# A Note on Adaptive Security in Hierarchical Identity-Based Encryption

Rishab Goyal *
University of Wisconsin-Madison, WI,
rishab@cs.wisc.edu

Venkata Koppula
IIT Delhi
kvenkata@iitd.ac.in

Mahesh Sreekumar Rajasree †
CISPA Helmholtz
srmahesh1994@gmail.com

**Abstract**

We present the first adaptively-secure HIBE, that does not rely on bilinear pairings or random oracle heuristics. Notably, we prove adaptive security of the Döttling-Garg HIBE scheme based on any selectively secure IBE system in the standard model. Combining this with known results, this gives *the first adaptively-secure HIBE system from a wide variety of standard assumptions such as CDH/Factoring/LWE/LPN*. We also extend our adaptively-secure HIBE system to satisfy full-anonymity, giving the *first adaptively-secure anonymous-HIBE under* CDH/LWE *assumption*. All our HIBE systems support unbounded length identities as well as unbounded number of recursive delegation operations.

## 1 Introduction

Traditional public-key encryption systems support data encryption for users, under the assumption that the recipient has established a public key. Identity-based encryption (IBE) [Sha85] is an advanced encryption system, which gets around the need for every user to establish its own individual public key.

In IBE, a central authority (e.g., government entity, non-profit organization, etc) is tasked with establishing a pair of master public-secret key pair $(\mathsf{MPK}, \mathsf{MSK})$. This reduces the burden on every user, as they do not need to generate and distribute their public keys. Rather, anybody can encrypt data for a user, given just their public identity $\mathsf{ID}$ (e.g., name, department, etc). Encryption needs only $\mathsf{MPK}$ and the target recipient's identity, $\mathsf{ID}$. Each user gets a secret decryption key $\mathsf{sk}_{\mathsf{ID}}$, corresponding to its identity $\mathsf{ID}$, for decrypting a ciphertext intended for it. Such keys are issued by the central authority, which authenticates user's identity and derives user-specific decryption key from master secret key, $\mathsf{MSK}$.

Hierarchical identity-based encryption (HIBE) [HL02, GS02] is a popular generalization of IBE. It was proposed to mirror an organizational hierarchy. It allows iterative key derivation, where a key authority can distribute its workload by delegating private key generation and identity authentication to lower-level authorities. In standard IBE, a single authority has to bear this burden; while in HIBE, a user with private key $\mathsf{sk}_{\mathsf{ID}}$ (for identity $\mathsf{ID}$) can act as an (intermediate) key authority, thus derive decryption keys for users under its hierarchy. By users under its hierarchy, we mean users whose identity are of the form '$\mathsf{ID}||?\cdots?$', where '?' could be any bit. For example, a CS department chair at a university could be given the key for

*Support for this research was provided by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.

†Work done while the author was at IIT Delhi. Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

identity $(\texttt{edu}, \texttt{univ}, \texttt{cs})$ allowing him to derive keys for identities $(\texttt{edu}, \texttt{univ}, \texttt{cs}, \texttt{username})$ corresponding to email addresses `username@cs.univ.edu`.

(H)IBE enable a powerful encryption functionality. They replace the notion of user-specific public keys with (natural) identities, and the entire system has just a "single" public key for an exponential number of users. Since early 2000s, numerous (H)IBE systems have been designed from a variety of cryptographic assumptions [DH76, Bon98, BR93, BF01, Reg05] with numerous features. The following is a partial list of such systems [BF01, Coc01, HL02, GS02, CHK03, BB04a, BB04b, Wat05, BBG05, BW06, BGH07, GPV08, SW08, AB09, GH09, ABB10a, ABB10b, LW10, CHKP12, BKP14, CGW15, DG17a, DG17b, BLSV18, WC23, GKRV24, HKK+24, GKK+25].

**Defining security.** The most desirable goal while designing (H)IBE is to prove *adaptive security assuming standard cryptographic hardness*. Adaptive security (also called full security) is defined as the ability to resist chosen-plaintext attacks. Such attackers: (1) *adaptively* select a target/challenge identity $\mathsf{ID}^*$ for which they receive a challenge ciphertext $\mathsf{CT}^*$, (2) can corrupt decryption keys for any polynomial number of identities $\{\mathsf{ID}_i\}_i$ as long as none of them match $\mathsf{ID}^*$. For HIBE, condition (2) is generalized to also require that none of the corrupted keys can be used to decrypt $\mathsf{CT}^*$ (either directly or by deriving a delegated key).

Adaptive security is the most natural approach to capture "real-world" threats. However, proving adaptive security is a very challenging problem! Thus, researchers often start with a weaker goal of *selective* security, which states that chosen-plaintext attacks can be resisted as long as the attacker selects $\mathsf{ID}^*$ *at the beginning*. While selective security is quite unnatural in practice (e.g., why would an attacker select $\mathsf{ID}^*$ before system is set up), it is a good starting point for obtaining theoretical feasibility results. A common practice in cryptography is to rely on complexity leveraging to boost to full security, but it comes at a heavy cost of sub-exponential security loss and stronger assumptions.

**The problem.** Over the last few decades, there has been tremendous progress in designing IBE systems with adaptive security [BF01, BB04b, Wat05, Gen06, GPV08, Wat09, ABB10a, LW10, CGW15]. This culminated in beautiful works by Döttling and Garg [DG17a, DG17b], who developed an approach to generically design *adaptively secure* IBE and *selectively secure* HIBE from any selectively secure IBE. This led to major developments in identity-based cryptography. Unfortunately, Döttling-Garg [DG17a, DG17b] and long line of follow-ups [BLSV18, DGHM18, GHMR18, GHM+19, GV20, GSW21, GGL24] left open the gap between selective and full security for HIBE. Thus, a major open problem in identity-based cryptography has been:

*Can we design adaptively secure HIBE from same assumptions as that required for adaptively secure IBE?*

To date, all known fully secure HIBE systems [Wat05, BW06, Wat09, LW10, LOS+10, LW11, CW13, BKP14, LP19, LP20] rely on clever algebraic tricks and/or complexity leveraging and/or random oracle heuristic. Moreover, prior works have identified important challenges and barriers in proving full security of HIBE [LW14, DG17b].

**Our results.** We close the gap between full and selective security in HIBE, answering the above question affirmatively. We show:

**Theorem 1.** Assuming a selectively secure IBE system, there exists a fully secure HIBE system.

Combining these with prior results [Coc01, BF01, AB09, ABB10a, ABB10b, DG17a, DG17b, BLSV18, DGHM18], we obtain the following as an immediate corollary:

**Corollary 1.** Assuming hardness of (at least one of) $X \in \{\mathsf{Factoring}, \mathsf{CDH}, k\text{-}\mathsf{Lin}, \mathsf{LWE}, \mathsf{exLPN}\}$, there exists a fully secure HIBE system.

We extend our techniques to anonymous HIBE [BDCOP04, BW06, KSW08], where the goal is to additionally hide recipient's identity in the ciphertext.

**Theorem 2.** Assuming hardness of $\mathsf{CDH}$ or $\mathsf{LWE}$, there exists a fully secure anonymous HIBE system.

We remark that all our (anonymous) HIBE systems support unbounded length identities, thus they do not need an a-priori bound on the depth of the hierarchy.

## 1.1 Technical Overview

In this section, we provide a high level overview of our techniques. We start by recalling the notion of HIBE and highlight the differences between selective and adaptive security for HIBE.

An HIBE scheme consists of five algorithms: $\mathsf{Setup}$, $\mathsf{Enc}$, $\mathsf{KeyGen}$, $\mathsf{Dec}$ and $\mathsf{Delegate}$. The setup algorithm outputs the master public key $\mathsf{mpk}$ and the master secret key $\mathsf{msk}$. The encryption algorithm takes as input a master public key $\mathsf{mpk}$, a message $m$ and an identity $\mathsf{id} \in \{0,1\}^*$, and outputs a ciphertext $\mathsf{ct}$. The key generation algorithm uses the master secret key $\mathsf{msk}$ to generate a secret key $\mathsf{sk}_{\mathsf{id}}$ for any identity $\mathsf{id} \in \{0,1\}^*$. The secret key $\mathsf{sk}_{\mathsf{id}}$ corresponding to identity $\mathsf{id}$ can be used to decrypt any ciphertext for $\mathsf{id}$. Additionally, $\mathsf{sk}_{\mathsf{id}}$ can be used to generate a secret key for any identity that contains $\mathsf{id}$ as a prefix (that is, $\mathsf{id}||\mathsf{id}'$), using $\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$. Such delegation operations can be iteratively performed on a key arbitrary many times.

Informally, security in HIBE is defined as an attacker's inability to bypass semantic security for any ciphertext, as long as it does not ask for the secret key associated with challenge identity, *or* any of the corrupted secret keys cannot be delegated to generate such a secret key. This is formally captured through a *selective* security game as follows.

**Selective security.** The attacker first selects a challenge identity $\mathsf{id}^*$, and sends it to the challenger. The challenger sets up the system and sends $\mathsf{mpk}$ to the attacker. Next, the attacker makes polynomially many secret key queries, and selects two challenge messages $m_0, m_1$. It is essential that all identities $\{\mathsf{id}_i\}_i$ for which the attacker receives a secret key are such that $\mathsf{id}_i \not\preceq \mathsf{id}^*$ (i.e., $\mathsf{id}_i$ is not a prefix of $\mathsf{id}^*$). Following this, the challenger creates a challenge ciphertext $\mathsf{ct}^*$, which is either an encryption of $m_0$ or $m_1$ for the identity $\mathsf{id}^*$. And, on receiving $\mathsf{ct}^*$, the attacker can make more secret key queries (with the same prefix-unsatisfiability constraint), and has to finally guess whether $m_0$ or $m_1$ was encrypted.

As discussed earlier, selective security does not capture usual real-world threats. Typically, an attacker would select a target identity $\mathsf{id}^*$ after it looks at $\mathsf{mpk}$ and corrupts some secret keys. Such an adaptive choice for selecting challenge identity is formalized via an *adaptive security* game. However, unlike IBE (and more generally FE) systems, defining adaptive security for *hierarchical* IBE is not as straightforward. The reason is, beyond giving the attacker the option to select $\mathsf{id}^*$ at a later point, one has to be careful about *how an attacker can corrupt a secret key.* Let us elaborate further.

*A simple, but incorrect, approach to define adaptive security:* To capture fully adaptive adversaries, consider an adversary that makes polynomially many secret key queries to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ oracle, before it commits to $\mathsf{id}^*$ and messages $m_0, m_1$. After this, the game proceeds as in the selective setting. That is, the adversary receives $\mathsf{ct}^*$ and makes polynomially many (post-challenge) secret key queries to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ again.

At first glance, the above seems sufficient to capture fully adaptive adversaries in HIBE. Unfortunately, that is not the case. This is because it does not use $\mathsf{Delegate}$ algorithm. As a result, we can trivially design HIBE systems that will be "adaptively" secure as per above generalization, but otherwise insecure!

For instance, consider a (contrived) scheme where $\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$ always outputs $\mathsf{sk}_{\mathsf{id}}$. Clearly, such an HIBE scheme just gives up the master secret key as part of any key delegation operation, thus if an adversary ever gets a chance to corrupt a "delegated" key, then the system can be trivially attacked. Such a definitional issue in hierarchical systems was first pointed by Shi and Waters [SW08]. They proposed a more sound approach to capture (full) adaptive security.

**Adaptive security.** An adaptive adversary makes three types of queries – $\mathsf{KeyGen}$, $\mathsf{Delegate}$, and $\mathsf{OutputKey}$. Using a $\mathsf{KeyGen}$ query, it can ask the challenger to create a secret key for any identity $\mathsf{id}$. Specifically, the challenger uses $\mathsf{msk}$ to create $\mathsf{sk}_{\mathsf{id}}$, but it does **not** send $\mathsf{sk}_{\mathsf{id}}$ to the attacker! Instead, the challenger sends a unique 'token' $t$ (say, query index). $\mathsf{Delegate}$ query is similar, wherein the adversary asks the challenger to derive a delegated key for an identity $\mathsf{id}'$. Specifically, the adversary provides a valid token $t$ (corresponding to some key $\mathsf{sk}_{\mathsf{id}}$), and $\mathsf{id}'$, and the challenger creates a key for $\mathsf{id}||\mathsf{id}'$ by running $\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$. Again, the challenger does **not** send the secret key, rather creates a new token $t'$ and sends $t'$ to the adversary.

At a high level, the purpose of $\mathsf{KeyGen}$/$\mathsf{Delegate}$ queries is to give the adversary the capability to 'initialize' honest users in an HIBE system. To actually corrupt keys/users, the adversary can make a third type of query, $\mathsf{OutputKey}$. The adversary supplies a valid token $t$ and receives the corresponding secret key. This

enables an attacker to fully corrupt any user's secret key in a real-world execution of HIBE, as a user's secret key could be generated by either KeyGen, or Delegate, or any iterative combination of these two algorithms.

By allowing an attacker to make such queries, one can properly capture a fully adaptive attacker. Note that each attacker can make these queries arbitrarily in the pre/post-challenge phase, with the only restriction that for all secret keys $\{\mathsf{sk}_{\mathsf{id}}\}_{\mathsf{id}}$ *'actually corrupted'* by the attacker, id is **not** a prefix of $\mathsf{id}^*$. That is, the attacker cannot trivially decrypt the challenge ciphertext.

We emphasize that this does **not** limit the attacker from making a KeyGen/Delegate corresponding to $\mathsf{id}^*$ (or its prefix). But it only says that the attacker cannot get their secret keys.

**Why is adaptive security much harder?** The main challenge in proving adaptive security can be understood as follows. Consider an adaptive adversary that asks the challenger to first create a secret key for $\mathsf{id}^*$ by making a KeyGen query. Afterwards, it makes a Delegate query on the token corresponding to $\mathsf{id}^*$. Eventually, it asks only to corrupt the delegated key. The point is such an (adaptive) adversary can confuse the challenger by making KeyGen/Delegate queries, both, 'along' and 'off' the path to challenge identity $\mathsf{id}^*$. This flexibility in asking the challenger to sample and store a key vs. actually corrupting the key makes it extremely challenging to design and prove adaptively secure HIBE.

*We emphasize that this flexibility is not an issue in the selective setting.* The reason is that the adversary commits to $\mathsf{id}^*$ at the beginning. Thus, the challenger can partition all KeyGen and Delegate queries in two categories – (a) secret keys for identities that are prefixes of $\mathsf{id}^*$, (b) that are **not**. This partition is deterministically known during the pre-challenge phase, thus adversary can no longer confuse the attacker. Note that the same distinction cannot be made for pre-challenge queries in the adaptive security game. Thus, the challenger has **no** way of finding/guessing whether the secret key it has to create as a response to KeyGen/Delegate (in the pre-challenge phase) will be corrupted or not.

This distinction is the source of hardness surrounding adaptively secure HIBE. Lewko and Waters [LW14] further studied this issue, and provided many barriers in extending typical proof strategies for adaptive security from IBE to HIBE. The most common approach in the literature to get around these barriers relies on the random oracle heuristic [BR93, GS02] or bilinear maps [Wat05, BW06, GH09, Wat09, LW10, LW11, BKP14, LP19, LP20]. Unfortunately, this only gives us a heuristic approach to adaptively secure HIBE, unless we want to restrict ourselves to algebraic groups supporting bilinear pairings. Our goal is to prove adaptive security in the standard model, without relying on heuristics or pairing-based assumptions.

## Our Approach

We circle back to the beautiful line of work, initiated by Döttling and Garg [DG17a, DG17b] (referred as DG17 henceforth). They proposed a novel non-black-box approach towards designing (hierarchical) identity-based encryption systems. We provide a new security proof to handle adaptive attackers in DG17-style HIBE constructions. Let us start by briefly reviewing DG17. (Any reader familiar with Döttling-Garg constructions can safely skip the next two pages.)

**Reviewing [DG17a, DG17b].** Döttling and Garg proposed a new primitive called one-time signature with encryption (OTSE). They showed that OTSE can be built from any selectively secure IBE (as well as other simple assumptions). And, interestingly, OTSE can be generically used to design adaptively secure IBE and selectively secure HIBE. The core technical idea behind their generic design was to compose garbled circuits with OTSE to enable deferred encryption. Below we provide a short recap.

*One-time signatures with encryption*: It consists of five algorithms – $\mathsf{Gen}, \mathsf{Setup}, \mathsf{Sign}, \mathsf{Enc}, \mathsf{Dec}$. The Gen algorithm takes as input a length parameter $\ell$ and outputs public parameters pp. Setup takes pp as input and outputs a signing-verification key pair $(\mathsf{sk}, \mathsf{vk})$, with only the requirement that vk's length should be independent of $\ell$. Using sk one can sign any $\ell$ bit string $x$, while Enc publicly encrypts a message $\alpha$ w.r.t. vk, index $i \in [\ell]$, and bit $b$. Correctness states that Dec recovers $\alpha$ from the ciphertext, when given a signature $\sigma$ on a string $x$ such that $x_i = b$. Formally, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, x)$ and $\mathsf{ct} \leftarrow \mathsf{Enc}((\mathsf{vk}, i, x_i), \alpha)$, then $\mathsf{Dec}((\mathsf{vk}, x, \sigma), \mathsf{ct}) = \alpha$. Security states that no polytime adversary can learn the plaintext, even if it gets a single signature on any $x$ with the restriction that $x_i \neq b$ (where $i, b$ are used to create ct). Succinctly, OTSE

is a generalization of one-time signatures that additionally support message encryption, with the signature serving as a decryption key.

*OTSE to IBE*: The main design principle in DG17-style IBE constructions is to implement a deferred-encryption paradigm using OTSE and garbled circuits. The IBE master public key acts as a succinct commitment to an exponential number of (regular) PKE public keys, and the master secret key succinctly encodes opening information for every public key (corresponding to each identity). Let us be more concrete.

