On the Power of Hierarchical Identity-Based Encryption

Mohammad Mahmoody* Ameer Mohammed†

August 16, 2015

Abstract

We prove that there is no fully black-box construction of collision-resistant hash functions (CRH) from hierarchical identity-based encryption (HIBE) with arbitrary polynomial number of identity levels. As a corollary we obtain a series of separations showing that none of the primitives implied by HIBE in a black-box way (e.g., IBE, CCA-secure public-key encryption) can be used in a black-box way to construct fully homomorphic encryption or any other primitive that is known to imply CRH in a black-box way. To the best of our knowledge, this is the first limitation proved for the power of HIBE.

Our proof relies on the reconstruction paradigm of Gennaro and Trevisan (FOCS 2000) and Haitner et al. (FOCS 2007) and extends their techniques for one-way and trapdoor permutations to the setting of HIBE. A technical challenge for our separation of HIBE stems from the *adaptivity* of the adversary who is allowed to obtain keys for different identities *before* she selects the attacked identity. Our main technical contribution is to show how to achieve compression/reconstruction in the presence of such adaptive adversaries.

Keywords: Hierarchical Identity-based Encryption, Collision Resistant Hashing, Homomorphic Encryption, Black-Box Separations.

*University of Virginia, mohammad@cs.virginia.edu. Supported by NSF CAREER award CCF-1350939.
†University of Virginia, am8zv@virginia.edu. Supported by University of Kuwait.

Contents

1 Introduction

Modern cryptography is based on well-defined hardness assumptions and formal proofs of security. For example, a sequence of fundamental work [Yao82, GM84, Rom90, HILL99, GGM86, LR88, IL89, NY89a, Nao91] led to constructions of private key encryption, pseudorandom generators, pseudorandom functions and permutations, bit commitment, and digital signatures solely based on the assumption that one-way function exists. On the other hand, cryptographic primitives such as public key encryption, oblivious transfer, and key agreement that are perhaps more "structured" are not known to be implied by one-way functions alone. The goal of founding cryptography on *minimal* assumptions has let to an extensive study of the power and limitations of cryptographic primitives. As a result, for every (newly introduced) primitive $\mathcal{P}$, researchers aim to answer two questions: (1) What are the minimal computational assumptions necessary for constructing $\mathcal{P}$? (2) What are the power and limitations of $\mathcal{P}$ as a computational assumption? In particular, what other basic cryptographic primitives could be constructed from $\mathcal{P}$?

Hierarchical identity-based encryption. In this work, we study the limitations of the power of *identity based encryption* as a strong form of encryption and its *hierarchical* variant. A traditional public-key encryption scheme allows Alice to send messages to Bob privately over a public channel knowing only Bob's public key. An identity-based encryption scheme [Sha84, BF03] does not require Alice to know a specific individual's public-key and allows Alice to encrypt messages for Bob by *only* knowing Bob's identity and a single master public key that is the same for all identities. Bob's decryption key can be generated using the (single) master secret key and Bob's (public) identity. The notion of IBE was first proposed by Shamir [Sha84], but the first fully functional IBE was first constructed by Boneh and Franklin [BF03] based on assumptions on bilinear maps.

A *hierarchical* identity based encryption scheme (see Definition 2.6) takes the versatility of IBE to the next level: each identity's decryption key can be considered as a master secret key on its own to generate decryption keys for "sub-identities". So as the name suggests it allows the delegation of encrypting power in a hierarchical structure of identities. HIBE was first defined and constructed in [GS02, HL02] where the security was based on the hardness of the Bilinear Diffie-Hellman problem in the random oracle model. Later, Boneh and Boyen [BB04] proposed a more efficient HIBE scheme in the standard (plain) model but only achieved selective-ID security. This construction was further improved in [BBG05], and in [ABB10] Agrawal et al. showed how to construct fully-secure efficient HIBE based on the learning with errors (LWE) assumption [Reg05].

Other than for their "prescribed" way of using them, IBE and specially HIBE are very powerful computational primitives for achieving other cryptographic tasks. For example it was shown by [BCHK06] that IBE can be used to obtain CCA secure public-key encryption in a black-box way, and Gentry et al. [GHRW14] showed a perhaps surprising application of IBE to garbling RAM programs. Canetti, Halevi, and Katz [CHK03] showed how to achieve forward-secure encryption scheme from IBE and HIBE. More recently, Naor and Ziv [NZ15] used HIBE in their construction of Primary-Secondary-Resolver Membership Proof Systems. In this work, we study the following question about HIBE as a cryptographic primitive/assumption:

> *What are the limitations of the power of hierarchical identity based encryption? In other words, what crypto primitives can or cannot be constructed from IBE/HIBE?*

The Black-Box Framework. We study our main question in the black-box framework of [IR89]. Impagliazzo and Rudich [IR89] were the first to develop such framework which enabled us to rule out the existence of an important and powerful class of reductions between primitives. The work of Reingold, Trevisan, and Vadhan [RTV04] formalized this framework further and established a taxonomy for the field. Many cryptographic constructions are black-box in the sense that (**1**) the algorithms implementing the construction of $\mathcal{Q}$ use another cryptographic primitive $\mathcal{P}$ (e.g., one-way functions) only as an oracle, and **(2)** the *security reduction* takes any adversary **Adv** who breaks $\mathcal{Q}^{\mathcal{P}}$ also as an oracle to reduce it to an attack against $\mathcal{P}$. Black-box constructions are also considered important due to their (typical) efficiency advantage over their non-black-box counterparts. Following the work of [IR89] a sequence of results known as "black-box separations" emerged in which limitation of the power of cryptographic primitives are proved with respect to black-box constructions/reductions. In this work we study the power of *fully* black-box reductions as defined in [RTV04] which is the most common form of black-box constructions used in cryptography.

1.1 Our Results

In this work we prove a black box separation result for hierarchical IBE which, to the best of our knowledge, is the first such result. Namely, we show that there is no fully black box construction of collision-resistant hash functions (CRH) from HIBE schemes.

Theorem 1.1 (Main Theorem)**.** *There is no black-box construction of collision-resistant hash functions from hierarchical identity-based encryption with an arbitrary polynomial number of levels.*

Separating homomorphic encryption from HIBE. A primary corollary of our main theorem above is that HIBE does not imply fully homomorphic encryption (FHE) in a black-box way. That is because by the result of Ishai, Kushilevitz, and Ostrovsky [IKO05] FHE implies CRHF in a black-box way. Theorem 1.1 together with the result of [BCHK06] implies that CCA secure public-key encryption does not imply CRH or FHE in a black-box way. Since CCA-secure public key encryption can be constructed from trapdoor permutations one might think that this separation follows form the work of Haitner et al. [HHRS07] who ruled out black-box constructions of CRH from trapdoor-permutations. However the construction of CCA secure encryption from TDP is *non-black-box* [NY90, Sah99, Lin06] due to its use of noninteractive zero knowledge proofs.[1]

The work of [IKO05] provides several other primitives (than FHE) whose existence also implies CRH in a black-box way. These primitives include: one-round private information retrieval (PIR) protocols as well as homomorphic one-way commitments. As a direct corollary, our main theorem above extends to all these primitives as well. Our separations also holds when the goal of the construction is to achieve statistically hiding commitment schemes with $o(n/\log n)$ round complexity (where n is the security parameter).[2] However, for simplicity of the presentation here we focus on the simpler primitive of CRH.

[1] We shall also note that even the techniques behind the proof of [HHRS07] do not seem to extend to a separation of CCA secure encryption from TDPs, even though CCA secure encryption could be constructed from TDP and random oracles [BR93]. The reason is that the "collision finding oracle" (see Definition 2.7) of [Sim98, HHRS07] prevents the random TDP oracle from being independent of the other subroutines of the oracle to be used as a random oracle.

[2] Statistically hiding commitment is known to be implied by CRH [DPP97, NY89b] and so proving separation for statistically hiding commitment is stronger than a similar result for CRH.

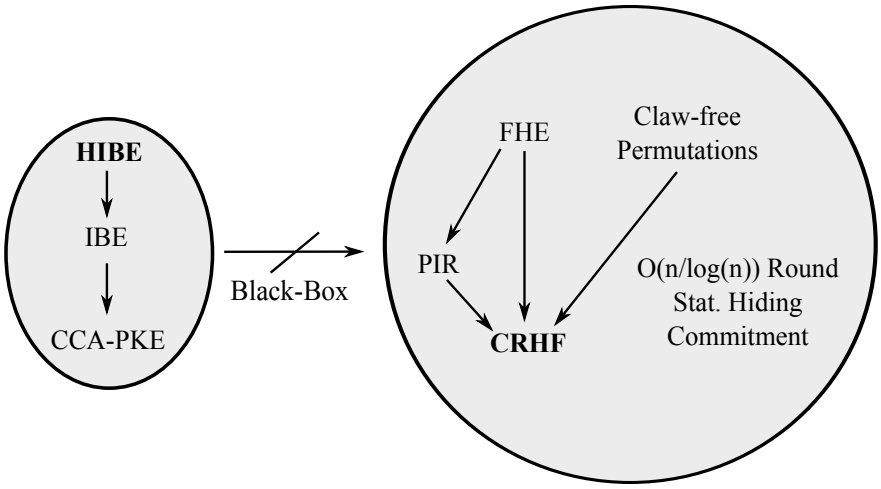

Figure 1: Our works shows that primitives on the left side do not imply any of the primitives on the right side in a black-box way.

Previous separations regarding IBE. In this work we present the first black-box separation for *hierarchical* IBE. However previous separations about IBE are known. The work of Boneh et al. [BPR+08] proved that there is no black-box construction of IBE from trapdoor permutations. Furthermore, Goyal et al. [GKLM12] studied the *limitations* of IBE and proved the first separation for IBE and separated it from fuzzy IBE schemes [SW05]. Our techniques are quite different from those of [GKLM12]. At a high level, [GKLM12] uses a random IBE oracle and aims at breaking any fuzzy IBE construction relative to this oracle with only a polynomial number of queries to this oracle, while our approach uses a random HIBE oracle together with the collision-finding oracle of [Sim98, HHRS07] (see Section 2.4). So the challenging part of our proof is to show that the random HIBE oracle remains secure in the presence of the collision-finding oracle rather than finding a poly-query attack to break CRH relative to this oracle.

Comparison with [AS15]. A corollary of our Theorem 1.1 for the case of IBE could be also derived [Vai] from the concurrent beautiful work of Asharov and Segev [AS15]. In particular Asharov and Segev [AS15] showed that there is no black-box construction of CRH from indistinguishability obfuscation (iO) and one-way functions, even if the iO primitive could be applied to OWF in a non-black-box way. Technically, they prove this result using an oracle O with OWF and iO subroutines while the iO sub-oracle could accept circuits with OWF gates in them. On the other hand Waters [Wat14] showed how to construct general (adaptively secure) functional encryption [BSW11] from iO and OWFs. Note that IBE (and not HIBE) is a special case of functional encryption. The construction of [Wat14] also uses OWF in a non-black-box way but only for applying the iO to circuits with OWF gates, therefore it can be implemented using the separating oracle of [AS15]. In other words, [AS15] shows how to construct an oracle relative to which the functional encryption (and thus IBE) construction of [Wat14] exists, but relative to the same oracle no CRH is secure.

As we will describe below, our proof is quite different from that of [AS15] and our compression techniques do not have a counterpart in [AS15]. While the result of [AS15] handles richer class of functional encryption schemes beyond IBE, our result extends to arbitrary levels of hierarchies.

1.2 Technical Overview

In this subsection we give an overview of our techniques in the proof of our main theorem. Here we focus on the case of IBE (i.e. HIBE with one hierarchy) since some of the main challenges already show up for the (adaptively secure) IBE, and after resolving them we can scale the proof up easily with the number of hierarchy levels.

We first need to recall tools and techniques from Haitner et al. [HHRS07] and [GT00] which we use as building blocks in our proof. Our starting point for the proof of Theorem 1.1 is the result of Haitner et al. [HHRS07] that separates CRH from trapdoor permutations. In their separation, they employed an oracle $O = (T, \mathrm{Sam}^T)$ where T is a random trapdoor permutation and Sam^T is (a variant of) the collision finding oracle of Simon [Sim98] (see Definition 2.7) which returns a collision (x, x') for any given input circuit C with T gates. It is easy to see that relative to O there is no secure construction of CRH that only uses T gates. To derive the separation it is enough to show that relative to O, T remains a secure TDP (see Lemma 2.8 for a proof). The main technical argument in [HHRS07] was to show that a random trapdoor permutation T remains secure in the presence of Sam^T. We will sketch some components of this proof first, and then we will discuss how to use these tools in addition to our new compression lemmas to prove our main separation.

