned}$$

We now focus on how to bound

$$\Pr_{\mathbf{Y}'}\left[\bigwedge_{i_1=1}^{\ell^{(i)}}\left(\langle \mathsf{id}^{(i)}_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right) \mid \bigwedge_{i_1=1}^{\ell^\star}\left(\langle \mathsf{id}^\star_{i_1}, \mathbf{y}_{i_1}\rangle = 0 \mod 2q\right)\right].$$

First, observe that $\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q$ is independent to $\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i'_1} \rangle = 0 \mod 2q$ if $i_1 \neq i'_1$. As a consequence, we only need to compute $\Pr_{\mathbf{Y}'}\left[\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q | \bigwedge_{k=1}^{\ell^\star} (\langle \mathsf{id}_k^\star, \mathbf{y}_k \rangle = 0 \mod 2q)\right]$.

To this end, we consider two cases: that $\ell^{(i)} > \ell^\star$ or $\ell^{(i)} \leq \ell^\star$. In the first case, for each $i_1 \in \{\ell^\star + 1, \ldots, \ell^{(i)}\}$ such that

$$\Pr_{\mathbf{Y}'}\left[\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q | \bigwedge_{k=1}^{\ell^\star} (\langle \mathsf{id}_k^\star, \mathbf{y}_k \rangle = 0 \mod 2q)\right] = \Pr_{\mathbf{Y}'}\left[\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q\right] = 1/2q.$$

For all indices $i_1 \in \ell^\star$, the same probability is either 1 or $1/(2q)$.

If $\ell^{(i)} \leq \ell^\star$, for each $i_1 \in \{1, \ldots, \ell^{(i)}\}$, we have

$$\Pr_{\mathbf{Y}'}\left[\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q | \bigwedge_{k=1}^{\ell^\star} (\langle \mathsf{id}_k^\star, \mathbf{y}_k \rangle = 0 \mod 2q)\right] = \Pr_{\mathbf{Y}'}\left[\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q | \langle \mathsf{id}_{i_1}^\star, \mathbf{y}_{i_1} \rangle = 0 \mod 2q\right]$$

which is 1 if $\mathsf{id}_{i_1}^{(i)} = \mathsf{id}_{i_1}^\star$ and $1/(2q)$ otherwise due to the fact stated above.

Define $\chi_1^{(i)} = \max(\ell^{(i)} - \ell^\star, 0)$ and $\chi_2^{(i)} = \#\{1 \leq i_1 \leq \min(\ell^\star, \ell^{(i)}) | \mathsf{id}_{i_1}^{(i)} \neq \mathsf{id}_{i_1}^\star\}$. Note that, by the restrictions imposed on $\mathsf{id}^{(i)}$, we have $\chi_1^{(i)} + \chi_2^{(i)} \geq 1$ for all $i \in \{1, \ldots, q\}$. Putting it all together, we find

$$\Pr_{\mathbf{Y}'}\left[\bigwedge_{i_1=1}^{\ell^{(i)}} \left(\langle \mathsf{id}_{i_1}^{(i)}, \mathbf{y}_{i_1} \rangle = 0 \mod 2q\right) | \bigwedge_{i_1=1}^{\ell^\star} \left(\langle \mathsf{id}_{i_1}^\star, \mathbf{y}_{i_1} \rangle = 0 \mod 2q\right)\right] = \frac{1}{(2q)^{\chi_1^{(i)} + \chi_2^{(i)}}} \leq \frac{1}{(2q)}$$

We can conclude that $\eta_{2q} \geq 1/(2^{\ell^\star + 1} q^{\ell^\star})$. Combining this bound with the bound on η, we get the statement of the Lemma. □

B.4 Adaptive Security: Proof of Theorem 3

Regarding condition (i) of partial lossiness, let RL_0 and RL_n be the games specified in Section 4.2. Let $\widehat{RL_0}$ and $\widehat{RL_n}$ be the games which are the same as RL_0 and RL_n except that they include the artificial abort stage described above. First, we claim that

$$\Pr[\widehat{RL_0} \Rightarrow 1] - \Pr[\widehat{RL_n} \Rightarrow 1] \leq n \cdot \left((d \cdot \mu + 1) \cdot \mathbf{Adv}^{\mathcal{P}\text{-BDH}_1}(\mathcal{B}_1) + q \cdot \mathbf{Adv}^{\mathcal{P}\text{-DDH}_2}(\mathcal{B}_2)\right)$$

The proof for this statement is almost identical to the proof for Lemma 1.