In their design, $\mathsf{mpk}$ contains a single OTSE verification key $\mathsf{vk}_\epsilon$ (here $\epsilon$ denotes the empty string), and $\mathsf{msk}$ contains the corresponding secret key $\mathsf{sk}_\epsilon$ and a PRF key $K$. The purpose of the PRF key is to deterministically expand $\mathsf{vk}_\epsilon$ into $2^n$ PKE public-secret key pairs, one for each ($n$-bit) identity. At a high level, the idea is to implicitly define a depth-$n$ complete binary tree, where each internal node is associated with an OTSE key. That is, let $v \in \{0,1\}^{\leq n}$ denote a unique identifier of any node in the tree. Every internal node $v \in \{0,1\}^{<n}$ is associated with an OTSE key pair as $(\mathsf{sk}_v, \mathsf{vk}_v) = \mathsf{Setup}(\mathsf{pp}; F(K, v))$. While leaf nodes are associated with a PKE key pair sampled as $(\mathsf{pke.pk}_v, \mathsf{pke.sk}_v) = \mathsf{PKE.Setup}(\mathsf{pp}; F(K, v))$. Crucially, all keys are sampled using this fixed PRF key $K$, thus deterministically fixed at setup time.

The secret key for an identity $\mathsf{id} \in \{0,1\}^n$ contains the PKE key $\mathsf{pke.sk}_{\mathsf{id}}$ for its corresponding leaf node, and a succinct opening to $\mathsf{pke.sk}_{\mathsf{id}}$ generated as a "chain" of OTSE signatures. In a bit more detail, for every $0 \leq i < n-1$, the key generator signs the message '$\mathsf{vk}_{\mathsf{id}[1:i]||0}||\mathsf{vk}_{\mathsf{id}[1:i]||1}$' under the OTSE key $\mathsf{sk}_{\mathsf{id}[1:i]}$. Here $\mathsf{id}[1:i]$ denotes the first $i$ bits of $\mathsf{id}$. And, for $i = n-1$, it signs '$\mathsf{pke.pk}_{\mathsf{id}[1:i]||0}||\mathsf{pke.pk}_{\mathsf{id}[1:i]||1}$' under $\mathsf{sk}_{\mathsf{id}[1:i]}$.

Observe that chaining OTSE signatures as above ensures they are deterministically linked, with two special features: (1) the size of the OTSE chain of signatures from root to leaf grows polynomially with $n$, and not $2^n$, and (2) this acts as an "opening" of $\mathsf{pke.pk}_{\mathsf{id}}$ under $\mathsf{vk}_\epsilon$. The point (2) is crucial in enabling efficient IBE-style encryption/decryption functionality as we explain next.

To encrypt $m$ for $\mathsf{id}$ under $\mathsf{mpk} = \mathsf{vk}_\epsilon$, the trick is to iteratively garble circuits starting from leaf to root. Specifically, the encryptor starts by garbling the circuit $\mathsf{PKE.Enc}(\cdot, m; r)$ where $m$ and $r$ (chosen uniformly at random) are hardcorded into the circuit. This results in a garbled circuit $\mathcal{T}$ along with the associated labels $\{\mathsf{lab}^T_{j,b}\}$. Next, it garbles the circuit $\{\mathsf{OTSE.Enc}((\cdot, \mathsf{id}_n \cdot \lambda + j, b), \mathsf{lab}^T_{j,b})\}_{j,b}$ to produce $\mathcal{Q}^{(1)}$ and its associated labels $\{\mathsf{lab}^{(1)}_{j,b}\}$. Note that this circuit has the $n$-th bit of recipient $\mathsf{id}$ hardcoded. Such a process is iteratively carried out, where the $i$-th circuit has $(n-i+1)$-th bit of $\mathsf{id}$ hardcoded, and appropriately encrypts wire labels for $(i-1)$-th circuit. Eventually, this way the encryptor computes the ciphertext which contains a sequence of $n+1$ garbled circuits: $\mathcal{Q}^{(n)}, \ldots, \mathcal{Q}^{(1)}$, and $\mathcal{T}$ along with wire keys $\{\mathsf{lab}^{(n)}_{j,\mathsf{mpk}_j}\}$ for $\mathcal{Q}^{(n)}$.

During decryption, an identity (secret) key is used to successively decrypt the encrypted wire labels followed up evaluating the garbled circuits to get encrypted wire labels for next circuit and so on. This process looks like a guided binary search over the OTSE tree. In further detail, the decryptor starts with $\{\mathsf{lab}^{(n)}_{j,\mathsf{mpk}_j}\}_j$ and evaluates $\mathcal{Q}^{(n)}$. This generates a sequence of OTSE ciphertexts, $\{\mathsf{OTSE.Enc}((\mathsf{vk}_\epsilon, \mathsf{id}_1 \cdot \lambda + j, b), \mathsf{lab}^{(n-1)}_{j,b})\}_{j,b}$. Now using OTSE signature on '$\mathsf{vk}_0||\mathsf{vk}_1$' under key $\mathsf{sk}_\epsilon$, the decryptor can obtain $\{\mathsf{lab}^{(n-1)}_{j,\mathsf{vk}_{\mathsf{id}_1},j}\}$. That is, wire keys corresponding to bits of $\mathsf{vk}_{\mathsf{id}_1}$. This process can be iteratively carried, where it next evaluates $\mathcal{Q}^{(n-1)}$ using these keys, then decrypts them using OTSE signature on '$\mathsf{vk}_{\mathsf{id}_1||0}||\mathsf{vk}_{\mathsf{id}_1||1}$' under key $\mathsf{sk}_{\mathsf{id}_1}$, and so on. Eventually, it obtains a PKE ciphertext encrypting the plaintext under $\mathsf{pke.pk}_{\mathsf{id}}$ which can be decrypted using $\mathsf{pke.sk}_{\mathsf{id}}$.

*Generalizing to hierarchical IBE*: DG17 showed that the above template can be easily extended to support hierarchical encryption and key generation. To handle hierarchical encryption (i.e., encryption for varying length identities), each internal node is associated with a PKE key (in addition to an OTSE key). To enable hierarchical key generation, the idea is to use delegatable PRFs [GGM86] (also called prefix-constrained PRFs [KPTZ13, BW13, BGI14]). Recall a delegatable PRF allows a key holder to construct a constrained key $K_v$ such that $K_v$ can be used to evaluate the PRF on any string that has $v$ as prefix. We summarize the main differences between the aforementioned IBE template and its hierarchical generalization below:

**Setup.** It is nearly identical with the only difference that the PRF key sampled is delegatable.

**Key Generation & Delegation.** Key generation proceeds as before, except now it signs the message '$\mathsf{vk}_{\mathsf{id}[1:i]||0}||\mathsf{vk}_{\mathsf{id}[1:i]||1}||\mathsf{pke.pk}_{\mathsf{id}[1:i]}$' instead. Basically, each node's PKE key is also signed. Additionally, each secret key also contains a delegated PRF key $K_{\mathsf{id}}$. Recall using $K_{\mathsf{id}}$ a user can evaluate the PRF on any input of the form $\mathsf{id}||\cdots$. This allows a secret key holder to further delegate. Because given $K_{\mathsf{id}}$, it can deterministically sample OTSE as well as PKE keys for all nodes underneath the $\mathsf{id}$-th node.

**Encryption & Decryption.** Both these algorithms work nearly identically with minor syntactic changes. The changes are simply due to the slightly larger message length (to incorporate signing the PKE key at each level as well) that are now signed using OTSE.

Delegatable PRFs can be designed via the seminal GGM PRF construction. Therefore, the above generalization leads to a hierarchical IBE system.

**Proving selective security.** To provide more insight into the hardness of proving adaptive security of HIBE, let us zoom in on how the proof of selective security goes. Recall in selective security, the adversary commits to challenge identity $\mathsf{id}^*$ at the beginning of the game. Since $\mathsf{id}^*$ is selectively available, the challenger can replace all the evaluations of the PRF along the path from the root ($v = \epsilon$) to the challenge node ($v = \mathsf{id}^*$) with truly random values. This leverages selective security of delegatable PRFs. Since the challenger does not need the full PRF key to answer any admissible secret key query, then one can rely on a step-by-simulation strategy for simulating the challenge ciphertext $\mathsf{ct}^*$ one garbled circuit at-a-time. In further detail, $\mathsf{ct}^*$ only contains labels corresponding to $\mathsf{vk}_\epsilon$ for the garbled circuit $\mathcal{Q}^{(n)}$. Thus, this can be simulated given OTSE encryptions of wire labels, under $\mathsf{vk}_\epsilon$, for $\mathcal{Q}^{(n-1)}$ and appropriate indices (i.e., $\{(\mathsf{id}_1 \cdot \lambda + j, b)\}$). Next, by relying on OTSE security, half of these OTSE ciphertexts can be replaced with garbage encryptions. That is, wire labels corresponding to only $\mathsf{vk}_{\mathsf{id}_1}$ are correctly encrypted. Following this process iteratively, $\mathsf{ct}^*$ can be fully simulated, where the final garbled circuit $\mathcal{T}$ is simulated using a regular PKE ciphertext under $\mathsf{pke.pk}_{\mathsf{id}^*}$, the challenge identity/node's PKE key.

**Why is it difficult to prove adaptive security?** Unfortunately, the above approach cannot handle adaptive attackers. There are two reasons – *first*, delegatable PRFs. The canonical approach for designing delegatable PRFs builds on the classic Goldreich-Goldwasser-Micali PRF construction [GGM86, KPTZ13, BW13, BGI14]. Until very recently [HKK23], it was not known if delegatable PRFs can be designed with adaptive security under standard polynomial time hardness. Thus, any approach to prove adaptive security of DG17-style HIBE construction failed at the first step.

Luckily, in a beautiful recent work by Hofheinz, Kastner, and Klein [HKK23], the problem of adaptively secure delegatable PRFs was resolved. They developed a new *rewinding*-based proof technique to prove adaptive security of GGM PRF under standard polynomial hardness of any PRG. Although their formulation of delegatable PRFs mildly departs from the abstraction necessary for the DG17-HIBE template[1], it is relatively easy to show that their proof techniques are sufficient to prove adaptive security of delegatable PRFs that are needed for the DG17-HIBE template. Therefore, by relying on adaptively secure delegatable PRFs, we can successfully implement the first step of the hybrid proof in the adaptive HIBE security experiment. But, this is where the *second* (and major) issue comes up.

**Step-by-step garbling simulation breaks down.** A major difference between the above IBE and HIBE templates is that an adversary never directly learns anything about the actual PRF evaluations in an IBE system (since identity keys only consist of OTSE signatures and PKE secret keys); whereas in HIBE, and adversary additionally learns (multiple) delegated PRF keys. While one might suppose that, by relying on adaptively secure delegatable PRFs, this difference would become meaningless, it is not the case!

DG17 (and followups) use the fact that a GGM-style PRF evaluation tree behaves as a tree of truly random values, as long as an attacker does not see any PRF evaluations. That is, an attacker never explicitly learns PRF values for any node, but only OTSE/PKE keys corresponding to those nodes. Thus, all these keys are as good as *truly randomly sampled keys* rather than pseudo-random. We stress that this ensures

[1]Standard GGM PRFs rely on a length-doubling PRGs; while for HIBE, we need a length-tripling PRG with the property that only the two-thirds of the PRF output needs to act as a delegated key in the future.

a step-by-step garbled circuit simulation strategy interleaved with OTSE security can be safely executed. Because for using OTSE security, we have the guarantee that OTSE keys are randomly sampled.

Crucially, this cannot be replicated in HIBE! Note that even having access to *adaptively secure* delegatable PRFs, we **cannot** argue that all intermediate GGM-style PRF evaluations are as good as random. This is because an adaptive attacker learns PRF evaluations for intermediate nodes due to the hierarchical nature. In other words, an attacker can query for $\mathsf{id} = 111$, and later for $\mathsf{id} = 11$. Since DG17-style keys must be deterministically generated from master key, thus the secret key for 11 should be able to generate/explain the key for 111. This is essentially why we cannot hope to get to a hybrid where the entire (GGM) PRF tree (or, just the HIBE secret keys which contain delegated PRF keys) are as good as random. This is true even if we assume adaptively secure delegatable PRFs. This is why DG17 and all follow-ups were stuck at selective security for HIBE.

The core technical hurdle towards adaptive security is that for the pre-challenge queries, we need to "somehow" guess the path from the root to the challenge identity and replace only the PRF evaluations (in the queried HIBE keys) along this path with truly random values. In the case of selective security, such guessing is not required because the challenge identity is committed at the beginning of the game (i.e., there are no 'real' pre-challenge queries since $\mathsf{id}^*$ is known in advance).

**Nested hybrids and a pebbling-style strategy.** Our main technical contribution is based on the "nested hybrids" technique [FKPR14, FJP15] and using an intricate *pebbling-style* argument to prove adaptive security of the above HIBE scheme *with only polynomial security loss*. At its core, we show it is sufficient to only guess the location of a "single node" in the path from the root to the challenge identity $\mathsf{id}^*$, while moving between consecutive hybrids. At first, it might seem that, guessing the location of a single node in the tree of polynomial depth, would lead to an exponential security loss. However, we notice that this can be avoided, and we could prove indistinguishability of consecutive hybrids using nested hybrids, incurring only a polynomial loss. Basically, we only need to guess the 'index' of the first $\mathsf{OutputKey}$ query that overlaps with the path from root to challenge identity[2]. If we can successfully guess this, then we can answer all pre-challenge HIBE key queries as well as correctly perform the step-by-step garbling simulation while leveraging adaptive security of delegatable PRFs. Clearly, this would lead to (only) an overall polynomial loss in the proof of adaptive security, since number of queries is going to be some polynomial. A crucial component for this reduction is to ensure that we end up relying on delegatable PRF security to switch PRF evaluation for exactly *one* node in the GGM tree. We use a pebbling strategy to correctly design nested hybrids to be able to reduce to adaptive security of delegatable PRFs. We provide a detailed description later in Section 3. Below we summarize the key points of our proof:

**Game 0.** We start with the regular HIBE adaptive security game.

**Game 1.** Next, we rely on the fact that all HIBE secret keys (under the DG17 template) can always be thought to be directly generated from $\mathsf{msk}$ by running the key generation algorithm. That is, the challenger in this game no longer needs to do any computation while answering any $\mathsf{KeyGen}/\mathsf{Delegate}$, but only when the attacker makes a $\mathsf{OutputKey}$ query, it actually runs the key generation algorithm to create the appropriate secret key. This heavily relies on the unique/deterministic key derivation property of the HIBE construction.

**Game** $(i+1)_{i=1}^{n}$**.** Suppose $n$ is the length of challenge identity, $\mathsf{id}^*$. Next, we consider a sequence of $n$ hybrid games, where we simulate the first $i$ garbled circuits $\mathcal{Q}^{(n)}, \ldots, \mathcal{Q}^{(n-i+1)}$, rather than creating them honestly.

**Game** $(n+2)$**.** Here, we additionally simulate the final garbled circuit $\mathcal{T}$ as well.

Until this point, the proof structure so far resembles the (iterated) step-by-step garbling simulation strategy of Döttling-Garg. However, the similarities end here and our proof needs to diverge. Let us dive into the

[2]We highlight that [DG17a] also relied on a similar guessing trick. However, as explained below, there is a major difference between IBE and HIBE. In IBE, an attacker never learns intermediate PRF values directly, but in HIBE, it does learn them since they are contained as part of the secret key.

indistinguishability of hybrids $i$ and $i+1$ (for $1 \leq i \leq n$). Ideally, we want to replicate [DG17a], and do the following:

**Step 1:** Replace $i$-th garbled circuit with a simulated garbled circuit since only half of its wires keys are revealed.

**Step 2:** Use adaptive security of delegatable PRFs to generate OTSE keys randomly for node corresponding to $i$-bit prefix of $\mathsf{id}^*$. (Thus, all nodes from root to here contain OTSE keys that are randomly sampled.)

**Step 3:** Rely on OTSE security to switch half of the ciphertexts (encrypting wire labels for $(i+1)$-th garbled circuit) to garbage.

However, there is a major hurdle. It is unclear how to handle pre-challenge key queries. The reason is during the above (internal) hybrids, we are changing all OTSE keys for all nodes in $\mathsf{id}^*[1:1], \mathsf{id}^*[1:2], \ldots, \mathsf{id}^*[1:i]$ to be fully random instead of pseudo-random. But $\mathsf{id}^*$ is unknown until challenge query is made by the attacker. Thus, how to figure out which OTSE keys to sample randomly instead of pseudo-randomly in the pre-challenge phase!? We could guess, but that would result in an exponential security loss. Our observation is that once we switch half of the OTSE ciphertexts in step 3, then we no longer need OTSE keys corresponding to $\mathsf{id}^*[1:i]$ to be randomly sampled, but sampling it pseudo-randomly is good enough. So, we start by adding another internal hybrid as follows:

**Step 4:** Use adaptive security of delegatable PRFs again to revert back OTSE keys for $\mathsf{id}^*[1:i]$ as pseudo-randomly sampled.

The advantage of above is that at any point in the proof, we now only need to sample a single OTSE key randomly. That is, OTSE keys for $\mathsf{id}^*[1:1], \mathsf{id}^*[1:2], \ldots, \mathsf{id}^*[1:i-1]$ could still be *pseudo-randomly* sampled, but only for $\mathsf{id}^*[1:i]$, we need to sample the corresponding OTSE key randomly. The above is reminiscent of pebbling strategies used in various other contexts [HJO+16, JW16].