Hardness of random permutations in the presense of Sam**.** The seminal work of [GGKT05] showed that a random permutation $\pi\colon \{0,1\}^n \mapsto \{0,1\}^n$ is hard to invert by introducing the compresson/reconstruction technique. For any supposed adversary A^π who inverts an $1/\operatorname{poly}(n)$ fraction of $\{0,1\}^n$, [GGKT05] showed how to represent the permutation π with $\Omega(n)$ bits *fewer* than it is takes to represent a random permutation. That would be sufficient to show that the probability that π is "easy" for such A is at most exponentially small $2^{-\Omega(n)}$.[3]

The work of [HHRS07] extended the result of [GGKT05] by showing that the hardness of random permutations π holds, even if we allow the adversary access the collision finding oracle Sam^π that takes as input any circuit C with π gates (that shrinks its input) and returns a collision for it (chosen from a specific distribution). This immediately implies a (fully) black-box separation for CRH from TDP since for every black-box construction $\mathcal{H}$ it gives an oracle (π, Sam^π) relative to which secure TDP exists (i.e. π) but the implementation $\mathcal{H}$ of CRH using π is insecure. This is indeed sufficient to derive the fully black-box separation [RTV04, GKM+00].

Extension to TDPs. [HHRS07] extended their result to trapdoor permutations using a similar argument to that of [GGK03] who also proved the hardness of random trapdoor permutation as follows. Let $T = (G, F, F^{-1})$ be a random trapdoor permutation in which G is a random permutation over $\{0,1\}^n$, $F[pk](\cdot)$ is a random permutation for every $pk \in \{0,1\}^n$ and $F^{-1}[sk](\cdot)$ be the inverse of $F[pk](\cdot)$ whenever $G(sk) = pk$. Let $A^{(T,\mathrm{Sam}^T)}$ be an adversary who inverts a random y under a random public key pk with probability $1/\operatorname{poly}(n)$. Then A is implicitly doing either of the following. **Case 1**: A finds $sk = G^{-1}(pk)$ for the given random pk. In this case A has inverted the random permutation G. **Case 2**: A succeeds at finding $x = F^{-1}[sk](y)$ *without* asking $G(sk) = pk$. Since A does not ask $G(sk) = pk$ the permutation $F[pk](\cdot)$ would be a random permutation inverted by A. In both cases the existence of A leads to an efficient-query algorithm inverting random permutations in the presence of Sam. However, as we saw this is impossible.

[3] In fact [GT00] achieves exponential compression and *doubly* exponentially small probability of success for a fixed A. This lets them do a union bound over all poly-sized circuits A when π is chosen at random.

1.2.1 Beyond TDP: The Case of IBE and HIBE

First, let us recall IBE informally (see Definition 2.6). An *identity-based encryption* (IBE) scheme for security parameter n and messages $\{0,1\}^n$ is defined using four PPTs (Setup, KeyGen, Enc, Dec):

- Setup(MSK) takes random master secret key and generates master public key MPK.
- KeyGen[MSK](id) generates a trapdoor td for a given identity id.
- Enc[MPK, id](x) encrypts x under identity id and outputs ciphertext y.
- Dec[MPK, id, td](y) decrypts y using the trapdoor td and gets back x.

Security variants. The basic CPA security of an IBE requires that no adversary can distinguish between the encryptions of any two messages of his choice even if he is allowed to *choose* the challenge identity id^* of the encryption after getting trapdoors for identities $id \neq id^*$ of his choice. Here we first focus on the basic CPA security for IBE, but our full proof handles the stronger notion of CCA secure IBE/HIBE. For simplicity in this exposition we assume that the IBE's adversary aims at decrypting a randomly selected message x (encrypted under challenge identity id^*).[4]

As in the case of TDPs, here our goal is to design an oracle $O = (U, \text{Sam}^U)$ such that U implements IBE/HIBE in way that it remains secure even in the presence of the Sam^U oracle. We first start by a direct generalization of the above arguments to oracles with more than one level of trapdoor, and then will see what aspects of the security of IBE will require us to change our oracle and the proof of hardness accordingly. Our first try is to use hierarchical random trapdoor permutations to implement, but to prove the full fledged adaptive security of IBE/HIBE we will change this oracle to use injective random functions.

Hierarchical random permutations. We first use a direct adaptation from random TDPs to implement our IBE oracle. Let $U = (S, G, F, F^{-1})$ be an oracle that we call a "random IBE oracle" defined and used as follows. S implements the setup and is a random permutation over $\{0,1\}^n$ that maps master secret key MSK to master public key MPK. $G[\text{MSK}](\cdot)$ implements the trapdoor generation and is a random permutation over $\{0,1\}^n$ that maps identities to their trapdoor. Finally $F[\text{MPK}, id](\cdot)$ and $F^{-1}[\text{MPK}, td])$ are random permutations over $\{0,1\}^n$ that are used for encryption and decryption.

Our goal here would be to show that any adversary A who breaks the IBE implemented by U by accessing $O = (U, \text{Sam}^U)$ will be forced to invert some random permutation (to derive the contradiction). Let us list the possible cases through which an adversary can win the security game:

- **Case 1**: A finds $\text{MSK} = S^{-1}(\text{MPK})$.
- **Case 2**: A does not find MSK but finds the trapdoor td^* for his identity of choice id^*.
- **Case 3**: A does not find the trapdoor for id^* but still inverts the challenge ciphertext y.

Problem with Case 2: adaptivity of adversary. Similarly to the case of random TDP, **Case 1** and **Case 3** will imply that A is indeed inverting a random permutation, which can only happen with negligible probability. However, **Case 2** does not imply so for two reasons:

1. While inverting id^* to find its trapdoor, the adversary A is allowed to obtain trapdoors $td \neq td^*$ for other identities $id \neq id^*$.
2. The adversary gets to *choose* id^* as opposed to being given a random one.

In the following we will show how to resolve the two issues above. First we will ignore the second issue above by working with a weaker security definition for IBE in which the adversary

[4] By the Goldreich Levin lemma [GL89] these two notions of security are equivalent in a black-box way.

does not choose id^* but rather is given a random one (but still gets to ask the trapdoor for other identities). And then we will describe our new oracle to handle both issues above.

Attacks against *random* challenge identity. Note that we are focusing on the scenario in which the adversary A^{U,Sam^U} breaks the IBE by causing Case 2 to occur with non-negligible probability. We are also assuming, for now, that the adversary does *not* choose an identity of his choice, bur rather finds the trapdoor of a *random* identity. Our goal is to reduce this case to when (a variant of) A is essentially inverting a random permutation using Sam oracle. Think of the permutation $G^{-1}[\mathrm{MPK}](\cdot)$ as the inverse of $G[\mathrm{MSK}]$. It can be verified that when A succeeds in Case 2, it is in fact inverting G^{-1} on a random point while he is able to ask some "inversion queries" to G^{-1} before "inverting" the random challenge id^* on its own.

To rule out successful attackers against random challenge identity we generalize the results of [GGKT05] for hardness of random permutations π to the setting in which the adversary is allowed to ask inversion queries to $\pi^{-1}(\cdot)$ in addition to direct queries to $\pi(\cdot)$. Namely, we show that a random permutation, is *adaptively* one-way [PPV08] even in the presence of the Sam oracle.

Lemma 1.2 (Informal: Adaptive/CCA hardness of random permutations in the presence of Sam)**.** *For any permutation π over $\{0,1\}^n$, define $O = (\pi, \pi^{-1})$ to be an oracle giving access to π in both directions. Let A be a* poly(n)*-query circuit that accepts a challenge $y^* \in \{0,1\}^n$ and whose goal is to find $x = \pi^{-1}(y^*)$ using O while all queries to O are allowed* except *for $\pi^{-1}(y^*)$. Then, with overwhelming probability over the choice of random permutation π it holds that: the probability of success for A is negligible, even if it is allowed to call the* Sam^O *oracle.*

We show that the argument of [GT00] for the case of "basic" (i.e., non-CCA) attackers does indeed extend to the CCA setting described in Lemma 1.2. (Lemma A.1 is a formalization of this lemma without the presence of the Sam oracle.) We also show that the $\mathrm{Sam}^{(\pi,\pi^{-1})}$ oracle will not help the attacker much, using the techniques of [HHRS07] (see Section 3 for a proof of this).

Fully adaptive attacks. The most challenging aspect of our proof handing general IBE attacks stems from the fact that here the adversary is allowed to *choose* the challenge identity. Note that such an adversary does not necessarily invert a "randomly sampled" identity id^* to win according to Case 2, and it has has full control over id^*. Unfortunately, we are not able to prove a variant of Lemma 1.2 in which the adversary chooses y^* itself since as soon as we sample and fix the permutation π, there is always a *trivial* adversary whose description depends on π and "knows" a pair $\pi(x) = y$ (through non-uniformity) and that proposes y as the challenge and inverts it!

Compression Amplification. The way we can rule out such attacks in the context of IBE/HIBE (for Case 2) is by relying on the fact that the adversary needs to succeed for a randomly given master public-key out of his choice. The idea is that even though one cannot prove a strong hardness of inverting π when the adversary chooses the inversion point y, we can still compress the description of π by $\approx \Omega(n)$ bits. Even though the above compression for the oracle achieved in Case 2 as described above is quite small (i.e., $\Omega(n)$ as opposed to the needed $2^{\Omega(n)}$ bits of compression) we show how to *amplify* this compression for our random IBE oracle. The key point is that the adversary A who has $1/\operatorname{poly}(n)$ chance of winning in Case 2 is still winning in Case 2 for $2^n/\operatorname{poly}(n)$ number of the master public keys given to him. As a result, we achieve $\Omega(n) \cdot 2^n/\operatorname{poly}(n)$ bits of

compression on the *total* representation of the random IBE oracle, which is sufficient to derive its hardness against poly-size (oracle-aided) circuits.

Using random injective functions for trapdoor generation. A seemingly minor technical obstacle against our compression amplification argument sketched above is that we need to represent the key (MPK or MSK) for each of the sub-oracles $G[\text{MSK}](\cdot)$ for which we achieve $\Omega(n)$ bits of compression, so that we can reconstruct the full oracle (S, G, F, F^{-1}). Unfortunately, if we do so, we will lose all the (little) compression that we could achieve for representation of $G[\text{MSK}](\cdot)$ (for many MSKs). To resolve this issue, we use random *injective* functions with large image length to generate the identity trapdoors. This enables us to gain compression for representation of each $G[\text{MSK}](\cdot)$ over which the attacker succeeds in finding matching $G[\text{MSK}](id^*) = td^*$ even if we explicitly represent the corresponding MSK. Lemma A.2 formally states and proves this simple, yet useful building block of our compression argument (when there is no Sam oracle) when we switch to use injective functions for generating identity trapdoors.

Extension to HIBE. Our proof sketched above scales well with the number of identity hierarchies. We will do so by expanding Case 2 of the analysis into many more subcases corresponding to each level. However, the fundamental difference between the first (master key general) and last (encryption/decryption) levels compared to other levels (generating identity trapdoors) remain different and is handled as described above using injective functions with long output.

Comparison to [CLMP13]. At an abstract level, our compression amplification technique described above allows us to achieve exponential compression for primitives that are of the "family" form (here we are interpreting the MPK as an index over which the (sub) primitive and the attack are launched) and can potentially be applied to more applications. In particular, we conjecture that our technique could give an alternative approach to that of Chung et al [CLMP13] whose goal was to show how to achieve hardness against non-uniform attackers (circuits) when the primitive is of the "family" form. [CLMP13] achieved this by employing an information theoretic lemma by Unruh [Unr07], while our approach uses the compression technique of [GT00].

2 Preliminaries

For any $n \in \mathbb{N}$, let Π_n be a family of permutations where $\pi \leftarrow \Pi_n$ is a random permutation mapping $\{0,1\}^n$ to $\{0,1\}^n$. Furthermore, let $\mathcal{F}_{n,m}$ be a family of *injective* functions where $f \leftarrow \mathcal{F}_{n,m}$ is random injective function mapping $\{0,1\}^n$ to $\{0,1\}^m$. For a set S by $x \leftarrow S$ we refer to the process of sampling x uniformly at random from S. We use $[i..j]$ for $\{i, \ldots, j\}$ and $[N]$ for $[1..N]$.

2.1 Black-Box Constructions

We use the following definition of black-box constructions due to Reingold et al. [RTV04]. Unless specified otherwise, in this work we use the terms black-box and fully black-box equivalently.

Definition 2.1. [Fully black-box constructions [RTV04]] A *fully black-box* construction of a primitive $\mathcal{Q}$ from a primitive $\mathcal{P}$ consists of two PPT algorithms (Q, S) as follows:

1. Implementation: For any oracle P that implements $\mathcal{P}$, Q^P implements $\mathcal{Q}$.

2. For any oracle P implementing $\mathcal{P}$ and for any oracle adversary A successfully breaking the security of Q^P, it holds that $S^{P,A}$ breaks the security of P.