We now claim that both $\Pr[d_{exp}^{\mathcal{A}} = 1 \mid \mathsf{REAL}] = \Pr[\widehat{RL_0} \Rightarrow 1]$ and

$$|\Pr[d_{exp}^{\mathcal{A}} = 1 \mid \mathsf{LOSSY}] - \Pr[\widehat{RL_n} \Rightarrow 1]| \in \mathsf{negl}(\varrho).$$

These statements imply condition (i) of partial lossiness.

The proof of the last two statements is identical to the proof for Theorem 4: when using the auxiliary input $\mathbf{y}^{(0)}$, $\mathsf{HF.Eval}$ will always be injective. On the other hand, when using $\mathbf{y}^{(1)}$, $\mathsf{HF.Eval}$ will be lossy for identities $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ such that $\langle \mathbf{y}_{i_1}^{(1)}, \mathsf{id}_{i_1} \rangle = 0$ for all $i_1 \in \{1, \ldots, \ell\}$, with lossiness $\lambda\left(\mathsf{HF.Eval}\big(\mathsf{pms}, \mathsf{mpk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), .\big)\right) \geq n - \log p = \omega$, and $\mathsf{HF.Eval}$ will be injective for the other identities.

Seeing that condition (ii) is fulfilled is straightforward: from Lemma 2, we know that $\eta_{low} \leq \Pr[d_1 = 1 \mid \mathsf{REAL}]$ and, by construction, $\eta_{low} \leq \Pr[d_2 = 1 \mid d_1 = 1 \wedge \mathsf{REAL}]$. This gives us the lower bound η_{low}^2 on $\Pr[d_{\neg abort} = 1 \mid \mathsf{REAL}]$, which is non-negligible. However, the value of ϵ_1 will not be this lower bound. Instead, we will show that another condition (which we give at the end of this proof) is satisfied and guarantees the existence of ϵ_1 and ϵ_2. The proof for this is actually very similar to the proof in [35]. Here, we briefly outline the details. First, due to Chernoff bounds we have the following:

$$\Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 1 \wedge \mathsf{REAL}] \geq \eta_{low}(1 - \frac{1}{4}\zeta)$$

and

$$\Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 0 \wedge \mathsf{REAL}] \leq \eta_{low}(1 + \frac{3}{8}\zeta)$$

We refer the reader to [35] for details on how to compute these bounds.

Let us now assume the existence of a PPT adversary $\mathcal{A}$ such that

$$\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] \; - \; \frac{1}{2} > \zeta$$

for some non-negligible ζ. This implies that we can write $\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] > 1/2 + \zeta$ and $\Pr[d_{\mathcal{A}} = 0 \mid \mathsf{REAL}] < 1/2 - \zeta$. Combining these inequalities with the two inequalities above (from Chernoff bounds) we obtain

$$\begin{aligned}\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] \cdot \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 1 \wedge \mathsf{REAL}] \\ - \; \Pr[d_{\mathcal{A}} = 0 \mid \mathsf{REAL}] \cdot \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 0 \wedge \mathsf{REAL}] > 2\delta\zeta \quad (22)\end{aligned}$$

with $\delta = 13\eta_{low}/16 = 13/\left(32 \cdot (2q\mu)^d\right)$. Now, let us observe that

$$\Pr[d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] \; \cdot \; \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 1 \wedge \mathsf{REAL}] = \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \wedge d_{\mathcal{A}} = 1 \mid \mathsf{REAL}]$$