There is a minor technical subtlety that we need to handle while implementing the above idea. Typically, delegatable PRFs do not allow PRF evaluation queries on any prefix of the challenge input. In the above approach, we would need honest PRF evaluations for all prefixes of the challenge input (i.e., $\mathsf{id}^*[1:i]$ in above case). This seems contradictory, but we show that this can be handled nearly generically. Basically, we create distinction between a PRF evaluation and what we consider a delegated PRF key. This way we allow an attacker to make PRF evaluation queries on prefixes of challenge input as well, but not delegation key queries for those prefixes. We show that the recent adaptively secure delegatable PRFs [HKK23] are still secure under such an expanded definition. Thus, it appears that the main technical hurdle has been resolved. However, there is one final issue.

*How to guess* $\mathsf{id}^*[1:i]$ *in the pre-challenge phase?* The issue is that, even with the above modification, how does the reduction know what is the value of $\mathsf{id}^*[1:i]$ in the pre-challenge phase? Our latest modifications remove the need to guess all prefixes of $\mathsf{id}^*[1:i]$, but we still need to know the index of this node at the $i$-th level of the tree. This is essential as it is possible that a pre-challenge query overlaps with $\mathsf{id}^*[1:i]$ (i.e., has the same $i$-bit prefix). In this case, we need to sample OTSE keys for $\mathsf{id}^*[1:i]$ in the pre-challenge phase itself. However, this still leads to a potentially super-polynomial security loss, when $i = \omega(\log \lambda)$. Our final observation is that we could instead rely on a sequence of "nested hybrids" where we only need to do some limited guessing about the adversary's adaptive choices. Basically, we define these internal hybrid to depend on a query index $q$, where we interpret $q$ as the smallest query index such that the $q$-th $\mathsf{OutputKey}$ query is the *first* key query that overlaps with the $i$-bit prefix of $\mathsf{id}^*$, i.e. $\mathsf{id}^*[1:i]$. Thus, if an attacker can selectively provide this information or we can guess in the reduction, then we could finish the security analysis with only a polynomial security loss. This is because any polytime attacker makes at most a polynomial number of corruption queries, and we are simply the index of the first one which satisfies an efficiently testable predicate. We provide more details later in the main body. Overall, the above strategy of coupling a pebbling-argument with nested hybrids and reducing to adaptive security of delegatable PRFs gives us an adaptively secure HIBE system (under the minimal assumption of selectively secure IBE).

### Fully-secure anonymous HIBE from CDH and LWE

An HIBE scheme is *anonymous* if an adversary cannot even learn the recipient's identity from the challenge ciphertext. In other words, the security game is similar to the adaptive HIBE security game, except the adversary sends *two* message-identity pairs: $(m_0, \mathsf{id}_0)$ and $(m_1, \mathsf{id}_1)$, and receives $\mathsf{Enc}(\mathsf{mpk}, m_b, \mathsf{id}_b)$ for a random bit $b$. Naturally, the adversary should not receive with any keys for identities that are prefixes of $\mathsf{id}_0$ or $\mathsf{id}_1$. We show how to design anonymous HIBE under the hardness of the computational Diffie-Hellman (CDH) or learning with errors (LWE) assumption.

*Anonymous HIBE from CDH.* Brakerski et al. [BLSV18] showed that anonymous IBE can be constructed from a weakly compact blind IBE and blind garbled circuits, and these objects can be based on CDH. We start by observing that their anonymous IBE can be easily extended by relying on delegatable PRFs similar to [DG17a]. Next, our strategy is to prove adaptive security of the resulting construction by using our pebbling strategy and nested hybrids techniques. This results in an anonymous unbounded HIBE scheme based on CDH. At a very high level, the main difference between this construction and our non-anonymous HIBE construction is purely in replacing tree-based composition of OTSE and garbled circuits, with blind IBE and blind garbled circuits. We provide the CDH-based adaptively secure anonymous HIBE in Section 4.

*Anonymous HIBE from LWE.* Goyal-Koppula-Waters and Wichs-Zirdelis introduced the notion of lockable obfuscation [GKW17, WZ17, GKVW20], and demonstrated how lockable obfuscation can be used to upgrade any attribute-based encryption scheme [SW05, GPSW06] to satisfy one-sided predicate hiding [KSW08, BW07, GKW17]. They designed such obfuscation schemes under the hardness of LWE assumption. We observe that their construction can also be used (as is) for upgrading HIBE to anonymous HIBE, while preserving adaptive security. Therefore, this gives us an adaptively secure anonymous HIBE under LWE. We highlight that despite the fact that lockable obfuscation can only obfuscate circuits of a-priori bounded size, our anonymous HIBE still supports unbounded length identities. The reason is that, in the GKW transformation, the encryptor performs obfuscation. Since the recipient's identity (therefore, the maximum secret key length that can be used to decrypt the corresponding ciphertext) is fixed at encryption time, thus we could still use lockable obfuscation without relying on circular security. For completeness, we provide the transformation in Appendix B.

We believe our techniques might also be useful in other adaptive security contexts, such as adaptive registration-based encryption [GHMR18, GHM+19, GV20] and other advanced systems such as adaptive delegatable attribute-based/functional encryption systems [BGG+14, BCG+17]. We leave these as interesting open problems.

## 1.2 Related Work

Gentry and Silverberg [GS02] presented hierarchical identity-based systems in the ROM under the hardness of the Bilinear Diffie-Hellman problem. Later, Waters [Wat05] improved this to adaptive security without random oracles, but could only support constant number of delegation levels. Boyen-Water [BW06] extended this to additionally design fully anonymous HIBE constructions from similar pairing assumptions. However, all these schemes could only be proven adaptively secure for identity hierarchies of constant depth as the security reductions suffered from exponential degradation in depth of hierarchy. Gentry-Halevi [GH09] presented the first fully secure HIBE with a tight proof of security, hence supporting polynomially many levels. This was later refined by Waters [Wat09] via the dual systems methodology. Several follow-up works [LOS+10, LW10, LW11, CW13, BKP14, LP19, LP20] improved this line of work designing fully secure (anonymous) HIBE with various improvements such as shorter ciphertexts, unbounded identities, tighter reductions, etc. Achieving adaptive security without bilinear maps has been a very difficult task, and has generally required more sophisticated primitives, such as indistinguishability obfuscation [BGI+01, GGH+16] or functional encryption [Wat15, ABSV15]. There are some exceptions [Tsa19, GLW21], but these do not support any key delegation.

This lack of diversity can also be attributed to many barriers known in the literature. Lewko-Waters [LW14] demonstrated that adaptive security cannot be proven (via black-box reductions) for schemes with certain checkability properties. Recently, Brakerski-Medina [BM23] extended this to cover rewinding reductions,

shedding light on why certain lattice-based ABE schemes remain unprovable for full adaptive security, even though no known adaptive attacks have been identified. In contrast, semi-adaptive security (a weaker notion) has been comparatively easier to achieve [BV16, GKW16], and any selectively secure scheme can be generically upgraded to achieve semi-adaptive security.

**Hierarchical encryption beyond HIBE.** Delegatable attribute-based encryption [GVW13, BGG+14] is a generalization of HIBE, where each secret key is associated with predicate $\phi$ and each ciphertext encodes an attribute $x$. Correctness states that the ciphertext can be decrypted iff $\phi(x) = 1$, and moreover, the delegation feature says that a secret key for predicate $\phi$ can be delegated to generate a secret key for any predicate of the form $\phi \wedge \psi$ for any other predicate $\psi$. In a concurrent work, Goyal and Yadugiri [GY25] designed a delegatable attribute-based encryption for general circuits by relying on general-purpose witness encryption.

# 2 Preliminaries

**Notation.** Let PPT denote probabilistic polynomial-time. We denote the set of all positive integers upto $n$ as $[n] := \{1, \ldots, n\}$ and $[n]_0 := \{0, 1, \ldots, n\}$. Also, we use $[i, j]$ to denote the set of all non-negative integers between $i$ and $j$ including $i, j$, i.e., $[i, j] := \{i, i+1, \ldots, j\}$. For any two binary string $x, y$, we use the notation $x \preceq y$ (or $x \in \mathsf{prefix(y)}$) to imply that $x$ is a prefix of $y$ and $x \,||\, y$ to denote $x$ concatenated with $y$. And, $x[i, j]$ denotes the substring $x[i] \,||\, x[i+1] \,||\, \ldots \,||\, x[j]$ when $i \leq j$ whereas $x[i, j] = \epsilon$ where $i > j$. Throughout this paper, unless specified, all polynomials we consider are positive polynomials. For any finite set $S$, $x \leftarrow S$ denotes a uniformly random element $x$ from the set $S$. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element $x$ drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of $n$ components, where each component is drawn independently from the distribution $\mathcal{D}$.

## 2.1 Delegatable Pseudorandom Functions

A delegatable pseudorandom function is a pseudorandom function $\mathsf{PRF} = (\mathsf{Setup}, \mathsf{Eval})$ with an additional deterministic algorithm $\mathsf{Delegate}(\cdot, \cdot)$ with the following description and properties.

$\mathsf{Setup}(1^\lambda, 1^m)$ : The setup algorithm takes as input the security parameter $\lambda$ and the output length $m$, and outputs a key $s$.

$\mathsf{Eval}(s, x)$ : The evaluation algorithm is a deterministic algorithm that takes as input a key $s$ and an input $x \in \{0,1\}^*$ and outputs a string $y \in \{0,1\}^m$.

$\mathsf{Delegate}(s, x)$ : The delegation algorithm is a deterministic algorithm that takes as input a key $s$ and an input $x \in \{0,1\}^*$ and outputs a key $s_x$.

**Correctness.** The scheme is correct if for all $\lambda, m \in \mathbb{N}$, $x, x' \in \{0,1\}^*$ and $s \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$, and $s_x = \mathsf{Delegate}(s, x)$, the following holds – $\mathsf{Eval}(s_x, x') = \mathsf{Eval}(s, x \,||\, x')$ and $\mathsf{Delegate}(s, x \,||\, x') = \mathsf{Delegate}(s_x, x')$

**Definition 2.1** (Adaptive Security)**.** Consider the following experiment:

1. The challenger randomly generates $s \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$.
2. In the pre-challenge phase, the adversary has oracle access to the functions $\mathsf{Delegate}(s, \cdot)$ and $\mathsf{Eval}(s, \cdot)$. Let $Q$ and $T$ be the set of queries made to each oracle by the adversary in this phase.
3. In the challenge phase, the adversary sends $x^* \in \{0,1\}^*$ such that $x^* \notin T$ and there does not exists an $x \in Q$ such that $x \preceq x^*$, i.e., $x$ is not a prefix of $x^*$. The challenger randomly picks $b \in \{0,1\}$. If $b = 0$, it returns $\mathsf{Eval}(s, x^*)$, else it returns a truly random value.
4. In the post-challenge phase, the adversary has oracle access to $\mathsf{Delegate}(s, \cdot)$ but is not allowed to query on any prefix of $x^*$. It also has access to $\mathsf{Eval}(s, \cdot)$ but is not allowed to query on $x^*$.

5. Finally, the adversary sends $b' \in \{0,1\}$ and wins if $b = b'$.

A PRF $\mathsf{PRF}$ is an adaptively secure delegatable pseudorandom function if for all $\mathsf{PPT}$ adversaries, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, m \in \mathbb{N}$, $\Pr[\text{adversary wins}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

**Remark 2.2.** Most prior works focused on delegatable PRF that can only be evaluated at the leaf nodes, i.e., only on $n$-bit strings. Hofheinz, Kastner, and Klein [HKK23] recently demonstrated that the GGM construction [GGM86] provides an adaptively secure delegatable PRF. We emphasize that the proof provided in [HKK23] can be extended to achieve adaptively secure delegatable PRF satisfying the above definition. More details are provided in Appendix A.

**Theorem 2.3.** Assuming one-way functions exist, there exists adaptively secure delegatable PRFs.

## 2.2 One-Time Signature with Encryption

A One-Time Signature with Encryption (OTSE) scheme consists of the following algorithms.

$\mathsf{Gen}(1^\lambda, 1^\ell)$ : The generation algorithm takes as input the security parameter $\lambda$ and a message length $\ell$ and outputs public parameter $\mathsf{pp}$.

$\mathsf{Setup}(\mathsf{pp})$ : The setup algorithm takes as input public parameter $\mathsf{pp}$ and outputs a verfication key $\mathsf{vk}$ and a signing key $\mathsf{sk}$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, x)$ : The signing algorithm takes as input public parameter $\mathsf{pp}$, a signing key $\mathsf{sk}$ and a message $x \in \{0,1\}^\ell$ and outputs a signature $\sigma$.

$\mathsf{Enc}(\mathsf{pp}, (\mathsf{vk}, i, b), m)$ : The encryption algorithm takes as input public parameter $\mathsf{pp}$, a verification key $\mathsf{vk}$, an index $i \in [\ell]$, a bit $b \in \{0,1\}$ and a plaintext $m$. It outputs a ciphertext $\mathsf{ct}$.

$\mathsf{Dec}(\mathsf{pp}, (\mathsf{vk}, x, \sigma), \mathsf{ct})$ : The decryption algorithm takes as input public parameters $\mathsf{pp}$, a verification key $\mathsf{vk}$, a signature $\sigma$, a message $x$ and a ciphertext $\mathsf{ct}$. It outputs $m$.

**Succinctness.** For $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, \ell)$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$, the size of $\mathsf{vk}$ is independent of $\ell$ and only depending on the security parameter $\lambda$. But, the size of $\mathsf{pp}$ can polynomial in both $\ell$ and $\lambda$.

**Correctness.** For correctness, we require that for all $\ell, \lambda \in \mathbb{N}$, message $x$, plaintext $m$ and $i \in [\ell]$, we have that $\mathsf{Dec}(\mathsf{pp}, (\mathsf{vk}, \sigma, x), \mathsf{ct}) = m$, where $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$ and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, (\mathsf{vk}, i, x_i), m)$.

**Security.** Consider the following experiment with a $\mathsf{PPT}$ adversary $\mathcal{A}$.

1. The challenger generates $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$ and sends $\mathsf{pp}$ to $\mathcal{A}$. $\mathcal{A}$ returns a message $x$.
2. The challenger generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(\mathsf{pp})$ and computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, x)$. It sends $(\mathsf{vk}, \sigma)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs an index $i \in [\ell]$ and two plaintexts $m_0, m_1$.
4. The challenger randomly samples $b \leftarrow \{0,1\}$ and computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, (\mathsf{vk}, i, 1 - x_i), m_b)$. It sends $\mathsf{ct}^*$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins if $b' = b$.

An OTSE scheme is said to selective secure if for all $\mathsf{PPT}$ adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[\mathcal{A} \text{ wins in the above experiment}] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

**Theorem 2.4** ([DG17a, Theorem 2])**.** Assuming selectively secure IBE, there exists secure OTSE schemes.

## 2.3 Blind Garbled Circuits

A blind garbling scheme $\mathsf{GC}$ consist sof three algorithms $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ with the following syntax:

$\mathsf{Garble}(1^\lambda, 1^n, 1^m, C)$. The garbling algorithm takes as input the security parameter $\lambda$, two parameters $n, m$ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$, and outputs a garbled circuit $\mathcal{C}$ and the associated labels $\{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}$.

$\mathsf{Eval}(\mathcal{C}, \{\mathsf{lab}_i\}_{i\in[n]})$. The evaluation algorithm takes as input a garbled circuit $\mathcal{C}$ and a set of labels $\{\mathsf{lab}_i\}_{i\in[n]}$, and outputs $y \in \{0,1\}^m$.

$\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, y)$. The simulation algorithm takes as input the security parameter $\lambda$, the size of a circuit $C$, the input length $n$ and a string $y \in \{0,1\}^m$, and outputs a simulated garbled circuit $\mathcal{C}$ and simulated labels $\{\mathsf{lab}_i\}_{i\in[n]}$.

**Correctness.** We say a garbling scheme satisfies correctness if for all $\lambda, n, m \in \mathbb{N}$, $C : \{0,1\}^n \to \{0,1\}^m, x \in \{0,1\}^n$, we have that $\mathsf{Eval}(\mathcal{C}, \{\mathsf{lab}_{i,x[i]}\}_{i\in[n]}) = C(x)$, where $(\mathcal{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, 1^n, 1^m, C)$.

**Simulation Security.** For all $\lambda, n, m \in \mathbb{N}$, circuits $C : \{0,1\}^n \to \{0,1\}^m$ and $x \in \{0,1\}^n$, the following two distributions are computationally indistinguishable.

$$\left\{\mathcal{C}, \{\mathsf{lab}_{i,x[i]}\}_{i\in[n]} \ : \ (\mathcal{C}, \{\mathsf{lab}_{i,b},\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, 1^n, 1^m, C)\right\} \approx_c$$

$$\left\{\mathcal{C}, \{\mathsf{lab}_i\}_{i\in[n]} \ : \ (\mathcal{C}, \{\mathsf{lab}_i\}_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, 1^n, C(x))\right\}$$

**Blindness property.** A garbling scheme is said to be blind if $\mathsf{Sim}(1^\lambda, 1^c, 1^n, U_m) \approx_c U_\ell$, where $\ell = |\mathcal{C}| + |\{\mathsf{lab}_i\}_{i\in[n]}|$ and $U_m, U_\ell$ is the uniform distribution over $\{0,1\}^m, \{0,1\}^\ell$, respectively.

**Theorem 2.5** ([BLSV18])**.** Assuming PRGs, there exists blind garbling schemes.

## 2.4 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme $\mathsf{IBE}$ for set of identity spaces $\mathcal{I} = \{\{0,1\}^n\}_{n\in\mathbb{N}}$ and message spaces $\mathcal{M}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter $\lambda$ and identity length $n$. It outputs the public parameters $\mathsf{mpk}$ and the master secret key $\mathsf{msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm takes as input the master secret key $\mathsf{msk}$ and an identity $\mathsf{id} \in \{0,1\}^n$. It outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the public parameters $\mathsf{mpk}$, a message $m \in \mathcal{M}$, and an identity $\mathsf{id} \in \{0,1\}^n$. It outputs a ciphertext $\mathsf{ct}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to m/\perp$. The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext $\mathsf{ct}$. It outputs either a message $m \in \mathcal{M}$ or a special symbol $\perp$.