Even though the notion above was formalized in [RTV04], the original work of Impagliazzo and Rudich were the first to note that black-box constructions "relativize"; namely they hold relative to any oracle. Thus to rule out black-box constructions it is sufficient to rule out relativizing constructions. The following argument has its roots in the work of Gertner et al. [GKM+00] and is a strengthening of this argument. Informally speaking it asserts that it is enough to choose the separating oracle after (and based on) a candidate construction. Another interpretation of this technique is known as the two-oracle technique of Hsiao and Reyzin [HR04]. Here we describe this lemma and prove it for sake of completeness. This lemma is implicitly used in the work of Haitner et al. [HHRS07]. Here we abstract out this lemma and use it in our proof of Section 3.

Lemma 2.2. *For any primitives $\mathcal{P}$ and $\mathcal{Q}$ let $O = (O_1, O_2)$ be a randomized oracle with two subroutines such that:*

1. *Primitive $\mathcal{P}$ could be implemented using (any sample for) the first part O_1.*
2. *Any fixed (computationally unbounded) $\mathrm{poly}(n)$-query adversary B who could call both of O_1 or O_2 could break the implementation of $\mathcal{P}$ relative to O_1 with probability $\mathrm{negl}(n)$ where n is the security parameter. This probability is over the selection of O as well as attacker B.*
3. *For any implementation Q of $\mathcal{Q}$ that only calls O_1 there is a $\mathrm{poly}(n)$-query attacker A who breaks Q^{O_1} with probability $\geq 1 - 1/n^2$ where the probability is over O and the attacker A.*

Then there is no fully black-box construction of $\mathcal{Q}$ from $\mathcal{P}$.

Breaking a primitive could mean different thing for different primitives, but in this paper we deal with $\mathcal{Q}$ being CRH whose attackers have to find collisions with non-negligible probability.

Proof. For sake of contradiction, suppose (Q, S) is a fully black-box construction of $\mathcal{Q}$ from $\mathcal{P}$. Sample $O = (O_1, O_2)$ and use the implementation of $\mathcal{P}$ that exists relative to O_1 to get implementation Q^{O_1} of $\mathcal{Q}$, and let A be the attacker who breaks this scheme (whose existence is guaranteed by the property 3 of O). Since A succeeds with probability $1 - 1/n^2$ and that $\sum_n 1/n^2 = O(1)$ by Borel-Cantelli lemma, with measure one over the choice of O, A succeeds in its attack for an infinite set of security parameters. In this case we call A a good adversary relative to O.

Now, consider $S^{P,A}$ where P is the implementation of $\mathcal{P}$ using O_1 and A is the above adversary. By the definition of fully black-box constructions, for any sampled O such that A is a good adversary relative to O, $S^{P,A}$ will break P^{O_1} also for an infinite sequence of security parameters. Therefore, with measure one over the choice of O, $S^{P,A}$ will break P^{O_1} for an infinite sequence of security parameters. But we will show below that this cannot happen.

Let us "merge" the algorithm A into S and consider $B^O = (S^A)^O$ as a new $\mathrm{poly}(n)$-query attacker who calls O and tries to break P^{O_1} directly. By property 2 of O, this attacker would have $\mathrm{negl}(n)$ chance of doing so. By an averaging argument, for each fixed security parameter, with probability $1 - \mathrm{negl}(n) \geq 1 - 1/n^2$ over the choice of O it holds that B^O breaks P^{O_1} with probability at most $\mathrm{negl}(n) = \alpha(n)$. By another application of Borel-Cantelli lemma, it follows that with measure one over the choice of O it holds that: the number of security parameters for which $B^O = (S^A)^O$ breaks P^{O_1} with probability more than $\alpha(n)$ is finite, which is a contradiction. □

2.2 Collision-resistant Hash Functions

In this work we define collision-resistant hash functions only for those that shrink their input by a factor of two. It is well known that any CRH with only one bit of shrinkage could be turned into one defined below. We use this definition as it simplifies some of the arguments needed for the separation.

Definition 2.3. A collision resistant hash function $H = \{h \mid h\colon \{0,1\}^m \times \{0,1\}^n \mapsto \{0,1\}^{n/2}\}$ for $m = \text{poly}(n)$ is a family of functions such that for any PPT adversary A there is a negligible function $\epsilon(n)$ such that:

$$\Pr_{d \leftarrow \{0,1\}^m}[A(d) = (x_1, x_2) \in \{0,1\}^{2n} \wedge x_1 \neq x_2 \wedge h_d(x_1) = h_d(x_2)] \leq \epsilon(n).$$

2.3 Hierarchical Identity-Based Encryption

Definition 2.4 (Identity vector)**.** For $i \geq 0$, an i-level *identity vector* $\text{ID}_i = \langle id_0, ..., id_i \rangle$ is a tuple of i identities, where $id_j \in \{0,1\}^* \ \forall \ j \in [0, i]$. Furthermore, the corresponding private-key for identity vector ID_i is given as td_{ID_i}. If $i < 0$, we let $\text{ID}_i = \epsilon$, the empty vector.

Definition 2.5 (Prefix vector)**.** We define a *prefix vector* for identity vector $\text{ID}_i = \langle id_0, ..., id_i \rangle$ as any tuple $\langle s_0, ..., s_j \rangle$ such that $j \leq i$ and $s_k = id_k$ for $0 \leq k \leq j$. We denote the set of all prefix vectors of ID_i as $pre(\text{ID}_i)$.

Definition 2.6 (Hierarchical identity-based encryption [HL02])**.** Given security parameter n and $n = \text{poly}(n)$, an l-depth *hierarchical identity-based encryption* (l-HIBE) scheme for messages in $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consists of $l+3$ PPT algorithms $(\text{Setup}, \{\text{KeyGen}_i\}_{i=1}^{l}, \text{Enc}, \text{Dec})$ defined as follows. (For simplicity and without loss of generality we assume that n is the security parameter as well as the length of the master secret and public keys.)

- $\text{Setup}(1^n)$ takes as input security parameter n and outputs a pair $(\text{MSK}, \text{MPK}) \in \{0,1\}^n \times \{0,1\}^n$. We let $\text{ID}_0 = \langle id_0 \rangle = \langle \text{MPK} \rangle$ and $td_{\text{ID}_0} = \langle td_0 \rangle = \text{MSK}$.
- For $i \in [1, l], \text{KeyGen}_i(\text{ID}_{i-1}, td_{\text{ID}_{i-1}}, id_i)$ takes as input the parent identity vector ID_{i-1}, the corresponding private-key $td_{\text{ID}_{i-1}}$ and identity id_i then outputs the corresponding i-level private key vector td_{ID_i}.[5]
- $\text{Enc}(\text{ID}_l, x)$ takes as input the full public identity vector ID_l, and a message $x \in \mathcal{M}$, and outputs ciphertext $y \in \mathcal{C}$.[6]
- $\text{Dec}(\text{ID}_l, td_{\text{ID}_l}, y)$ takes as input the identity vector ID_l, the corresponding private-key vector td_{ID_l}, and ciphertext $y \in \mathcal{C}$, then returns the message $x \in \mathcal{M}$.

Correctness: Given any $(\text{MSK}, \text{MPK}) \leftarrow \text{Setup}(1^n)$, an HIBE scheme must satisfy the following correctness condition for all $x \in \mathcal{M}$ and all $(\text{ID}_l, td_{\text{ID}_l})$ where td_{ID_l} is the corresponding private-key of the identity vector $\text{ID}_l = \langle \text{MPK}, id_1, ..., id_l \rangle$ obtained through iterative call to KeyGen_i it holds that $\text{Dec}(\text{ID}_l, td_{\text{ID}_l}, \text{Enc}(\text{ID}_l, x)) = x$.

[5] Note that we define a key generation algorithm for each level (as opposed to a single algorithm) in order to simplify our HIBE construction using our ideal oracle.

[6] Some of the subsequent definitions of HIBE use a more general definition in which one can encrypt messages under partial identity vectors $\text{ID}_i = \langle id_0, ..., id_i \rangle$ of depth $i < \ell$. Our impossibility result directly extends to this more general setting as well. However, for simplicity of the presentation here we focus on the original definition of [HL02].

Security: An HIBE scheme is said to be CCA secure if for all adaptive PPT adversaries A:

$$\Pr[\mathrm{CCA}_A^{\mathrm{HIBE}}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$$

where $\mathrm{CCA}_A^{\mathrm{HIBE}}$ is shown in Figure 2. In Step 2, A can adaptively ask key generation queries for ID_i to oracle H_{MSK} which returns td_{ID_i}, the private-key associated with this identity vector, by recursively applying $td_{\mathrm{ID}_i} = \mathrm{KeyGen}_i(\mathrm{ID}_{i-1}, td_{\mathrm{ID}_{i-1}}, id_i)$ given that $td_{\mathrm{ID}_0} = \mathrm{MSK}$. Its chosen identity ID_l^A must not be asked as a query to H_{MSK}. Furthermore, A can adaptively ask decryption queries $D_{\mathrm{MSK}}(\mathrm{ID}_l, c)$ to decrypt ciphertext $c \in \mathcal{C}$ with respect to any identity ID_l. In Step 4, A can still issue queries to H_{MSK} but only for identities ID_i that are not in $pre(\mathrm{ID}_l^A)$, and it can still issue queries to D_{MSK} but not for inputs (ID_l^A, c), where c is the challenge ciphertext.

Experiment $\mathrm{CCA}_A^{\mathrm{HIBE}}(1^n)$:
1. $(\mathrm{MSK}, \mathrm{MPK}) \leftarrow \mathrm{Setup}(1^n)$
2. $(x_0, x_1, \mathrm{ID}_l^A) \leftarrow A^{H_{\mathrm{MSK}}(.), D_{\mathrm{MSK}}(.,.)}(\mathrm{MPK})$
3. $c \leftarrow \mathrm{Enc}(\mathrm{ID}_l^A, x_b)$ where $b \xleftarrow{\$} \{0,1\}$
4. $b' \leftarrow A^{H_{\mathrm{MSK}}(.), D_{\mathrm{MSK}}(.,.)}(c)$
5. Output $(b = b')$

Figure 2: The $\mathrm{CCA}_A^{\mathrm{HIBE}}$ Experiment

Note that, for $l = 0$, this reduces to a standard CCA secure public-key encryption system, and for $l = 1$ this reduces to a CCA-secure IBE scheme.

Definitional Variations: The standard CCA security of HIBE as given in the previous definition can be weakened in multiple ways. We present here some variations of the security definition that we might refer to later, noting only the differences from the original definition.

- CPA (resp. CCA1): The adversary's capabilities are limited to chosen plaintext (resp. non-adaptive chosen ciphertext) attacks.
- rID-CCA/rID-CPA: Instead of having the adversary choose ID_l^A, the target identity will be chosen uniformly at random by the challenger and provided to the adversary along with MPK.
- OW-CCA/OW-CPA: Instead of distinguishing between two ciphertexts, the goals of the adversary here is to "invert" the given challenge ciphertext and find the corresponding (randomly selected) message.
  These notions could be combined into notions like OW-rID-CPA, OW-CCA, etc.

Black-box construction of HIBE. The definition of a black-box construction of CRH from HIBE (in its various security forms) could be derived from Definitions 2.1 and 2.6.

2.4 Collision Finding (Sam) Oracle

In this section, we define the collision finding oracle Sam of [Sim98, HHRS07]. Roughly speaking and in its simplest form, Sam is a (possibly inefficient) algorithm that accepts some description of a circuit C and outputs (x, x'), both uniformly distributed, such that $C(x) = C(x')$. This oracle

was originally introduced by Simon [Sim98] and then was extended into the nested Sam oracle of Haitner et al. [HHRS07].[7]

Definition 2.7 (Collision-finding oracle [Sim98, HHRS07])**.** For any arbitrary oracle O, the algorithm $\mathrm{Sam}_r^O(C)$ for C with input length n samples a uniformly random $x \in \{0,1\}^n$ and then (after sampling x) samples another uniformly random point x' *conditioned on* $C(x) = C(x')$.[8] It then returns (x, x'). Note that this oracle is *randomized* but the randomness r_C is independent for each circuit C (and is sampled only once). The randomness of Sam for each query is provided by the randomized function $r(C) = r_C$ that for each circuit query C provides a random point in the inputs of C as well as a random permutation over the input space of C. The first is used to sample x and then the random permutation is enumerated till we get a collision x'.

It is easy to see that using Sam one can efficiently find collisions for any circuit C whose output length m is smaller than its input length n. Specifically, if $n > m$ then one guarantees the existence of some pair (x, x') such that $C(x) = C(x')$ (i.e. a collision), which results in Sam successfully returning such a pair. The following lemma provides a general tool to use Sam for separating primitives from CRH.

Lemma 2.8. *Let $\mathcal{P}$ be any primitive and $\mathcal{Q}$ represent the collision-resistant hash function primitive. Let $U = (O, \mathrm{Sam}^O)$ be a randomized oracle with two subroutines such that:*

1. *Primitive $\mathcal{P}$ could be implemented using O.*
2. *Any fixed (even computationally unbounded) $\mathrm{poly}(n)$-query adversary B who could call both of O and Sam^O could break the implementation of $\mathcal{P}$ relative to O with probability $\mathrm{negl}(n)$ where n is the security parameter.*

Then there is no fully black-box construction of collision-resistant hash functions from $\mathcal{P}$.