and

$$\begin{aligned}&\Pr[d_{\mathcal{A}} = 0 \mid \mathsf{REAL}] \cdot \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid d_{\mathcal{A}} = 0 \wedge \mathsf{REAL}] = \\ &= \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \wedge d_{\mathcal{A}} = 0 \mid \mathsf{REAL}] \\ &= \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid \mathsf{REAL}] \cdot \Pr[d_{\mathcal{A}} = 0 \mid d_{\neg abort}^{\mathcal{A}} = 1 \wedge \mathsf{REAL}] \\ &= \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid \mathsf{REAL}] \cdot (1 - \Pr[d_{\mathcal{A}} = 1 \mid d_{\neg abort}^{\mathcal{A}} = 1 \wedge \mathsf{REAL}]) \\ &= \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid \mathsf{REAL}] - \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \wedge d_{\mathcal{A}} = 1 \mid \mathsf{REAL}]\end{aligned}$$

Combining these two equalities with inequality (22), we obtain

$$2\delta\zeta < 2\Pr[d_{\neg abort}^{\mathcal{A}} = 1 \wedge d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] - \Pr[d_{\neg abort}^{\mathcal{A}} = 1 \mid \mathsf{REAL}].$$

Dividing this inequality by 2 and using the fact that

$$\Pr[d^{\mathcal{A}}_{\neg abort} = 1 \wedge d_{\mathcal{A}} = 1 \mid \mathsf{REAL}] = \Pr[d^{\mathcal{A}}_{\neg abort} = 1 \mid \mathsf{REAL}] \cdot \Pr[d_{\mathcal{A}} = 1 \mid d^{\mathcal{A}}_{\neg abort} = 1 \wedge \mathsf{REAL}],$$

we obtain the relation

$$\delta\zeta < \Pr[d^{\mathcal{A}}_{\neg abort} = 1 \mid \mathsf{REAL}] \cdot \left(\Pr[d_{\mathcal{A}} = 1 \mid d^{\mathcal{A}}_{\neg abort} = 1 \wedge \mathsf{REAL}] - \frac{1}{2} \right)$$

Finally, this shows that there exist ϵ_1 and ϵ_2 such that (ii) and (iii) are satisfied and that their product is δ.

□

C Adaptive-id Secure Deterministic (H)IBE

A hierarchical identity-based deterministic encryption scheme (HIB-DE) is a tuple of efficient algorithms $\mathsf{HIB\text{-}DE} = (\mathsf{HIB\text{-}DE.Setup}, \mathsf{HIB\text{-}DE.MKg}, \mathsf{HIB\text{-}DE.Kg}, \mathsf{HIB\text{-}DE.Del}, \mathsf{HIB\text{-}DE.Enc}, \mathsf{HIB\text{-}DE.Dec})$. The setup algorithm $\mathsf{HIB\text{-}DE.Setup}$ takes as input a security parameter $\varrho \in \mathbb{N}$, the (constant) number of levels in the hierarchy $d \in \mathbb{N}$, the length of the identities $\mu \in \mathsf{poly}(\varrho)$ and the length of the plaintexts $s \in \mathsf{poly}(\varrho)$, and outputs a set of global public parameters pms, which specifies an identity space IdSp and the necessary mathematical objects and hash functions. The master key generation algorithm $\mathsf{HIB\text{-}DE.MKg}$ takes as input pms and outputs a master public key mpk and a master secret key msk. The key generation algorithm $\mathsf{HIB\text{-}DE.Kg}$ takes as input pms, msk and a hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in \mathsf{IdSp}$, for some $\ell \geq 1$ and outputs a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$. The delegation algorithm $\mathsf{HIB\text{-}DE.Del}$ takes as input pms, msk, a hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$ for it, and an additional identity $\mathsf{id}_{\ell+1}$; the output is a secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell,\mathsf{id}_{\ell+1})}$ for the hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell, \mathsf{id}_{\ell+1})$ iff $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell, \mathsf{id}_{\ell+1}) \in \mathsf{IdSp}$. The evaluation algorithm $\mathsf{HIB\text{-}DE.Enc}$ takes as input pms, msk, a hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ and a value $m \in \{0,1\}^s$; the result of the evaluation is denoted as C. Finally, the inversion algorithm $\mathsf{HIB\text{-}DE.Dec}$ takes as input pms, msk, a hierarchical identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, a secret key $\mathbf{SK}_{\mathsf{id}}$ for it and an evaluation C, and outputs a value $\tilde{m} \in \{0,1\}^s$.