**Correctness.** We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies correctness if for all $\lambda, n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $\mathsf{id} \in \{0,1\}^n$, $m \in \mathcal{M}$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = m$.

**Definition 2.6.** We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is fully/ adaptive secure if for any stateful PPT adversary $\mathcal{A}$ and polynomial $\mathsf{poly}(\cdot)$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$ and $n = \mathsf{poly}(\lambda)$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} 1^n \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all identities $\mathsf{id}$ queried by $\mathcal{A}$ satisfy $\mathsf{id} \neq \mathsf{id}^*$.

**Definition 2.7** (Weakly Compact Blind IBE [BLSV18])**.** An IBE scheme $\mathsf{IBE} = (\mathsf{Gen}, \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a weakly compact blind IBE if

1. **Public Parameters:** The $\mathsf{Gen}$ algorithm takes as input the security parameter $1^\lambda$ and the number of identities $1^T$ and generates the public parameter $\mathsf{pp}$, which is used in $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ algorithms.

2. **Weak Compactness:** Size of the master public key is $|\mathsf{mpk}| = O(T^{1-\epsilon}\mathsf{poly}(\lambda))$ where $T = 2^n$ is the number of identities.

3. **Encryption Decomposition:** The encryption algorithm $\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m; r)$ can be decomposed as $\mathsf{Enc}_1(\mathsf{pp}; r) \,||\, \mathsf{Enc}_2(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}, m; r)$.

4. **Blindness:** For any stateful PPT adversary $\mathcal{A}$ and polynomial $\mathsf{poly}(\cdot)$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$ and $n = \mathsf{poly}(\lambda)$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{pp},\mathsf{msk},\cdot)}(\mathsf{ct}_1, \mathsf{ct}_2) = b : \begin{array}{c} 1^n \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, 1^n); (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}) \\ b \leftarrow \{0,1\}; \mathsf{id}^* \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{pp},\mathsf{msk},\cdot)}(1^\lambda, n, \mathsf{mpk}) \\ m \leftarrow \mathcal{M}; (\mathsf{ct}_1, \mathsf{ct}_2) = \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{id}^*, m) \\ \text{if } b = 1, \mathsf{ct}_2 \leftarrow \{0,1\}^{|\mathsf{ct}_2|} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

   where $\mathcal{A}$ can query the oracle $\mathsf{KeyGen}$ on $\mathsf{id}^*$.

**Theorem 2.8** ([BLSV18])**.** Assuming the hardness of $\mathsf{CDH}$, there exists weakly compact blind IBE schemes.

## 2.5 Hierarchical Identity-Based Encryption

An unbounded Hierarchical Identity-Based Encryption (HIBE) scheme is a generalisation of IBE for the set of identity spaces $\mathcal{I} = \{0,1\}^*$ and message spaces $\mathcal{M}$ that consists of five polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ with the following syntactical changes and additions:

$\mathsf{Setup}(1^\lambda)$ : The setup algorithm takes only the security paramater as input.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}), \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$ : The identity could be any bit string, and the syntax for $\mathsf{KeyGen}$ and $\mathsf{Enc}$ is identical to that in IBE.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$ : The syntax for $\mathsf{Dec}$ is identical to that in IBE.

$\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}') \to \mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'}$. On input a secret key $\mathsf{sk}_{\mathsf{id}}$ and an identity $\mathsf{id}' \in \{0,1\}^*$, the delegate algorithm outputs a secret key $\mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'}$.

**Correctness.** We say an HIBE scheme $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ satisfies correctness if for all $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, any non-empty sequence of $\ell$ identities $\mathsf{id}_i \in \{0,1\}^{n_i}$ for $\ell \geq 1, i \in [\ell]$ where $n_i \geq 1$, message $m \in \mathcal{M}$, $\mathsf{sk}_1 \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}_1)$, and $\mathsf{sk}_{i+1} \leftarrow \mathsf{Delegate}(\mathsf{sk}_i, \mathsf{id}_{i+1})$ for $i \in [\ell - 1]$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}_1 \,||\, \ldots \,||\, \mathsf{id}_\ell, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{ct}) = m$.

**Definition 2.9** (Adaptive security)**.** We say an HIBE scheme $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is fully/adaptive secure if for any stateful PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{O(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}^*) \leftarrow \mathcal{A}^{O(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the oracle $O(\mathsf{msk}, \cdot)$ is a *stateful* oracle initialized with parameter $t := 1$, and takes as input a tuple $(\mathsf{id}, \mathsf{ind}, \mathsf{mode}) \in \{0,1\}^* \times \mathbb{N} \times \{\mathsf{StoreKey}, \mathsf{OutputKey}, \mathsf{DelegateKey}\}$, and answers each query as follows:

- If $\mathsf{mode} = \mathsf{StoreKey}$, then the challenger generates $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, stores $(t, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ and replies with $(t, \perp)$. It also updates $t := t + 1$.
- If $\mathsf{mode} = \mathsf{DelegateKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}', \mathsf{sk}_{\mathsf{id}'})$ for some identity $\mathsf{id}'$. If no such tuple exists, it outputs $\perp$. Otherwise, it generates $\mathsf{sk}_{\mathsf{id}' \,||\, \mathsf{id}} \leftarrow \mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}'}, \mathsf{id})$, stores $(t, \mathsf{id}' \,||\, \mathsf{id}, \mathsf{sk}_{\mathsf{id}' \,||\, \mathsf{id}})$ and replies with $(t, \perp)$. It also updates $t := t+1$.
- If $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})$. If no such tuple exists or if $\mathsf{id}$ is a prefix of $\mathsf{id}^*$ (that is, $\mathsf{id} \in \mathsf{prefix}(\mathsf{id}^*)$), it outputs $\perp$. Otherwise, it replies with $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$.

Note that $\mathcal{A}$ must also not have received any queries for any prefixes of $\mathsf{id}^*$ in the pre-challenge-query phase.

**Definition 2.10** (Anonymity)**.** We say an HIBE scheme $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is anonymous and fully/adaptive secure if for any stateful PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{O(\mathsf{msk},\cdot)}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}_0^*, \mathsf{id}_1^*) \leftarrow \mathcal{A}^{O(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}_b^*, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $|\mathsf{id}_0^*| = |\mathsf{id}_1^*|$, and the oracle $O(\mathsf{msk}, \cdot)$ is same as in Definition 2.9 except if $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})$. If no such tuple exists or if $\mathsf{id}$ is a prefix of $\mathsf{id}_0^*$ or $\mathsf{id}_1^*$ (that is, $\mathsf{id} \in \mathsf{prefix}(\mathsf{id}_0^*) \cup \mathsf{prefix}(\mathsf{id}_1^*)$), it outputs $\perp$. Otherwise, it replies with $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$.

# 3 Adaptively Secure Unbounded HIBE

In this section, we present an adaptively secure unbounded HIBE scheme. As discussed earlier, our construction is identical to the selectively secure HIBE scheme by [DG17a]. We start by describing the construction (mostly) verbatim from [DG17a]. Any reader familiar with their construction can safely skip Section 3.1 and move to Section 3.2.

## 3.1 [DG17a] HIBE

The construction relies the following ingredients.

- $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme.
- $\mathsf{SS} = (\mathsf{SS.Gen}, \mathsf{SS.Setup}, \mathsf{SS.Sign}, \mathsf{SS.Enc}, \mathsf{SS.Dec})$ be an OTSE scheme.
- $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ be a garbling scheme.
- $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval}, \mathsf{PRF.Delegate})$ be a delegatable PRF scheme.

For simplicity, we will assume that the verification keys $\mathsf{ss.vk}$ of the OTSE scheme and public keys $\mathsf{pke.pk}$ of the PKE scheme are of length $\lambda$. We also assume that the setup algorithms of both the OTSE and the PKE scheme require $\lambda$ random bits. As a consequence of this, we require a PRF with output length $2\lambda$ and denote $\mathsf{PRF.Eval}(s, x) = \mathsf{PRF.Eval}_1(s, x) \,||\, \mathsf{PRF.Eval}_2(s, x)$ where $|\mathsf{PRF.Eval}_i(s, x)| = \lambda$. First, we define the $\mathsf{Node}$ function that will be used in the $\mathsf{KeyGen}$ and $\mathsf{Delegate}$ functions.

$\mathsf{Node}(\mathsf{pp}, v, s)$ : The function takes as input public parameters $\mathsf{pp}$, a node identifier $v \in \{0,1\}^*$ and a PRF key $s$. It first computes a pair of verification and signing key $(\mathsf{ss.vk}_v, \mathsf{ss.sk}_v) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}_1(s, v))$. Similarly, it computes $\mathsf{ss.vk}_{v\|0}$ and $\mathsf{ss.vk}_{v\|1}$ and generates $(\mathsf{pke.pk}_v, \mathsf{pke.sk}_v) \leftarrow \mathsf{PKE.Setup}(1^\lambda; \mathsf{PRF.Eval}_2(s, v))$. Note that if $s$ is a delegated key, we have to truncate $v$ accordingly to achieve correctness.

It sets $x_v = \mathsf{ss.vk}_{v\|0} \,||\, \mathsf{ss.vk}_{v\|1} \,||\, \mathsf{pke.pk}_v$ and computes $\mathsf{ss}.\sigma_v \leftarrow \mathsf{SS.Sign}(\mathsf{pp}, \mathsf{ss.sk}_v, x_v)$. It generates a delegate key $s_v = \mathsf{PRF.Delegate}(s, v)$. It returns $(\mathsf{ss.vk}_v, x_v, \mathsf{ss}.\sigma_v, \mathsf{pke.sk}_v, s_v)$.

$\mathsf{Setup}(1^\lambda)$ : The setup algorithm takes as input a security parameter $1^\lambda$. It generates a PRF key $s \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^{2\lambda})$, $\mathsf{pp} \leftarrow \mathsf{SS.Gen}(1^\lambda, 1^{3\lambda})$ and computes $(\mathsf{ss.vk}_\epsilon, \cdot, \cdot, \cdot, \cdot) \leftarrow \mathsf{Node}(\mathsf{pp}, \epsilon, s)$.

It sets $\mathsf{mpk} = (\mathsf{pp}, \mathsf{ss.vk}_\epsilon)$ and $\mathsf{msk} = s$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ : Let $\mathsf{msk} = s$ be a PRF key and $\mathsf{id}$ be an identity such that $n = |\mathsf{id}|$. For $j = 0$ to $n$, it computes $(\cdot, x_{\mathsf{id}[1,j]}, \sigma_{\mathsf{id}[1,j]}, \mathsf{sk}_{\mathsf{id}[1,j]}, s_{\mathsf{id}[1,j]}) = \mathsf{Node}(\mathsf{pp}, \mathsf{id}[1,j], s)$. It outputs $\mathsf{sk}_{\mathsf{id}} := (\{(\sigma_v, x_v)\}_v, \mathsf{pke.sk}_{\mathsf{id}}$[3], $s_{\mathsf{id}})$ where $v \in \{\mathsf{id}[1,j]\}_{j\in[n]_0}$.

$\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$ : Let $\mathsf{sk}_{\mathsf{id}} = (\{(\sigma_{\mathsf{id}[1,j]}, x_{\mathsf{id}[1,j]})\}_{j\in[n]_0}, \mathsf{sk}_{\mathsf{id}}, s_{\mathsf{id}})$ be an HIBE key and $\mathsf{id}'$ be an identity such that $n' = |\mathsf{id}'|$ and $n = |\mathsf{id}|$. For $j = 1$ to $n'$, it computes

$$(\cdot, x_{\mathsf{id}\,||\,\mathsf{id}'[1,j]}, \sigma_{\mathsf{id}\,||\,\mathsf{id}'[1,j]}, \mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'[1,j]}, s_{\mathsf{id}\,||\,\mathsf{id}'[1,j]}) = \mathsf{Node}(\mathsf{pp}, \mathsf{id}'[1,j], s_{\mathsf{id}})$$

It then computes $s_{\mathsf{id}\,||\,\mathsf{id}'} = \mathsf{PRF.Delegate}(s_{\mathsf{id}}, \mathsf{id}')$ and outputs $(\{(\sigma_v, x_v)\}, \mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'}, s_{\mathsf{id}\,||\,\mathsf{id}'})$ where $v \in \{(\mathsf{id}\,||\,\mathsf{id}')[1,j]\}_{j\in[n+n']_0}$.

$\mathsf{Enc}(\mathsf{mpk}, m, \mathsf{id})$ : The encryption algorithm takes as input a master public key $\mathsf{mpk} = (\mathsf{pp}, \mathsf{ss.vk}_\epsilon)$, a message $m$ and an identity $\mathsf{id}$. Let

- $T_m(\mathsf{pk})$ be a circuit that computes $\mathsf{PKE.Enc}(\mathsf{pk}, m)$.
- $Q_{\beta\in\{0,1,2\},\{z_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}}(\mathsf{vk})$ be a circuit that computes $\{\mathsf{SS.Enc}(\mathsf{pp}, (\mathsf{vk}, \beta\cdot\lambda+i, b), z_{i,b})\}_{i\in[\lambda],b\in\{0,1\}}$.

Let $|\mathsf{id}| = n$. The algorithm starts by generating $(\mathcal{T}, \left\{\mathsf{lab}_{i,j}^T\right\}) \leftarrow \mathsf{GC.Garble}(1^\lambda, T_m)$. Then, it generates $(\mathcal{Q}^{(n)}, \mathsf{lab}_{i,j}^{(n)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, Q_{2,\left\{\mathsf{lab}_{i,j}^T\right\}})$.

For $k = n-1$ to 0, it computes $(\mathcal{Q}^{(k)}, \mathsf{lab}_{i,j}^{(k)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, Q_{\mathsf{id}[k+1],\left\{\mathsf{lab}_{i,j}^{(k+1)}\right\}})$. Finally, it returns $\left(\left\{\mathsf{lab}_{i,\mathsf{ss.vk}_\epsilon[i]}^{(0)}\right\}_{i\in[\lambda]}, \left\{\mathcal{Q}^{(k)}\right\}_{k\in[n]_0}, \mathcal{T}\right)$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$ : The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{id}} = (\{(\sigma_k, x_k)\}_{k\in[n]_0}, \mathsf{sk}_{\mathsf{id}}, s_{\mathsf{id}})$ where $n = |\mathsf{id}|$ is the length of $\mathsf{id}$ and a ciphertext $\mathsf{ct} = (\left\{\mathsf{lab}_i^{(0)}\right\}_{i\in[\lambda]}, \left\{\mathcal{Q}^{(k)}\right\}_{k\in[n]_0}, \mathcal{T})$. It sets $\mathsf{vk}^{(0)} := \mathsf{ss.vk}_\epsilon$ and for $k = 0$ to $n$,

1. it computes $\{\mathsf{ss.ct}_{i,j}\} \leftarrow \mathsf{GC.Eval}(\mathcal{Q}^{(k)}, \{\mathsf{lab}_i^{(k)}\}_{i\in[\lambda]})$.
2. if $(k \neq n)$, it sets $\mathsf{vk}^{(k+1)} := x_k[\mathsf{id}[k+1]]$ where $x_k$ is a 3 block of $\lambda$ length strings. Else, it sets $\mathsf{vk}^{(n+1)} := x_n[2]$.
3. it computes $\mathsf{lab}_i^{(k+1)} \leftarrow \mathsf{SS.Dec}(\mathsf{pp}, (\mathsf{vk}^{(k)}, x_k, \sigma_k), \mathsf{ss.ct}_{i,\mathsf{vk}^{(k+1)}[i]})$.

It computes $\mathsf{pke.ct} \leftarrow \mathsf{GC.Eval}(\mathcal{T}, \{\mathsf{lab}_i^{(n+1)}\}_{i\in[\lambda]})$ and sets $m \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pke.ct})$. It return $m$.

It important to note that both $\mathsf{KeyGen}$ and $\mathsf{Delegate}$ algorithms are *deterministic*. Also, from the correctness of the delegation of the PRF, we can easily show that $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}\,||\,\mathsf{id}') = \mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$.

[3]It is not important to keep $\mathsf{pke.sk}_{\mathsf{id}}$ as a part of $\mathsf{sk}_{\mathsf{id}}$ because $s_{\mathsf{id}}$ can be used to compute $\mathsf{PRF.Eval}(s, \mathsf{id})$ by evaluating $\mathsf{PRF.Eval}(s_{\mathsf{id}}, \epsilon)$. However, we include it for clarity. For the syntax of the PRF, refer to Appendix A

**Correctness.** Let $\mathsf{ct} = (\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}_\epsilon[i]}\}_{i\in[\lambda]}, \{\mathcal{Q}^{(k)}\}_{k\in[n-1]_0}, \mathcal{T})$ be an encryption of message $m$ for the identity $\mathsf{id}$. The decryption process starts by evaluating the garbled circuit $\mathcal{Q}^{(0)}$ on $\{\mathsf{lab}^{(0)}_i\}_{i\in[\lambda]}$ where $\mathsf{lab}^{(0)}_i = \mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}_\epsilon[i]}$. Therefore, from the correctness of the garbling scheme we have $\mathsf{ss.ct}_{i,j} = \mathsf{SS.Enc}(\mathsf{pp}, (\mathsf{ss.vk}_\epsilon, \mathsf{id}[1]\cdot\lambda + i, j), \mathsf{lab}^{(1)}_{i,j})$ for all $i \in [\lambda], j \in \{0,1\}$. Next, the algorithm sets $\mathsf{vk}^{(1)} := x_0[\mathsf{id}[1]]$ where $x_0 = \mathsf{ss.vk}_0 \,||\, \mathsf{ss.vk}_1 \,||\, \mathsf{pke.pk}_\epsilon$. Therefore, $\mathsf{vk}^{(1)} = \mathsf{ss.vk}_{\mathsf{id}[1]}$. Next, it computes $\mathsf{lab}^{(1)}_i \leftarrow \mathsf{SS.Dec}(\mathsf{pp}, (\mathsf{vk}^{(0)}, x_0, \sigma_0), \mathsf{ss.ct}_{i,\mathsf{vk}^{(1)}[i]})$. Now, from the correctness of the OTSE scheme, we get $\mathsf{lab}^{(1)}_i = \mathsf{lab}^{(1)}_{i,\mathsf{ss.vk}_{\mathsf{id}[1]}[i]}$. Repeating the same argument, we will obtain $\mathsf{lab}^{(n+1)}_i = \mathsf{lab}^{T}_{i,\mathsf{pke.pk}_{\mathsf{id}}[i]}$. Again, using the correctness of the garbling scheme, we obtain $\mathsf{pke.ct} = \mathsf{PKE.Enc}(\mathsf{pke.pk}_{\mathsf{id}}, m)$ by evaluating $\mathsf{pke.ct} \leftarrow \mathsf{GC.Eval}(\mathcal{T}, \{\mathsf{lab}^{(n+1)}_i\}_{i\in[\lambda]})$. Finally, from the PKE scheme's correctness, we get $m$ by decrypting $\mathsf{pke.ct}$ using $\mathsf{pke.sk}_{\mathsf{id}}$.