Proof. The lemma almost directly follows from Lemma 2.2; we just have to prove the third property needed by Lemma 2.2. In fact, for any implementation Q of CRH that only calls O there is a 1-query attacker A who breaks Q^O with probability $1 - \mathrm{negl}(n)$. All A does is to take d as the index of hash function, turn $h_d(\cdot)$ into a $\mathrm{poly}(n)$-size circuit C with input length n, and call Sam^O over C and outputs the result. It is easy to see that since h is shrinking its input by a factor of two, with $1 - \mathrm{negl}(n)$ probability over the first sampled point x_1, the number of "siblings" of x_1 relative to the function $h_d(\cdot)$ are exponentially large, and therefore the two colliding points (x_1, x_2) returned by Sam^O will be different points with probability $1 - \mathrm{negl}(n)$. □

3 Separating Hierarchical IBE from Collision Resistant Hashing

In this section we will formally prove our main Theorem 1.1. Namely, we will prove that there exists no fully black-box construction of collision-resistant hash functions from l-level hierarchical identity-based encryption for any polynomial $l = \mathrm{poly}(n)$.

Theorem 3.1. *For any security parameter n and an arbitrary polynomial number of levels $\ell = \mathrm{poly}(n)$ there is no fully black-box construction of collision-resistant hash functions from ℓ-level OW-CCA secure hierarchical identity-based encryption scheme.*

[7] In this work we focus on using the simpler collision finding oracle that is *not* interactive. However, all of our separation results hold with respect to the stronger Sam oracle of "low" (i.e. $o(n/\log n)$) as well. We refer the proof for the more general case to the full version of the paper.

[8] Note that the returned "collision" (x, x') is *not* necessarily distributed like a uniformly sampled collision from all possible collisions.

Corollary of Theorem 3.1. The above theorem, together with the result of Boneh et al. [BCHK06] shows the separation of (standard) CCA-secure HIBE from CRH. In particular, we apply Theorem 3.1 for $\ell + 1$ levels of identity with one-way CCA security (in fact one-way CPA also suffices). Then using Goldreich-Levin lemma and bit-by-bit encryption, we can achieve CPA security as a black-box, and the result of [BCHK06] gives us an CCA secure HIBE for ℓ levels of identity relative to the same oracle.

To prove Theorem 3.1, we will state and use the following claim, which shows the existence of a separating oracle for which a secure implementation of a hierarchical IBE exists.

Claim 3.2. *There exists a randomized oracle $U = (O, \mathrm{Sam}^O)$ such that the following holds:*

1. *An implementation P of OW-CCA-secure l-HIBE exists relative to O.*
2. *Any $\mathrm{poly}(n)$-query adversary A with access to U can break P only with negligible probability.*

Proof of Theorem 3.1. Using Claim 3.2, we can assume the existence of oracle $U = (O, \mathrm{Sam}^O)$ for which OW-CCA-secure l-HIBE exists. Thus, since such an oracle exists, an immediate application of Lemma 2.8 yields that there is no black-box construction of collision-resistant hash functions from OW-CCA-secure l-HIBE. □

We now dedicate the rest of this section to proving Claim 3.2. We first start in Section 3.1 by giving a formal description of the first subroutine O of our separating oracle, which represents an (idealized) random hierarchical trapdoor injective function, so that we will later use it to implement HIBE in Section 3.2. The proof proper starts in Section 3.3 where we will use the randomized oracle $U = (O, \mathrm{Sam}^O)$ such that **(1)** a OW-CCA-secure l-level HIBE implementation exists relative to O, and **(2)** the HIBE implementation remains secure against any $\mathrm{poly}(n)$-query computationally unbounded adversary even after adding Sam^O. The first part is proven in Section 3.3.2 while the second part is discussed in Section 3.3.3.

3.1 Description of Oracle O

In this section, we describe the distribution of our oracle O, which will be used to show that with overwhelming probability over the choice of this oracle, ℓ-level OW-CCA HIBE exists relative to it. In the following definition we will use our notation of identity vectors as defined in Definition 2.4 but, for simplicity of presentation, all of our identities will be strings of length n. Our proof can be directly extended to handle the case of unbounded-length identities as well, but for sake of the simplicity of the presentation we focus on the case of bounded-length identities; see Section B for a sketch of the modifications needed for the case of unbounded identity lengths.

Definition 3.3 (Random Hierarchical Trapdoor Injective Functions)**.** For any security parameter integer $n \in \mathbb{N}$ and number of hierarchies ℓ, let $m = 10n \cdot \ell$. Our random hierarchical injective functions oracle O_n consists of $2\ell + 3$ subroutines: $\{h_0^{-1}, g_1, h_1^{-1}, \ldots, g_{\ell+1}, h_{\ell+1}^{-1}\}$ which are distributed as follows. Note that even though the functions $h_i(\cdot)$ are not publicly available as subroutines of O we still use them to describe the subroutines of O more easily.

- The key generation oracle $h_0^{-1}(\cdot) := S(\cdot)$, the encryption oracle $h_{\ell+1}^{-1}[\mathrm{ID}_\ell](\cdot) := F[\mathrm{ID}_\ell](\cdot)$ and the decryption oracle $g_{\ell+1}[\mathrm{ID}_{\ell-1}, td_\ell](\cdot) := F^{-1}[\mathrm{ID}_\ell](\cdot)$ are random permutations over $\{0,1\}^n$.
- For $i \in [1..\ell]$, and index $\mathrm{ID}_{i-1} = (\mathrm{MPK}, id_1, \ldots, id_{i-1}) \in \{0,1\}^{i \cdot n}$, let $h_i[\mathrm{ID}_{i-1}](\cdot)$: $\{0,1\}^n \mapsto \{0,1\}^m$ be a random injective function. We define $h_i^{-1}[\mathrm{ID}_{i-1}](td_i) = id_i$ if $h_i[\mathrm{ID}_{i-1}](id_i) = td_i$ for some $id_i \in \{0,1\}^n$ and we define $h_i^{-1}[\mathrm{ID}_{i-1}](td_i) = \perp$ if no such id_i exists.

- For $i \in [1..\ell]$ and $(\mathrm{ID}_{i-2}, td_{i-1}) \in \{0,1\}^{(i-1)n+m}$ function $g_i[\mathrm{ID}_{i-2}, td_{i-1}](\cdot)\colon \{0,1\}^n \mapsto \{0,1\}^m$ is defined as follow: For given input id_i, if $h_{i-1}^{-1}[\mathrm{ID}_{i-2}](td_{i-1}) = id_{i-1}$ for some $id_{i-1} \neq \perp$, then $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i) = h_i[\mathrm{ID}_{i-1}](id_i)$. If no such id_{i-1} exists, then $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i) = \perp$.

Our actual oracle (for applying Lemma 2.8) will be (O, Sam^O) where O is sampled from the distribution of the oracles of Definition 3.3, and Sam^O is the Sam oracle as defined in Section 2.4 where the input circuits to Sam^O are allowed to have O-gates.

3.2 Implementing ℓ-level HIBE Using Oracle O

Here we show how to use the oracle O of Definition 3.3 to implement ℓ-level HIBE. Then we will turn into proving the security which is the main part of the proof. Although the *security* of the sampled O is intuitive, due to the *fully random* nature of the permutations used in the implementation, since our actual oracle also has Sam^O attached to it, the proof of security becomes nontrivial.

Intuition. We use the subroutine $h_0^{-1}(\cdot)$ to generate the master public key. We use $g_i(\cdot)$ to generate the ID's of the i'th layer, and we use $h_{\ell+1}^{-1}(\cdot)$ to encrypt and $g_{\ell+1}(\cdot)$ to decrypt. Therefore, for sake of clarity of the presentation, we will use the following alternative names for the subroutines of the first and last layers: $h_0^{-1}(\cdot) = S(\cdot), h_{\ell+1}^{-1}[\cdot](\cdot) = F[\cdot](\cdot), g_{\ell+1}[\cdot](\cdot) = F^{-1}[\cdot](\cdot)$. We will also treat the master public key as the identity of level zero, the ciphertexts as identity of the level $\ell+1$, and the plaintexts as the trapdoors of the level $\ell+1$.

Note that we elected to use random trapdoor *injective functions* to represent $h_i^{-1}[.](.)$ in O as opposed to using random trapdoor permutations as one would first naturally assume. This is to prevent the adversary from trivially breaking the scheme using a call to the $h_i^{-1}[.](.)$ subroutines. Specifically, if we used a trapdoor permutation, since the adversary can *choose* the challenge identity, it can first call $h_i^{-1}[.](.)$ for any $i \in [1..l]$ to find an identity for his own (say randomly selected) trapdoor of level i and announce that specific identity as the one he will use in the attack! Therefore, it was crucial that either we remove the subroutines $h_i^{-1}[.](.)$ from O or change the oracle in way that mitigates such an attack. We thus chose to use random injective functions with a sparse range to make it hard for an adversary to discover a valid trapdoor td_i such that $h_i^{-1}[\mathrm{ID}_{i-1}](td_i) \neq \perp$ for any ID_{i-1}.

Construction 3.4 (Implementing ℓ-level HIBE Using Oracle O)**.** For any security parameter n, and oracle O_n sampled according to Definition 3.3, we will implement an l-HIBE scheme as follows. Our message space and the identities of each level are all $\{0,1\}^n$. To get unbounded message length, we will use bit-by-bit encryption after the scheme is turned into an CPA secure scheme. For larger $\mathrm{poly}(n)$ identity lengths, we will change the security parameter n into $\mathrm{poly}(n)$ in the first step of the construction. As described below, for any identity vector ID_i, we will represent its trapdoor td_{ID_i} as $(\mathrm{ID}_{i-1}, td_i)$ for the "correct" td_i. The algorithms for the constructed scheme work as follows:

- $\mathrm{Setup}(1^n)$: Choose $\mathrm{MSK} \in \{0,1\}^n$ uniformly at random then get $\mathrm{MPK} = S(\mathrm{MSK})$. We let $\mathrm{ID}_0 = \langle id_0 \rangle = \langle \mathrm{MPK} \rangle$ and $td_{\mathrm{ID}_0} = td_0 = \mathrm{MSK}$. Output $(\mathrm{MSK}, \mathrm{MPK})$.
- For $i \in [1..l], \mathrm{KeyGen}_i(\mathrm{ID}_{i-1}^*, td_{\mathrm{ID}_{i-1}}, id_i)$ where $\mathrm{ID}_{i-1}^* = \langle id_0^*, ..., id_{i-1}^* \rangle$: Parse $td_{\mathrm{ID}_{i-1}} = (\mathrm{ID}_{i-2}, td_{i-1})$. If $\mathrm{ID}_{i-2}^* = \mathrm{ID}_{i-2}$ then set $td_i = g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i)$ and output $td_{\mathrm{ID}_i} = (\mathrm{ID}_{i-1}, td_i)$. Otherwise, output $\perp$.
- $\mathrm{Enc}(\mathrm{ID}_l, m)$: Output $F(\mathrm{ID}_{l-1}, id_l, m)$.
- $\mathrm{Dec}(\mathrm{ID}_l^*, td_{\mathrm{ID}_l}, c)$: Parse $td_{\mathrm{ID}_l} = (\mathrm{ID}_{l-1}, td_l)$. If $\mathrm{ID}_{l-1}^* = \mathrm{ID}_{l-1}$ then output $F^{-1}(\mathrm{ID}_{l-1}, td_l, c)$. Otherwise, output $\perp$.

3.3 Security of Implemented HIBE Relative to O

We prove that the constructed HIBE of Construction 3.4 (using the oracle O of Definition 3.3) is OW-CCA secure relative to (O, Sam^O). The proof has the following two steps:

1. **Compression and Reconstruction.** Assuming the existence of any (deterministic) adversary A who can break the OW-CCA security of the constructed HIBE (using O of Definition 3.3) with probability $\epsilon \geq 1/\operatorname{poly}(n)$ and $q = \operatorname{poly}(n)$ queries, we show how to (**1**) compress O to represent it using a "fewer" bits than is necessary to represent a general sampled O, and (**2**) show how to reconstruct O. This compression is relative to the fixed adversary A and both compression and reconstruction heavily depend on A. The number of bits saved in the representation of O will directly imply a bound on the number of such oracles that A can successfully attack. This would imply that with overwhelming probability over the choice of O it will not be a good oracle for A's attack. As usual with reconstruction-type arguments, the bound obtained with this argument allows us to even do a union bound over all possible adversaries that are implemented as circuits of polynomial size. Thus, with overwhelming probability no efficient attacker would exist.