A HIB-DE satisfies the property of correctness if

$$\mathsf{HIB\text{-}DE.Dec}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{id}, \mathbf{SK}_{\mathsf{id}}, \mathsf{HIB\text{-}DE.Enc}\big(\mathsf{pms}, \mathsf{mpk}, \mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), m\big)\big) \;=\; m,$$

for any $m \in \{0,1\}^s$, any $\mathsf{pms}, (\mathsf{mpk}, \mathsf{msk})$ generated by $\mathsf{HIB\text{-}DE.Setup}$ and $\mathsf{HIB\text{-}DE.MKg}$, any hierarchical identity $(\mathsf{id}_1, \ldots, \mathsf{id}_\ell) \in \mathsf{IdSp}$ and any secret key $\mathbf{SK}_{(\mathsf{id}_1,\ldots,\mathsf{id}_\ell)}$ generated either by running $\mathsf{HIB\text{-}DE.Kg}\big(\mathsf{pms}, \mathsf{msk}, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)\big)$ or by applying the delegation algorithm $\mathsf{HIB\text{-}DE.Del}$ to secret keys of shorter hierarchical identities.

Bellare *et al.* [7] gave several definitions for deterministic encryption and proved them equivalent. In the case of *block sources*[9] [10], Boldyreva *et al.* [10] proved that single-challenge security (called PRIV1 security in [5, 10]) is equivalent to multi-challenge security (referred to as PRIV security [5], where the adversary is given a vector of challenge ciphertexts) in the sense of indistinguishability-based definitions. The simplified indistinguishability-based notion, called PRIV1-IND hereafter, is somewhat handier to work with and we thus use this one.

We define PRIV1-IND-ID security, the natural analogue of PRIV1-IND security in the IBE scenario. The security notion is defined via the following experiment between a challenger and an adversary. Some instructions depend on whether we are in the *selective* or *adaptive* security case.

[9] Informally, a block source is a distribution of message vectors where each component has high min-entropy conditionally on previous ones.

Definition 3. *We define* $\mathit{Guess}^{\mathcal{A}}_{\mathsf{HIB\text{-}DE}}(M)$*, for a random variable* M *as follows:*