**Efficiency.** The master public key $\mathsf{mpk}$ is a pair of public parameters and verification key of the OTSE scheme for a message length of $3\lambda$. Therefore, the size of the master public key is $|\mathsf{mpk}| = \mathsf{poly}(\lambda)$.

The master secret key is a PRF key, therefore, $|\mathsf{msk}| = \mathsf{poly}(\lambda)$. A secret key (as well as delegated secret key) for an identity $\mathsf{id}$ of length $n$ consists of $n$ pairs of signature-message (where the length of the message is $3\lambda$), a secret key of the PKE scheme and a delegated PRF key. Therefore $|\mathsf{sk}_{\mathsf{id}}| = n \cdot \mathsf{poly}(\lambda)$.

A ciphertext $\mathsf{ct}$ associated with a message $m$ and an identity $\mathsf{id}$ of length $n$ consists of a garbled circuit for $T_m(\cdot)$ whose size is $|m| \cdot \mathsf{poly}(\lambda)$, $n$ garbled circuits of $Q_{\beta,\{z_{i,b}\}}(\cdot)$ of size $n \cdot \mathsf{poly}(\lambda)$ and $\lambda$ many labels for a garbled circuit. Therefore, $|\mathsf{ct}| = (|m| + n) \cdot \mathsf{poly}(\lambda)$.

## 3.2 Proof of Adaptive Security

**Theorem 3.1.** Assuming $\mathsf{PKE}$ is a CPA secure PKE scheme, $\mathsf{SS}$ is a secure OTSE scheme, $\mathsf{GC}$ is a secure garbling scheme and $\mathsf{PRF}$ is an adaptively secure PRF, then the above construction is adaptively secure.

*Proof.* The proof of adaptive security proceeds via the following sequence of hybrids.

- Hybrid $H_0$: This is the original adaptively secure HIBE game.
- Hybrid $H_1$: This is same as $H_0$ except that when $\mathcal{A}$ queries the oracle $O(\mathsf{msk}, \cdot)$ on $(\mathsf{id}, \mathsf{ind}, \mathsf{mode})$, the challenger does the following.
  - If $\mathsf{mode} = \mathsf{StoreKey}$, then the challenger does not generate the secret key but stores $(t, \mathsf{id}, \perp)$ and replies with $(t, \perp)$. It also updates $t := t + 1$.
  - If $\mathsf{mode} = \mathsf{DelegateKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}', \cdot)$ for some identity $\mathsf{id}'$. If no such tuple exists, it outputs $\perp$. Otherwise, it does not generate the delegated secret key but stores $(t, \mathsf{id}' \,||\, \mathsf{id}, \perp)$ and replies with $(t, \perp)$. It also updates $t := t + 1$.
  - If $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}, \cdot)$. If no such tuple exists or if $\mathsf{id}$ is a prefix of $\mathsf{id}^*$ (that is, $\mathsf{id} \in \mathsf{prefix}(\mathsf{id}^*)$), it outputs $\perp$. Otherwise, if $(\mathsf{ind}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ exists, it replies with $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$. Else, it will compute $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, stores it and outputs $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$.
- For $k = 0$ to $n$ (where $n = |\mathsf{id}^*|$),
  - Hybrid $H_{2,k}$: In this hybrid, $\left(\mathcal{Q}^{(h)}, \left\{\mathsf{lab}^{(h)}_{i,\mathsf{ss.vk}_{\mathsf{id}^*[1,h]}[i]}\right\}\right)$ are generated by using simulation algorithm on $\left\{\mathsf{ss.ct}^{(h)}_{i,j}\right\} := Q_{\mathsf{id}[h+1],\left\{\mathsf{lab}^{(h+1)}_{i,j}\right\}}(\mathsf{ss.vk}_{\mathsf{id}^*[1,h]})$ $\left(\text{in the case when } h = n, \left\{\mathsf{ss.ct}^{(n)}_{i,j}\right\} := Q_{2,\left\{\mathsf{lab}^{T}_{i,j}\right\}}(\mathsf{ss.vk}_{\mathsf{id}^*})\right)$, for all $h \in [k]_0$.

- Hybrid $H_3$: In this hybrid, $\left(\mathcal{T}, \left\{\mathsf{lab}^{T}_{i,\mathsf{pke.pk}_{\mathsf{id}^*}[i]}\right\}\right)$ are generated by using the simulation algorithm on $T_m(\mathsf{pke.pk}_{\mathsf{id}^*}) = \mathsf{PKE.Enc}(\mathsf{pke.pk}_{\mathsf{id}^*}, m)$.
- Hybrid $H_4$: In this hybrid, rather than using $\mathsf{PRF.Eval}_2(s; \mathsf{id}^*)$ to generate $(\mathsf{pke.pk}_{\mathsf{id}^*}, \mathsf{pke.sk}_{\mathsf{id}^*})$, it uses truly random coins in $\mathsf{PKE.Setup}$ algorithm.

**Analysis:** Let $p_{\mathcal{A},H}$ denote probability of $\mathcal{A}$ outputting $b' = b$ in Hybrid $H$. We will show that this probability is almost the same in every game.

**Lemma 3.2.** For all PPT adversaries $\mathcal{A}$, $p_{\mathcal{A},H_0} = p_{\mathcal{A},H_1}$.

*Proof.* This follows directly from the fact that $\mathsf{KeyGen}$ and $\mathsf{Delegate}$ algorithms are deterministic and $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id} \,||\, \mathsf{id}') = \mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}} \,||\, \mathsf{id}')$. □

We use the notation $H_{2,-1}$ to denote hybrid $H_1$.

**Lemma 3.3.** Assuming that $\mathsf{GC}$ is a secure garbling scheme, $\mathsf{PRF}$ is a secure delegatable PRF scheme and $\mathsf{SS}$ is a secure OTSE scheme, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n]_0$, $|p_{\mathcal{A},H_{2,k}} - p_{\mathcal{A},H_{2,k-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* For the case $k = 0$, observe that only $\left\{\mathsf{lab}^{(0)}_{i,\mathsf{ss.vk}_\epsilon[i]}\right\}$ is present in the challenge ciphertext. So, we can immediately replace $\mathsf{GC.Garble}$ with $\mathsf{GC.Sim}$ to generate $\mathcal{Q}^{(0)}$ and the simulated labels and use them in the ciphertext. Since, $\mathsf{GC}$ is a secure garbling scheme, $|p_{\mathcal{A},H_{2,0}} - p_{\mathcal{A},H_{2,-1}}| = \mathsf{negl}(\lambda)$.

For the case when $k > 0$, we will use nested hybrids technique along with a pebbling-style argument as follows. For a particular $k$ and the adversary $\mathcal{A}$, let $|p_{\mathcal{A},H_{2,k}} - p_{\mathcal{A},H_{2,k-1}}| = \epsilon$. Since $\mathcal{A}$ is a PPT adversary, we can bound the number of (pre-challenge) key queries it makes — let us call it $Q = Q(\lambda)$. We now consider a modified adversary $\mathcal{A}'$ that uses $\mathcal{A}$ to distinguish hybrid $H_{2,k}$ from $H_{2,k-1}$. At a high level, $\mathcal{A}'$ will attempt to guess the index $q$ such that the first $\mathsf{Node}(\mathsf{pp}, \mathsf{id}^*[1,k], \cdot)$ computation is performed either in the $q$-th $\mathsf{OutputKey}$ query or during the generation of challenge ciphertext. The complete description of $\mathcal{A}'$ is as follows:

1. $\mathcal{A}'$ starts by randomly choosing a number $q$ from $[Q+1]$.
2. It then interacts with the challenger and $\mathcal{A}$ by relaying messages between them.
3. Until the $q$-th query made by $\mathcal{A}$, $\mathcal{A}'$ maintains a set $S$ containing all identifiers $v \in \{0,1\}^k$ of length $k$ that the challenger would have visited using $\mathsf{Node}(\mathsf{pp}, \cdot, \cdot)$. It stores the $q$-th $v$ as $\mathsf{guess}$ and checks whether $\mathsf{guess} \in S$. If so, $\mathcal{A}'$ aborts. Otherwise, it continues the game.
4. After receiving the challenge ciphertext, it checks if $\mathsf{guess}$ is a prefix of $\mathsf{id}^*$. If not, it aborts the game. Otherwise, it continues the game until the end by simulating $\mathcal{A}$ and returning $\mathcal{A}$'s response $b'$ as its final response.

**Claim 3.4.** $p_{\mathcal{A}',H_{2,k}} \geq \dfrac{p_{\mathcal{A},H_{2,k}}}{Q+1}$ and $p_{\mathcal{A}',H_{2,k-1}} \geq \dfrac{p_{\mathcal{A},H_{2,k-1}}}{Q+1}$.

*Proof.* We will prove the first equality, as the second equality follows analogously. Since, $\mathcal{A}'$ aborts the game if $\mathsf{guess}$ is not the correct prefix of $\mathsf{id}^*$, therefore, the probability of $\mathcal{A}'$ winning in the hybrid $H_{2,k}$ is $p_{\mathcal{A}',H_{2,k}} \geq p_{\mathcal{A},H_{2,k}} \cdot \Pr\left[\mathsf{guess} \preceq \mathsf{id}^*\right]$.

Recall that $\mathcal{A}$ makes $Q$ key queries. Therefore, at most $Q+1$ distinct nodes with identifiers of length $k$ are visited by the invocation of the $\mathsf{Node}(\mathsf{pp}, \cdot, \cdot)$ algorithm until the challenge ciphertext is generated. These visits occur either when the challenger responds to the key queries of the adversary or while generating the challenge ciphertext.

At the beginning of the game, $\mathcal{A}'$ randomly selects an index $q$ from $[1, Q+1]$. The goal of $\mathcal{A}'$ is to guess the first query index where the corresponding identifier $v$ matches the prefix of the challenge identity

$\mathsf{id}^*[1:k]$. Since there are at most $Q+1$ such indices, the probability that $\mathcal{A}'$ correctly selects the first such index is at least $1/(Q+1)$. Thus, we get that $p_{\mathcal{A}',H_{2,k}} \geq \frac{p_{\mathcal{A},H_{2,k}}}{Q+1}$. □

Our goal is to show that $|p_{\mathcal{A}',H_{2,k}} - p_{\mathcal{A}',H_{2,k-1}}| = \mathsf{negl}(\lambda)$. We achieve this by considering additional internal hybrids as follows. Recall that $\mathsf{guess}$ is the $q$-th identifier $v$ of length $k$ that the challenger would have visited through $\mathsf{Node}(\mathsf{pp},\cdot,\cdot)$.

- Hybrid $H_{2,k-1,1}$: In this hybrid, $\mathcal{A}'$ sends $q$ to the challenger at the beginning of the game. The challenger, rather than using $\mathsf{PRF.Eval}_1(s,\mathsf{guess})$ to generate $(\mathsf{ss.vk}_{\mathsf{guess}},\mathsf{ss.sk}_{\mathsf{guess}})$, it uses a truly random coins in $\mathsf{SS.Setup}$ algorithm.
- Hybrid $H_{2,k-1,2}$: This is similar to the previous hybrid, except that the labels $\mathsf{lab}^{(k)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}$ are replaced with $\mathbf{0}$ to generate $\mathsf{ss.ct}^{(k-1)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}$ for all $i \in [\lambda]$.
- Hybrid $H_{2,k-1,3}$: In this hybrid, $\mathsf{ss.vk}_{\mathsf{guess}},\mathsf{ss.sk}_{\mathsf{guess}}$ are generated using the $\mathsf{PRF.Eval}_1(s,\mathsf{guess})$ algorithm in $\mathsf{SS.Setup}$ algorithm.

**Claim 3.5.** Assuming that $\mathsf{PRF}$ is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{2,k-1,1}} - p_{\mathcal{A}',H_{2,k-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* This directly follows from the PRF security. Observe that the PRF is used only during the oracle queries (while executing $\mathsf{KeyGen}$) and while generating the challenge ciphertext (to generate verification keys $\mathsf{ss.vk}_{\mathsf{id}^*[1,i]}, \forall i \in [n]_0$). From Definition 2.1, the PRF adversary is allowed to query for evaluation of the PRF at any node (except at $x^*$) whereas delegation queries are not allowed along the path from root to the target point $x^*$. In the current hybrid, the target point $x^*$ is $\mathsf{guess}$ and if $\mathcal{A}'$ does not abort (i.e., its guess is correct which would imply that $\mathsf{guess} \preceq \mathsf{id}^*$), then $\mathsf{KeyGen}$ algorithm would require delegated PRF keys for nodes that are not in the path from root to $\mathsf{guess}$. Therefore, an adversary $\mathcal{A}'$ that can distinguish these two hybrids can be used to build an adversary $\mathcal{B}$ that break the PRF security by faithfully simulating the HIBE game.

1. The reduction $\mathcal{B}$ obtains $q$ from $\mathcal{A}'$ and simulates the HIBE game without generating the PRF key $s$.
2. $\mathcal{B}$ responds to the queries from $\mathcal{A}'$ using the $\mathsf{Delegate}(s,\cdot)$ and $\mathsf{Eval}(s,\cdot)$ oracle provided by the PRF challenger. In the $q$-th index of query, it will send $\mathsf{guess}$ to the PRF challenger and receives $r$ – which is either a truly random string or $\mathsf{Eval}(s,\mathsf{guess})$. It uses $r$ in the computation of $\mathsf{Node}(\mathsf{pp},\mathsf{guess},\cdot)$ to generate $(\mathsf{ss.vk}_{\mathsf{guess}},\mathsf{ss.sk}_{\mathsf{guess}})$ and continues to simulate the game.
3. In the challenge phase, we know that if $\mathcal{A}'$ does not abort, then $\mathsf{guess}$ will be a prefix of $\mathsf{id}^*$, thus $\mathcal{B}$ simulates the rest of the HIBE game honestly and returns $\mathcal{A}'$ response.

Therefore, any if $\mathcal{A}'$ has non-negligible advantage between distinguishing these hybrids, then $\mathcal{B}$ can break adpative security of delegatable PRFs. This concludes the proof of the claim. □

**Claim 3.6.** Assuming that $\mathsf{SS}$ is a secure OTSE scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{2,k-1,2}} - p_{\mathcal{A}',H_{2,k-1,1}}| = \mathsf{negl}(\lambda)$.

*Proof.* Recall that in this hybrid, instead of encrypting $\mathsf{lab}^{(k)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}$ using $\mathsf{ss.vk}_{\mathsf{id}^*[1,k-1]}$ (which is a verification key at the $(k-1)^{th}$ level of the tree), $\mathbf{0}$ are being encrypted to generate $\mathsf{ss.ct}^{(k-1)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}}$. We will show that an adversary $\mathcal{A}'$ that can distinguish between the two hybrids can be used to break the OTSE security.

1. The reduction $\mathcal{B}$ obtains $q$ from $\mathcal{A}'$.

2. The OTSE challenger sends $\mathsf{pp}$ to $\mathcal{B}$. $\mathcal{B}$ runs the setup algorithm to obtain $(\mathsf{mpk}, \mathsf{msk})$ and sends $\mathsf{mpk}$ to $\mathcal{A}'$.

3. $\mathcal{B}$ responds to the queries from $\mathcal{A}'$ normally until the $q$-th $\mathsf{ss.vk}$ has to be generated for some $\mathsf{id}' \in \{0,1\}^k$. It generates $(\mathsf{ss.vk}_{\mathsf{id}' \,||\, d}, \mathsf{ss.sk}_{\mathsf{id}' \,||\, d}) \leftarrow \mathsf{SS.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}_1(s, \mathsf{id}' \,||\, d))$ for $d \in \{0,1\}$ and $(\mathsf{pke.pk}_{\mathsf{id}'}, \mathsf{pke.sk}_{\mathsf{id}'}) \leftarrow \mathsf{PKE.Setup}(1^\lambda; \mathsf{PRF.Eval}_2(s, \mathsf{id}'))$ and sends $x_{\mathsf{id}'} := (\mathsf{ss.vk}_{\mathsf{id}' \,||\, 0} \,||\, \mathsf{ss.vk}_{\mathsf{id}' \,||\, 1} \,||\, \mathsf{pke.pk}_{\mathsf{id}'})$ to the OTSE challenger. The challenger returns $\mathsf{vk}^*$ along with $\sigma^*$. $\mathcal{B}$ sets $\mathsf{ss.vk}_{\mathsf{id}'} := \mathsf{vk}^*$ and $\sigma_{\mathsf{id}'} = \sigma^*$.