2. **Adding** Sam^O**.** The second step of the proof shows that adding Sam^O (and allowing A to call it) does not interfere with the compression and reconstruction. The argument of this step is identical to that of [HHRS07] but we provide our sketch of the proof in later in this section.

3.3.1 Formalizing the Adversary A

Without loss of generality, we assume that A is a *deterministic* adversary who asks q queries (to O and the challenger combined) and wins against a fixed oracle O with probability $\geq \epsilon$:

$$\Pr_{(\mathrm{MPK},y)}[A(\mathrm{MPK}, y) = (id_1^*, \ldots, id_\ell^*, x) \mid F(\mathrm{MPK}, id_1^*, \ldots, id_\ell^*, x) = y] \geq \epsilon$$

where A is participating in the OW-CCA security game, i.e. no vector of identities $(id_1^*, \ldots, id_\ell^*)$ is given to the adversary and he is the one who chooses them, but A is given a random master public key MPK and a random ciphertext y and he wants to invert y in a CCA attack. We can assume A is deterministic since we are working with non-uniform adversaries and we will prove that our oracle is secure against all circuits, and a non-uniform attacker can always fix its randomness to the "best" value.

Notation. Throughout this section, we might occasionally use the simplifying notation in which $\mathrm{MPK} = id_0^*$, $y = id_{\ell+1}^*$, and so the full identity vector of the attack is simply $\mathrm{ID}_{\ell+1}^* = (\mathrm{MPK}, id_1^*, \ldots, id_\ell^*, y)$, but the first and last components of this vector are not chosen by the adversary but are selected at random. In addition we use td_i^* to denote the corresponding trapdoor for id_i^* with respect to the prefix ID_{i-1}^*. So we will also have the selected $\mathrm{MSK} = td_0^*$ and $x = td_{\ell+1}^*$ for x the inverse of y.

Putting A in *canonical* form. We will modify A as follows.

1. Whenever A wants to output x as its final answer, it queries the encryption on x by calling $F[\mathrm{ID}_\ell^*](x)$.

2. Whenever A is about to ask the query $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i)$ (which returns td_i) A will first ask the query $h_{i-1}^{-1}[\mathrm{ID}_{i-3}, id_{i-2}](td_{i-1})$ from O (that returns id_{i-1} corresponding to td_{i-1} for prefix ID_{i-2}). This modification potentially increases the number of queries asked by A by a factor of two which we can safely ignore (and assume that A is in canonical form to start with).

Now we define the following events, which somehow capture the "level" in which the adversary finds a relevant trapdoor. This "trapdoor" could be finding the message x itself (which as described before, is interpreted as the "trapdoor" for its corresponding challenge ciphertext y), or it could be finding the relevant master secret key MSK (which we interpret as the trapdoor of the "identity" MPK), or it could be finding a trapdoor somewhere in between for id_i for $i \in [\ell]$. Note that finding trapdoor for smaller i is a "stronger" attack that lets A find the relevant trapdoors for bigger i and eventually invert y.

Definition 3.5 (Events E_i)**.** For $i \in [0..\ell+1]$ we say that the event E_i has happened during the execution of A in the (OW-CCA) security game, if A calls the query $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*)$ and receives an answer other than $\perp$. We also let E_{-1} be an empty event (and so $\neg E_{-1}$ holds with probability one).

The first canonical modification of A implies that the success probability of A (and the notation we use to denote x as $td_{\ell+1}^*$ and $\mathrm{MPK} = id_0^*$) simply means that what we are assuming about A's success attack is equivalent to saying:

$$\Pr_{\mathrm{ID}_{\ell+1}^*}[E_{\ell+1}] \geq \epsilon.$$

In the following we assume this is the case.

Lemma 3.6. *For events E_i's defined according to Definition 3.5, there exists $i \in [0..\ell+1]$ such that $\Pr_{\mathrm{ID}_{\ell+1}^*}[E_i \wedge \neg E_{i-1}] \geq \epsilon/(\ell+2)$.*

Proof. It holds that:

$$E_{\ell+1} \subseteq \bigcup_{i \in [0..\ell+1]} (E_i \wedge \neg E_{i-1} \wedge \cdots \wedge \neg E_0) \subseteq \bigcup_{i \in [0..\ell+1]} (E_i \wedge \neg E_{i-1}) .$$

Also note that $\Pr[E_{\ell+1}] \geq \epsilon$. Therefore, there should exists an index i for which $\Pr[E_i \wedge \neg E_{i-1}] \geq \epsilon/(\ell+2)$. □

Fixing parameters. In the rest of the proof, we fix i to the value that satisfies Lemma 3.6, and we let $\epsilon' = \epsilon/(\ell+2)$. However, we will *not* always fix the master public-key MPK nor the challenge ciphertext y using an averaging argument. Whether or not we fix either of them will depend on i.

The sub-oracle $h(\cdot) = \{h_0(\cdot), \ldots, h_{\ell+1}(\cdot)\}$. So far we only defined $h^{-1}(\cdot)$ without referring to $h(\cdot)$ (which was not a subroutine provided as part of the oracle O). Here we introduce this subroutine module and allow A to call it in a "restricted" way. To elaborate, note that in the CCA security game, the adversary can call the oracles $H_{\mathrm{MSK}}(.)$ and $D_{\mathrm{MSK}}(.,.)$, which allow him get the trapdoors for any identity as long as it is not a prefix of the challenge identity, and get decryption of any message other than challenge ciphertext y. Both of these queries are special cases of queries that

are the inverse of $h^{-1}(\cdot)$. Namely, for $i \in [1..\ell+1]$ let $h_i(\mathrm{ID}_i) = h_i[\mathrm{ID}_{i-1}](id_i)$ be defined to be equal to td_i whenever $h_i^{-1}[\mathrm{ID}_{i-2}, id_{i-1}](td_i) = h_i^{-1}[\mathrm{ID}_{i-1}](td_i) = id_i$. Then any query of the adversary A to *both* of the oracles $H_{\mathrm{MSK}}(.), D_{\mathrm{MSK}}(.,.)$ is a special case of a query to $h_i(\cdot)$ for some $i \in [1..\ell+1]$. For simplicity, we will even define h_0 even though the adversary is not calling such queries directly (since the MPK $= id_0^*$ is given by the challenger and is fixed). The restriction on adversary's queries to $H_{\mathrm{MSK}}(.), D_{\mathrm{MSK}}(.,.)$ translates into a natural restriction on how he accesses the oracle $h(\cdot)$: none of these queries are allowed to be a prefix of $\mathrm{ID}_{\ell+1}^*$.

3.3.2 Step 1: Compression and reconstruction of O without the presence of Sam^O

We now begin the first step of the proof by showing how we can use a fixed adversary A (with the behaviour and capabilities that were described in 3.3.1) to compress the description of oracle O.

Full representation of O with no compression. To represent a general oracle O fully (while there is no attacker) without redundant information, it suffices to represent only the injective oracles $h_i[\mathrm{ID}_{i-1}](\cdot)$ for all $i \in [0..\ell+1]$ and all $\mathrm{ID}_{i-1} \in \{0,1\}^{in}$. Note that for $i = \{0, \ell+1\}$ these injective functions happen to be a permutation as well! Now any query to $h_i^{-1}[\mathrm{ID}_{i-1}](\cdot)$ can be answered using its corresponding explicitly represented inverse function $h_i[\mathrm{ID}_{i-1}](\cdot)$. To answer the $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i)$ queries, we employ induction on i. Recall that the master public key generation $S(\cdot)$ sub-routine of O is the same as $h_0^{-1}(\cdot)$. Now, for $i \in [1..\ell+1]$ and a given query $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i)$, we can first find the relevant identity of td_{i-1} by looking up the value of $h_{i-1}^{-1}[\mathrm{ID}_{i-2}](td_{i-1})$, whose answer is represented in the description of the permutation $h_{i-1}[\mathrm{ID}_{i-2}](\cdot)$, and get id_{i-1}. This will enable us to find $h_i(\mathrm{ID}_i) = h_i[\mathrm{ID}_{i-1}](id_i)$, which is also the answer to $g_i[\mathrm{ID}_{i-2}, td_{i-1}](id_i)$.

Intuition behind the compression of O relative to A. Here we describe the high level idea behind how to compress O relative to A, using the ideas described in Lemma A.1 and Lemma A.2. At a very high level we will compress O as follows.

1. If $i = l+1$ (i.e. adversary wins by inverting $y = id_{l+1}$ without finding any trapdoor for any of the identities he proposes): In this case, we apply a similar idea to Lemma A.1 and compress O by representing it using three pieces of information: the description of the fixed master public key $id_0^* = \mathrm{MPK}^*$ that maximizes the adversary's success probability, the full description of $h_i^{-1}[\mathrm{ID}_{i-1}](\cdot)$ for all $i \in [0..\ell]$ and all $\mathrm{ID}_{i-1} \in \{0,1\}^{in}$, and the "compressed" description of $h_{l+1}^{-1}[\mathrm{ID}_l^*] = F[\mathrm{ID}_l^*]$ where $\mathrm{ID}_l^* = \langle id_0^*, id_1^*, ..., id_l^* \rangle$ for some adversarially chosen identities $(id_1^*, ..., id_l^*)$.

   The idea behind compressing $h_{l+1}^{-1}[\mathrm{ID}_l^*](\cdot)$ is as follows. Note that the identity vector ID_l^* (and its corresponding trapdoor) determines a single permutation that is described in *different directions* by $h_{l+1}^{-1}[\mathrm{ID}_l^*](\cdot)$, $h_{l+1}[\mathrm{ID}_l^*](\cdot)$, and $g_{l+1}[\mathrm{ID}_{l-1}^*, td_l^*](\cdot)$. The main difference between these three permutations is that $h_{l+1}^{-1}[\mathrm{ID}_l^*](\cdot)$ provides the access from trapdoors to identities, while the other two provide the access in the opposite direction. The algorithm A is "inverting" a random ciphertext id_{l+1} with respect to the identity vector ID_l^*, and it finds its related td_{l+1} with probability ϵ'. Now, if we can show that A's access to the three permutations $h_{l+1}^{-1}[\mathrm{ID}_l^*](\cdot)$, $h_{l+1}[\mathrm{ID}_l^*](\cdot)$, and $g_{l+1}[\mathrm{ID}_{l-1}^*, td_l^*](\cdot)$ does not let him find td_{l+1} "trivially" we can apply the compression algorithm of Lemma A.1. The queries that let A find td_{l+1}

trivially are the two queries $h_{l+1}[\mathrm{ID}_l^*](id_{l+1})$ and $g_{l+1}[\mathrm{ID}_{l-1}^*, td_l^*](id_{l+1})$. However, we already know that none of these queries are asked by A (before he asks $h_{l+1}^{-1}[\mathrm{ID}_l^*](td_{l+1})$). The reason that A is not asking $h_{l+1}[\mathrm{ID}_l^*](\cdot)$ is that $h(\cdot)$ is not a subroutine publicly available as part of oracle O, and is only provided due to the CCA nature of the attack, yet this particular query $h_{l+1}[\mathrm{ID}_l^*](id_{l+1})$ is prohibited to be asked by A since it violates the CCA attack's requirements (asking this query is akin to allowing the adversary to query the oracle with the challenge!). In addition, the reason that A is not asking $g_{l+1}[\mathrm{ID}_{l-1}^*, td_l^*](id_{l+1})$ is that if he does so, he would be asking the query $h_l^{-1}[\mathrm{ID}_{l-1}^*](td_l^*)$ right before that, which means the event E_l is happening (which we already assumed is not happening). Therefore, the behaviour of A lets us apply the compression algorithm of Lemma A.1 to compress the description of $h_{l+1}^{-1}[\mathrm{ID}_l^*](\cdot)$.

2. If $i = 0$ (i.e. adversary wins by finding the master secret key): In this case, we apply a similar idea to Lemma A.1 and compress O by representing it using three pieces of information: the description of the fixed challenge ciphertext $id_{l+1}^* = y^*$ that maximizes the adversary's success probability, the full description of $h_i^{-1}[\mathrm{ID}_{i-1}](\cdot)$ for all $i \in [1..\ell+1]$ and all $\mathrm{ID}_{i-1} \in \{0,1\}^{in}$, and the "compressed" description of h_0^{-1} (which corresponds to S). The same idea that was described for $i = (l+1)$ applies here as well except that we need not concern ourselves with queries that could be used to trivially find td_0.

3. If $i \in [1..\ell]$: This is the part of the proof for the OW-CCA security game that differs from the other two cases significantly. First we will fix y to whatever that maximizes the winning probability of the adversary. Now, the only remaining randomness (over which the adversary wins with some probability $\geq \epsilon'$) is the randomly selected master public key. We call a MPK *good* (for adversary) if the adversary manages to make $(E_i \wedge \neg E_{i-1})$ hold given this MPK (and the fixed challenge ciphertext y).