0. *The challenger* $\mathcal{C}$ *chooses global parameters* pms *by running* $\mathit{HIB\text{-}DE.Setup}$*. The parameters* pms *are given to* $\mathcal{A}$*, who replies by choosing a hierarchical identity* $\mathsf{id}^\dagger = (\mathsf{id}_1^\dagger, \ldots, \mathsf{id}_{\ell^\dagger}^\dagger)$*, for some* $\ell^\dagger \leq d$*.*
1. *The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathit{HIB\text{-}DE.MKg}(\mathsf{pms})$ *and sends* mpk *to* $\mathcal{A}$*. Also, two lists* $QS \leftarrow \emptyset$*,* $IS \leftarrow \emptyset$ *are initialized.*
2. $\mathcal{A}$ *is allowed to make a number of adaptive queries for hierarchical identities* $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ *(where hierarchical identities are encoded as hierarchical vectors* $(\vec{X}_1, \ldots, \vec{X}_\ell)$*).*
   - **Create-key***:* $\mathcal{A}$ *provides* id *and the challenger* $\mathcal{C}$ *creates a private key* SK_{id} *by running* $\mathit{HIB\text{-}DE.Kg}(\mathsf{pms}, \mathsf{msk}, \mathsf{id})$*. The list* QS *is updated as* $QS = QS \cup \{\mathsf{id}\}$*.*
   - **Create-delegated-key***:* $\mathcal{A}$ *provides* $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ *and* $\mathsf{id}_{\ell+1}$ *such that* $\mathsf{id} \in QS$*. The challenger* $\mathcal{C}$ *then computes* $SK_{\mathsf{id}'}$ *for* $\mathsf{id}' = (\mathsf{id}_1, \ldots, \mathsf{id}_{\ell+1})$ *by running the delegation algorithm* $\mathit{HIB\text{-}DE.Del}\big(\mathsf{pms}, \mathsf{mpk}_\beta, SK_{\mathsf{id}}, \mathsf{id}_{\ell+1}\big)$*. The list* QS *is updated as* $QS = QS \cup \{\mathsf{id}'\}$*.*
   - **Reveal-key***:* $\mathcal{A}$ *provides* id *with the restriction that if* $\mathcal{A}$ *is selective, then* $\mathsf{id} \not\leq \mathsf{id}^\dagger$*.* $\mathcal{C}$ *returns* $\perp$ *if* $\mathsf{id} \notin QS$*. Otherwise,* SK_{id} *is returned to* $\mathcal{A}$ *and the list* IS *is updated as* $IS = IS \cup \{\mathsf{id}\}$*.*
3. *The adversary* $\mathcal{A}$ *outputs a hierarchical identity* $\mathsf{id}^\star = (\mathsf{id}_1^\star, \ldots, \mathsf{id}_{\ell^\star}^\star)$*. In the selective setting, we impose* $\ell^\star = \ell^\dagger$ *and* $\mathsf{id}_j^\star = \mathsf{id}_j^\dagger$ *for each* $j \in \{1, \ldots, \ell^\star\}$*. In the adaptive case, no element of* IS *is a prefix of* $\mathsf{id}^\star$*.*
4. *The challenger encrypts a message* m *sampled from the given message distribution* M*. The resulting ciphertext is sent to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs a bit* $b' \in \{0, 1\}$*, which is the output of the experiment.*

Recall that a random variable X over $\{0,1\}^s$ is called a (t, s)-source if $H_\infty(X) \geq t$, where $H_\infty(X)$ is the min-entropy of X; $H_\infty(X) = -\log(\max_x P_X(x))$. We now give the identity-based version of IND-PRIV1 security of [10].

Definition 4. *An* s*-bit encryption scheme* $\mathsf{HIB\text{-}DE}$ *is PRIV1-IND-ID secure for* (t, s)*-sources if for any* (t, s)*-sources* M_0 *and* M_1 *and all polynomial time adversaries* $\mathcal{A}$*, the PRIV1-IND-ID advantage*

$$Adv^{\text{priv1-ind-id}}_{\mathsf{HIB\text{-}DE}}(\mathcal{A}, M_0, M_1) = \Pr[\mathit{Guess}^{\mathcal{A}}_{\mathsf{HIB\text{-}DE}}(M_0) = 1] - \Pr[\mathit{Guess}^{\mathcal{A}}_{\mathsf{HIB\text{-}DE}}(M_1) = 1]$$

of $\mathcal{A}$ *against* $\mathsf{HIB\text{-}DE}$ *is negligible.*

C.1 Universal HF implies deterministic encryption

As it can be seen from the definition, a hierarchical identity-based encryption scheme is very close to an HIB-TDF, even syntactically. Indeed, as shown in [10], in the public key setting a natural way to construct a deterministic DE encryption scheme is by defining the algorithms of DE as their natural counterparts in some lossy LTDF. Boldyreva *et al.* [10] show that if the lossy function is also universal, this construction is PRIV1-IND secure. For functions not satisfying this property, the result follows also directly by an extension of the Crooked Leftover Hash Lemma of Dodis and Smith [19] given by Boldyreva *et al.* for this purpose. Similarly, in the identity-based setting, one can construct a hierarchical deterministic identity-based encryption scheme HIB-DE from HF, a lossy HIB-TDF, by defining the algorithms of HIB-DE from those of HF in the natural way (in this case, we say that HIB-DE is defined by HF). We will show that with the additional requirement of

universality, such a construction is secure in some natural extension of PRIV1-IND security. To do without this extra condition, it is possible to use the same techniques of Boldyreva *et al.*.