4. In the challenge phase, we know that if $\mathcal{A}'$ does not abort, then $\mathsf{guess}$ will be a prefix of $\mathsf{id}^*$, thus $\mathcal{B}$ can generate the challenge ciphertext as follows. For simulating $(\mathcal{Q}^{(k-1)}, \left\{\mathsf{lab}^{(k-1)}_{i,\mathsf{ss.vk}_{\mathsf{id}^*[1:k]}[i]}\right\})$, it sends $(\mathsf{lab}^{(k)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}, \mathbf{0})$ to obtain $\mathsf{ss.ct}^{(k-1)}_{i,1-\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}$ for each $i$. The rest of the game can be simulated by $\mathcal{B}$.

Therefore, any if $\mathcal{A}'$ has non-negligible advantage between distinguishing these hybrids, then $\mathcal{B}$ can break OTSE security. This concludes the proof of the claim. □

**Claim 3.7.** Assuming that $\mathsf{PRF}$ is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{2,k-1,3}} - p_{\mathcal{A}',H_{2,k-1,2}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Claim 3.5. □

**Claim 3.8.** Assuming that $\mathsf{GC}$ is a secure garbling scheme, there exits a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, |p_{\mathcal{A}',H_{2,k}} - p_{\mathcal{A}',H_{2,k-1,2}}| = \mathsf{negl}(\lambda)$.

*Proof.* Observe that only $\left\{\mathsf{lab}^{(k)}_{i,\mathsf{ss.vk}_{\mathsf{id}^*[1,k]}[i]}\right\}$ are used in $H_{2,k-1,3}$. Therefore, using the security of the garbling scheme, we get $|p_{\mathcal{A}',H_{2,k}} - p_{\mathcal{A}',H_{2,k-1,2}}| = \mathsf{negl}(\lambda)$. □

Combining the above equalities using triangular inequality, we have $|p_{\mathcal{A}',H_{2,k}} - p_{\mathcal{A}',H_{2,k-1}}| = \mathsf{negl}(\lambda)$. Using the above claims and the triangular equality, we get $\epsilon/(Q+1) = \mathsf{negl}(\lambda)$. This implies that $\epsilon = \mathsf{negl}(\lambda)$ because $Q$ is a polynomial in $\lambda$. □

**Lemma 3.9.** Assuming that $\mathsf{GC}$ is a secure garbling scheme, $\mathsf{PRF}$ is a secure delegatable PRF scheme and $\mathsf{SS}$ is a secure OTSE scheme, for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_{2,n}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Lemma 3.3. □

**Lemma 3.10.** Assuming that $\mathsf{PRF}$ is a secure delegatable PRF scheme, for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n]_0$, $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof follows from a guessing argument and the security of the PRF scheme. □

**Lemma 3.11.** Assuming that $\mathsf{PKE}$ is a secure PKE scheme, for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},H_4} \leq 1/2 + \mathsf{negl}(\lambda)$.

*Proof.* The proof employs a similar guessing technique used in Claim 3.6. In this case, $\mathcal{B}$ is a adversary that tries to breaks the CPA security of the PKE scheme.

1. The PKE challenger sends $\mathsf{pk}^*$ to $\mathcal{B}$. $\mathcal{B}$ runs the setup algorithm to obtain $(\mathsf{mpk}, \mathsf{msk})$ and sends $\mathsf{mpk}$ to $\mathcal{A}$. Also, $\mathcal{B}$ randomly picks $i^* \leftarrow [Q+1]$.

2. $\mathcal{B}$ responds to the queries from $\mathcal{A}$ normally until the $(i^*)^{th}$ $\mathsf{pke.pk}$ has to be generated for some $\mathsf{id}' \in \{0,1\}^k$. Rather than using the PRF to generate the public key, it sets $\mathsf{pke.pk}_{\mathsf{id}'} := \mathsf{pk}^*$ and continues simulating the rest of the game till the challenge phase.

3. In the challenger phase, if $\mathsf{id}'$ is not equal to $\mathsf{id}^*$, then $\mathcal{B}$ will return a random bit as its response and abort the game. Else, it will generate the challenge ciphertext as follows. For simulating $(\mathcal{T}^{(k)}, \left\{\mathsf{lab}^{(T)}_{i,\mathsf{pke.pk}_{\mathsf{id}^*}}\right\})$, it sends $(m_0, m_1)$ to obtain $\mathsf{pke.ct}^* = \mathsf{PKE.Enc}(\mathsf{pke.pk}_{\mathsf{id}^*}, m_d)$ for some randomly chosen $d \in \{0,1\}$. The rest of the game can be simulated by $\mathcal{B}$.

Observe that the probability of the reduction guessing the correct index $i^*$ is at least $1/(Q+1)$. Therefore, $p_{\mathcal{A},H_4} \leq \frac{1}{2} + \frac{1}{\Pr[\mathsf{id}'=\mathsf{id}^*]}\left|p_{\mathcal{B},PKE} - \frac{1}{2}\right| = \frac{1}{2} + \mathsf{negl}(\lambda)$, where $p_{\mathcal{B},PKE}$ is the probability of $\mathcal{B}$ winning the PKE game. □

Using the above lemmas and triangular inequality, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. □

Using Theorem 2.4, Theorem 2.3 and Theorem 3.1, we have following corollary.

**Corollary 3.12.** Assuming the existence of selectively secure IBE, there exists adaptively secure unbounded HIBE.

# 4 Adaptively Secure Anonymous Unbounded HIBE from CDH

In this section, we will present an adaptively secure anonymous unbounded HIBE scheme. Our construction closely follows the selectively secure IBE scheme from [BLSV18], with the key difference being that we use a delegatable PRF to achieve delegation. We start by describing the construction (mostly) verbatim from [BLSV18].

## 4.1 [BLSV18] HIBE

The construction requires the following primitives.

- $\mathsf{IBE} = (\mathsf{IBE.Gen}, \mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ be a weakly blind IBE. The IBE scheme has the property that it can support $T = T(\lambda)$ identities with a master public key of size $S = S(\lambda)$ bits. By compactness, we choose $T = poly(\lambda)$ large enough so that $S < T/4$.
- $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ be a blind garbling scheme.
- $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval}, \mathsf{PRF.Delegate})$ be a delegatable PRF scheme.

For simplicity, we assume that the setup and the key generation algorithm of IBE scheme require $\lambda$ random bits.

$\mathsf{Gen}(1^\lambda)$ : The generation algorithm takes as input the security parameter $1^\lambda$ and computes $\mathsf{pp} \leftarrow \mathsf{IBE.Gen}(\lambda, 1^T)$.

$\mathsf{Setup}(\mathsf{pp})$ : The setup algorithm takes as input a public parameters $\mathsf{pp}$. It generates a PRF key $s \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^\lambda)$ and computes $(\mathsf{mpk}^{(\epsilon)}, \mathsf{msk}^{(\epsilon)}) \leftarrow \mathsf{IBE.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}(s, \epsilon))$.

It sets $\mathsf{mpk} := \mathsf{mpk}^{(\epsilon)}$ and $\mathsf{msk} := s$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$ : The key generation algorithm takes as input a public parameters $\mathsf{pp}$, a master secret key $\mathsf{msk} = s$ and an identity $\mathsf{id}$ such that $n = |\mathsf{id}|$. For $k = 0$ to $n$,

1. it sets $\beta := \mathsf{id}[1:k]$ and $\gamma := \mathsf{id}[1:k+1]$.
2. it computes $(\mathsf{mpk}^{(\beta)}, \mathsf{msk}^{(\beta)}) \leftarrow \mathsf{IBE.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}(s, \beta))$.
3. it computes $(\mathsf{mpk}^{(\gamma)}, \mathsf{msk}^{(\gamma)}) \leftarrow \mathsf{IBE.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}(s, \gamma))$.
4. if $k < n$, it generates $\mathsf{sk}_{k,i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp}, \mathsf{msk}^{(\beta)}, \mathsf{id}[k+1] \,||\, i \,||\, \mathsf{mpk}^{(\gamma)}[i])$ where $i \in [S]$. Else, it generates $\mathsf{sk}_n \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp}, \mathsf{msk}^{(\mathsf{id})}, \epsilon)$.

It generate $s_{\mathsf{id}} \leftarrow \mathsf{PRF.Delegate}(s, \mathsf{id})$. It finally returns $\Big( \Big\{ \mathsf{mpk}^{(\mathsf{id}[1:k])} \Big\}_{k \in [n]_0}, \{\mathsf{sk}_{k,i}\}_{k \in [n-1]_0, i \in [S]}, \mathsf{sk}_n, {}^{4} s_{\mathsf{id}} \Big)$.

$\mathsf{Delegate}(\mathsf{pp}, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$ : The delegation algorithm takes as input a public parameters $\mathsf{pp}$, an HIBE key $\mathsf{sk}_{\mathsf{id}} = (\Big\{ \mathsf{mpk}^{(\mathsf{id}[1:k])} \Big\}_{k \in [n]}, \{\mathsf{sk}_{k,i}\}_{k \in [n-1]_0, i \in [S]}, \mathsf{sk}_n, s_{\mathsf{id}})$ and an identity $\mathsf{id}'$ such that $n' = |\mathsf{id}'|$ and $n = |\mathsf{id}|$. For $j = 0$ to $n'$,

1. it sets $\beta := \mathsf{id} \,||\, \mathsf{id}'[1 : j+1]$ and $\gamma := \mathsf{id} \,||\, \mathsf{id}'[1 : j+1]$.
2. it computes $(\mathsf{mpk}^{(\beta)}, \mathsf{msk}^{(\beta)}) \leftarrow \mathsf{IBE.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}(s_{\mathsf{id}}, \mathsf{id}'[1 : j]))$.
3. it computes $(\mathsf{mpk}^{(\gamma)}, \mathsf{msk}^{(\gamma)}) \leftarrow \mathsf{IBE.Setup}(\mathsf{pp}; \mathsf{PRF.Eval}(s_{\mathsf{id}}, \mathsf{id}'[1 : j+1]))$.
4. if $j < n'$, it generates $\mathsf{sk}_{n+j,i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp}, \mathsf{msk}^{(\beta)}, \mathsf{id}'[j+1] \,||\, i \,||\, \mathsf{mpk}^{(\gamma)}[i])$ where $i \in [S]$. Else, it generates $\mathsf{sk}_{n+n'} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp}, \mathsf{msk}^{(\mathsf{id} \,||\, \mathsf{id}')}, \epsilon)$.

It generate $s_{\mathsf{id} \,||\, \mathsf{id}'} \leftarrow \mathsf{PRF.Delegate}(s_{\mathsf{id}}, \mathsf{id}')$. It finally returns the following set – $\Big( \Big\{ \mathsf{mpk}^{((\mathsf{id} \,||\, \mathsf{id}')[1:j])} \Big\}_{j \in [n+n']_0}, \{\mathsf{sk}_{j,i}\}_{j \in [n+n'-1]_0, i \in [S]}, \mathsf{sk}_{n+n'}, s_{\mathsf{id} \,||\, \mathsf{id}'} \Big)$.

$\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, m, \mathsf{id})$ : The encryption algorithm takes as input a public parameters $\mathsf{pp}$, a master public key $\mathsf{mpk} = \mathsf{mpk}^{(\epsilon)}$, a message $m$ and an identity $\mathsf{id}$. Let

- $T_{m,r}(\mathsf{mpk})$ be a circuit that computes $\mathsf{IBE.Enc}_2(\mathsf{pp}, \mathsf{mpk}, \epsilon, m; r)$.
- $Q_{\beta \in \{0,1\}, \{z_{i,b}, r_{i,b}\}_{i \in [\lambda], b \in \{0,1\}}}(\mathsf{mpk})$ be a circuit that computes $\{\mathsf{IBE.Enc}_2(\mathsf{pp}, \mathsf{mpk}, \beta \| i \| b, z_{i,b}; r_{i,b})\}_{i \in [\lambda], b \in \{0,1\}}$.

Let $n = |\mathsf{id}|$. The algorithm starts by generating $r$ and $\Big\{ r_{i,b}^{(k)} \Big\}$ uniformly at random, for all $i \in [S], b \in \{0,1\}, k \in [n-1]_0$. It then

- generates $(\mathcal{T}, \Big\{ \mathsf{lab}_{i,j}^{(n)} \Big\}) \leftarrow \mathsf{GC.Garble}(1^\lambda, T_{m,r})$ and $\mathsf{ct}_n \leftarrow \mathsf{IBE.Enc}_1(\mathsf{pp}; r)$.
- for $k = n-1$ to 0, it computes the set $(\mathcal{Q}^{(k)}, \Big\{ \mathsf{lab}_{i,j}^{(k)} \Big\}) \leftarrow \mathsf{GC.Garble}(1^\lambda, Q_{\mathsf{id}[k+1], \Big\{ \mathsf{lab}_{i,j}^{(k+1)}, r_{i,j}^{(k+1)} \Big\}})$ and $\mathsf{ct}_{i,j}^{(k)} \leftarrow \mathsf{IBE.Enc}_1(\mathsf{pp}; r_{i,j}^{(k)})$.

Finally, it returns

$$(\mathsf{ct}_0, \mathsf{ct}_1) = \left( \left( \Big\{ \Big\{ \mathsf{ct}_{i,j}^{(k)} \Big\}_{i \in [S], b \in \{0,1\}} \Big\}_{k \in [n]_0}, \mathsf{ct}_n \right), \left( \Big\{ \mathsf{lab}_{i, \mathsf{mpk}[i]}^{(0)} \Big\}, \Big\{ \mathcal{Q}^{(k)} \Big\}_{k \in [n-1]_0}, \mathcal{T} \right) \right)$$

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$ : The decryption algorithm takes as input a public parameters $\mathsf{pp}$, a secret key $\mathsf{sk}_{\mathsf{id}} = (\Big\{ \mathsf{mpk}^{(\mathsf{id}[1:j])} \Big\}_{j \in [n]_0}, \{\mathsf{sk}_{k,i}\}_{k \in [n-1]_0, i \in [S]}, \mathsf{sk}_n, s_{\mathsf{id}})$ for the identity $\mathsf{id}$ (and $n = |\mathsf{id}|$) and a ciphertext $\mathsf{ct}$ as follows

$$(\mathsf{ct}_0, \mathsf{ct}_1) = \left( \left( \Big\{ \Big\{ \mathsf{ct}_{i,j}^{(k)} \Big\}_{i \in [S], b \in \{0,1\}} \Big\}_{k \in [n]_0}, \mathsf{ct}_n \right), \left( \Big\{ \mathsf{lab}_i^{(0)} \Big\}, \Big\{ \mathcal{Q}^{(k)} \Big\}_{k \in [n-1]_0}, \mathcal{T} \right) \right)$$

For $k = 0$ to $n-1$,

1. it computes $\{\mathsf{ct}_{i,j}\}_{i \in [S], b \in \{0,1\}} \leftarrow \mathsf{GC.Eval}(\mathcal{Q}^{(k)}, \{\mathsf{lab}_i^{(k)}\}_{i \in [\lambda]})$.
2. it computes $\mathsf{lab}_i^{(k+1)} \leftarrow \mathsf{IBE.Dec}(\mathsf{pp}, \mathsf{sk}_{k,i}, \mathsf{ct}_{i,\beta_i}^{(k)}, \mathsf{ct}_{i,\beta_i})$ for all $i \in [S]$ where $\beta_i = \mathsf{mpk}^{(\mathsf{id}[1:k+1])}[i]$.

[4] It is not important to keep $\mathsf{sk}_n$ as a part of $\mathsf{sk}_{\mathsf{id}}$ because $s_{\mathsf{id}}$ can be used to compute $\mathsf{PRF.Eval}(s, \mathsf{id})$ by evaluating $\mathsf{PRF.Eval}(s_{\mathsf{id}}, \epsilon)$. However, we include it for clarity. For the syntax of the PRF, refer to Appendix A

It computes $\mathsf{ct} \leftarrow \mathsf{GC.Eval}(\mathcal{T}, \{\mathsf{lab}_i^{(n)}\}_{i\in[S]})$ and sets $m \leftarrow \mathsf{IBE.Dec}(\mathsf{pp}, \mathsf{sk}_n, \mathsf{ct}_n, \mathsf{ct})$. It returns $m$.

It important to note that both $\mathsf{KeyGen}$ and $\mathsf{Delegate}$ algorithms are *deterministic*. Also, from the correctness of the delegation of the PRF, we can easily show that $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}\,||\,\mathsf{id}') = \mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$.

**Correctness.** Let $(\mathsf{ct}_0, \mathsf{ct}_1) = \left(\left(\left\{\left\{\mathsf{ct}_{i,j}^{(k)}\right\}_{i\in[S],b\in\{0,1\}}\right\}_{k\in[n]_0}, \mathsf{ct}_n\right), \left(\left\{\mathsf{lab}_{i,\mathsf{mpk}[i]}^{(0)}\right\}, \{\mathcal{Q}^{(k)}\}_{k\in[n-1]_0}, \mathcal{T}\right)\right)$ be an encryption of message $m$ for the identity $\mathsf{id}$. The decryption process starts by evaluating the garbled circuit $\mathcal{Q}^{(0)}$ on $\{\mathsf{lab}_i^{(0)}\}_{i\in[\lambda]}$ where $\mathsf{lab}_i^{(0)} = \mathsf{lab}_{i,\mathsf{mpk}^{(\epsilon)}[i]}^{(0)}$. Therefore, from the correctness of the garbling scheme we have $\mathsf{ct}_{i,j} = \mathsf{IBE.Enc}_2(\mathsf{pp}, \mathsf{mpk}^{(\epsilon)}, \mathsf{id}[1]\,||\,i\,||\,j, \mathsf{lab}_{i,j}^{(1)})$ for all $i \in [\lambda], j \in \{0,1\}$. Next, the algorithm sets $\beta_i = \mathsf{mpk}^{(\mathsf{id}[1])}[i]$. Next, it computes $\mathsf{lab}_i^{(1)} \leftarrow \mathsf{IBE.Dec}(\mathsf{pp}, \mathsf{sk}_{0,i}, \mathsf{ct}_{i,\beta_i}^{(0)}, \mathsf{ct}_{i,\beta_i})$ for all $i \in [S]$. Now, from the correctness of the IBE scheme, we get $\mathsf{lab}_i^{(1)} = \mathsf{lab}_{i,\mathsf{mpk}^{(\mathsf{id}[1])}[i]}^{(1)}$. Repeating the same argument, we will obtain $\mathsf{lab}_i^{(n+1)} = \mathsf{lab}_{i,\mathsf{pke.pk}_{\mathsf{id}}[i]}^{T}$. Again, using the correctness of the garbling scheme, we obtain $\mathsf{ct} = \mathsf{IBE.Enc}(\mathsf{mpk}^{(\mathsf{id})}, m)$ by evaluating $\mathsf{ct} \leftarrow \mathsf{GC.Eval}(\mathcal{T}, \{\mathsf{lab}_i^{(n+1)}\}_{i\in[\lambda]})$. Finally, from the IBE scheme's correctness, we get $m$ by decrypting $\mathsf{ct}$ using $\mathsf{sk}_{\mathsf{id}}$.