   We compress O as follows. We represent each "subtree" of the oracle O that correspond to different MPKs differently. For MPKs that are not good, we will give a full representation. However, for each good MPK, we will represent the part of O that corresponds to this MPK in a compressed manner using the basic compression algorithm of Lemma A.2 applied to the *single* injective function $h_i[\mathrm{ID}_{i-1}^*](\cdot)$. We will also represent (ID_{i-1}^*) (for this particular MPK) but the number of bits that we save by compressing $h_i[\mathrm{ID}_{i-1}^*](\cdot)$ is more than $|\mathrm{ID}_{i-1}^*|$ because $m \gg n \cdot \ell$. Finally, since the adversary is succeeding for all good MPK's and there are super-polynomially many of them, our compression algorithm compresses the description of O by a super-polynomial number of bits.

Now we proceed with stating the formal claims that we will focus on from now on for proving that there exists a secure HIBE with respect to O. In particular, we will have two claims: one for handling the two identical cases of $i = 0$ and $i = l+1$ (where an adversary would find the corresponding $\mathrm{MSK}^* = td_0^*$ or $x^* = td_{l+1}^*$), and another for treating the case of $i \in [1..l]$ (where an adversary finds an intermediate trapdoor) as mentioned during the intuition.

Claim 3.7. *Let O be an l-level random hierarchical trapdoor injective function oracle and n be the security parameter. Let A be a q-query circuit that accepts a master public key $\mathrm{MPK}^* \in \{0,1\}^n$, chooses an identity vector $\mathrm{ID}_l^* = \langle \mathrm{MPK}^*, id_1^*, ..., id_l^* \rangle$, then receives a challenge ciphertext $y^* \leftarrow F[\mathrm{ID}_l^*](x^*) = h_{l+1}^{-1}[\mathrm{ID}_l^*](x^*)$ for a random $x^* \in \{0,1\}^n$. Then for $q \leq 2^{n/5}$, $\epsilon' \geq 2^{-n/5}$, and*

$i \in \{0, l+1\}$ *(and large enough n) we have:*

$$\Pr_{O}\left[\Pr_{\substack{\text{MPK}\leftarrow\{0,1\}^n \\ x^*\leftarrow\{0,1\}^n}}\left[\text{ID}_l^* \leftarrow A^O(\text{MPK}) : A^O(\text{MPK}, h_{l+1}^{-1}[\text{ID}_l^*](x^*)) = td_i^*\right] \geq \epsilon'\right] \leq 2^{-2^{n/2}}$$

Therefore, the oracle O can be represented using $\alpha - 2^{n/2}$ bits where α is the number of bits required to represent a random O.

Claim 3.8. *Let O be an l-level random hierarchical trapdoor injective function oracle and n be the security parameter. Let A be a q-query circuit that accepts a master public key $\text{MPK}^* \in \{0,1\}^n$, chooses an identity vector $\text{ID}_l^* = \langle \text{MPK}^*, id_1^*, ..., id_l^* \rangle$, then receives a challenge ciphertext $y^* \leftarrow F[\text{ID}_l^*](x^*) = h_{l+1}^{-1}[\text{ID}_l^*](x^*)$ for a random $x^* \in \{0,1\}^n$. Then for $q \leq 2^{n/5}$, $\epsilon' \geq 2^{-n/5}$, $m = 10nl$, and $i \in [1, l]$ (and large enough n) we have:*

$$\Pr_{O}\left[\Pr_{\substack{\text{MPK}\leftarrow\{0,1\}^n \\ x^*\leftarrow\{0,1\}^n}}\left[\text{ID}_l^* \leftarrow A^O(\text{MPK}) : A^O(\text{MPK}, h_{l+1}^{-1}[\text{ID}_l^*](x^*)) = td_i^*\right] \geq \epsilon'\right] \leq 2^{-2^{3n/5}}$$

Therefore, the oracle O can be represented using $\alpha - 2^{3n/5}$ bits where α is the number of bits required to represent a random O.

Proof of Claim 3.7. We show here the compression of the oracle in case $i = 0$ or $i = (l+1)$ since these two cases are identical in nature. As described before, the compressed representation of O will contain the fixed i, the fixed master public key or challenge ciphertext, as well as full representation of permutations $h_j^{-1}[\text{ID}_{j-1}](\cdot)$ for all $j \in [0..\ell+1]$ and all $\text{ID}_{j-1} \neq \text{ID}_{i-1}^*$. In the following we describe how to represent the description of $h_i^{-1}[\text{ID}_{i-1}^*](\cdot)$ by describing the encoding and decoding algorithms.

Encoder: Fix i and let $c = |i - (l+1)|$. Fix id_c^* such that Lemma 3.6 is satisfied. Note that id_c^* represents the fixed master public key if $i = (l+1)$, and represents the fixed ciphertext challenge when $i = 0$. Let $N = 2^n$ and let $I \subseteq \{0,1\}^n$ be the set of i-level identities $id_i^* \in \{0,1\}^n$ for which A can successfully find their corresponding trapdoor td_i^* (so $|I| \geq \epsilon' N$), and let $Y = \emptyset$. The encoder works as follows:

1. Remove the lexicographically first element $\widetilde{id_i^*}$ from I and put it in Y
2. Run $A^O(id_c^*, \widetilde{id_i^*})$. If A asks query:
   - $h_i^{-1}[\text{ID}_{i-1}^*](td_i^*) = id_i^*$ and $id_i^* \in I$ then remove id_i^* from I
   - $h_i[\text{ID}_{i-1}^*](id_i^*)$ and $id_i^* \in I$ then remove id_i^* from I

   Note that since event E_{i-1} does not happen here, A will not call $g_i[\text{ID}_{i-2}^*, td_{i-1}^*](\cdot)$
3. If $|I| \neq \emptyset$, go back to Step 1. Otherwise go to the next step.
4. Output the following:
   - Description of i

- Description of id_c^*
- The compressed representation of $h_i^{-1}[\mathrm{ID}_{i-1}^*]$ which consists of:
  - Description of $Y \subseteq I$
  - Description of $X = h_i[\mathrm{ID}_{i-1}^*](Y)$
  - Description of permutation $Z = \{(id_i^*, h_i[\mathrm{ID}_{i-1}^*](id_i^*) \mid id_i^* \in \{0,1\}^n \backslash Y\}$
- The full representation of all the other permutations $\mathcal{H} = \{h_j^{-1}[\mathrm{ID}_{j-1}] \mid j \in [0,..,l+1], \mathrm{ID}_{j-1} \neq \mathrm{ID}_{i-1}^*\}$

Decoder: Given A, the descriptions of X, Y, Z, and $\mathcal{H}$, and the description of i and id_c^*, the decoder can reconstruct O as follows:

1. Remove first lexicographically ordered id_i^* from Y and call it $\widetilde{id_i^*}$
2. Run $A^O(id_c^*, \widetilde{id_i^*})$. If A asks query:
   - $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*)$ for any $td_i^* \in \{0,1\}^n$:
     - If $td_i^* \notin X$: value of $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*)$ is given by Z.
     - If $td_i^* \in X$ and $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*) <_{lex} \widetilde{id_i^*}$: value of $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*)$ would have been precomputed before this call.
     - If $td_i^* \in X$ and $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*) = \widetilde{id_i^*}$: A has *hit* $\widetilde{id_i^*}$ and found its corresponding trapdoor, so we set $h_i^{-1}[\mathrm{ID}_{i-1}^*](td_i^*) = \widetilde{id_i^*}$.
   - $h_i[\mathrm{ID}_{i-1}^*](id_i^*) = td_i^*$ for any $id_i^* \in \{0,1\}^n \backslash \widetilde{id_i^*}$
     - If $td_i^* \notin X$: value of $h_i[\mathrm{ID}_{i-1}^*](id_i^*)$ is given by Z.
     - If $td_i^* \in X$ and $id_i^* <_{lex} \widetilde{id_i^*}$: value of $h_i[\mathrm{ID}_{i-1}^*](id_i^*)$ would have been precomputed before this call and can be inferred from the description of $h_i^{-1}[\mathrm{ID}_{i-1}^*]$
   - $h_j^{-1}[\mathrm{ID}_{j-1}](td_j)$ for any $td_j \in \{0,1\}^n$, and either $j \neq i$ or $\mathrm{ID}_{j-1} \neq \mathrm{ID}_{i-1}^*$: the result id_j can be obtained using the given full representation of $h_j^{-1}[\mathrm{ID}_{j-1}]$
   - $h_j[\mathrm{ID}_{j-1}](id_j)$ for any $id_j \in \{0,1\}^n$, and either $j \neq i$ or $\mathrm{ID}_{j-1} \neq \mathrm{ID}_{i-1}^*$: the result td_j can be obtained using the given full representation of $h_j^{-1}[\mathrm{ID}_{j-1}]$
   - $g_j[\mathrm{ID}_{j-2}, td_{j-1}](id_j)$ for any $td_j \in \{0,1\}^n$, and either $j \neq i$ or $\mathrm{ID}_{j-1} \neq \mathrm{ID}_{i-1}^*$: due to the canonical behaviour of A, $h_{j-1}^{-1}[\mathrm{ID}_{j-2}](td_{j-1})$ will be called first to get id_{j-1}. Then we can find the desired trapdoor td_j using $h_j[\mathrm{ID}_{j-1}](id_j)$ whose answer is represented in $h_j^{-1}[\mathrm{ID}_{j-1}]$.
3. If $|Y| = \emptyset$ then stop. Otherwise go to step 1.

Since for each id_i^* that is inserted into Y we remove at most q from I, the size of Y is at least $a := |I|/(q+1) \geq \epsilon' N/(q+1)$. Let $Enc(O)$ represent the size (in bits) of the compressed oracle. The only difference in the description of the permutations between $Enc(O)$ and O is that in the compressed oracle we are saving on the representation of $h_i^{-1}[\mathrm{ID}_{i-1}^*]$. Specifically, while $h_i^{-1}[\mathrm{ID}_{i-1}^*]$ requires $\log N!$ bits to be fully represented in O, we only need $2\log\binom{N}{a} + \log((N-a)!)$ to represent the compressed $h_i^{-1}[\mathrm{ID}_{i-1}^*]$, which consists of X, Y and Z. Furthermore, we need

$n+\log(l+2) = O(n+l)$ to represent id_c^* and i. Thus, the amount of bits we save in our compression is:

$$\log N! - 2\log\binom{N}{a} - \log((N-a)!) - O(n+l)$$

and since $l = \text{poly}(n)$, the overhead we incur due to representing the index i and identity id_c^* in the compressed oracle is relatively insignificant. In particular, the fraction of oracles O on which A can do ϵ'-well is at most:

$$\begin{aligned}
\frac{2^{|Enc(O)|}}{2^{|O|}} &= 2^{2\log\binom{N}{a}+\log((N-a)!)+O(n+l)-\log N!} \\
&= \frac{\binom{N}{a}^2 (N-a)!}{N!} \cdot 2^{O(n+l)} \\
&= \frac{\binom{N}{a}}{a!} \cdot 2^{O(n+l)} \\
&\leq \left(\frac{Ne^2}{a^2}\right)^a \cdot 2^{O(n+l)}
\end{aligned}$$

If we let $q \leq 2^{n/5}$ and $\epsilon' \geq 2^{-n/5}$ we get that $a \geq \frac{2^{-n/5}2^n}{2^{n/5}+1} \geq 2^{3n/5}/2$. So the upper bound reduces to $\left(\frac{(4)2^n e^2}{2^{6n/5}}\right)^a 2^{O(n+l)} = \left(\frac{4e^2}{2^{n/5}}\right)^a 2^{O(n+l)} \leq 2^{-a+O(n+l)} \leq 2^{-2^{3n/5-1}+O(n+l)} \leq 2^{-2^{n/2}}$ for sufficiently large n. □

Proof of Claim 3.8. We show here the compression of the oracle in case $i \in [1..l]$. In the following we describe how to represent the description of the injective function $h_i[\text{ID}_{i-1}^*](\cdot)$ by describing the encoding and decoding algorithms.