More specifically, universality requires that if LHF is the lossy sibling of a (ω, δ)-lossy HIB-TDF function, then $\mathsf{HF.Setup}$, $\mathsf{LHF.MKg}$ and $\mathsf{HF.Eval}(\mathsf{pms}, \mathsf{mpk}, \mathsf{id}^\star, \cdot)$ induce a universal hash function when we have a lossy function for $\mathsf{id}^\star$, i.e., $\lambda\left(\mathsf{HF.Eval}\big(\mathsf{pms}, \mathsf{mpk}_1, (\mathsf{id}_1^\star, \ldots, \mathsf{id}_{\ell^\star}^\star), \cdot\big)\right) \geq \omega$. It is not difficult to see that both [9] and our construction induce a universal hash in the lossy mode.

Theorem 5. *If* HF *is a universal* (ω, δ)*-lossy HIB-TDF function with input* $\{0,1\}^n$*, then the hierarchical deterministic encryption function* $\mathsf{HIB\text{-}DE}$ *defined by it is secure for poly-time sampleable* (t, s)*-sources such that* $t \geq n - \omega + 2\log(1/\epsilon)$ *for some negligible* ϵ*. In particular, for any two* (t, s)*-sources* M_0 *and* M_1 *and for every PRIV1-IND-ID adversary* $\mathcal{B}$ *against* $\mathsf{HIB\text{-}DE}$ *there exists a PPT adversary* $\mathcal{A}$ *against* HF *such that*

$$\mathbf{Adv}^{\text{lossy}}_{\mathsf{HF},\mathsf{LHF},\mathcal{P},\omega,\zeta}(\mathcal{A}) \geq \frac{1}{3} \cdot \delta \cdot \mathbf{Adv}^{\text{priv1}-\text{ind}-\text{id}}_{\mathsf{HIB\text{-}DE}}(\mathcal{B}, M_0, M_1) - \nu(\varrho).$$

Proof. We use similar ideas as those of [10]. Namely, our proof goes as follows: assume that there is an adversary $\mathcal{B}$ that breaks PRIV1-ID-IND security of the deterministic encryption scheme. Namely, there exist two (t, s)-sources M_0, M_1 and a non-negligible ζ such that

$$\mathbf{Adv}^{\text{priv1-ind-id}}_{\mathsf{HIB\text{-}DE}}(\mathcal{B}, M_0, M_1) = \Pr[\mathsf{Guess}^{\mathcal{B}}_{\mathsf{HIB\text{-}DE}}(M_0) = 1] - \Pr[\mathsf{Guess}^{\mathcal{B}}_{\mathsf{HIB\text{-}DE}}(M_1) = 1] \geq \zeta.$$

Then, we build an adversary $\mathcal{A}$ that breaks the security of the underlying HF, that is, $\mathcal{A}$ will interact with a challenger according to the lossy or the real experiment and will tell the difference between both scenarios with non-negligible probability.

$\mathcal{A}$ forwards an identity $\mathsf{id}^\dagger$ to its challenger, which is some random identity in the adaptive case. When the challenger runs the setup and gives the output to $\mathcal{A}$, $\mathcal{A}$ forwards this information to $\mathcal{B}$. When $\mathcal{B}$ asks for a secret key for a hierarchical identity id, $\mathcal{A}$ forwards the query to the experiment and forwards the reply to $\mathcal{B}$. At some point, $\mathcal{B}$ outputs $\mathsf{id}^\star$. Adversary $\mathcal{A}$ then forwards $\mathsf{id}^\star$ to its challenger, chooses $\gamma \leftarrow \{0,1\}$ at random and encrypts $m_\gamma \xleftarrow{R} M_\gamma$ under the identity $\mathsf{id}^\star$. Note that this corresponds to an execution of $\mathsf{Guess}^{\mathcal{B}}_{\mathsf{HIB\text{-}DE}}(M_\gamma)$. After some more secret key queries, $\mathcal{B}$ outputs a bit $\gamma' \in \{0,1\}$ and $\mathcal{A}$ outputs $d_{\mathcal{A}} = 1$ if $\gamma = \gamma'$ and $d_{\mathcal{A}} = 0$ otherwise.