**Efficiency.** The master public key $\mathsf{mpk}$ is a master public key of a WCIBE scheme for $\mathsf{poly}(\lambda)$ many identities. Therefore, the size of the master public key is $|\mathsf{mpk}| = \mathsf{poly}(\lambda)$.

The master secret key is a PRF key, therefore, $|\mathsf{msk}| = \mathsf{poly}(\lambda)$. A secret key (as well as delegated secret key) for an identity $\mathsf{id}$ of length $n$ consists of $n$ pairs of master public key-secret key of the WCIBE scheme and a delegated PRF key. Therefore $|\mathsf{sk}_{\mathsf{id}}| = n \cdot \mathsf{poly}(\lambda)$.

A ciphertext $\mathsf{ct}$ associated with a message $m$ and an identity $\mathsf{id}$ of length $n$ consists of a garbled circuit for $T_{m,r}(\cdot)$ whose size is $|m| \cdot \mathsf{poly}(\lambda)$, $n$ garbled circuits of $Q_{\beta,\{z_{i,b},r_{i,b}\}}(\cdot)$ of size $n \cdot \mathsf{poly}(\lambda)$, $\lambda$ many labels for a garbled circuit and $n+1$ ciphertexts associated with the WCIBE scheme. Therefore, $|\mathsf{ct}| = (|m|+n)\cdot\mathsf{poly}(\lambda)$.

## 4.2 Proof of Adaptive Anonymous Security

**Theorem 4.1.** Assuming $\mathsf{IBE}$ is a secure blind IBE scheme, $\mathsf{GC}$ is a secure garbling scheme and $\mathsf{PRF}$ is an adaptively secure PRF, then the above construction is an adaptively secure anonymous unbounded HIBE scheme.

*Proof.* We now proceed to sketch the proof that the above scheme is an adaptively secure anonymous unbounded HIBE scheme.

- Hybrid $H_0$: This is the original adaptively secure anonymous HIBE game where $b$ is the challenge bit. We will use $\mathsf{id}^* := \mathsf{id}_b^*$ and $m := m_b$ in the rest of the proof.
- Hybrid $H_1$: This is same as $H_0$ except that when $\mathcal{A}$ queries the oracle $O(\mathsf{msk}, \cdot)$ on $(\mathsf{id}, \mathsf{ind}, \mathsf{mode})$, the challenger does the following.
  - If $\mathsf{mode} = \mathsf{StoreKey}$, then the challenger does not generate the secret key but stores $(t, \mathsf{id}, \perp)$ and replies with $(t, \perp)$. It also updates $t := t + 1$.
  - If $\mathsf{mode} = \mathsf{DelegateKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}', \cdot)$ for some identity $\mathsf{id}'$. If no such tuple exists, it outputs $\perp$. Otherwise, it does not generate the delegated secret key but stores $(t, \mathsf{id}'\,||\,\mathsf{id}, \perp)$ and replies with $(t, \perp)$. It also updates $t := t + 1$.
  - If $\mathsf{mode} = \mathsf{OutputKey}$, then the challenger first checks if there exists a key tuple of the form $(\mathsf{ind}, \mathsf{id}, \cdot)$. If no such tuple exists or if $\mathsf{id}$ is a prefix of $\mathsf{id}^*$ (that is, $\mathsf{id} \preceq \mathsf{id}^*$), it outputs $\perp$. Otherwise, if $(\mathsf{ind}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ exists, it replies with $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$. Else, it will compute $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, stores it and outputs $(\mathsf{ind}, \mathsf{sk}_{\mathsf{id}})$.

- For $k = 0$ to $n - 1$ (where $n = |\mathsf{id}^*|$),
  - Hybrid $H_{2,k}$: In this hybrid, $(\mathcal{Q}^{(k)}, \left\{\mathsf{lab}_i^{(k)}\right\})$ are generated by using the simulation algorithm on the output of $Q_{\mathsf{id}^*[k+1], \left\{\mathsf{lab}_{i,j}^{(k+1)}, r_{i,j}^{(k+1)}\right\}}(\mathsf{mpk}^{(\mathsf{id}^*[1,k])})$.
- Hybrid $H_3$: In this hybrid, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ are generated by using the simulation algorithm on the output of $T_{m,r}(\mathsf{mpk}^{(\mathsf{id}^*)})$.
- Hybrid $H_4$: In this hybrid, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ is replaced with a random string.
- For $k = n - 1$ to $0$ (where $n = |\mathsf{id}^*|$),
  - Hybrid $H_{5,k}$: In this hybrid, $(\mathcal{Q}^{(k)}, \left\{\mathsf{lab}_i^{(k)}\right\})$ is replaced with a random string.

**Analysis:** Let $p_{\mathcal{A},H}$ denote probability of $\mathcal{A}$ outputting $b' = b$ in Hybrid $H$. We will show that this probability is almost the same in every game.

**Lemma 4.2.** For all PPT adversaries $\mathcal{A}$, $p_{\mathcal{A},H_0} = p_{\mathcal{A},H_1}$.

*Proof.* This follows directly from the fact that the distribution of keys returned by $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id} \,||\, \mathsf{id}')$ is identical to the distribution of keys generated by $\mathsf{Delegate}(\mathsf{sk}_{\mathsf{id}} \,||\, \mathsf{id}')$. □

We use the notation $H_{2,-1}$ to denote the hybrid $H_1$.

**Lemma 4.3.** Assuming that GC is a secure garbling scheme, PRF is a secure delegatable PRF scheme and IBE is a secure WCIBE scheme, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{2,k}} - p_{\mathcal{A},H_{2,k-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Lemma 3.3, where we construct a modified adversary $\mathcal{A}'$ that guesses an index $q$ and define multiple internal hybrids. First, we replace the PRF evaluation with a truly random value, then modify certain IBE ciphertexts to encrypt $\mathbf{0}$, revert back to the PRF evaluation, and finally use the simulation mode of the garbling scheme to generate the garbled circuit and necessary labels. Indistinguishability is argued based on the security of the GC, PRF, and IBE schemes. □

**Lemma 4.4.** Assuming that GC is a secure garbling scheme, PRF is a secure delegatable PRF scheme and IBE is a secure WCIBE scheme, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_3} - p_{\mathcal{A},H_{2,n-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Lemma 4.3. □

**Lemma 4.5.** Assuming that GC is a secure blind garbling scheme, PRF is a secure delegatable PRF, IBE is a secure WCIBE scheme, for all PPT adversarier $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Lemma 3.3. Consider the modified adversary $\mathcal{A}'$ that attempts to guess the index $q \in [Q+1]$ such that the first $\mathsf{Node}(\mathsf{pp}, \mathsf{id}^*, \cdot)$ is performed where $Q$ is the number of key queries made by $\mathcal{A}$. Our goal is to show that $|p_{\mathcal{A}',H_4} - p_{\mathcal{A}',H_3}| = \mathsf{negl}(\lambda)$. To achieve this, we will utilise the following internal hybrids. Recall that guess is the $q$-th distinct $v$ of length $n$ that the challenger would have visited through $\mathsf{Node}(\mathsf{pp}, \cdot, \cdot)$.

- Hybrid $H_{3,1}$: In this hybrid, rather than using $\mathsf{PRF.Eval}(s; \mathsf{guess})$ to generate $(\mathsf{mpk}^{(\mathsf{guess})}, \mathsf{msk}^{(\mathsf{guess})})$, it uses a truly random coins in IBE.Setup algorithm.

- Hybrid $H_{3,2}$: In this hybrid, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ are generated by using the simulation algorithm on the output of $T_{t,r}(\mathsf{mpk}^{(\mathsf{guess})})$ where $t$ is a randomly generated message.
- Hybrid $H_{3,3}$: In this hybrid, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ are generated by using the simulation algorithm on a random output string, i.e., $\mathsf{GC.Sim}(1^\lambda, 1^{|T|}, 1^\lambda, t)$ where $t$ is a random string of length $|\mathsf{ibe.ct}|$.
- Hybrid $H_{3,4}$: In this hybrid, $(\mathsf{mpk}^{(\mathsf{guess})}, \mathsf{msk}^{(\mathsf{guess})})$ are generated using the $\mathsf{PRF.Eval}(s; \mathsf{guess})$ algorithm.

**Claim 4.6.** Assuming that PRF is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{3,1}} - p_{\mathcal{A}',H_3}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the adaptive security of the delegatable PRF. □

**Claim 4.7.** Assuming that IBE is a secure CPA scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{3,2}} - p_{\mathcal{A}',H_{3,1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof follows directly from the CPA security of the IBE scheme. In hybrid $H_{3,1}$, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ are generated by using the simulation algorithm on the output of $T_{m,r}(\mathsf{mpk}^{(\mathsf{guess})})$, i.e., $\mathsf{IBE.Enc}_2(\mathsf{pp}, \mathsf{mpk}^{(\mathsf{guess})}, m; r)$. Whereas, in $H_{3,2}$, $m$ replaced by a truly random string $t$. It is easy to see that an adversary distinguish $H_{3,2}$ from $H_{3,1}$ can be used to break CPA security of IBE scheme. □

**Claim 4.8.** Assuming that IBE is a secure WCIBE scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{3,3}} - p_{\mathcal{A}',H_{3,2}}| = \mathsf{negl}(\lambda)$.

*Proof.* This follows from the blindness property of the IBE scheme. In hybrid $H_{3,2}$, $(\mathcal{T}, \left\{\mathsf{lab}_i^{(n)}\right\})$ are generated by using the simulation algorithm on the output of $T_{t,r}(\mathsf{mpk}^{(\mathsf{guess})})$, i.e., $\mathsf{IBE.Enc}_2(\mathsf{pp}, \mathsf{mpk}^{(\mathsf{guess})}, t; r)$ where $t$ is a random string. Whereas, in $H_{3,3}$, this is replaced by a truly random string. It is easy to see that an adversary distinguish $H_{3,3}$ from $H_{3,2}$ can be used to break the blindness property of the IBE scheme. □

**Claim 4.9.** Assuming that PRF is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_{3,4}} - p_{\mathcal{A}',H_{3,3}}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the adaptive security of the delegatable PRF. □

**Claim 4.10.** Assuming that GC is a secure blind garbling scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|p_{\mathcal{A}',H_4} - p_{\mathcal{A}',H_{3,4}}| = \mathsf{negl}(\lambda)$.

*Proof.* This follows directly from the blindness property of the GC scheme. □

Combining above claims using triangular inequality, we obtain $|p_{\mathcal{A}',H_4} - p_{\mathcal{A}',H_3}| = \mathsf{negl}(\lambda)$. Using the fact that $p_{\mathcal{A}',H_4} = p_{\mathcal{A},H_4}/(Q+1)$, $p_{\mathcal{A}',H_3} = p_{\mathcal{A},H_3}/(Q+1)$ and $Q$ is a polynomial in $\lambda$, we obtain $|p_{\mathcal{A},H_4} - p_{\mathcal{A},H_3}| = \mathsf{negl}(\lambda)$. □

We use the notation $H_{5,-1}$ to denote the hybrid $H_4$.

**Lemma 4.11.** Assuming that GC is a secure blind garbling scheme, PRF is a secure delegatable PRF, IBE is a secure WCIBE scheme, for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A},H_{5,k}} - p_{\mathcal{A},H_{5,k-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The proof is similar to Lemma 3.3 where the internal hybrids are as follows. Recall that guess is the $q$-th distinct $v$ of length $k$ that the challenger would have visited through $\mathsf{Node}(\mathsf{pp}, \cdot, \cdot)$.

- Hybrid $H_{5,k,1}$: In this hybrid, rather than using $\mathsf{PRF.Eval}(s, \mathsf{guess})$ to generate $\mathsf{mpk}^{(\mathsf{guess})}, \mathsf{msk}^{(\mathsf{guess})}$, it uses a truly random coins in IBE.Setup algorithm.

- Hybrid $H_{5,k,2}$: In this hybrid, $(\mathcal{Q}^{(k)}, \left\{\mathsf{lab}_i^{(k)}\right\})$ are generated by using the simulation algorithm on a random output string, i.e., $\mathsf{GC.Sim}(1^\lambda, 1^{|Q|}, 1^\lambda, t)$ where $t$ is a random string of length $2\lambda \cdot |\mathsf{ibe.ct}|$.
- Hybrid $H_{5,k,3}$: In this hybrid, $\mathsf{mpk}^{(\mathsf{guess})}, \mathsf{msk}^{(\mathsf{guess})}$ are generated using the $\mathsf{PRF.Eval}(s, \mathsf{id}^*[1,k])$ in $\mathsf{IBE.Setup}$ algorithm.

**Claim 4.12.** Assuming that $\mathsf{PRF}$ is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A}',H_{5,k-1,1}} - p_{\mathcal{A}',H_{5,k-1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the adaptive security of the delegatable PRF. □

**Claim 4.13.** Assuming that $\mathsf{IBE}$ is a secure WCIBE scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A}',H_{5,k-1,2}} - p_{\mathcal{A}',H_{5,k-1,1}}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the blindness property of the IBE scheme. □

**Claim 4.14.** Assuming that $\mathsf{PRF}$ is a secure delegatable PRF scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A}',H_{5,k-1,3}} - p_{\mathcal{A}',H_{5,k-1,2}}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the adaptive security of the delegatable PRF. □

**Claim 4.15.** Assuming that $\mathsf{GC}$ is a secure blind garbling scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, k \in [n-1]_0$, $|p_{\mathcal{A}',H_{5,k}} - p_{\mathcal{A}',H_{5,k-1,3}}| = \mathsf{negl}(\lambda)$.

*Proof.* The follows from the blindness property of the garbling scheme. □

Using the above claim and the triangular inequality, we get that $|p_{\mathcal{A},H_{5,k}} - p_{\mathcal{A},H_{5,k-1}}| = \mathsf{negl}(\lambda)$. □

Observe that in the last hybrid, the challenge ciphertext $\mathsf{ct}^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*)$ does not contain any information of $\mathsf{id}_b^*$ or $m_b$. This is because $\mathsf{ct}_0^*$ are generated without using $\mathsf{id}_b^*$ or $m_b$ whereas, all the entities in $\mathsf{ct}_1^*$ have been replaced with a truly random string. Therefore, the probability of any adversary $\mathcal{A}$ that wins this hybrid game is exactly $\frac{1}{2}$. Using this fact, the above lemmas and triangular inequality, for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $p_{\mathcal{A},0} \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. □

Using Theorem 2.8 and Theorem 4.1, we have the following theorem.

**Theorem 4.16.** Assuming the hardness of $\mathsf{CDH}$, there exists adaptively secure anonymous unbounded HIBE scheme.

# References


[AB09] Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. *Manuscript, July*, 3, 2009.

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h) ibe in the standard model. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 553–572. Springer, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 98–115. Springer, 2010.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Annual Cryptology Conference*, pages 657–677. Springer, 2015.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 223–238. Springer, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Annual International Cryptology Conference*, pages 443–459. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 440–456. Springer, 2005.

[BCG+17] Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev. Hierarchical functional encryption. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 506–522. Springer, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 533–556. Springer, 2014.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption-without pairings. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 647–657. IEEE, 2007.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *International workshop on public key cryptography*, pages 501–519. Springer, 2014.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (hierarchical) identity-based encryption from affine message authentication. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 408–425. Springer, 2014.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–564. Springer, 2018.

[BM23] Zvika Brakerski and Stav Medina. Limits on adaptive security for attribute-based encryption. *Cryptology ePrint Archive*, 2023.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *International algorithmic number theory symposium*, pages 48–63. Springer, 1998.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: unbounded attributes and semi-adaptive security. In *Annual International Cryptology Conference*, pages 363–384. Springer, 2016.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Annual International Cryptology Conference*, pages 290–307. Springer, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*, pages 280–300. Springer, 2013.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system abe in prime-order groups via predicate encodings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 595–624. Springer, 2015.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 255–271. Springer, 2003.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25:601–639, 2012.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding: 8th IMA International Conference Cirencester, UK, December 17–19, 2001 Proceedings 8*, pages 360–363. Springer, 2001.

[CW13] Jie Chen and Hoeteck Wee. Fully,(almost) tightly secure ibe and dual system groups. In *Annual Cryptology Conference*, pages 435–460. Springer, 2013.

[DG17a] Nico Döttling and Sanjam Garg. From selective ibe to full ibe and selective hibe. In *Theory of Cryptography Conference*, pages 372–408. Springer, 2017.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Annual international cryptology conference*, pages 537–569. Springer, 2017.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In *IACR International Workshop on Public Key Cryptography*, pages 3–31. Springer, 2018.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*, pages 601–620. Springer, 2015.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained prfs. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 82–101. Springer, 2014.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, pages 445–464. Springer, 2006.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[GGL24] Rachit Garg, Rishab Goyal, and George Lu. Dynamic collusion functional encryption and multi-authority attribute-based encryption. In *IACR International Conference on Public-Key Cryptography*, pages 69–104. Springer, 2024.