Encoder: Fix i and y such that Lemma 3.6 is satisfied. Let $N = 2^n, M = 2^m$ and let $I \subseteq \{0,1\}^n$ be the set of master public keys $\text{MPK} \in \{0,1\}^n$ for which A can successfully find *some* ID_i^* such that id_i^* was obtained by calling $h_i^{-1}[\text{ID}_{i-1}^*](td_i^*)$ without any prior invocation to $h_i[\text{ID}_{i-1}^*](id_i^*)$. Thus, $|I| \geq \epsilon' N$. Initialize the set $Y = \emptyset$. The encoder works as follows:

1. Remove the lexicographically first element $\widetilde{\text{MPK}}$ from I and put it in Y
2. Run $A^O(\widetilde{\text{MPK}}, y)$. If A asks query:
   - $h_i[\text{ID}_{i-1}^*](id_i^*)$ and $id_0^* \in I$ then remove id_0^* from I

   Note that since event E_{i-1} does not happen here, A will not call $g_i[\text{ID}_{i-2}^*, td_{i-1}^*](\cdot)$
3. If $|I| \neq \emptyset$, go back to Step 1. Otherwise go to the next step.
4. Output the following:
   - Description of i and y
   - Description of $Y \subseteq I$
   - For each $\text{MPK} = id_0^* \in Y$:
     - Description of ID_{i-1}^* on which A was successful

- The compressed representation of $h_i[\mathrm{ID}^*_{i-1}]$ which consists of:
  - Description of point id^*_i
  - Description of the injective function $h'_i[\mathrm{ID}^*_{i-1}] : [N-1] \to [M]$ on all points except id^*
  - The query index $k \in [q]$ during which $h_i^{-1}[\mathrm{ID}^*_{i-1}](td^*_i) = id^*_i$ is called
- The full representation of all the other injective functions $\mathcal{H} = \{h_j[\mathrm{ID}_{j-1}] \mid \mathrm{ID}_{j-1} \neq \mathrm{ID}^*_{i-1}\}$

- The full representation of all injective functions for "bad" MPK: $\mathcal{R} = \{h_j[\mathrm{ID}_{j-1}] \mid id_0 \notin Y\}$

Decoder: Given A, the descriptions of $i, y, Y, \mathcal{H}, \mathcal{R}$, and the $|Y|$ compressed representations of $h_i[\mathrm{ID}^*_{i-1}]$ (including ID^*_{i-1} the query index k for each representation), the decoder can reconstruct O as follows:

1. Remove first lexicographically ordered MPK from Y and call it $\widetilde{\mathrm{MPK}}$. Let ID^*_{i-1} be the target identity that specifies which function has been compressed.

2. Reconstruct all the answers of $h_i(\mathrm{ID}^*_{i-1})$ using $h'_i[\mathrm{ID}^*_{i-1}]$ except for $h_i(\mathrm{ID}^*_{i-1})(id^*_i)$ which is yet to be determined

3. Run $A^O(\widetilde{\mathrm{MPK}}, y)$. If A asks query:

   - $h_i^{-1}[\mathrm{ID}^*_{i-1}](td^*_i)$ for any $td^*_i \in \{0,1\}^m$:
     - If this is the k^{th} query then we have found the corresponding id^*_i so set $h_i[\mathrm{ID}^*_{i-1}](id^*_i) = td^*_i$
     - Otherwise answer using $h'_i[\mathrm{ID}^*_{i-1}]$
   - $h_i[\mathrm{ID}^*_{i-1}](id^*_i) = td^*_i$ for any $id^*_i \in \{0,1\}^n$: answer using $h'_i[\mathrm{ID}^*_{i-1}]$
   - $h_j^{-1}[\mathrm{ID}_{j-1}](td_j)$ for any $td_j \in \{0,1\}^m$, and $id_0 \in Y$ and $\mathrm{ID}_{j-1} \neq \mathrm{ID}^*_{i-1}$: answer using the given full representation from $\mathcal{H}$. The same applies for $h_j[\mathrm{ID}_{j-1}]$ queries.
   - $h_j^{-1}[\mathrm{ID}_{j-1}](td_j)$ for any $td_j \in \{0,1\}^m$, and $id_0 \notin Y$: answer using the given full representation from $\mathcal{R}$. The same applies for $h_j[\mathrm{ID}_{j-1}]$ queries.
   - $g_j[\mathrm{ID}_{j-2}, td_{j-1}](id_j)$ for any $td_j \in \{0,1\}^m$: due to the canonical behaviour of A, $h_{j-1}^{-1}[\mathrm{ID}_{j-2}](td_{j-1})$ will be called first to get id_{j-1}. Then we can find the desired trapdoor td_j using $h_j[\mathrm{ID}_{j-1}](id_j)$ whose answer is represented in the description of $h_j^{-1}[\mathrm{ID}_{j-1}]$.

4. If $|Y| = \emptyset$ then stop. Otherwise go to step 1.

Since for each MPK that is inserted into Y we remove at most q from I, the size of Y is at least $a := |I|/(q+1) \geq \epsilon' N/(q+1)$. Let $Enc(O)$ represent the size (in bits) of the compressed oracle. The only difference in the description of the injective functions between $Enc(O)$ and O is that in the compressed oracle, we are saving on the representation of $h_i[\mathrm{ID}^*_{i-1}]$ for the master public keys represented in Y. Specifically, for each MPK $\in Y$, we are compressing a single injective function from requiring $\alpha_{N,M} = \Pi_{i\in[N]}(M-i-1)$ bits to $\alpha_{N-1,M}$ bits whilst incurring an overhead of at

most $n(l+1)+\log(q)$ to represent ID_i^* and the query index k. Thus if $q < 2^{n/5}$ and $m = 10nl$ and $l = \mathrm{poly}(n)$, we have that, for each good MPK, the net savings (in bits) of:

$$\begin{aligned}\log(\alpha_{N,M}) - \log(\alpha_{N-1,M}) - nl - n - \log(q) &= \log(\alpha_{N,M}/\alpha_{N-1,M}) - nl - n - \log(q) \\ &= \log(M - N - 1) - nl - n - \log(q) \\ &\geq \log(M/2) - nl - n - \log(2^{n/5}) \\ &\geq m - 1 - nl - n - (n-1) \\ &= 10nl - nl - 2n \geq 6nl\end{aligned}$$

Since we have $a \geq \epsilon' N/(q+1)$ "good" master public keys and given that $q < 2^{n/5}$ and $\epsilon' > 2^{-n/5}$, the total number of savings we get is at least $a \times 6nl \geq \dfrac{2^{3n/5}}{2}(6nl) \geq 2^{3n/5}$ for sufficiently large n. □

3.3.3 Step 2: Adding the Sam^O oracle

The second step of the proof shows that giving access to the oracle Sam^O to A does not interfere with the compression and reconstruction procedures. The argument of this step is identical to that of [HHRS07]. However, we sketch the steps of this argument for sake of completeness. Our goal here is to show that the same compression level of the oracle O (relative to which the adversary "succeeds" with non-negligible probability) could be obtained even when we add the Sam^O oracle (with an *arbitrary* fixed randomness) and allow A to call it. This would show that, with high probability over the choice of the oracle O and the randomness of the oracle Sam^O the implementation of HIBE using O remains secure.

Looking ahead, the only change would be that this time we need to use the *augmented* query complexity of the attacker instead, and we lose a factor of 3 in the success probability of A. Therefore, the hardness of the constructed HIBE using the sampled oracle O would be almost the same as before (as a parameter of ϵ and the augmented query complexity q). The augmented query complexity of an attacker A is equal to its standard query complexity to the oracle O plus the total number of *indirect* O queries in the form of O gates that are planted in circuits that are queried to from Sam^O by the adversary. The new modified proof goes through the following two steps.

1. First note that the job of the adversary is essentially to "hit" the preimage of the challenge ciphertext y or an identity id_i^*, or the master public key. This event could be either as a result of a direct query to O or as a result of an *indirect* query to O through a circuit C asked to Sam. The exact hitting event that the adversary is looking for depends on which case $E_i \wedge \neg E_{i-1}$ we are focusing on, but let us assume we are dealing with a fixed i and the adversary is able to make the event $E_i \wedge \neg E_{i-1}$ happen with a decent chance by "hitting" the trapdoor td_i^* of id_i^*. An *indirect hitting* of td_i^* would happen iff the adversary sends a circuit C to $\mathrm{Sam}^O(C)$ with O gates in it and returns a collision (x, x') and either of $C(x)$ or $C(x')$ happens to hit td_i^*.

   A crucial argument due to [HHRS07] shows that one can always modify the attacker to ask a few more queries so that it hits its goal td_i^* *directly* (before it happens indirectly) with a probability that is at most a factor of 3 less than the the total probability of hitting it (directly or indirectly). The intuition behind this argument is that the distributions of the

two points x and x' are both uniform over the inputs of C (even though they are distributed in a correlated way). So, if the adversary chooses a random point x'' and evaluates $C(x'')$ before asking C from the Sam^O oracle, it keeps the chance of hitting td_i^* directly at least half of hitting it indirectly![9]

In the second part of the argument below we will safely assume that the event $E_i \wedge \neg E_{i-1} \wedge \neg SamHit$ is happening with non-negligible probability, while $SamHit$ refers to to an the event that td_i^* is being hit first indirectly through a Sam query by the adversary.

2. The second part of the proof shows that if we start with the guarantee that $E_i \wedge \neg E_{i-1} \wedge \neg SamHit$ is happening with a noticeable probability, we can achieve the *same* compression of the oracle O *even if we fix the randomness of Sam*. This argument indeed holds because of the way our compression and decompression algorithms work. Note that at the heart of our compression and decompression algorithms we basically run the adversary over different inputs till it hits a special point. What is crucial in these arguments is that while we have not hit the final point of interest we can still continue the execution of the adversary and hope that the answer to the current queries are already reconstructed. Now if we have the guarantee that no $\text{Sam}^O(C)$ query by adversary is hitting td_i^* indirectly, and if we have already fixed the randomness of the Sam oracle, we can still run the encoding and decoding algorithms with almost no change. Namely, suppose C is a circuit query to Sam. The first thing Sam does is to run C on a random (but not fixed) input x. We have the guarantee that the execution of $C(x)$ does not encounter any query whose answer is not already reconstructed. Moreover, the second point x' is the lexicographically first input to C such that $C(x) = C(x')$ where x' is being chosen from a random permutation (that is also fixed!) over the inputs of C. To find the same x' while doing the reconstruction, all we have to do is to run C over all inputs one by one using the same permutation order (that is now fixed) till we manage to finish an execution $C(x')$ that happens to output the same $C(x)$. This means that we can run the same encoding and decoding algorithms even in the presence of Sam oracle.

Proof of Claim 3.2. Given the implementation of the HIBE scheme using O in Construction 3.4, we prove the first part of the claim, by referring to Claims 3.7 and 3.8. In particular, the combined claims show that for any given adversary of the HIBE scheme whose goal is to invert its challenge ciphertext for an identity vector of its choice, the probability of doing so is negligible in the security parameter when it is trying to invert an identity at level i. Thus, a union bound over all possible $i \in [l]$, where $l = \text{poly}(n)$ still results in negligible probability of success. The second part of the claim (that is, that the HIBE is secure even in the presence of Sam^O) follows by extension from the discussion in Section 3.3.3, and in particular from the techniques of [HHRS07]. □

Acknowledgement. We thank Vinod Vaikuntanathan for very useful discussions.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume

[9] The actual argument is more subtle, but the main idea is the linearity of expectation over different probabilities.

6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer Berlin Heidelberg, 2010. 1

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. Cryptology ePrint Archive, Report 2015/341, 2015. http://eprint.iacr.org/. 3

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and JanL. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 2004. 1

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pages 440–456, Berlin, Heidelberg, 2005. Springer-Verlag. 1

[BCHK06] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, December 2006. 1, 2, 12

[BF03] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. 1

[BPR+08] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *FOCS*, pages 283–292. IEEE Computer Society, 2008. 3

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 2

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011. 3

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003. 1

[CLMP13] Kai-Min Chung, Huijia Lin, Mohammad Mahmoody, and Rafael Pass. On the power of nonuniformity in proofs of security. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 389–400. ACM, 2013. 7

[DPP97] Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, pages 163–194, 1997. Preliminary version in CRYPTO '93. 2

[GGK03] Gennaro, Gertner, and Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2003. 4

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.*, 35(1):217–246, 2005. 4, 6

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. 1

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Garbled ram revisited, part i. Cryptology ePrint Archive, Report 2014/082, 2014. http://eprint.iacr.org/. 1

[GKLM12] Vipul Goyal, Virendra Kumar, Satya Lokam, and Mohammad Mahmoody. On black-box reductions between predicate encryption schemes. In Ronald Cramer, editor, *Theory of Cryptography*, volume 7194 of *Lecture Notes in Computer Science*, pages 440–457. Springer Berlin Heidelberg, 2012. 3

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The Relationship between Public Key Encryption and Oblivious transfer. In *FOCS*, pages 325–335, 2000. 4, 8

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pages 25–32. ACM, 1989. 5

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 1

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '02, pages 548–566, London, UK, UK, 2002. Springer-Verlag. 1

[GT00] Rosario Gennaro and Luca Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic constructions. In *FOCS*, pages 305–313, 2000. 4, 6, 7, 27

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - A tight lower bound on the round complexity of statistically-hiding commitments. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 669–679. IEEE Computer Society, 2007. 2, 3, 4, 6, 8, 10, 11, 14, 22, 23

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 1

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In LarsR. Knudsen, editor, *Advances in Cryptology EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer Berlin Heidelberg, 2002. 1, 9

[HR04] Hsiao and Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO: Proceedings of Crypto*, 2004. 8

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *In Proceedings of the 2nd Theory of Cryptography Conference*, pages 445–456, 2005. 2

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235, 1989. 1

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 44–61. ACM Press, 1989. 2

[Lin06] Yehuda Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *J. Cryptology*, 19(3):359–377, 2006. 2

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988. 1

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. 1

[NY89a] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989. 1

[NY89b] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43. ACM Press, 1989. 2

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *In Proc. of the 22nd STOC*, pages 427–437. ACM Press, 1990. 2

[NZ15] Moni Naor and Asaf Ziv. Primary-secondary-resolver membership proof systems. In *Theory of Cryptography*, pages 199–228. Springer, 2015. 1

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive One-Way Functions and Applications. In *CRYPTO*, pages 57–74, 2008. 6

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2005. 1

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990. 1

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004. 2, 4, 7, 8

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 543–553, 1999. 2

[Sha84] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53, 1984. 1

[Sim98] Daniel R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In *EUROCRYPT*, pages 334–345, 1998. 2, 3, 4, 10, 11

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT*, pages 457–473, 2005. 3

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 205–223. Springer, 2007. 7

[Vai] Vinod Vaikunthanatan. Personal communication. Personal Communication. 3

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014. http://eprint.iacr.org/. 3

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91. IEEE, 1982. 1

A Compression Building Blocks

A.1 Basic Compression Lemmas

Our first compression lemma is almost identical to the celebrated reconstruction lemma of Gennaro and Trevisan [GT00] with a big difference being that in our lemma, the adversary is allowed to call the permutation π in *both* directions except the query $\pi^{-1}(y)$ for the challenge y which is not allowed. Thus, we call this a CCA attack. We will rely on the CCA nature of this lemma when we apply it for the case of HIBE in which the adversary can ask trapdoors for arbitrary identities of his choice before trying to invert the challenge ciphertext. The proof is very similar to that of [GT00]. We go over the proof of [GT00] and show that it holds for the stronger version in which the adversary can access the permutation in both directions.