The security analysis is very similar to the one in Section 2.2. A simple argument shows that

$$\Pr[\gamma' = \gamma |\ \mathsf{REAL}] - \frac{1}{2} = \Pr[d_{\mathcal{A}} = 1 |\ \mathsf{REAL}] - \frac{1}{2} \geq \zeta/2,$$

since $\mathcal{A}$ perfectly simulated the experiment $\mathsf{Guess}^{\mathcal{B}}_{\mathsf{HIB\text{-}DE}}(M_\gamma)$ with $\mathcal{B}$. On the other hand, in the LOSSY setting when $\mathsf{id}^\star$ is lossy, the advantage of $\mathcal{B}$ in guessing γ is negligible because the universality property means that the output distribution of the encryption algorithm is independent of the input distribution, and this holds regardless of whether $d^{\mathcal{A}}_{\neg abort} = 1$, therefore:

$$\Pr[d_{\mathcal{A}} = 1 |\ \mathsf{LOSSY} \wedge d^{\mathcal{A}}_{\neg abort} = 1] \leq \frac{1}{2} + \nu,$$

for some negligible ν. At this point the analysis follows as in the analysis of the IND-ID-CPA case (proof of Theorem 1).

Related Work

In a recent, independent work [37], Xie, Xue and Zhang gave a lattice-based deterministic IBE construction in the auxiliary input setting [14]. While their scheme does provide adaptive security in the auxiliary input setting, it was described as a single-level IBE. Our results are incomparable to theirs: on one hand, we do not consider auxiliary inputs; on the other hand, we are concerned with pairing-based schemes in a hierarchical setting and focus on a more powerful primitive.

D On Hedged (H)IBE

Bellare *et al.* [6] studied the problem of designing *hedged* public key encryption schemes, which remain secure even if the randomness used for encryption is relatively bad. They consider two notions of hedged security: non-adaptive and adaptive. A non-adaptive attacker can make only a single challenge encryption query, whereas an adaptive attacker can make up to q challenge adaptive encryption queries, all of them for the same public key.

Bellare *et al.* give in [6] different constructions of public key encryption scheme enjoying non-adaptive hedged security. For instance, their construction RtD combines an IND-CPA secure encryption scheme and a deterministic PRIV1-IND secure encryption scheme to achieve non-adaptive hedged security. Then they prove a general result stating that a public key encryption scheme which enjoys both anonymity and non-adaptive hedged security, already enjoys adaptive hedged security. They show how to provide the necessary anonymity property by using universal lossy trapdoor functions.

The notion of non-adaptive hedged security, with only one challenge encryption query, can be easily extended to the identity-based setting, as well as the non-adaptively secure constructions in [6]. In particular, since our new notion of partial lossiness for (H)IB-TDFs implies both IND-CPA secure IBE and PRIV1-IND secure (H)IB deterministic encryption, we have that the existence of (H)IB-TDFs enjoying partial lossiness implies (through the identity-based version of RtD) non-adaptive hedged (H)IBE secure against adversaries who choose the challenge identity in an adaptive way.

Regarding the notion of adaptive hedged security, the extension to the identity-based setting is more challenging, because the adversary could choose different identities for each of the q challenge adaptive encryption queries. The partial lossiness definitions in [9] and in this paper do not seem to imply any positive result in that case, because these definitions consider only one challenge (lossy) identity. Maybe more general definitions for the partial lossiness notion of (H)IB-TDFs are needed to overcome this obstacle, in the line of all-but-N trapdoor functions [25] and all-but-many trapdoor functions [27]. We leave the problem of designing (H)IBE schemes with adaptive hedged security as an interesting open problem.