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

[GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *Theory of Cryptography Conference*, pages 437–456. Springer, 2009.

[GHM+19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In *IACR international workshop on public key cryptography*, pages 63–93. Springer, 2019.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: removing private-key generator from ibe. In *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part I 16*, pages 689–718. Springer, 2018.

[GKK+25] Rishab Goyal, Fuyuki Kitagawa, Venkata Koppula, Ryo Nishimaki, Mahesh Sreekumar Rajasree, and Takashi Yamakawa. Non-committing identity based encryption: Constructions and applications. In *IACR International Conference on Public-Key Cryptography*, pages 158–193. Springer, 2025.

[GKRV24] Rishab Goyal, Venkata Koppula, Mahesh Sreekumar Rajasree, and Aman Verma. Incompressible functional encryption. *Cryptology ePrint Archive*, 2024.

[GKVW20] Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In *Theory of Cryptography - TCC 2020 - 18th International Conference*, 2020.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography Conference*, pages 361–388. Springer, 2016.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–621. IEEE, 2017.

[GLW21] Rishab Goyal, Jiahui Liu, and Brent Waters. Adaptive security via deletion in attribute-based encryption: Solutions from search assumptions in bilinear groups. In *Theory and Application of Cryptology and Information Security - ASIACRYPT 2021 - 27th International Conference*, 2021.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98. ACM, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8*, pages 548–566. Springer, 2002.

[GSW21] Rishab Goyal, Ridwan Syed, and Brent Waters. Bounded collusion abe for tms from ibe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 371–402. Springer, 2021.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I 40*, pages 621–651. Springer, 2020.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

[GY25] Rishab Goyal and Saikumar Yadugiri. Delegatable ABE with full security from witness encryption. Cryptology ePrint Archive, Paper 2025/407, 2025.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In *Annual International Cryptology Conference*, pages 149–178. Springer, 2016.

[HKK23] Dennis Hofheinz, Julia Kastner, and Karen Klein. The power of undirected rewindings for adaptive security. In *Annual International Cryptology Conference*, pages 725–758. Springer, 2023.

[HKK+24] Goichiro Hanaoka, Shuichi Katsumata, Kei Kimura, Kaoru Takemure, and Shota Yamada. Tighter adaptive ibes and vrfs: Revisiting waters' artificial abort. In *Theory of Cryptography Conference*, pages 124–155. Springer, 2024.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *International conference on the theory and applications of cryptographic techniques*, pages 466–481. Springer, 2002.

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of yao's garbled circuits. In *Theory of Cryptography Conference*, pages 433–458. Springer, 2016.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, pages 146–162. Springer, 2008.

[LOS+10] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 62–91. Springer, 2010.

[LP19] Roman Langrehr and Jiaxin Pan. Tightly secure hierarchical identity-based encryption. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 436–465. Springer International Publishing, 2019.

[LP20] Roman Langrehr and Jiaxin Pan. Unbounded hibe with tight security. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 129–159. Springer, 2020.

[LW10] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography Conference*, pages 455–479. Springer, 2010.

[LW11] Allison Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 547–567. Springer, 2011.

[LW14] Allison Lewko and Brent Waters. Why proving hibe systems secure is difficult. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 58–76. Springer, 2014.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, 2005.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*, pages 47–53. Springer, 1985.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[SW08] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *International Colloquium on Automata, Languages, and Programming*, pages 560–578. Springer, 2008.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-cnf from lwe. In *Annual International Cryptology Conference*, pages 62–85. Springer, 2019.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 114–127. Springer, 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Annual international cryptology conference*, pages 619–636. Springer, 2009.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Annual Cryptology Conference*, pages 678–697. Springer, 2015.

[WC23] Huangting Wu and Sherman SM Chow. Anonymous (hierarchical) identity-based encryption from broader assumptions. In *International Conference on Applied Cryptography and Network Security*, pages 366–395. Springer, 2023.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 600–611, 2017.

# A Proof of Theorem 2.3

The construction is identical to that provided in the work of Garg and Döttling [DG17a], and we present it here for completeness. For convenience, we will assume that the length of keys and the outputs of the PRF is $\lambda$ whereas the length of the inputs is unbounded. Let $\mathsf{PRG}$ be a length-tripling pseudorandom generator with seed length $\lambda$. We write $\mathsf{PRG}(s) = \mathsf{PRG}_0(s) \,||\, \mathsf{PRG}_1(s) \,||\, \mathsf{PRG}_2(s)$ where $|\mathsf{PRG}_i(s)| = \lambda$, for all $i \in \{0,1,2\}$. The construction for the strong-adaptively secure PRF is as follows.

- $\mathsf{Setup}(1^\lambda)$ : The setup algorithm outputs a random key $k \leftarrow \{0,1\}^\lambda$.
- $\mathsf{Eval}(k, x)$ : The evaluation on input a key $k$ and string $x$ such that $|x| = d$, outputs

$$\mathsf{PRG}_2(\mathsf{PRG}_{x_d}(\mathsf{PRG}_{x_{d-1}}(\ldots(\mathsf{PRG}_{x_1}(k))\ldots))).$$

- $\mathsf{Delegate}(k, x)$ : On input a key $k$ and string $x$ such that $|x| = d$, the delegation algorithm outputs

$$\mathsf{PRG}_{x_d}(\mathsf{PRG}_{x_{d-1}}(\ldots(\mathsf{PRG}_{x_1}(k))\ldots)).$$

**Correctness.** Correctness follows immediately from the construction.

**Security.** As the proof is quite similar to the proof presented by Hofheinz, Kastner, and Klein [HKK23], we only provide the specifics where the proof varies, leaving it to the readers to refer to [HKK23] for the proof details (the changes are marked in red). The main theorem is as follows.

**Theorem A.1** (analogous to [HKK23, Theorem 1])**.** Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be a PRG. Then, for every $\mathsf{PRF}$ adversary $\mathcal{A}$ that runs in time $t_\mathcal{A}$, makes at most $Q_\mathcal{A}$ queries, and the challenge query is of length $d$, for every $\gamma \in (0,1]$, there is a $\mathsf{PRG}$ adversary $\mathcal{B}$ that runs in time $t_\mathcal{B}$, makes at most $Q_\mathcal{B}$ oracle queries[5] to the PRG challenger, such that

$$\mathsf{Adv}^{\mathsf{PRG}}_\mathcal{B} \geq \frac{1}{2(d+1)} \cdot \left(\mathsf{Adv}^{\mathsf{PRF}}_\mathcal{A} - \gamma\right), \text{ where}$$

$$\mathcal{T} \leq ((d+3) \cdot Q_\mathcal{A} + 2), \quad t_\mathcal{B} \leq (2 \cdot \ln(2 \cdot \mathcal{T}/\gamma)) \cdot t_\mathcal{A}, \quad Q_\mathcal{B} \leq 2 \cdot \ln(2 \cdot \mathcal{T}/\gamma) \cdot \mathcal{T}^3/\gamma$$

Compared to [HKK23, Theorem 1], the only changes are (a) we need a length tripling PRG instead of a length doubling one, (b) the tree depth $d$ is not fixed, instead it depends on the adversary's challenge input's length, (c) we need to replace $d$ with $(d+1)$ in the advantage bound, and the bound on $\mathcal{T}$ since there are $(d+1)$ levels of PRG computation in our setting. Using the theorem, if $\mathsf{Adv}^{\mathsf{PRF}}_\mathcal{A} \geq \epsilon$ (where $\epsilon$ is some non-negligible function), then we can use $\gamma = \epsilon/2$ to get a $\mathsf{PPT}$ algorithm that breaks PRG security with non-negligible advantage.

[5] $\mathcal{B}$ has access to an oracle that, on each invocation, returns either a freshly generated truly random string or $\mathsf{PRG}(s)$ for a newly sampled random seed $s$.

**Proof Sketch.** We recall two notations from [HKK23] which will be needed for our proof sketch. First, the view of an $\mathcal{A}$, denoted by $\mathsf{view} = (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$, captures the PRG evaluations that are triggered by each delegation and evaluation query made by $\mathcal{A}$. Each delegation query $x$ results in the generation of $\left\{(k_{x'||0}, k_{x'||1}, k_{x'||2})\right\}_{x' \prec x}$ by the challenger, whereas, each evaluation query $x$ results in the generation of $\left\{(k_{x'||0}, k_{x'||1}, k_{x'||2})\right\}_{x' \preceq x}$ by the challenger. These are denoted by $\{(\mathsf{PRG}, x')\}_{x'}$. The term $\mathsf{lastpre}_t(\mathsf{view})$ on a $\mathsf{view} = (\mathsf{ev}_1, \ldots, \mathsf{ev}_{\mathcal{T}})$ is the largest index $t' \leq t$ such that event $\mathsf{ev}_{t'}$ (is a $(\mathsf{PRG}, x)$ event) is on the path along $x^*$, i.e., $x \prec x^*$.

In [HKK23], the authors define a sequence of $d+1$ hybrid games, where Game 0 corresponds to the adaptive security game, and in Game $d$, the adversary's advantage is 0. We will follow the same hybrids. The hybrids are defined using a sampler (defined in Algorithm 1). This is identical to Algorithm 12 in [HKK23], except that we need to sample three keys in Step 8 instead of two keys. At a high level, the difference between the $i^{th}$ and $(i+1)^{th}$ hybrids is that the sampler *strategically* replaces the PRG evaluation at level $i+1$ along the path of $x^*$ with truly random values, while ensuring that the newly sampled view remains consistent with the distribution associated with the adversary's view.

**Algorithm 1** Sampler $\mathcal{D}^{\mathsf{view}}_{i,b_{\mathtt{prf}}}$ (analogous to [HKK23, Algorithm 12])

1: $\mathsf{view}_0 \leftarrow \mathcal{D}^{\mathsf{view}}_{\mathtt{prf},b_{\mathtt{prf}}}$
2: $\mathcal{T} := \mathsf{len}(\mathsf{view}_0)$
3: **for** $t := 1$ to $\mathcal{T}$ **do**
4: Write $\mathsf{view}_{t-1} = (\mathsf{ev}_{t-1,0}, \ldots, \mathsf{ev}_{t-1,\mathcal{T}})$
5: **repeat**
6: Rewind adversary to point $t$
7: **if** $\mathsf{ev}_{t-1,t} = (\mathsf{PRG}, x)$ with $|x| \leq i$ and $\mathsf{lastpre}_t(\mathsf{view}_{t-1}) = t$ **then**
8: Sample fresh $k_{x||0}, k_{x||1}, k_{x||2} \leftarrow \{0,1\}^\lambda$
9: Resample from point $t+1$ to obtain
$\mathsf{view}_t = (\mathsf{ev}_{t-1,0}, \ldots, \mathsf{ev}_{t-1,t-1}, \mathsf{ev}_{t-1,t}, \mathsf{ev}_{t,t+1}, \ldots, \mathsf{ev}_{t,\mathcal{T}})$
10: **else**
11: Resample from point $t$ to obtain
$\mathsf{view}_t = (\mathsf{ev}_{t-1,0}, \ldots, \mathsf{ev}_{t-1,t-1}, \mathsf{ev}_{t,t}, \ldots, \mathsf{ev}_{t,\mathcal{T}})$
12: **end if**
13: **until** $\mathsf{lastpre}_t(\mathsf{view}_t) = \mathsf{lastpre}_t(\mathsf{view}_{t-1})$ and $\mathsf{len}(\mathsf{view}_t) = \mathsf{len}(\mathsf{view}_{t-1})$
14: **end for**
15: Return $\mathsf{view}_{\mathcal{T}}$

# B Anonymous HIBE using Lockable Obfuscation

In this section, we revisit the concept of lockable obfuscation introduced by Goyal, Koppula, and Waters [GKW17] and present their transformation in the HIBE setting, (mostly) verbatim from [GKW17]. The transformation takes a regular HIBE scheme and applies lockable obfuscation to derive an anonymous HIBE scheme.

## B.1 Lockable Obfuscation

Let $n, m, d$ be polynomials, and $\mathcal{C}_{n,m,d}(\lambda)$ be the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bit input and $m(\lambda)$ bit output. A lockable obfuscator for $\mathcal{C}_{n,m,d}$ consists of algorithms $\mathsf{Obf}$ and $\mathsf{Eval}$ with the following syntax. Let $\mathcal{M}$ be the message space.

- $\mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \to \widetilde{P}$. The obfuscation algorithm is a randomized algorithm that takes as input the

security parameter $\lambda$, a program $P \in \mathcal{C}_{n,m,d}$, message $\mathsf{msg} \in \mathcal{M}$ and 'lock string' $\alpha \in \{0,1\}^{m(\lambda)}$. It outputs a program $\widetilde{P}$.

- $\mathsf{Eval}(\widetilde{P}, x) \to y \in \mathcal{M} \cup \{\perp\}$. The evaluator is a deterministic algorithm that takes as input a program $\widetilde{P}$ and a string $x \in \{0,1\}^{n(\lambda)}$. It outputs $y \in \mathcal{M} \cup \{\perp\}$.

**Correctness** For correctness, we require that if $P(x) = \alpha$, then the obfuscated program $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha)$, evaluated on input $x$, outputs $\mathsf{msg}$, and if $P(x) \neq \alpha$, then $\widetilde{P}$ outputs $\perp$ on input $x$.

**Security** Let $n, m, d$ be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $\mathcal{C}_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure if there exists a PPT simulator $\mathtt{sim}$ such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following function is bounded by $\mathsf{negl}(\cdot)$:

$$\left| \Pr \left[ \mathcal{A}_1(\tilde{P}_b, \mathsf{state}) = b : \begin{array}{c} (P \in \mathcal{C}_{n,m,d}, \mathsf{msg} \in \mathcal{M}, \mathsf{state}) \leftarrow \mathcal{A}_0(1^\lambda) \\ b \leftarrow \{0,1\}, \alpha \leftarrow \{0,1\}^{m(\lambda)} \\ \tilde{P}_0 \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{msg}, \alpha) \\ \tilde{P}_1 \leftarrow \mathtt{sim}(1^\lambda, 1^{|P|}, 1^{|\mathsf{msg}|}) \end{array} \right] - \frac{1}{2} \right|$$

## B.2 Transformation

Let $\mathsf{HIBE} = (\mathsf{HIBE.Setup}, \mathsf{HIBE.KeyGen}, \mathsf{HIBE.Delegate}, \mathsf{HIBE.Enc}, \mathsf{HIBE.Dec})$ be a hierarchical identity based encryption scheme for message spaces $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ with decryption circuit of depth $d(\lambda)$ and secret key space $\{0,1\}^{\ell(\lambda)}$. Also, let $\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ be a lockable obfuscator for circuit class $\mathcal{C}_{\ell,k,d}$ (i.e., the class of depth $d(\lambda)$ circuits with $\ell(\lambda)$ bit input and $k(\lambda)$ bit output). For simplicity, assume that $\mathcal{M}_\lambda = \{0,1\}^{k(\lambda)}$. Below we describe our construction. For notational convenience, let $\ell = \ell(\lambda), k = k(\lambda)$ and $d = d(\lambda)$.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm runs $\mathsf{HIBE.Setup}$ to generate master public key and master secret key as $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{HIBE.Setup}(1^\lambda)$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm runs $\mathsf{HIBE.KeyGen}$ to generate the secret key as $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{HIBE.KeyGen}(\mathsf{msk}, \mathsf{id})$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'}$. The delegation generation algorithm runs the $\mathsf{HIBE.Delegate}$ algorithm to generate the delegated secret key as $\mathsf{sk}_{\mathsf{id}\,||\,\mathsf{id}'} \leftarrow \mathsf{HIBE.Delegate}(\mathsf{sk}_{\mathsf{id}}, \mathsf{id}')$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm chooses a random string $\alpha \leftarrow \{0,1\}^k$ and encrypts it as $\mathsf{ct}_\alpha \leftarrow \mathsf{HIBE.Enc}(\mathsf{mpk}, \mathsf{id}, \alpha)$. Next, it obfuscates the decryption circuit with message $m$ and the lock $\alpha$ as $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, \mathsf{HIBE.Dec}(\cdot, \mathsf{ct}_\alpha), m, \alpha)$. Finally, it outputs the ciphertext as $\mathsf{ct} = \widetilde{P}$.
- $\mathsf{Dec}(\mathsf{sk}_C, \mathsf{ct}) \to m$ or $\perp$ . Let $\mathsf{ct} = \widetilde{P}$. The decryption algorithm evaluates the obfuscated program on input $\mathsf{sk}_C$, and outputs $\mathsf{Eval}(\widetilde{P}, \mathsf{sk}_C)$.

**Correctness and Efficiency.** Correctness follows directly from correctness of underlying objects. Moreover, we can note that since the since of $\mathsf{HIBE.Dec}(\cdot, \mathsf{ct}_\alpha)$ circuit is fixed at encryption time, therefore we could lockable obfuscation for appropriate depth circuit. Thus, even if the underlying HIBE schee supports unbounded length identities, then so will our anonymous HIBE scheme.

**Security.** The proof of security follows from adaptive security on underlying HIBE and lockable obfuscation security. Basically, first one could switch $\mathsf{ct}_\alpha$ to encrypt a garbage value instead of $\alpha$. This follows from adaptive security of underlying HIBE. Then we can use lockable obfuscation security to simulate $\widetilde{P}$ since $\alpha$ is no longer used anywhere else. Thus, the final HIBE ciphertext is simply a simulated program, which is independent of recipient $\mathsf{id}$. This concludes the adaptive security proof for the above anonymous HIBE scheme.

**Theorem B.1.** Assuming the hardness of LWE, there exists adaptively secure anonymous unbounded HIBE scheme.