Lemma A.1 (CCA hardness of random permutations)**.** *For any $\pi \in \Pi_n$, define $O = (\pi, \pi^{-1})$ to be an oracle for which one can request $\pi(x)$ and $\pi^{-1}(y)$ for any $x, y \in \{0,1\}^n$. Let A be a q-query circuit that accepts a challenge $y^* \in \{0,1\}^n$ and whose goal is to find $x = \pi^{-1}(y^*)$ using O while all queries to O are allowed except for $\pi^{-1}(y^*)$. Then for $q \leq 2^{n/5}$, $\epsilon \geq 2^{-n/5}$ (and large enough n) the oracle O can be represented using $\alpha - 2^{n/2}$ bits where α is the number of bits required to represent a random permutation over $\{0,1\}^n$. Therefore:*

$$\Pr_{\pi \leftarrow \Pi_n, O:=(\pi,\pi^{-1})} \left[\Pr_{y \leftarrow \{0,1\}^n} \left[A^O(y) = \pi^{-1}(y) \right] \geq \epsilon \right] \leq 2^{-2^{n/2}}$$

Proof. Let $N = 2^n$. We use a variant of the reconstruction lemma to find an upper bound on the fraction of oracles $O = (\pi, \pi^{-1})$ for which adversary A can ϵ-invert π. For that, we define an encoder and decoder as follows:

Encoder: Let $I \subseteq \{0,1\}^n$ be the set of strings $y \in \{0,1\}^n$ for which A successfully inverts π (so $|I| \geq \epsilon N$), and let $Y = \emptyset$. The encoder works as follows:

1. Remove the lexicographically first element $\widetilde{y}$ from I and put it in Y
2. Run $A^O(\widetilde{y})$. If A asks query:
   - $\pi(x) = y$ and $y \in I$ then remove y from I
   - $\pi^{-1}(y)$ and $y \in I$ then remove y from I
3. If $|I| \neq \emptyset$, go back to step 2. Otherwise go to the next step.
4. Output the following compressed representation of O, which consists of:
   - Description of $Y \subseteq I$
   - Description of $X = \pi^{-1}(Y)$
   - Description of permutation $Z = \{(y, \pi^{-1}(y) \mid y \in \{0,1\}^n \backslash Y\}$

Decoder: Given A, Y, X, and Z, the decoder can reconstruct O as follows:

1. Remove first lexicographically ordered y from Y and call it $\widetilde{y}$
2. Run $A^O(\widetilde{y})$. If A asks query:
   - $\pi(x)$ for any $x \in \{0,1\}^n$:
     - If $x \notin X$: value of $\pi(x)$ is given by Z.
     - If $x \in X$ and $\pi(x) <_{lex} \widetilde{y}$: value of $\pi(x)$ would have been precomputed before this call.
     - If $x \in X$ and $\pi(x) = \widetilde{y}$: A has *hit* $\widetilde{y}$ and found its inverse, so we set $\pi(x) = \widetilde{y}$.
   - $\pi^{-1}(y) = x$ for any $y \in \{0,1\}^n \backslash \widetilde{y}$
     - If $x \notin X$: value of $\pi^{-1}(y)$ is given by Z.
     - If $x \in X$ and $y <_{lex} \widetilde{y}$: value of $\pi^{-1}(x)$ would have been precomputed before this call.
3. If $|Y| = \emptyset$ then stop. Otherwise go to step 1.

To find an upper bound on the success probability of A, we first compute the total number of oracles $O = (\pi, \pi^{-1})$, which is determined by the number of bits needed to represent permutation π. Hence, $|O| = \log(N!)$.

Now, given A and the compressed representation of π, which consists of X, Y, Z, we show how many bits are required to represent the encoded oracle. Assuming A asks q queries, we would remove at most q elements from I for each y we insert into Y. Hence, $a := |Y| \geq \epsilon N/(q+1)$.

So, for each of X and Y, we need $\log\binom{N}{a}$ bits, and for Z we need $\log((N-a)!)$ bits. Thus, $|Enc(O)| = 2\log\binom{N}{a} + \log((N-a)!)$. The fraction of oracles O on which A can do ϵ-well is at most:

$$\begin{aligned}\frac{2^{|Enc(O)|}}{2^{|O|}} &= 2^{2\log\binom{N}{a}+\log((N-a)!)-\log N!} \\ &= \frac{\binom{N}{a}^2(N-a)!}{N!} \\ &= \frac{\binom{N}{a}}{a!} \\ &\leq \left(\frac{Ne^2}{a^2}\right)^a.\end{aligned}$$

If we let $q \leq 2^{n/5}$ and $\epsilon \geq 2^{-n/5}$ we get that $a \geq \dfrac{2^{-n/5}2^n}{2^{n/5}+1} \geq 2^{3n/5}/2$. So the upper bound reduces to $\left(\dfrac{(4)2^n e^2}{2^{6n/5}}\right)^a = \left(\dfrac{4e^2}{2^{n/5}}\right)^a \leq 2^{-a} = 2^{-2^{3n/5-1}} \leq 2^{-2^{n/2}}$ for sufficiently large n. □

Our second compression lemma asserts the simple fact that for a bounded-query attacker it is hard to hit the images of an "expanding" random injective function. Namely, if $f \leftarrow \mathcal{F}_{n,m}$ is a random injective function from $\{0,1\}^n$ to $\{0,1\}^m$, then for large $m \gg n$, a q query attacker has "small" chance of finding an image of f, even if he is allowed to call f in both directions, as long as he does not cheat by asking $f(x)$ first and then announcing $f(x)$ as the image. The small probability of A doing so is described through compressing the description f by at least $m-2n$ bits relative to the adversary A. We emphasize to bound the success provability of the adversary in this game a simpler argument based on "lazy evaluation" of the oracle can be used. However, we need the compression/decmopression procedures to use them inside the proof of our main result.

Lemma A.2 (Hardness of hitting images of injective random functions)**.** *For any $f \in \mathcal{F}_{n,m}$, define $O = (f, f^{-1})$ to be an oracle for which one can request $f(x)$ and $f^{-1}(y)$ for any $x \in \{0,1\}^n, y \in \{0,1\}^m$, and if $f^{-1}(y)$ is undefined then $\perp$ is returned. Let A be a q-query deterministic algorithm that gets access to O and at some point manages to query $f^{-1}(y^*)$ for some y^* to get x^* before asking $f(x^*)$. Then, for such fixed adversary A (and for large enough n) any such oracle O (that A can successfully hit some image y^* without asking $f(x^*) = y^*$) can be uniquely represented by $\alpha - \log(2^m - 2^n + 1) + n + \log q$ many bits where α is the number of bit required to represent an arbitrary function $f \in \mathcal{F}_{n,m}$. As a result for $q < 2^{n/5}$ and $m > 2n$, the function f can be represented by $\alpha - m + 2n$ bits.*

Proof. Let $N = 2^n, M = 2^m$. We first provide a description of the encoding algorithm.

Encoder: Output the following compressed representation of O:

- Description of x^*
- Description of the injective function $f([N]\setminus\{x\})$; namely the injective function $f(\cdot)$ excluding without the point x^* in its domain. This can be represented as an injective function from $[N-1]$ to $[M]$ which we call g.

- The index $j \in [q]$ of the query $f^{-1}(y^*)$ among the all q queries of A.

Before describing the decoding algorithm, we analyze how much the encoder algorithm saved in the representation of the oracle O. The number of injective functions from $[N]$ to $[M]$ is equal to $\alpha_{N,M} = \Pi_{i\in[N]}(M-i-1)$. So by representing g as a function over $[N-1]$ we save $\log(\alpha_{N,M}/\alpha_{N-1,M}) = \log(M-N-1)$ many bits which is at least $\log(M/2) = m-1$ for $m > 2n$. On the other hand we need to represent x^* that takes n bits and also $j \in [q]$ that takes $\log(q) < \log(2^{n/5}) < n/5 < n-1$ bits. So the total number of bits we saved in the representation of O is at least $m-1-n-(n-1) = m-2n$.

Decoder: Given x^*, $g\colon [N-1] \to [M]$, and j, the decoder can reconstruct O as follows:

1. First use x^* and g to "reconstruct" all of the answers of $f(\cdot)$ except for $f(x^*)$. Note that by only looking at the description of $g(\cdot)$ it is not clear what is the point x^* whose image is unknown (since we represent g simply as a function over $[N-1]$ to save bits of representation).

2. Run A^O for j steps till it asks $f^{-1}(y^*)$ as its j'th query. At this point we have found $y^* = f(x^*)$ and will output the full description of $f(\cdot)$. Note that in the process of executing A till its j'th query we would not encounter the query $f(x^*)$ be asked by A and so the execution could go on till the j'th query of A.

□

B Extension to HIBE with Unbounded-length Identities

In this section, we present a brief justification on why the main theorem can be easily extended to work with the more standard notion of HIBE oracles, which can support mapping identities of *unbounded* polynomial length to trapdoors (as opposed to restricting the identity lengths to an a-priori fixed polynomial in the security parameter).

Referring to Definition 3.3 of our oracle O, we note that, for a fixed number of hierarchies l and security parameter n, we have a *single* set of random injective function for each level in the hierarchy. That is, for any $i \in [1,..,l]$, we have 2^{in} functions of the form $h_i[\mathrm{ID}_{i-1}](.)$, where each function maps an i-level identity $id_i \in \{0,1\}^n$ to an i-level trapdoor $td_i \in \{0,1\}^m$ and each identity vector ID_{i-1} is of length in[10]. Recall that we set $m = 10nl$ in order to offset the overhead of representing the identity prefix (which is of length at most nl bits) by having a comparably large savings in the compression of the function.

For the case where we have unbounded-length identities, we have that, for any identity vector $\mathrm{ID}_{i-1} = \langle id_0, ..., id_{i-1}\rangle$ and each $j \in [0,..,i]$, the identity $id_j \in \{0,1\}^{n'_j}$ has length $n'_j \geq n$, which can be an unbounded polynomial in the security parameter. We now need a set of injective functions for each $i \in [1,..l]$ and for *each* identity vector ID_i of length $N_i := \sum_{j=0}^{i} n'_j = poly(n)$. That is, for any $i \in [1,..l]$ we define $h_{i,N}[\mathrm{ID}_{i-1}](.)$ to be a random injective function from $id_i \in \{0,1\}^{n'_i}$ to $td_i \in \{0,1\}^m$ where $m = 10 \cdot N_{i-1}$. As a result, we now have $2^{N_{i-1}} + n'_i$ random injective functions for every i.

[10]For simplicity, we only consider the function h in the oracle. Adapting the function g to work in this extended proof proceeds in a similar fashion.

Notice that the proof remains almost identical when we use this oracle. In particular, the length of the target identity vector ID^*_{i-1} that needs to be represented in the compressed oracle is now of length N_{i-1}, but since $m = 10N_{i-1}$, the amount of savings in the compression more than compensates for the representation of ID^*_{i-1}. Moreover, similar to the main proof described for fixed-length identities, the compressed oracle contains the full representation of ALL other functions except $h_i[\mathrm{ID}^*_{i-1}] : \{0,1\}^{n_i} \rightarrow \{0,1\}^m$, where we provide its description on all points except for id^*, for which the adversary can find its inverse.